TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Systemarchitektur, Professur für Rechnernetze

Master-Arbeit

ENTWICKLUNG EINES GAMIFICATION-KONZEPTS FÜR CROWD-BASIERTE SOFTWAREENTWICKLUNG

Felix Hanspach
Geboren am: 30. April 1990 in Dresden
Matrikelnummer: 3734637
Immatrikulationsjahr: 2010

zur Erlangung des akademischen Grades

Master of Science (M.Sc.)

Betreuer
Dr. Tenshi Hara
Dr. Iris Braun
Betreuender Hochschullehrer
Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill

Eingereicht am: 21. August 2017

TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Systemarchitektur Professur Rechnernetze

Aufgabenstellung für die Master-Arbeit

THEMA: Entwicklung eines Gamefication-Konzepts für Crowd-basierte Softwareentwicklung

Name:	Hanspach, Felix	Studiengang:	Master Informatik (PO 2010)
Matrikel-Nummer:	3734637	Projekt/Fokus:	Service and Cloud Computing
verantwortlicher Hochschullehrer:	Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill		
involvierte Mitarbeit	Dr.-Ing. Tenshi Hara, Dr.-Ing. Iris Braun		
Beginn am:	3.2.2017	einreichen bis:	14.7.2017

ZIELSTELLUNG

Motivation ist ein zentraler Baustein für Crowdsourcing. Wenn es einem Crowdsourcing-Anbieter nicht gelingt seine Dienstleister zu motivieren, so wird die kritische Crowd-Größe nicht erreicht. Neben monetären Anreizen hat sich bereits die Verwendung spieltypischer Elemente, sogenanntes Gamification, als großer Motivationswerkzeug bewährt. Hierbei werden Erfahrungspunkte, Erfolge, Ranglisten, etc. aus dem von dem Spiel-Kontext auf einen realen Kontext übertragen. Neben der Motivation, Crowdsourcing-Aufgaben zu übernehmen und zu lösen, bietet Gamification weiterhin eine Möglichkeit bestimmtes Nutzerverhalten stärker zu belohnen. So kann der Anbieter bestimmtes Verhalten seiner Nutzer beeinflussen. Zusätzlich können Crowdsourcing-Anbieter, durch die hinter der Gamification versteckten Metriken, Nutzer besser einordnen. Gamification ist bereits sowohl im Bereich der Softwareentwicklung, als auch im Bereich des Crowdsourcing verbreitet.

Ziel der Arbeit ist ein Konzept, welches die speziellen Anforderungen Crowd-basierter Softwareentwicklung durch Gamification unterstützt. Dieses soll einerseits für den Nutzer motivierend sein, andererseits zu wertvollen Metriken für den Crowdsourcing-Anbieter führen.

Um das Konzept zu erarbeiten, soll mittels eine umfangreiche State-of-the-Art-Analyse, insbesondere im Gebiet der Gamifacation Metrics, geklärt werden, welche Metriken für eine Motivation durch Gamification essentiell sind, und welche Metriken für den Crowdsourcing-Anbieter notwendig oder hilfreich sind. Die Definition einer Schnittmenge ist sinnvoll. Auf Basis der definierten Metriken soll anschließend geklärt werden, was im Kontext von Crowd-basierter Softwareentwicklung belohnenswert ist, insbesondere aber, wie dadurch die abgegebene Codequalität verbessert werden kann. Abschließen soll sich das Konzept der Belohnungszuweisung zuwenden und sinnvolle Darstellungs- und Zuteilungswerkzeuge eruieren.

Das Konzept in Form einer prototypischen Plattform umgesetzt werden. Hierbei soll besonderes Augenmerk auf die Modularität und Erweiterbarkeit gelegt werden, sodass eine Erweiterung mithilfe von Crowd-basierter-Softwareentwicklung unproblematisch möglich ist.

SCHWERPUNKTE

- Analyse und Auswahl bestehender Gamification-Methoden und -Metriken,
- Definieren von Anforderungen,
- Definieren von Bewertungskriterien für die Anforderungserfüllung,
- Konzept zur Übertragung auf Crowd-basierter Softwareentwicklung,
- prototypische Umsetzung des Konzepts,
- Evaluation und Auswertung der Ergebnisse.

Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill
(verantwortlicher Hochschullehrer)

SELBSTSTÄNDIGKEITSERKLÄRUNG

Hiermit versichere ich, dass ich die vorliegende Masterarbeit mit dem Titel **„Entwicklung eines Gamification-Konzepts für Crowd-basierte Softwareentwicklung“** selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie alle wörtlich oder sinngemäß übernommenen Gedanken und Zitate als solche kenntlich gemacht habe. Ich erkläre ferner, dass ich die vorliegende Arbeit an keiner anderen Stelle als Prüfungsarbeit eingereicht habe oder einreichen werde.

Felix Hanspach
Dresden, 21.08.2017

INHALTSVERZEICHNIS

1. Einleitung 7
   1.1. Motivation . . . 7
   1.2. Problemstellung . . . 9
   1.3. Zielsetzung . . . 10
   1.4. Struktur dieser Arbeit . . . 10

2. Verwandte Arbeiten / State of the Art 11
   2.1. Gamification . . . 11
      2.1.1. Begriff . . . 11
      2.1.2. Psychologischer Einfluss durch Gamification . . . 13
      2.1.3. Typische Spielelemente . . . 13
      2.1.4. Achievements/Erfolge . . . 15
      2.1.5. Erfahrungspunkte und Level . . . 16
      2.1.6. Probleme von Gamification . . . 17
   2.2. Gamification Frameworks . . . 17
      2.2.1. Octalysis . . . 17
      2.2.2. Spielertypen nach Bartle und Personas . . . 19
   2.3. Gamification Services . . . 20
      2.3.1. OpenBadges . . . 21
      2.3.2. Bunchball Nitro . . . 22
      2.3.3. Badgeville . . . 23
      2.3.4. getbadges . . . 24
   2.4. Zusammenfassung . . . 25
   2.5. Anforderungsanalyse . . . 26
      2.5.1. Funktionale Anforderungen . . . 26
      2.5.2. Nicht-Funktionale Anforderungen . . . 28

3. Kontext 29
   3.1. Crowdsourcing Plattform . . . 29
      3.1.1. Softwareentwicklungsprozess innerhalb der Plattform . . . 29
      3.1.2. Crowd-Entwickler . . . 31
      3.1.3. Datengrundlage . . . 32

4. Konzept 33
4.1. Gamification Konzept . . . 33
4.1.1. Ziele des Gamification-Ansatzes . . . 33
4.1.2. Performance-Metriken von Crowd-Entwicklern . . . 34
4.1.3. Erfahrungspunkte und Level . . . 38
4.1.4. Erfolge . . . 40
4.1.5. Ranglisten . . . 43
4.1.6. Zusammenfassung . . . 44
4.2. Implementierungskonzept . . . 45
4.2.1. Rohdatenbank . . . 47
4.2.2. Processing Function . . . 48
4.2.3. Live Komponente . . . 49
4.2.4. Zusammenfassendes Beispiel . . . 50

5. Implementierung 53
5.1. Vorgehen und Technologieauswahl . . . 53
5.2. PGE-Leader . . . 54
5.2.1. Implementierung des PGE-Leaders . . . 54
5.3. Rohdatenbank . . . 59
5.3.1. Trigger . . . 59
5.4. Processing Functions . . . 59
5.4.1. Funktionsweise . . . 59
5.4.2. Implementierung . . . 60
5.4.3. Deployment . . . 60
5.4.4. Kommunikation . . . 60
5.4.5. Sicherheit . . . 60
5.4.6. Beispiel . . . 61
5.5. Live-Komponente und Frontend . . . 62
5.5.1. Sicherheit . . . 62
5.5.2. Frontend . . . 63
5.6. Infrastruktur und Deployment . . . 64

6. Evaluation 67
6.1. Technische Evaluation . . . 67
6.1.1. Evaluation der aufgestellten Anforderungen . . . 67
6.1.2. Nicht-Funktionale Anforderungen . . . 70
6.1.3. Laufzeitmessungen . . . 71
6.1.4. Erweiterbarkeit . . . 72
6.1.5. Sicherheit . . . 73
6.2. Nutzerevaluation . . . 73
6.2.1. Durchführung . . . 73
6.2.2. Ergebnisse . . . 76
6.2.3. Octalysis Ergebnisse . . . 79
6.2.4. Bewertung und Fazit . . . 80
6.2.5. Verbesserte Farbwahl . . . 81

7. Zusammenfassung und Ausblick 83
7.1. Zusammenfassung 83
7.1.1. Gamification-Konzept 83
7.1.2. Gamification Service 84
7.2. Ausblick 85

Literatur 87

Abbildungsverzeichnis 91

Tabellenverzeichnis 93

Quellcodeverzeichnis 95

A. Personas i

B. Evaluationsergebnisse v
B.1. Ergebnisse vi
B.1.1. Proband 1 vi
B.1.2. Proband 2 vii
B.1.3. Proband 3 viii
B.1.4. Proband 4 ix
B.1.5. Proband 5 x
B.1.6. Proband 6 xi
B.1.7. Proband 7 xii

C. Vollständiger Fragebogen der Evaluation xiii
C.1. Online-Fragebogen xiii

1. EINLEITUNG

1.1. MOTIVATION

Eine große Anzahl Nutzer zu erreichen und zu binden, ist eines der wichtigsten Erfolgskriterien von Webplattformen. Eine gute User Experience ist hierbei eine wichtige Voraussetzung für den Erfolg eines Produkts (Jetter und Gerken 2006). Die Verwendung von Elementen aus Spielen in einem fremden Kontext hat sich als ein Verfahren herausgestellt, um dieses Ziel zu erreichen (Deterding 2015). Diese Methode ist seit dem Jahr 2010 als ***Gamification*** bekannt und wird seitdem in vielen wissenschaftlichen (z.B. Domínguez u. a. (2013), Jones u. a. (2014) und (Hamari und Koivisto 2015)) und industriellen[1][2][3][4] (z.B. Montola u. a. (2009), Herzig (2014)) Projekten angewendet. In diesen wird durch Gamification ein positiver Effekt erzielt (Hamari u. a. 2014).

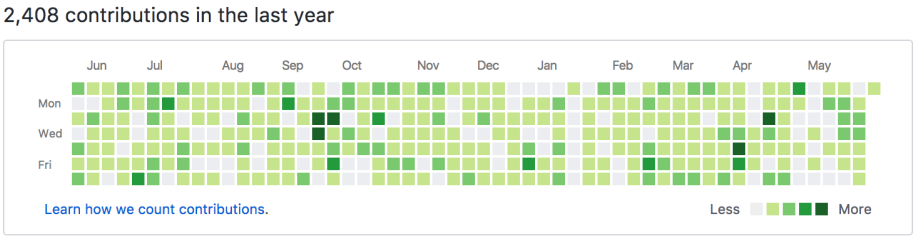


Abbildung 1.1.: Die Darstellung von Contributions auf Github ist ebenfalls eine Art der Gamification. Visualisiert wird die taggenaue Häufigkeit der Beiträge in Github-Projekten.

In dieser Arbeit soll ein Gamification-Konzept für eine Crowd-basierte Outsourcing-Plattform für Softwareentwicklung entstehen. Ein motivierendes Gamification-Konzept ist stark von der Domäne abhängig und entsteht nicht durch das schlichte Hinzufügen von Elementen aus Spielen, sondern erfordert ein strukturiertes Vorgehen und ein präzise definiertes Ziel.

[1] Beispiel: Duolingo `https://de.duolingo.com/` (Letzter Abruf: 17.08.2017)
[2] Beispiel: Nike+ `http://nikeplus.com/` (Letzter Aufruf: 17.08.2017)
[3] Beispiel: Swarm (Foursquare) `https://www.swarmapp.com/` (Letzter Aufruf: 17.08.2017)
[4] Beispiel: Untappd `https://untappd.com/` (Letzter Aufruf: 17.08.2018)

Ob durch den Aktivitätsgraphen von Github[5] (Abbildung 1.1), oder die Abzeichen von StackOverFlow[6] (Abbildung 1.2) – die Nutzung von Motivationselementen aus Spielen hat sich bereits in vielen Bereichen der Softwareentwicklung etabliert. Das Gamification-Konzept dieser Arbeit soll jedoch nicht nur die Motivation der Nutzer stärken, die Plattform zu verwenden, sondern gleichzeitig selbst aufgestellte, ausgewählte Performance-Metriken für Crowd-Entwickler optimieren. Dadurch soll neben der User Experience auch die Leistung der Crowd-Entwickler mittelfristig verbessert werden.

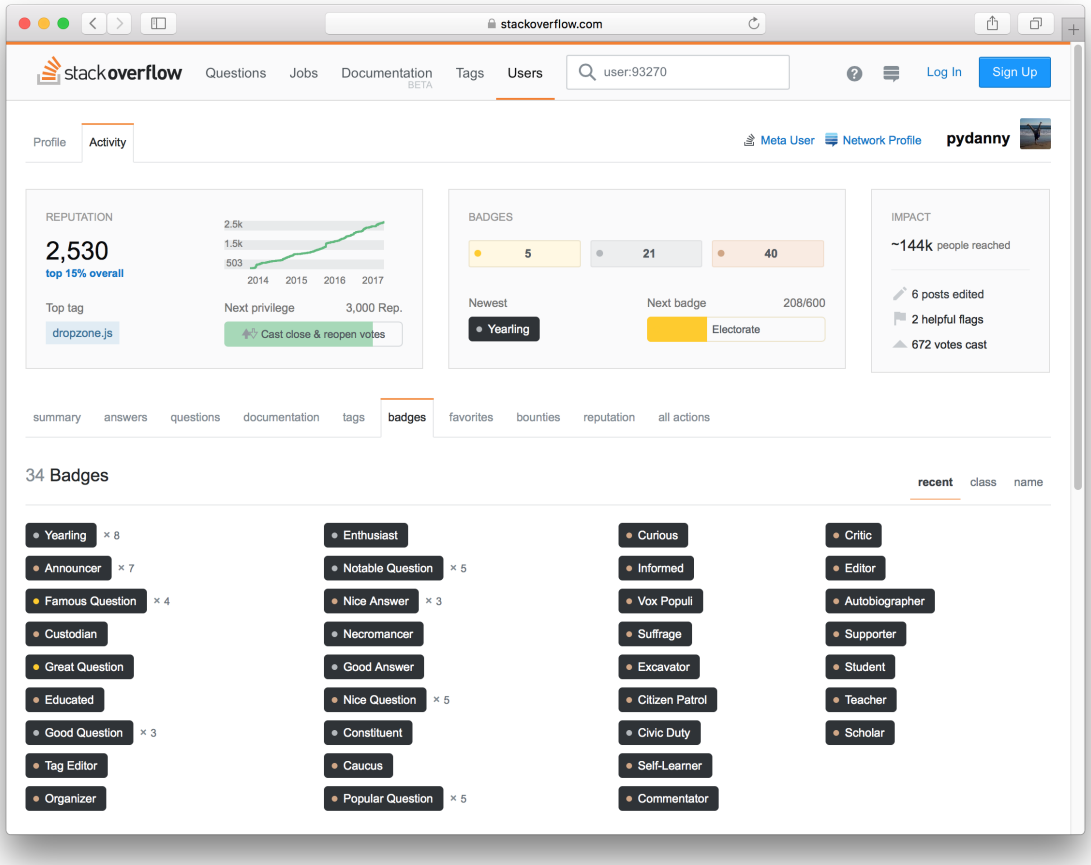


Abbildung 1.2.: Die Frage- und Antwortplattform StackOverFlow bietet ebenfalls Badges/Achievements und andere Gamification-Mechaniken zur Motivation ihrer Nutzer an.

[5] http://github.com/ (Letzter Abruf: 17.08.2017)
[6] http://stackoverflow.com/ (Letzter Auruf: 17.08.2017)

1.2. PROBLEMSTELLUNG

Der Kontext der Arbeit ist eine Crowd-basierte Outsourcing-Plattform für Softwareentwicklung (vgl. Kapitel 3.1). Auf dieser können Crowd-Entwickler (beispielsweise Informatik-Studenten) reale Aufgaben von Softwareunternehmen bearbeiten und durch deren Lösung Geld verdienen. Die fachliche Korrektheit des abgegebenen Quellcodes wird durch natürlichsprachliche Tests sichergestellt. Auf der Basis des Quellcodes der gelösten Aufgabe werden Qualitätsanalysen durchgeführt und Metriken berechnet. Das Gamification-Konzept dieser Arbeit soll die User Experience der Plattform-Prozesse verbessern. Die aufgestellten Metriken sollen hierbei gleichzeitig für den Plattformbetreiber die Leistung des Crowd-Entwicklers beschreiben.
Um ein Gamification-Konzept für die Plattform zu erstellen, kann auf ein *Gamification Framework* zurückgegriffen werden. Diese Frameworks bieten eine strukturierte Vorgehensweise für die Erstellung des theoretischen Gamification-Konzepts. Für die Umsetzung des Konzepts wird ein *Gamification Service* benötigt, der die Funktionalitäten des Konzepts bereitstellen kann. Diesen Zusammenhang stellt Abbildung 1.3 dar. Hierfür muss entweder ein fertiger Service ausgewählt, oder eine eigene Implementierung vorgenommen werden. Da für die Bereitstellung der Gamification auf die vollständigen Codeanalysen zurückgegriffen werden soll und diese zur Fertigstellung dieser Arbeit noch in Entwicklung sind, werden Konzept und Service unabhängig voneinander ausgearbeitet und evaluiert.
Das Resultat dieser Arbeit wird ein Gamification-Konzept sein, welches auf die Crowdsourcing Plattform anwendbar ist. Zusätzlich wird ein Gamification Service ausgewählt oder implementiert, mit dem das Konzept umsetzbar ist. Beide Teile werden abschließend evaluiert.

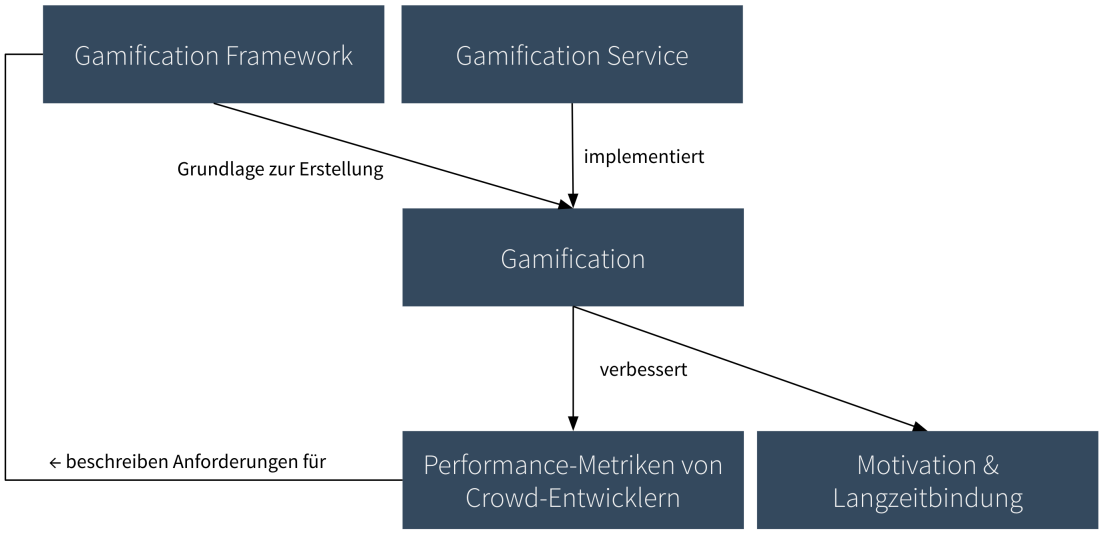


Abbildung 1.3.: Relationen zwischen Gamification, Frameworks, Services, sowie Motivation und Metriken.

1.3. ZIELSETZUNG

Das Ziel dieser Arbeit besteht aus zwei Unterpunkten, die erfüllt werden sollen. Einerseits soll ein Gamification-Konzept basierend auf Performance-Metriken für eine Crowd-basierte Outsourcing-Plattform entwickelt werden. Dafür sollen Metriken zur Bewertung der Crowd-Entwickler aufgestellt und mit einem Gamification-Konzept verbunden werden. Die Verbesserung dieser Metriken soll durch das entstandene Konzept motiviert werden.
Andererseits müssen die notwendigen Komponenten für das Gamification-Konzept von einem Gamification-Service bereitgestellt werden. Dafür müssen die Anforderungen analysiert werden und entweder ein bereits vorhandener Service gefunden und angepasst, oder neu implementiert werden. Im Falle einer Eigenimplementierung ist besonderer Wert auf Praxistauglichkeit und Generizität zu legen, damit der entwickelte Prototyp ohne großen Mehraufwand in den Produktionsbetrieb integriert werden kann.
Beide Teile werden abschließend unabhängig voneinander evaluiert. Für die Evaluation des Gamification-Konzeptes wird eine Nutzerbefragung durchgeführt, um zu beurteilen, ob potentielle Crowd-Entwickler das Konzept ansprechend finden. Der Gamification-Service wird auf technischer Ebene evaluiert, indem getestet wird, ob alle gewünschten Funktionalitäten erfüllt werden können.

1.4. STRUKTUR DIESER ARBEIT

Nach der Einleitung und Motivation wird in Kapitel 2 der aktuelle Stand der Technik vorgestellt. Dabei werden einerseits Grundlagen zum Begriff *Gamification* (Abschnitt 2.1.1), sowie übliche Spielelemente (Abschnitt 2.1.3) vorgestellt. Um ein Gamification-Konzept zu entwerfen, werden weiterhin Gamification-Frameworks vorgestellt, die den Entstehungsprozess strukturieren und vereinfachen sollen. Diese werden kurz vorgestellt und mit einem abschließenden Fazit eingeordnet. Für die Umsetzung des Gamification-Konzepts werden auf dieser Basis Anforderungen aufgestellt, die für dessen Umsetzung erforderlich sind. Auf dieser Basis werden vorhandene Gamification-Services auf Tauglichkeit für die Verwendung untersucht. Diese werden abschließend verglichen und auf dieser Basis die Anforderungen für eine Eigenentwicklung in Abschnitt 2.5 aufgestellt.
Um den Kontext der Arbeit zu verdeutlichen, wird in Kapitel 3 die Crowdsourcing-Plattform vorgestellt, in deren Rahmen das Gamification-Konzept Anwendung finden soll. Das entstandene Gamification-Konzept wird in Kapitel 4.1 vorgestellt. Dabei werden zuerst Performance-Metriken für Crowd-Entwickler ausgearbeitet, welche dann die Basis für das Konzept an sich bilden. Weiterhin wird in diesem Kapitel in Abschnitt 4.2 das grundlegende Konzept des entwickelten Gamification-Services vorgestellt. Details zur Implementierung des Services werden in Kapitel 5 vorgestellt und diskutiert.
Das entstandene Konzept und der entwickelte Gamification-Service werden in Kapitel 6 durch eine technische Evaluation und eine Nutzerbefragung bewertet. Dabei wird auch der entstandene Service mit den in Abschnitt 2.5 aufgestellten Anforderungen verglichen und offene Punkte und Mängel ausgeführt. Der Entstehungsprozess wird in Kapitel 7 zusammengefasst. Weiterhin wird ein Ausblick auf die zukünftige Weiterentwicklung gegeben.

2. VERWANDTE ARBEITEN / STATE OF THE ART

Dieses Kapitel beschreibt verwandte Arbeiten zu Gamification Konzepten und Frameworks.. Zuerst wird auf Gamification im Allgemeinen eingegangen und der Begriff erläutert. Um ein Gamification-Konzept auf ein Projekt anzuwenden gibt es eine Reihe sogenannter Gamification-Frameworks. Diese sind Konzepte zum strukturierten Vorgehen bei der Entwicklung eines Gamification-Konzepts für eine bestimmte Domäne. Im dritten Teil des Kapitels werden vorhandene Gamification Services vorgestellt und auf deren Eignung im Kontext einer Crowd-basierten Softwareentwicklungs-Plattform überprüft.

2.1. GAMIFICATION

2.1.1. BEGRIFF

Als Gamification wird die Methode bezeichnet,bewährte Elemente aus Spielen, wie Level, Ranglisten, Erfahrungspunkte oder Erfolge in einem spielfremden Kontext zu verwenden. Gamification wird genutzt um spielerisch die Motivation der Nutzer einer Plattform (oder App, Software, usw.) und damit die Langzeitbindung zu erhöhen und gleichzeitig die User-Experience (UX) zu verbessern. Zusätzlich können Qualität und Quantität der Ergebnisse von Nutzeraufgaben durch Gamification aufgewertet werden (Morschheuser u. a. 2017).

Abbildung 2.1.: Google Suchen nach Stichwort *Gamification* der letzten 10 Jahre Quelle: `https://trends.google.de/trends/explore?date=2007-01-01%202017-08-20&q=Gamification` (Abgerufen am 20.08.2017).

Gamification ist ein relativ neues Konzept, welches jedoch in den letzten Jahren immer häufiger angewendet wird. Der Begriff kann bis ins Jahr 2008 zurückverfolgt werden. Eine größere Verbreitung fand erst seit 2010 statt (Deterding u. a. 2011a). Dieser Trend wird auch anhand der weltweiten Google-Suchanfragen innerhalb der letzten zehn Jahre deutlich (Abbildung 2.1). Insbesondere durch den Erfolg der standort-basierten Empfehlungsplattform *Foursquare*[1], die 2010 weltweit startete, gewann Gamification an immer größerer Bedeutung. (Deterding u. a. 2011b) Mittlerweile ist Gamification in vielen Branchen (Gesundheit, Sport, Ausbildung, Marketing und Vertrieb, und Crowdsourcing) vertreten.

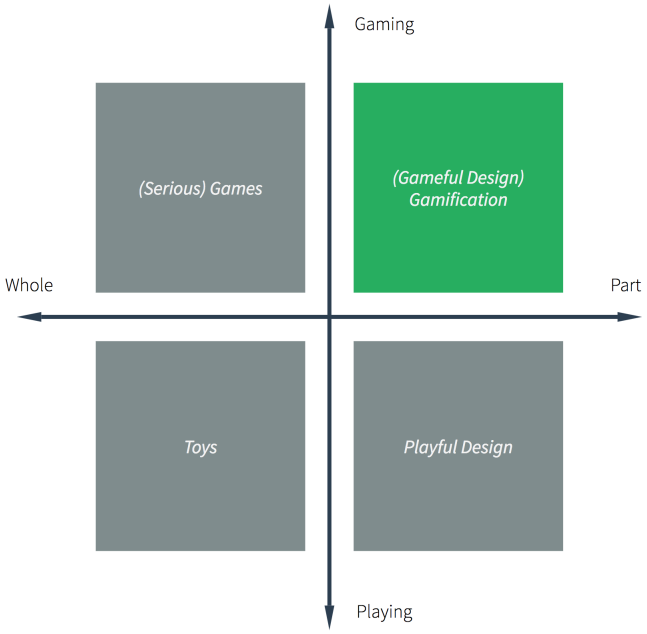


Abbildung 2.2.: Positionierung von Gamification zwischen Serious Games und Playful Design nach Deterding u. a. (2011b)

Eine präzise und abgrenzende Definition des Gamification-Begriffs – insbesondere von *Serious Games* (also Spiele mit einem ernsten Hintergrund wie der Wissensvermittlung, beispielsweise Flugsimulationen) und Playful Design (spielzeugähnliches Design) – bieten Deterding u. a. (2011b):

Definition 1 *Gamification refers to the use (rather than the extension) of design (rather than game-based technology or other game- related practices) elements (rather than full-fledged games) characteristic for games (rather than play or playfulness) in non-game contexts (regardless of specific usage intentions, contexts, or media of implementation).*

Diese Definition von Gamification wird innerhalb dieser Arbeit verwendet. Die Positionierung von Gamification zwischen Serious Games, Playful Design und Toys stellt Abbildung 2.2 schematisch dar.

[1] `http://foursquare.com`. Später wurde die Gamification-Komponente in die ebenfalls standort-basierte App *Swarm* ausgelagert (`http://swarmapp.com`) Letzter Aufruf: 05.06.2017

2.1.2. PSYCHOLOGISCHER EINFLUSS DURCH GAMIFICATION

Die psychologischen Hintergründe von Gamification sind nicht Teil dieser Arbeit und werden deshalb in diesem Abschnitt nur exemplarisch beschrieben.
Obwohl Gamification als wissenschaftliches Thema relativ neu ist, sind die Einflüsse von Gamification-Konzepten mittlerweile weitgehend untersucht. Die Meta-Studie von Hamari u. a. (2014) analysierte 24 empirische Studien hinsichtlich der Fragestellung *"Does Gamification Work?"*. Zu diesem Zweck entwickelten die Autoren ein konzeptuelles Rahmenmodell von Gamification, in dem sie drei Teilbereiche von Gamification unterschieden (Abbildung 2.3). Die Wirkungen von Gamification basieren darauf, dass die verwendeten Elemente ein motivierendes Angebot schaffen (*Motivational Affordance*). Dieses resultiert in psychologischen Auswirkungen (*Psychological Outcomes*), welche wiederum Änderungen im Verhalten nach sich ziehen können (*Behavioral Outcomes*). Es wurde daher untersucht, auf welche Art und Weise in den verwendeten Studien Motivation implementiert wurde, welche psychologischen Auswirkungen gemessen und welche Verhaltensänderungen festgestellt wurden. Die meisten (21) der untersuchten Studien bewerteten den Erfolg der Gamification anhand des veränderten Nutzerverhaltens. Die Mehrheit der untersuchten Studien zeigten positive Auswirkungen der Gamification wie beispielsweise Freude oder Arbeitseifer. Aus den Studien wird damit deutlich, dass wenn ein Gamification-Konzept richtig angewendet wird, das Verhalten der Nutzer positiv beeinflusst werden kann, sodass beispielsweise Lerneffekte begünstigt werden.

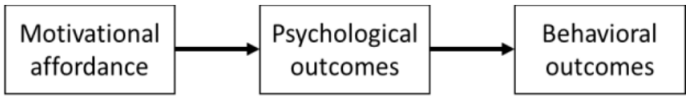


Abbildung 2.3.: Einfluss durch Gamification (Hamari u. a. 2014)

2.1.3. TYPISCHE SPIELELEMENTE

Spiele sind grundsätzlich vielfältig und unterschiedlich aufgebaut, wodurch es schwer ist, typische Elemente von Spielen im Allgemeinen zu definieren. Eine häufige Intention beim Spieldesign ist es jedoch, den Fortschritt und die Errungenschaften des Spielers auf bestimmte Arten und Weisen zu visualisieren.

Häufig werden solche Elemente durch Gamification-Konzepte eingesetzt. Viele dieser Elemente stammen in abgewandelter Form aus einem Kontext, der unabhängig von Video-Spielen ist und wurden bereits früher verwendet, um Leistungen zu bewerten. Beispiele hierfür sind Sportwettbewerbe (Ranglisten, Medaillen, Preise, etc.) oder das Militär (Medaillen, besondere Ränge, Titel, etc.). (Zichermann und Cunningham 2011)

Welche Elemente aus Spielen für Gamification im Bildungs-Kontext häufig genutzt werden, untersuchte eine Meta-Analyse von Ed und Hutchison (2014). Dabei kristallisierten sich acht Gamification-Elemente heraus, welche mehrheitlich verwendet wurden:

1. Achievements/Erfolge/Badges
   Achievements/Erfolge sind eines der Standardelemente von Gamification. Sie sind Errungenschaften werden die durch Handlungen im Spielgeschehen freigeschaltet werden können. Erfolge sind im Abschnitt 2.1.4 detailliert beschrieben.

2. Punkte
   Punkte fungieren als Messwert für den Erfolg eines Spielers. Sie verdeutlichen dem Spieler seinen aktuellen Stand und machen ihn vergleichbar mit anderen Spielern. Die Arten und Systeme, durch die Punkte genutzt werden, unterscheiden sich stark zwischen Spielen. In vielen Spielen erhält die vom Spieler gesteuerte Figur Erfahrungspunkte für abgeschlossene Aufgaben und Missionen. Erfahrungspunkte werden häufig für Gamification eingesetzt, um den Fortschritt des Spielers anzuzeigen. Näheres zu Erfahrungspunkten und Spielerstufen wird in Abschnitt 2.1.5 beschrieben.

3. Levels/Stages
   Levels werden genutzt, um dem Spieler einen Sinn für den Fortschritt innerhalb des Spiels zu vermitteln. Typischerweise sind die Anfangslevel einfach und werden zunehmend komplexer und schwerer zu erreichen. Sie verlangen mehr Können und/oder mehr zeitlichen Aufwand.

4. Ranglisten
   Ranglisten schaffen eine Vergleichbarkeit unter den Spielern. Sie regen den Eifer an, mehr zu erreichen und innerhalb der Rangliste aufzusteigen.

5. Preise und Belohnungen
   Preise und Belohnungen haben sich als gutes Motivationskonzept bewährt (Betts u. a. 2013). Häufige kleine Belohnungen werden als motivierender wahrgenommen, als wenige große Preise.

6. Fortschrittsanzeigen
   Fortschrittsanzeigen können dem Nutzer ein Bild von Fortschritt seines Erfolgs vermitteln und damit messbar machen. Der Fortschritt kann dabei beispielsweise innerhalb eines Levels, der Fortschritt eines Erfolges, oder Ähnliches sein.

7. Storylines
   Insbesondere bei Lernanwendungen haben sich Storylines bewährt. Der Nutzer wird hierbei in eine Geschichte einbezogen, in der Aufgaben gelöst werden müssen. Durch diese kann Spannung auf den Anfang und das Ende einer Aufgabe gelegt werden und damit die Motivation des Nutzers aufrechterhalten werden (Kapp 2012).

8. Feedback
   Direktes und schnelles Feedback hat sich als ein wichtiges Spielelement erwiesen, da es erforderlich ist um einen *Flow*-Zustand (der Zustand völliger Vertiefung in ein Problem nach Csikszentmihalyi (2000)) zu erreichen.

2.1.4. ACHIEVEMENTS/ERFOLGE

Achievements haben sich als eines der Standardelemente von Gamification-Konzepten etabliert (Hamari 2011). Die Bezeichnung *Achievement* ist zu einer Art Industriestandard geworden, seit große Spieleplattformen wie Xbox-Live oder Steam diesen adaptiert haben (Hamari 2011). Innerhalb des Spiel-Kontexts beschreiben Achievements eine Errungenschaft, die separat vom eigentlichen Spielgeschehen durch bestimmte Handlungen und unter bestimmten Bedingungen erreichbar sind. Hamari (2011) bietet ein Framework zur Definition und zum Aufbau von Achievements und gleichzeitig eine Definition des Begriffs an:

Definition 2 *Achievements are goals in an achievement/reward system (different system than the core game) whose fulfilment is defined through activities and events in other systems (commonly in the core game).*

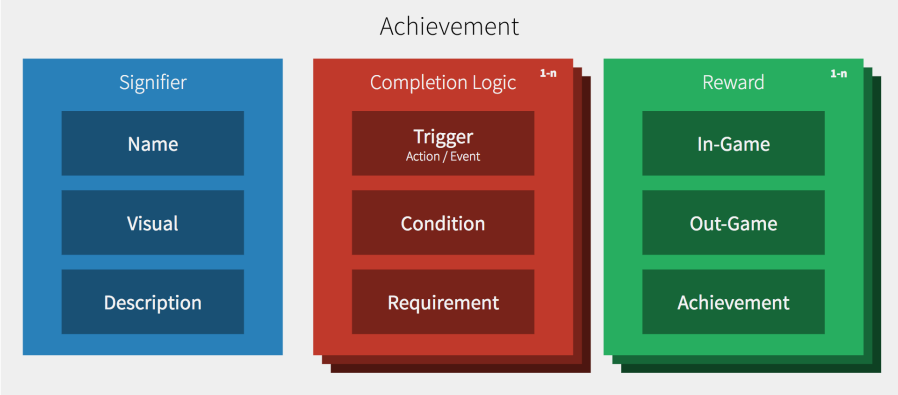


Abbildung 2.4.: Achievement-Modell nach Hamari (2011)

Hamari beschreibt das Achievement-Modell bestehend aus drei Teilen: *Signifier*, *Completion Logic* und *Reward*. Jeder der drei Teile besteht wiederum aus drei Komponenten (Abbildung 2.4).

Der *Signifier* beschreibt den sichtbaren Teil des Erfolges. Er besteht aus einem Namen, einer Beschreibung und einem Visual. Dieses ist ein, zum Erfolg zugehöriges, Bild oder Icon.

Die *Completion Logic* beschreibt, wann ein Erfolg durch den Spieler errungen wurde. Die Logik wird in drei Teile unterteilt: Den Trigger, der den Auslöser angibt durch den der Nutzer den Erfolg erhält. Die Condition (Bedingung), unter der Trigger ausgelöst wird und ein Requirement, welches vorher erfüllt sein muss (Beispielsweise ein anderer Erfolg). Ein Erfolg kann aus einem oder mehreren *Completion Logic* Komponenten bestehen.

Der dritte Teil ist die *Reward*-Komponente. Die Belohnungen werden in drei Teile unterteilt: In-Game, Out-Game und Achievement-Game. Die Belohnungen können also innerhalb des Spielkontext sein (z.B. Gegenstände im Spiel, Titel, etc.), außerhalb des Spiels (beispielsweise Belohnungen in anderen Spielen, oder in der realen Welt) oder Belohnungen innerhalb des Achievement-Games (z.B. Erfolgspunkte für Ranglisten). Ein Erfolg kann viele *Reward*-Komponenten enthalten und damit viele Belohnungen gewähren.

BEISPIEL

Abbildung 2.5.: Beispielerfolg im Online-Rollenspiel *World of Warcraft*

Beispielhaft zeigt Abbildung 2.5 einen Erfolg im Online-Rollenspiel *World of Warcraft*[2]. Die drei Signifier sind der Titel des Erfolges, darunter die Beschreibung und das Bild des Drachen links. Der Trigger wird durch einen grünen Balken, inklusive Fortschritt (hier: 50/50) dargestellt. Zusätzliche Conditions oder Requirements sind nicht vorhanden. Als Belohnung für den Erfolg erhält der Spieler den Titel „Jenkins", sowie 10 Punkte im Achievement-Game.

Ein Beispiel für reale Belohnungen durch Gamification bietet die in Abschnitt 2.1.1 erwähnte standardbasierte Swarm-App von Foursquare. In der App kann der Nutzer an Orten (Hotels, Flughäfen, Cafes, Restaurants, etc.) „einchecken" und damit bestätigen, dass er diese besucht hat. Der Check-In ist nur möglich, wenn er sich an dem Ort befindet. Besucht der Nutzer einen Ort mehrfach können einerseits Erfolge („Sticker") freigeschaltet werden (z.B. *Zehn Check-Ins in Hotels*), jedoch auch echte Belohnungen wie Rabatte in teilnehmenden Geschäften aktiviert werden.

2.1.5. ERFAHRUNGSPUNKTE UND LEVEL

Erfahrungspunkte und Spieler-Stufen (Level) bieten die Möglichkeit, den Gesamtfortschritt des Spielers deutlich zu machen. Das Konzept ist vor allem bekannt aus Computer-Rollenspielen, die stark darauf aufbauen, den Charakter zu entwickeln und sich mit ihm zu identifizieren. Die Spielfigur erhält typischerweise Erfahrungspunkte für Aktionen im Spiel (besiegte Gegner, abgeschlossene Aufgaben, etc.). Hat der Spieler eine gewisse Anzahl Erfahrungspunkte erlangt, steigt er ein Level auf. Die Anzahl der Erfahrungspunkte wird wieder auf Null zurückgesetzt. Die einzelnen Erfahrungspunkte sind oftmals nicht nach außen sichtbar. Das erreichte Level ist jedoch für andere Spieler einsehbar. Oft können beim Level-Aufstieg weiterhin Fähigkeiten an die Spielfigur vergeben werden, sodass diese mit jedem weiteren Level an Macht gewinnt.
Üblicherweise sind die Level-Übergänge so gewählt, dass es mit zunehmenden Level schwieriger (oder zeitlich aufwändiger) ist, eine höhere Stufe zu erreichen.
Die Berechnung des Levels kann auf verschiedene Arten erfolgen. Jedes Level verfügt über eine Mindestanzahl an Erfahrungspunkten, die der Charakter erreichen muss und einen maximalen Erfahrungswert, bei dem die Figur das nächste Level erreicht. Ein naiver Ansatz speichert die Level-Übergänge und weist diese ihrem zugehörigen Level zu. Probleme bei dieser Lösung sind potentielle Inkonsistenzen innerhalb der Level-Übergänge. Weiterhin sind dadurch nur begrenzt viele Level abbildbar. Ein besserer Ansatz ist die Abbildung durch eine Level-Funktion, durch die zu jedem Erfahrungspunktewert ein Level bestimmt werden kann.

[2] `http://worldofwarcraft.com` (Letzter Aufruf: 17.08.2017)

2.1.6. PROBLEME VON GAMIFICATION

> »*I'm using a new App to track my running and I'm annoyed how they spam me with stars, achievements and congratulations like I'm a 5 year old.* «
> — Ron Gilbert (Spieldesigner und Erfinder von *Monkey Island*) auf Twitter[3]

Bei einem Gamification-Ansatz besteht immer die Gefahr, dass Nutzer diesen nicht akzeptieren oder mögen. Insbesondere dann kann die benötigte Allgegenwärtigkeit des Gamification Ansatzes schnell lästig werden. So wurde auch in der Beschreibung von Montola u. a. (2009) erwähnt, dass einige Nutzer den beschriebenen Gamification Ansatz nicht verstanden und deshalb störend fanden. Laut der Meta-Analyse von Hamari u. a. (2014) reagiert jedoch der Großteil der Nutzer positiv auf Gamification.

2.2. GAMIFICATION FRAMEWORKS

Die Entwicklung eines Gamification-Konzepts ist komplexer als eine einfache Portierung von Spielelementen in einen fremden Kontext (Chou 2016), da insbesondere der sinnvolle Einsatz dieser Elemente ausschlaggebend ist. Für den Entwurf eines solchen Gamification Konzepts gibt es eine Reihe von etablierten Frameworks. Auf eine Reihe dieser Frameworks wird in diesem Kapitel eingegangen. Die Frameworks werden beschrieben und in einem abschließenden Fazit bezüglich der Zielsetzung bewertet.

2.2.1. OCTALYSIS

Das Octalysis Framework wurde von Yu-Kai Chou veröffentlicht (Chou 2016). Chous Ansatzpunkt ist, dass eine gute Gamification nicht durch das bloße Hinzufügen spieltypischer Elemente wie Erfahrungspunkte, Erfolge und Leader-Boards entsteht. Vielmehr sei es nötig, das anzusprechen, was Motivation antreibt. Schließlich gibt es auch eine Vielzahl an Spielen, die diese Features haben, aber nicht motivierend sind. Er bezeichnet dieses Vorgehen als Human-Focused Design:

> »*Gamification is design that places the most emphasis on human motivation in the process. In essence, it is Human-Focused Design (as opposed to „function-focused design"))* «

Um die Frage zu beantworten, durch *was* genau ein Gamification-Konzept motivierend wird, beschreibt Chou acht zentrale *Core Drives*, die genau die Anreize wiedergeben sollen, welche die Motivation im Nutzer auslöst. Die acht Core-Drives sind hierbei nicht positiv oder negativ, sondern beschreiben lediglich, dass sie eine Motivation nach sich ziehen.

[3] `https://twitter.com/grumpygamer/status/871730822574985217` (Letzter Aufruf: 20.06.2017)

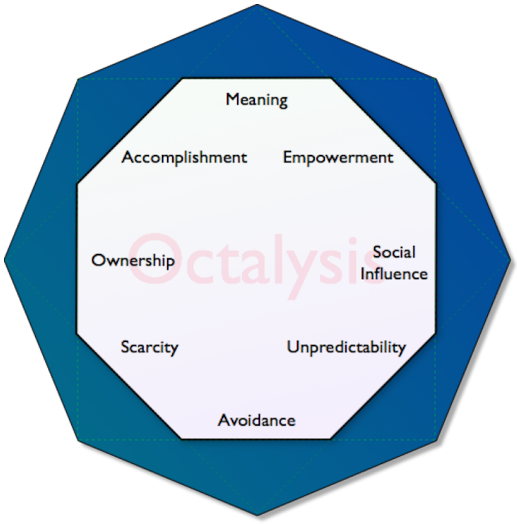


Abbildung 2.6.: Anordnung der acht Core-Drives des Octalysis Frameworks

1. Epic meaning and calling
   Dieser Core-Drive beschreibt das Empfinden der Wichtigkeit der zu erledigenden Aufgabe und die Rolle des Nutzers. Wichtig ist, dass der Anwender den Eindruck bekommt genau der Richtige, der „Auserwählte", für einen Sachverhalt von größter Wichtigkeit zu sein.

2. Development and accomplishment
   Dieser Punkt beschreibt den Fortschritt und den Lerneffekt des Anwenders. An diesem Punkt können Gamification-Systeme meist gut ansetzen.

3. Empowerment of creativity and feedback
   Inhalt dieses Core-Drives ist die Belohnung kreativen Verhaltens.

4. Ownership and possession
   Durch *ownership and possession* wird die Motivation des Sammelns und des Besitzes beschrieben.

5. Social influence and relatedness
   Dieser Punkt spricht soziale Werte wie Akzeptanz, soziales Feedback, Neid und Konkurrenzverhalten an.

6. Scarcity and impatience
   Die reine Seltenheit einer Belohnung oder Aufgabe ist oft ein motivierender Faktor, ebenso wie die Notwendigkeit auf eine Aufgabe oder Belohnung warten zu müssen.

7. Unpredictability and curiosity
   Unvorhersehbarkeit und Neugierde sind der Ausgangspunkt der Motivation, ein Buch zu lesen oder einen Film zu Ende zu sehen. Gleichzeitig ist es auch, was Spielsucht ausmacht.

8. Loss and avoidance
   Die Angst vor Verlust und dessen Vermeidung ist oftmals ebenso ein motivierender Faktor.

Die acht Core-Drives des Octalysis Frameworks werden in einem Achteck angeordnet und jeder Kante genau einer der Core-Drives zugeordnet (Abbildung 2.6). Die oberen drei Drives bilden eher positive Motivatoren. Scarcity, Unpredictability und Avoidance prägen die eher negativen Motivatoren aus. Im Sinne der Motivation an sich ist keiner der Core-Drives gut oder schlecht aufzufassen, die negativen Drives, können jedoch auf Dauer die Motivation mindern. Auf der linken Seite sind eher logische Faktoren, auf der rechten Seite sind emotionale und kreative Faktoren ausgeprägter.
Zu jedem der Core-Drives wird zur Evaluation ein Wert zwischen 0 - 10 vergeben und quadriert:

$$\text{octalysis\_score} = \sum_{i=1}^{8} \left(\text{Core Drive}_i\right)^2 \tag{2.1}$$

Die Summe dieser Werte ergibt den Score des Gamification Konzepts. Das Octalysis Modell bewertet dadurch die starke Ausprägung einzelner Core Drives stärker, als die mittelmäßige Ausprägung sämtlicher Drives.

FAZIT

Das Octalysis Framework ist gut zur Evaluierung des eigenen Gamification-Konzepts geeignet. Es bietet jedoch nur wenige Vorgaben, um bei der Entwicklung eines solchen Konzepts zu unterstützen. Die einzelnen Core-Drives bieten eine sinnvolle Auswahl an Ansatzpunkten und sollten betrachtet werden, um die Motivation der Nutzer in diesen Bereichen zu verbessern. Das Octalysis Framework ist sehr generisch und bietet keine Vorgaben, auf welche Art die Gamification implementiert werden soll.

2.2.2. SPIELERTYPEN NACH BARTLE UND PERSONAS

Richard Bartle klassifizierte - ursprünglich für Multiplayer-Computerspiele - verschiedene Spielertypen, um darzustellen welche Ziele unterschiedliche Spieler erreichen möchten (Bartle 1999). Diese Spielertypen können genutzt werden, um besser zu verstehen, was den einzelnen Spieler motiviert (Zichermann und Cunningham 2011). Bartle definierte ursprünglich vier verschiedene Spielertypen. Als Zuordnung nach dem Modell von Bartle erhält man keine eindeutige Zuweisung zu einer Gruppe, sondern einen Anteil, zu wie viel Prozent man den vier Gruppen entspricht. Die vier Spielertypen werden im Folgenden beschrieben.

EXPLORERS

Explorers reizt insbesondere das Unbekannte. Sie haben großes Interesse am Entdecken neuer Inhalte. Die entdeckten Inhalte können hierbei auch beispielsweise Spielfehler sein. *Explorers* werden schnell demotiviert, wenn sie merken, dass die zukünftig erreichbaren Inhalte Wiederholungen der bisher erlebten sind.

ACHIEVERS

Achievers versuchen möglichst viel zu erreichen, viele Punkte zu sammeln und hohe Level zu erlangen. Achievers suchen den Vergleich zu anderen Mitspielern und versuchen höhere Ränge zu erreichen. Sie suchen Anerkennung durch andere Spieler, indem sie Dinge erreichen, die nur ein kleiner Anteil der Spieler erreicht hat.

SOCIALIZERS

Socializers bilden die größte Gruppe der Spielertypen (Zichermann und Cunningham 2011). Sie werden insbesondere durch die soziale Interaktion mit anderen Spielern motiviert. Für sie ist das Spiel ein Katalysator für soziale Interaktionen. Die meisten bekannten Gesellschaftsspiele (Poker, Rommé, Monopoly, etc) sind für Socializers zugeschnitten.

KILLERS

Killers bilden die kleinste Gruppe unter den Spielertypen (Zichermann und Cunningham 2011). Sie ähneln in ihrem Verlangen nach Erfolgen den *Achievers*. Zusätzlich ist spielt es für sie jedoch eine Rolle, dass ein anderer Spieler verliert.

Ein Gamification-Konzept sollte so erstellt werden, dass alle Spielergruppen nach ihren individuellen Bedürfnissen belohnt werden können, sodass jede Gruppe ausreichend Motivation geboten wird. (Zichermann und Cunningham 2011) Weiterhin können sie verwendet werden, um Personas[4] zu erstellen. Personas werden häufig als Mittel benutzt, um ein Gamification-Konzept zu erstellen (Morschheuser u. a. 2017). Das Personenprofil kann dann genutzt werden, um die Anforderungsanalyse zu unterstützen und die Bedürfnisse der Persona zu erfüllen.

2.3. GAMIFICATION SERVICES

In diesem Abschnitt werden vorhandene Gamification Services untersucht und überprüft, ob sie sich für einen Einsatz in einer Crowd-basierten Outsourcing-Plattform für Software-Entwicklung eignen. Die Auswahl der untersuchten Services erfolgte nach deren Bekanntheitsgrad und dem Stand ihrer Entwicklung.
Damit ein Service verwendet werden kann, müssen folgende Kriterien erfüllt werden:

1. Generische Anbindung an fremde Plattform
2. Definition eigener Erfolge auf Basis selbst definierter Metriken
3. Definition eines Erfahrungspunkte-Systems mit individuellen Punktebelohnungen je nach Güte der abgeschlossenen Aufgabe auf Basis selbst definierter Metriken
4. Anbindung von selbst entwickelten komplexen Metrik-Berechnungen
5. Lizenz, die Verwendung in eigenem Projekt erlaubt

Der Gamification-Service der SAP Cloud Plattform (Herzig u. a. 2012) steht nicht außerhalb der SAP-Plattform zur Verfügung und wird deshalb nicht im Einzelnen betrachtet.

[4] Personas sind Personen-Profile, die den Charakter und die Eigenschaften einer (fiktiven) Person innerhalb der Zielgruppe darstellen.

2.3.1. OPENBADGES

OpenBadges[5] ist ein 2012 gestartetes, quelloffenes Projekt der Mozilla Foundation. Es wurde durch die MacArthur Foundation finanziert. OpenBadges ermöglicht es, Abzeichen („Badges") zu erstellen und diese den Nutzern beim Erreichen bestimmter Anforderungen zu vergeben. Dabei zielt OpenBadges insbesondere auf Bildungseinrichtungen.
Die Badges können in ein Backpack hochgeladen und dort veröffentlicht werden. Beispiele für diese Backpack-Plattformen sind Badgr[6] oder OpenBadges Backpack [7] auf denen sämtliche hochgeladenen OpenBadges eines Nutzers dargestellt, sortiert und veröffentlicht werden können. Die Badges bestehen aus PNG-Bildern und können über eine REST-Schnittstelle angefragt und angezeigt werden. Dadurch ist es für Webseiten oder Blog-Betreiber einfach, OpenBadges auf ihrer Webseite anzuzeigen. Die Badges können dann genutzt werden, um besuchte Kurse oder erlangte Fähigkeiten nach außen zu zeigen und beispielsweise bei Bewerbungen anzugeben. (Abbildung 2.7)
OpenBadges ist insbesondere im Bildungsbereich verbreitet. So bietet beispielsweise die Lernplattform moodle[8] seit Version 2.5 eine Integration an.

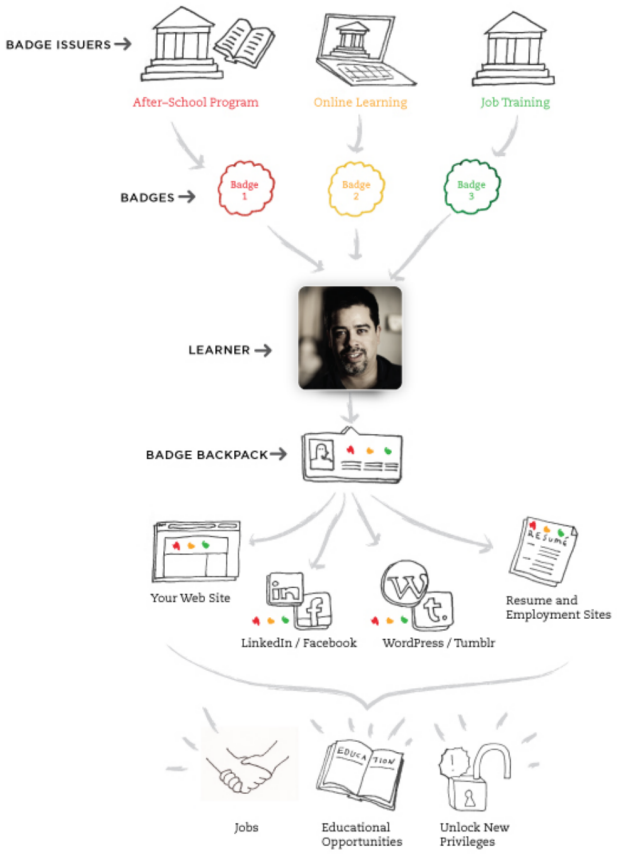


Abbildung 2.7.: Funktionsweise von OpenBadges (Bildquelle: Goligoski 2012)

[5] https://openbadges.org/ (Letzter Abruf: 17.06.2017)
[6] https://info.badgr.io/ (Letzter Abruf: 17.06.2017)
[7] https://backpack.openbadges.org/ (Letzter Abruf: 17.06.2017)
[8] https://moodle.org/ (Letzter Abruf: 17.06.2017)

FAZIT

Das OpenBadges Projekt bietet einen plattformübergreifenden Service zum Erstellen, Verteilen und Anzeigen von Abzeichen. Der Dienst wird bereits von vielen Organisationen genutzt (3000 - Stand 2017). Durch den limitierten Funktionsumfang - OpenBadges bietet ausschließlich Schnittstellen für Badges - ist jedoch die Kombination mit einem weiteren Gamification-Service sinnvoll. Die Plattform bietet insbesondere gute Funktionalitäten für Nutzer, um errungene Erfolge nach außen sichtbar zu machen.

2.3.2. BUNCHBALL NITRO

Bunchball Nitro[9] ist ein cloud-basierter Gamification-Service für den Enterprise-Bereich. Laut eigener Website bietet Nitro eine Vielzahl vordefinierter Gamification-Elemente wie Benutzerprofile, Challenges, Levels, Erfolge, Ranglisten, sowie Aktivitätsanzeigen und Notifications und ist dabei komplett anpassbar. Das Interface zur Erstellung eigener Regeln wird in Abbildung 2.8. Um die Daten aus eigenen Anwendungen oder Plattformen für Gamification nutzbar zu machen werden APIs verwendet. Zusätzlich bietet Nitro fertige Integrationen für Business-Plattformen wie SalesForce an (Schulten 2014).

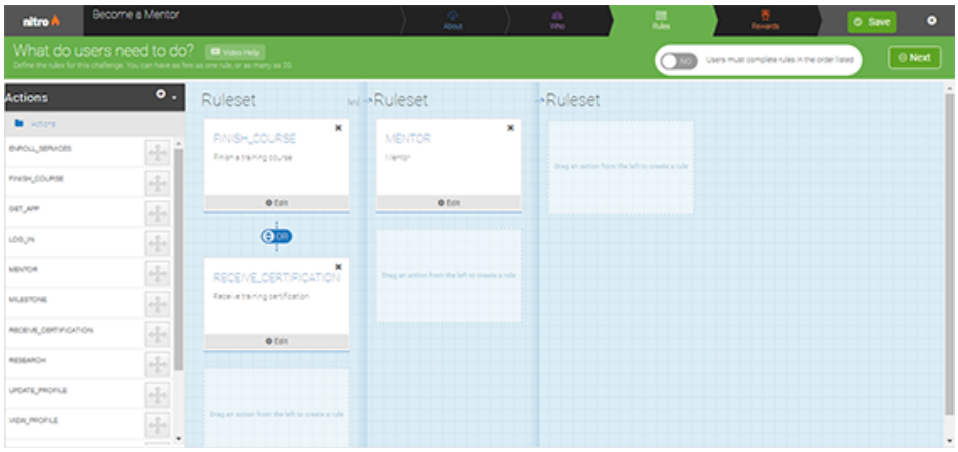


Abbildung 2.8.: Interface zur Regelerstellung in Bunchball Nitro (Bildquelle: `http://www.bunchball.com/products/nitro` (Abgerufen: 18.08.2017))

FAZIT

Die Features von Bunchball Nitro kommen den gewünschten Features sehr nahe, jedoch ist die Plattform eine rein kommerzielle Software und kann daher nicht als Gamification-Service für die Crowdsourcing-Plattform genutzt werden. Inwieweit die Metriken, die die Basis des Gamification Konzepts verwendet werden veränderbar sind, kann nicht abgeschätzt werden.

[9] `https://www.bunchball.com/products/nitro` (Letzter Aufruf 17.08.2017)

2.3.3. BADGEVILLE

Badgeville[10] ist eine Plattform für Gamification im Enterprise-Bereich. Diese kann über eine API angesteuert werden und bietet somit die Möglichkeit der Integration in eine fremde Software oder Plattform. Die Plattform zeichnet hierbei alle gewünschten Aktionen auf und berechnet auf deren Basis personalisiertes Feedback und Belohnungen. Diese können einerseits innerhalb der eigenen, angebundenen Anwendung angezeigt werden, andererseits bietet Badgeville die Möglichkeit einer externen Visualisierung. Laut Website kann Badgeville stark personalisiert werden. Es gibt die Möglichkeit eigene Metriken, Workflows, Systeme anzubinden. Weiterhin sind Integrationen in eine Reihe von verbreiteten Business-Anwendungen wie SalesForce oder SharePoint vorhanden.

FAZIT

Obwohl die Features, die auf der Website zur Badgeville-Plattform genannt werden, überzeugen, ist der Service nicht für den Anwendungsfall dieser Arbeit nutzbar. Die Anwendung kann ausschließlich entgeltlich genutzt werden und ist daher keine Alternative zur eigenen Implementierung eines Gamification-Services. Inwieweit die Metriken anpassbar sind, kann nur schwer abgeschätzt werden.

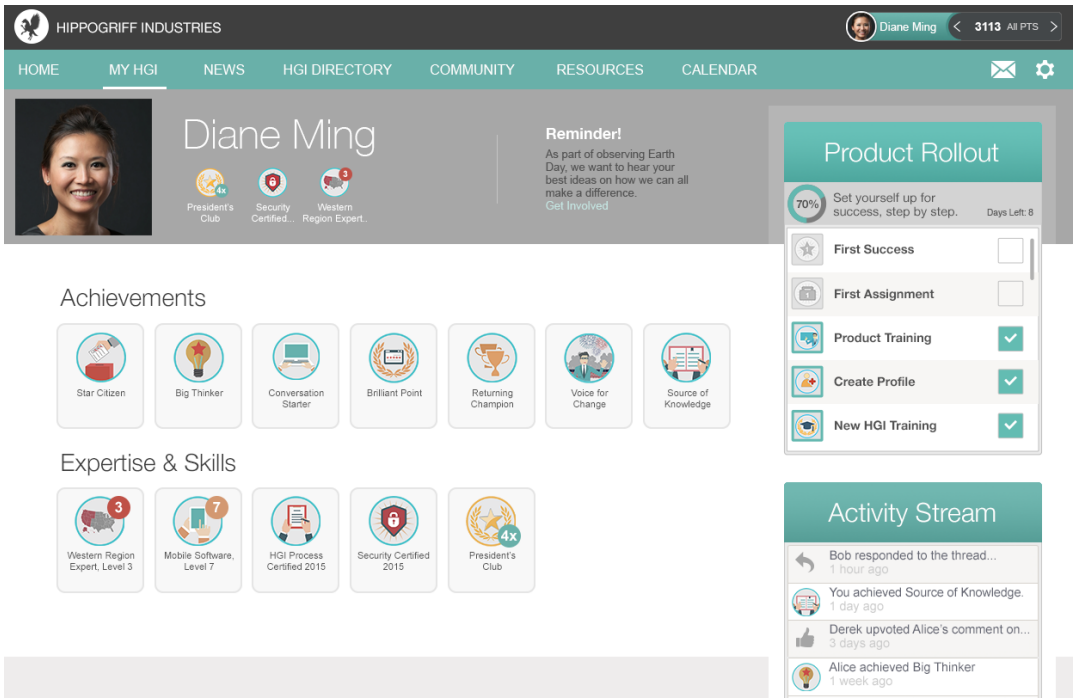


Abbildung 2.9.: Beispiel eines Badgeville Profils (Bildquelle: `https://badgeville.com/` (Letzter Aufruf: 15.08.2017))

[10] `https://badgeville.com/` (Letzter Abruf: 30.06.2017)

2.3.4. GETBADGES

Getbadges[11] ist eine Gamification-Plattform speziell für Softwareentwicklungsprojekte und deren Entwicklungsteams. Die Plattform bietet Integrationen für eine Reihe von Entwickler-Tools und Plattformen (Github, Gitlab, Slack, Jira, Bitbucket etc.) und Continuous Integration-Tools (Jenkins, TeamCity, TravisCI, etc.) an. Durch die durch die Integrationen gesammelten Daten werden die Softwareentwicklungsprozesse auf ein Spiel übertragen (Abbildung 2.10). Das Team wird dann für geschlossene Tickets, Commits oder erfolgreiche Builts belohnt und erreicht gemeinsam Ziele. Dadurch wird die Motivation des gesamten Entwicklerteams gesteigert.

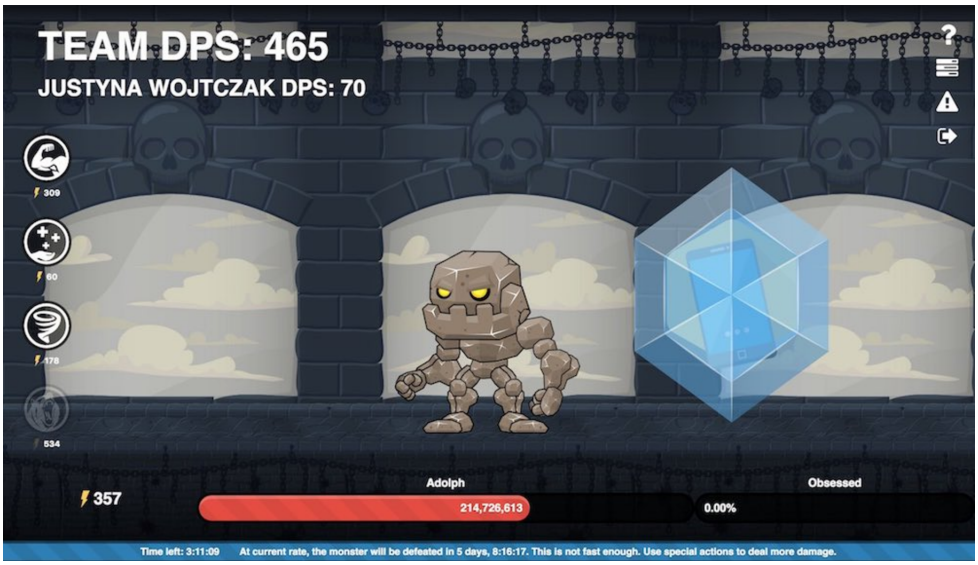


Abbildung 2.10.: Eine Quest in einem Softwareprojekt auf der getbadges Plattform (Bildquelle: `https://getbadges.io/` (Letzter Aufruf: 15.08.2017))

Zusätzlich zu den Teamaufgaben können die einzelnen Nutzer Erfahrungspunkte erhalten, Level aufsteigen und persönliche Erfolge (Badges) sammeln. Dadurch bietet die Plattform zusätzlich einen kompetitiven Ansatz, der den Ehrgeiz fördern soll.
Getbadges ist für open-source Projekte kostenlos. In anderen Projekten kann die Plattform nur entgeltlich genutzt werden. Daher kann es nicht für die Umsetzung des Gamification-Konzepts der Crowdsourcing-Plattform verwendet werden.

FAZIT

Die getbadges-Plattform bietet durch die komplett eigene Plattform einen interessanten Ansatzpunkt. Sie ist durch die Nähe zu den Softwareentwicklungsprozessen eine gute Quelle für Anregungen in Bezug auf Gamification im Softwareentwicklungs-Bereich. Dennoch zielt die Plattform mit Entwickler-Teams auf eine andere Zielgruppe und kann damit keinen Gamification-Service für den Einsatzzweck der Arbeit ersetzen.

[11] `https://getbadges.io/` (Letzter Aufruf: 30.06.2017)

2.4. ZUSAMMENFASSUNG

In diesem Kapitel wurden eine Definition von Gamification und die Grundlagen der wichtigsten Gamification-Elemente eingeführt. Um ein strukturiertes Vorgehen bei der Erstellung eines Gamification-Konzepts zu ermöglichen, wurden weiterhin exemplarisch Gamification-Frameworks und verwandte Konzepte (wie Bartles Spielertypen) vorgestellt. Diese vereinfachen das Vorgehen bei der Ausarbeitung des Konzepts. Abschließend wurden fertige und etablierte Gamification Services beispielhaft vorgestellt und auf Eignung bezüglich der Problemstellung überprüft. Die Ergebnisse werden in der folgenden Tabelle dargestellt.

	OpenBadges	Badgeville	GetBadges	Nitro
Generische Anbindung	Ja	Ja	Nein	Ja
Eigene Erfolge	Ja	Ja	Ja	Ja
Erfahrungspunkte	Nein	Nein	Ja	Ja
komplexe Metriken	Ja	Ja	Nein	Nein
geeignete Lizenz	Ja	Nein	Nein	Nein

Tabelle 2.1.: Vergleich von etablierten Gamification-Services anhand der aufgestellten Kriterien

Keiner der vorgestellten Services ist hinreichend für den Anwendungsfall innerhalb einer Crowd-basierten Outsourcing-Plattform für Softwareentwicklung geeignet. Vier der untersuchten Lösungen sind rein properitäre Services, speziell für den Enterprise-Bereich und aufgrund ihres Bezahlmodells ungeeignet. Weiterhin stellt die Anbindung eigener Metriken und Analysen, welche anhand der gesammelten Daten durchgeführt werden sollen, ein Problem für einige der Services dar. Lediglich die Plattform OpenBadges wäre für eine Erweiterung geeignet, bietet jedoch keine Abstraktionen für Gamification-Elemente wie Erfahrungspunkte oder Notifications bereit.
Aus diesen Gründen wurde die Entwicklung eines eigenen Gamification-Services entschieden. Die feingranulare Anforderungsanalyse wird in Kapitel 2.5 durchgeführt.

2.5. ANFORDERUNGSANALYSE

In diesem Abschnitt werden die Anforderungen an den generischen Gamification Service aufgestellt. Diese sind mithilfe der MoSCoW-Methode (Miranda 2011) priorisiert und werden in vier Kategorien unterteilt: **Must-Have**, **Should-Have**, **Could-Have** und **Won't-Have**. Alle Must-Have Kriterien bilden das Minimale Produkt (Minimum Viable Product - MVP) und sind die Anforderungen ohne die der Service nicht existieren kann. Should-Have und Could-Have sind Erweiterungen des MVP, wobei Should-Have höher priorisiert wird als Could-Have, und damit zuerst umgesetzt werden sollte. Won't-Have Features sind wichtig, jedoch nicht zeitkritisch und sind nicht Teil der aktuellen Iteration.

2.5.1. FUNKTIONALE ANFORDERUNGEN

In diesem Kapitel werden die aufgestellten funktionalen Anforderungen dargestellt und erläutert.

MUST-HAVE KRITERIEN

ID	Name	Beschreibung
M10	Generische (Web-)Schnittstelle für den Empfang von Rohdaten	Um verschiedene Typen Rohdaten in Empfang zu nehmen, muss eine generische Schnittstelle angeboten werden.
M20	Durchsuchbare und strukturierte Speicherung der Rohdaten	Die Rohdaten müssen durchsuchbar gespeichert werden, sodass diese auch bei großen Datenmengen effizient verarbeitet werden können.
M30	Möglichkeit zur Berechnung von Metriken anhand strukturierter Rohdaten	Über die in *M10* definierte Schnittstelle werden generische strukturierte Daten (beispielsweise im JSON-Format) übertragen. Basierend auf den Rohdaten werden Analysen durchgeführt, die für die Gamification Elemente benötigt werden.
M40	Speicherung berechneter Daten	Ebenso wie die Rohdaten, müssen auch die verarbeiteten Daten gespeichert werden. Die Menge der Daten ist hierbei signifikant kleiner als die der Rohdaten, jedoch ist die Zugriffszeit beim Laden der Daten von größerer Relevanz.
M50	Abstraktion von Achievements/Badges	Als typisches Gamification-Element wurden Achievements bzw. Badges gewählt, da diese sehr generisch verwendet werden können. Achievements bestehen aus einem Titel, einem beschreibenden Text, einem Bild und einem Zahlenwert (*Signifier*) und einem Auslöser (*Trigger*). Der Zahlenwert kann gleichzeitig für Ranglisten verwendet werden.

M51	Signifier	Der Signifier eines Achievements beschreibt die sichtbaren Elemente des Erfolgs. Er soll aus einem Titel, einem beschreibenden Text und einem zugehörigen Bild bestehen.
M52	Trigger	Der Trigger ist die Logik, die erfüllt sein muss, um das Achievement freizuschalten.

SHOULD-HAVE KRITERIEN

ID	Name	Beschreibung
S10	Abstraktion weiterer Gamification Elemente	Um die grundlegenden Gamification Features anbieten zu können, ist es nötig, übliche Spiel-Konzepte zu abstrahieren und anbieten zu können.
S11	Erfahrungspunkte und Level	Erfahrungspunkte sind ein Maß für den Langzeiterfolg des Spielers. (siehe Abschnitt 2.1.3)
S12	Ranglisten	Ranglisten sind ein gutes Mittel, um Vergleichbarkeit zwischen den Spielern zu schaffen und den Ehrgeiz anzuregen (siehe Abschnitt 2.1.3; Xu (2011))
S13	Aktivitätsanzeige	Aktivitätsanzeigen bieten die Möglichkeit die letzten Errungenschaften des Nutzers darzustellen und verbessert damit das direkte Feedback (Xu 2011).
S20	Live-Benachrichtigungen	Direktes Nutzerfeedback ist insbesondere wichtig, um den Nutzer auf das Gamification System hinzuweisen und dessen Neugierde zu wecken (Montola u. a. 2009; Herzig u. a. 2012).

COULD-HAVE KRITERIEN

ID	Name	Beschreibung
C10	Nutzerprofile	Nutzerprofile erlauben das öffentliche Einsehen der Nutzerdaten und sind sinnvoll um den Benutzern untereinander eine Vergleichbarkeit zu bieten (Xu 2011).
C20	Rechtesystem	Einige Nutzer möchten möglicherweise nicht, dass alle Informationen im Nutzerprofil freigegeben sind.

WON'T-HAVE KRITERIEN

ID	Name	Beschreibung
W10	Natürlichsprachliche Definition der Funktionen	Eine Domänen-spezifische Sprache (DSL) für die Definitionen der einzelnen Analysefunktionen ist sinnvoll und ermöglicht eine sehr einfache Erweiterung. Durch eine Verwendung natürlichsprachlicher Definitionen wird es möglich, die komplexen Berechnungen ohne tiefgreifendes Vorwissen zu komponieren. Eine solche DSL ist längerfristig vorgesehen, wird jedoch aufgrund der komplexen Problemstellung vorerst nicht implementiert.
W20	Anbindung von OpenBadges	Die Anbindung an etablierte externe Services ist sinnvoll, da ein größeres Publikum angesprochen werden kann.

2.5.2. NICHT-FUNKTIONALE ANFORDERUNGEN

ID	Name	Beschreibung
NF10	Schnelles Feedback für den Nutzer	Schnelles Feedback ist für Gamification insbesondere wichtig, da dem Nutzer die Kausalität der Gamification-Elemente bewusst gemacht werden muss (Montola u. a. 2009; Herzig u. a. 2012).
NF20	Skalierbarkeit bei vielen parallelen Anfragen	Um keine Beschränkung hinsichtlich der Nutzerzahl zu geben, ist es nötig, viele parallele Anfragen bearbeiten zu können.
NF30	Robustheit bei häufigen Anfragen	Das System sollte gehäuften Anfragen standhalten, um zuzulassen, dass viele Daten aufgezeichnet und verarbeitet werden.
NF40	Einfache Erweiterbarkeit	Auf einfache Erweiterbarkeit wird besonderen Wert gelegt, um den Service möglichst generisch und praxistauglich zu entwickeln.
NF50	Schutz vor Datenverlust	Datenverlust ist kritisch, da eine Reihe von Gamificationelementen auf dem bisherigen Datenverlauf aufbaut.

3. KONTEXT

Das Ziel dieser Arbeit ist es, ein Gamification-Konzept für eine Crowd-basierte Plattform für Softwareentwicklung zu erarbeiten. Dieses soll die Nutzer motivieren, Aufgaben innerhalb der Plattform zu lösen, die User Experience verbessern und damit gleichzeitig die Nutzer stärker an die Plattform binden. Weiterhin soll das Gamification-Konzept in Metriken resultieren, mit deren Hilfe der Plattformbetreiber seine Nutzer bewerten und einordnen kann. In diesem Kapitel werden die Grundlagen zur Crowdsourcing-Plattform und zu Gamification erläutert.

3.1. CROWDSOURCING PLATTFORM

Der in dieser Arbeit entwickelte Gamification-Service soll innerhalb einer Crowdsourcing-Plattform für Softwareentwicklung mit automatischer Abnahme genutzt werden. Im folgenden wird die Idee hinter der Plattform erläutert.
Ziel der Plattform ist es, Softwareentwicklungs-Aufgaben mithilfe der Crowd parallel und damit sowohl zeit- als auch kosteneffizient lösen zu können. Um die Plattform dabei effizient und fair zu gestalten, sollen die Crowd-Entwickler grundsätzlich für, innerhalb eines Zeitlimits, korrekt gelöste Aufgaben bezahlt werden. Um diese Ziele erreichen zu können, wurde der folgende Prozess ausgearbeitet. Dieser ist an den Test-driven Development (TDD) Ansatz angelehnt. Abbildung 3.1 stellt diesen Workflow schematisch dar. Die Abbildung zeigt, wie der Softwareentwicklungsprozess durch Einsatz der Plattform, parallel zur internen Entwicklung genutzt werden kann. Schnittstelle zwischen interner und externer Entwicklung bildet dabei ein Branching-Modell des Versionsverwaltungs-Tools Git[1].

3.1.1. SOFTWAREENTWICKLUNGSPROZESS INNERHALB DER PLATTFORM

Der Softwareentwicklungsprozess beginnt, wie üblich, mit einem Auftrag. Dieser wird durch speziell geschulte Software-Architekten analysiert und auf Umsetzbarkeit geprüft. Wurde zusammen mit dem Auftraggeber (beispielsweise durch Agile Workshops) eine gemeinsame Lösung gefunden, beginnen die Architekten mit der Aufteilung der Aufgaben und dem Erstellen der automatischen Abnahmetests. Die Abnahmetests werden in Form

[1] `http://git-scm.com/` (Letzter Abruf: 03.07.2017)

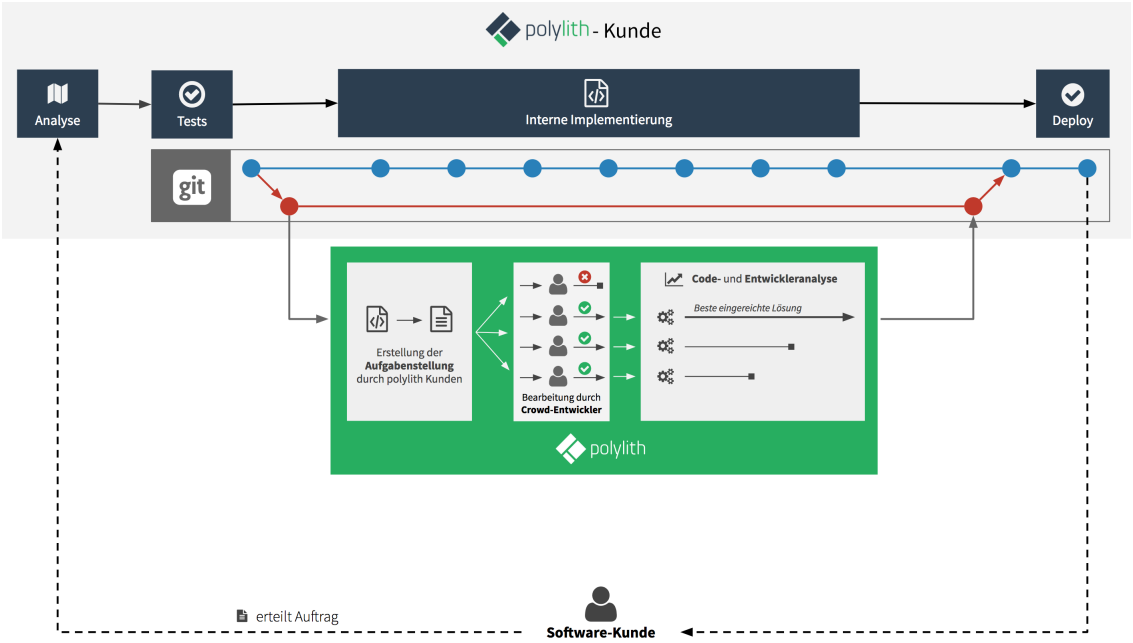


Abbildung 3.1.: Schematische Darstellung des test-getriebenen Crowdsourcing Ansatzes der polylith-Plattform als Ergänzung des Software-Entwicklungszyklus beim polylith-Kunden

natürlichsprachlicher Tests geschrieben und bilden gleichzeitig die Aufgabenstellung der Aufgabe. Dadurch sind Aufgabenstellung und Abnahmekriterien jederzeit konsistent und klar ersichtlich. Ist die Aufgabe hinreichend getestet, wird sie auf der Plattform zur Bearbeitung freigeschaltet. Eine Aufgabe ist hinreichend gut getestet, wenn sie im Rahmen des Plattformbetriebs die Testanforderungen erfüllt. Die Bestimmung, wann eine Aufgabe diese Bedingung erfüllt, ist nicht Teil dieser Arbeit.
Sobald eine Aufgabe veröffentlicht wurde, kann sie von einem oder mehreren Crowd-Entwicklern angenommen werden. Die Anzahl der Entwickler die gleichzeitig die Aufgabe bearbeiten dürfen, richtet sich nach den Vorgaben der Kunden. Mehr gleichzeitige Entwickler gewährleisten eine schnellere Bearbeitung und eine höhere Erfolgsquote, sind aber gleichzeitig teurer (da jede korrekt abgegebene Aufgabe auch bezahlt wird). Das Lösen der Crowdsourcing-Aufgabe findet direkt innerhalb der Plattform in einer bereitgestellten Web-IDE statt (Abbildung 3.2). Dadurch hat der Crowd-Entwickler keinerlei Aufwand beim Aufsetzen einer lokalen Entwicklungs- und Testumgebung. Weiterhin kann dadurch der Entwicklungszyklus jeder Aufgabe analysiert werden und beispielsweise auch Betrugsversuche (durch Kopieren einer korrekten Lösung) aufgedeckt werden. Zusätzlich wird der Entwickler durch den Editor unterstützt, indem unterstützende Grafiken (z.B. Klassendiagramme) eingeblendet werden oder Quellcode, der für die Lösung der Aufgabe nicht relevant ist, ausgeblendet wird. Die Web-IDE fügt den Quellcode des Projekts im Hintergrund wieder zusammen und ermöglicht dadurch die Bearbeitung eines kleineren Projekts für den Crowd-Entwickler.
Vollendet der Crowd-Entwickler die Aufgabe und erfüllt alle Tests, kann sie abgegeben werden. Bei der Abgabe wird der Quellcode einer erneuten Qualitätsüberprüfung unterzogen. Hierbei werden nicht-funktionale Qualitätsmerkmale wie Lesbarkeit, Doppelter Quellcode, Verschachtelungstiefe und weitere Code Smells (vgl. Fowler und Beck (1999)) untersucht.

Der Nutzer soll bei besserer Erfüllung dieser Kriterien zusätzlich unentgeltlich, in Form eines Gamification-Ansatzes, belohnt werden. Dadurch sollen sich die Entwicklungsfähigkeiten der einzelnen Crowd-Entwickler verbessern. Durch die Qualitätsanalyse wird weiterhin die Notwendigkeit eines zusätzlichen manuellen Reviews erkannt.
Sind alle Aufgaben durch die Crowd gelöst, können diese zu einem gemeinsamen Endergebnis zusammengefasst werden. Aufgrund des test-getriebenen Ansatzes erfüllt diese Lösung exakt die Anforderungen, die zuvor in den Tests aufgestellt wurden. Das fertige Ergebnis kann nach einem letzten Qualitätssicherungsschritt dem Auftraggeber übergeben werden.

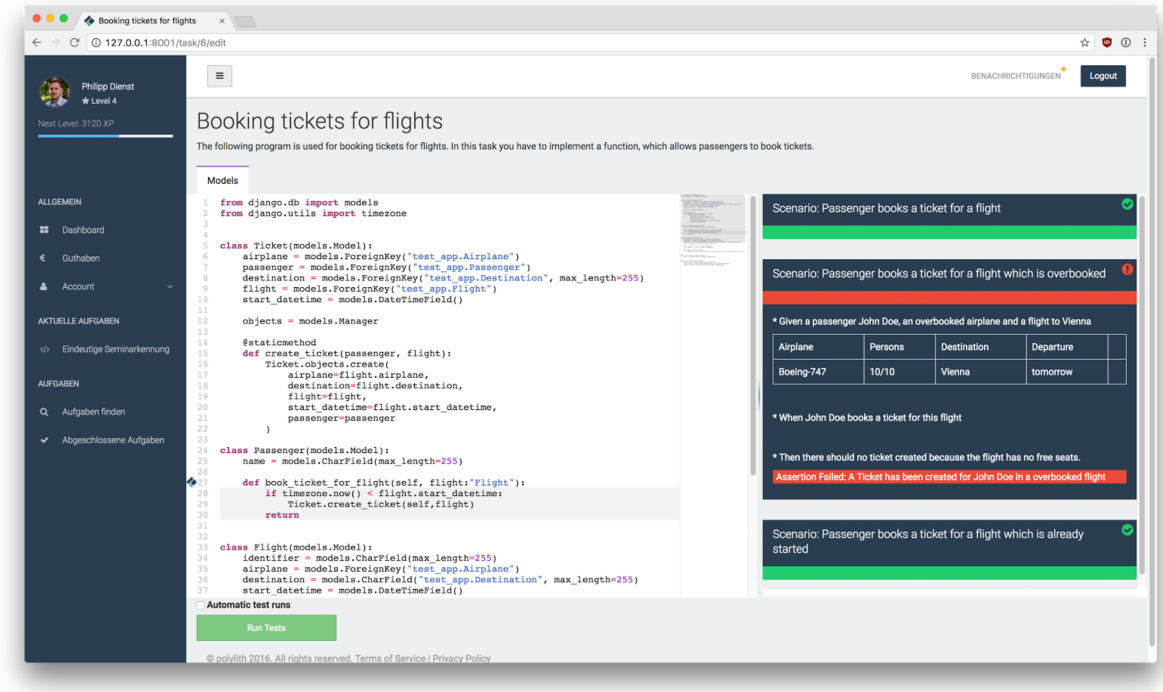


Abbildung 3.2.: Prototyp des Code-Editors der Plattform mit integrierten Hilfen, wie dem Ausblenden von (für diese Aufgabe) unnötigem Quellcode Bildquelle: Dienst (2017)

3.1.2. CROWD-ENTWICKLER

Die Zielgruppe der Crowd-Entwickler der Plattform sind in erster Linie Informatikstudenten. Diese sind noch keine fertig ausgebildeten Fachkräfte, verfügen aber trotzdem schon über das nötige Fachwissen, um abgegrenzte Programmieraufgaben zu lösen. Der Code-Editor unterstützt diese bei der Lösung der Aufgabe, die Plattform verhindert gleichzeitig durch automatisierte Tests die Abgabe von fachlich falschem Quellcode.
Die Hauptzielgruppe der Plattform bildet somit eine Gruppe von Entwicklern, die nur wenig oder keine Praxiserfahrung haben. Dies erfordert ein Qualitätssicherungskonzept, durch das auch nicht-funktionale Qualität überprüft wird. Die Crowd-Entwickler brauchen weiterhin Feedback, um zu erfahren, an welchen Stellen Verbesserungsbedarf vorhanden ist. An dieser

Stelle soll das Gamification-Konzept dieser Arbeit greifen. Einerseits soll dadurch der Spaß am Lösen der Programmieraufgaben erhöht werden, andererseits sollen die Fähigkeiten der Crowd-Entwickler spielerisch verbessert werden.

3.1.3. DATENGRUNDLAGE

Eine wichtige Datengrundlage bildet das *Entwickler-Tracking* der Plattform. Durch die Verwendung eines Web-Code Editors, wird die Aufgabe vollständig innerhalb der Plattform gelöst. Dies bedeutet zwar erheblichen Aufwand für die Infrastruktur, resultiert jedoch auch in einer großen Menge an Rohdaten, die für Analysen zur Verfügung stehen. Die Plattform kann damit sämtliche Zwischenstände, die bei der Entstehung der Lösung erstellt werden, analysieren. Hierbei ist auch die Verbindung zu den programmatischen Tests von Bedeutung. Zu jedem geschrieben Quellcode ist dadurch eine Zuordnung zu einem Testfall möglich (Abbildung 3.3). Weiterhin kann durch den Entstehungsverlauf bestimmtes Nutzerverhalten erkannt werden. So kann beispielsweise die Nutzung von Refactorings festgestellt werden, ohne, dass der Nutzer das dafür vorgesehene Refactoring-Tool des Editors verwendet.
Die aufgezeichneten und teils ausgewerteten Rohdaten bilden die Grundlage für das Gamification-Konzept. Ohne die Analysen des Entwickler-Trackings können viele beschriebene Konzepte (vgl. Abschnitt 4.1.2) nicht umgesetzt werden.

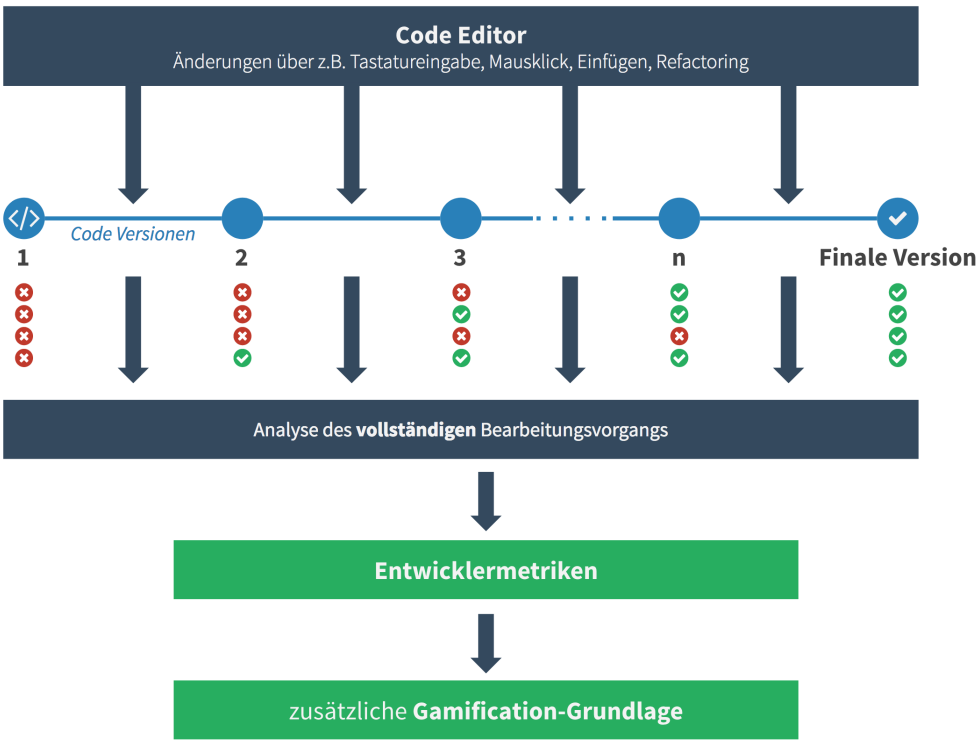


Abbildung 3.3.: Durch den Workflow in der polylith Plattform können Daten über den genauen Ablauf der Softwareentwicklung aufgezeichnet und verarbeitet werden. Diese werden ausgewertet und als zusätzliche Gamification-Grundlage verwendet. Die Entwicklermetriken bilden gleichzeitig eine interne Bewertungsgrundlage der Crowd-Entwickler

4. KONZEPT

In diesem Kapitel wird das Konzept, welches anhand der Anforderungsanalyse in Abschnitt 2.5 ausgearbeitet wurde, beschrieben. Das Konzept soll zugeschnitten auf die Crowdsourcing-Plattform (beschrieben in Kapitel 3.1) erarbeitet werden. Das Kapitel spaltet sich in zwei Teile: das Gamification-Konzept und das Implementierungskonzept.

4.1. GAMIFICATION KONZEPT

4.1.1. ZIELE DES GAMIFICATION-ANSATZES

Um ein sinnvolles Gamification-Konzept zu entwerfen ist es wichtig, das Verhalten zu definieren, welches vom idealen Nutzer erwünscht wird. Inwieweit der Benutzer dieses Verhalten erfüllt, wird durch die in diesem Kapitel vorgestellten Metriken gemessen. Das Gamification-Konzept hat vier grundlegende Ziele:

1. Die Verbesserung der abgegebenen Code-Qualität,
2. Die Bildung einer Langzeitmotivation,
3. Die interne Bewertung der Entwickler anhand der durch die Gamification erlangten Metriken und
4. Die Exploration und Nutzung der Plattform-Features.

Um die, durch den Entwickler eingereichte Code-Qualität zu verbessern und gleichzeitig eine Vergleichbarkeit unter den Entwicklern zu erreichen, werden als Basisdaten die, durch den Code-Editor aufgezeichneten Entstehungsdaten verwendet. Anhand dieser Daten werden Metriken gebildet, mit deren Hilfe die Performance des Entwickler gemessen werden kann. Die Gamification soll den Nutzer hierbei so belohnen, dass die Metriken maximiert werden.
Die aufgezählten Rohdaten der genannten Metriken werden vom Code-Editor der Plattform und zugehörigen Berechnungskomponenten zur Verfügung gestellt. Die Beschreibung der Bestimmung der Rohdaten ist nicht Teil dieser Arbeit und wird separat in Blume (2017) behandelt.

4.1.2. PERFORMANCE-METRIKEN VON CROWD-ENTWICKLERN

In diesem Abschnitt werden Metriken erläutert, die für die Crowdsourcing-Plattform relevant sind. Die Metriken sollen sowohl eine Vergleichsgrundlage für das Verhalten des Nutzers auf der Plattform im Allgemeinen (Logins, Nutzung von Plattformfeatures, Teilen auf sozialen Netzwerken, etc.), als auch die Performanz des Entwicklers beim Lösen der Aufgaben (Code-Qualität, Schnelligkeit bei der Bearbeitung, Verlässlichkeit, etc.) beschreiben. Die beschriebenen Metriken sind aus der Perspektive des Anbieters der Crowdsourcing Plattform beschrieben. Die beschriebenen Metriken sollen die Grundlage für das Gamification Konzept bilden. Weitere Metriken zur Code-Qualität und Details zu deren Bestimmung werden in Blume (2017) beschrieben. Die aufgezählten Metriken können hierbei stets durch weitere Metriken ergänzt werden.
Die hier vorgestellten Metriken sollen durch das, in dieser Arbeit vorgestellte Gamification-Konzept verbessert werden.

VERLÄSSLICHKEIT

Die Verlässlichkeit eines Entwicklers beschreibt, welchen Prozentanteil der angenommenen Aufgaben korrekt abgenommen wurde. Für die Crowd-Sourcing Plattform ist dies eine der wichtigsten Metriken, da sie beschreibt, mit welcher Wahrscheinlichkeit eine angenommene Aufgabe von einem Entwickler erledigt wird. Da das einfache Verhältnis der gelösten und angenommenen Aufgaben wenig Aussagekraft hat, solange die Anzahl der eingereichten Aufgaben sehr niedrig ist, wird diese Metrik als Vektor zusammen mit der Anzahl der eingereichten Aufgaben gebildet.

$$reliability = \left(\frac{\text{Anzahl korrekt gelöster Aufgaben}}{\text{Anzahl angenommener Aufgaben}}, \text{Anzahl eingereichte Lösungen} \right)^T \tag{4.1}$$

AKTIVITÄT

Die Aktivität beschreibt, wie häufig ein Nutzer die Plattform nutzt, d.h. sich einloggt, Aufgaben annimmt, usw. Idealerweise loggt sich ein Benutzer regelmäßig ein und löst regelmäßig Aufgaben. Die Aktivität von einem Zeitraum kann gewichtet durch die Gewichte w_a und w_l für die Anzahl der abgegebenen Aufgaben a sowie die Anzahl der Logins l berechnet werden, indem ein normalisierter Score über die zu bewertenden Unterzeiträume N bestimmt wird:

$$activity = \frac{\sum_{i=0}^{N}(w_a a_i + w_l l_i)}{N} \tag{4.2}$$

Eine höhere Aktivität kann gut durch Gamification-Mechaniken motiviert werden. Insbesondere die Langzeitmotivation beim Lösen der Aufgaben ist eine der Kernaufgaben des Motivations-Konzepts.
Zusätzlich ist die reine Häufigkeit von Abgaben wichtig. Nutzer die, insbesondere in einem zeitnahem Zeitraum, mehr Aufgaben abgeben und damit aktiver sind, sollten zusätzlich motiviert werden.
Die oben beschriebene Metrik kann nach Bedarfsfall beliebig zur Messung anderer Aktivitäten verwendet oder erweitert werden. So kann beispielsweise die Social-Media Aktivität eines Nutzers, oder angenommene, jedoch nicht gelöste Aufgaben in die Metrik einfließen.

AUSREIZUNG DER BEARBEITUNGSZEIT

Um die Planung der Aufgaben auf der Plattform einfacher zu gestalten, ist es hilfreich möglichst frühzeitig eine funktionierende Lösung zu einer Aufgabe eingereicht zu bekommen. Aus diesem Grund sollte die Ausreizung minimiert werden. Die Metrik wird beschrieben durch:

$$timely\_score = \frac{\text{Gesamte Ausschreibungszeit}}{\text{verbleibende Ausschreibungszeit nach Abgabe}} * 100\% \tag{4.3}$$

AKZEPTANZ DER ABGEGEBENEN AUFGABE

Die Akzeptanz der abgegebenen Aufgaben beschreibt, wie viele der eingereichten, korrekten Lösungen letztendlich ausgewählt als beste Lösung wurden und in das Endprodukt integriert sind:

$$acceptance = \frac{\text{Anzahl für Lösung ausgewählte Aufgaben}}{\text{Anzahl korrekt gelöster Aufgaben}} \tag{4.4}$$

Die Akzeptanz ist intern eine wichtige Metrik, da es eine Zusammenfassung der abgegebenen Code-Qualität darstellt. Der Crowd-Entwickler bekommt keinen Einblick, ob die abgegebene Aufgabe ausgewählt wird, da das die Motivation verringern könnte. Aus diesem Grund kann das Gamification Konzept diese Metrik nicht direkt beeinflussen. Indirekt können die Metriken verbessert werden, die dazu führen, dass eine Lösung nicht ausgewählt wurde. Durch Verbesserung der Metriken auf Quellcodebasis (z.B. Refactoring Score, Lesbarkeit, etc.) kann somit indirekt die Akzeptanz erreicht werden.

REFACTORING SCORE

Der Refactoring Score wird exemplarisch für Scores auf Quellcode-Ebene detaillierter ausgeführt. Die Entwicklung zusätzlicher komplexer Bewertungsmechanismen für den abgegebenen Quellcode innerhalb der Crowdsourcing-Plattform werden von Blume (2017) beschrieben.

Auf Quellcode-Ebene kann überprüft werden, inwiefern ein Crowd-Entwickler schlechten Code selbstständig erkennt und nachbessert. Bei der Entstehung von Quellcode lassen sich qualitativ schlechte Passagen nicht vermeiden. Das Ausbessern solcher *Code Smells* und Anti-Patterns wird *Refactoring* genannt. (Fowler und Beck 1999) Um die Performanz eines Entwicklers zu messen, kann überprüft werden, ob der Entwickler solche *Code Smells* behebt. Die untersuchten *Smells* und Refactorings sind aus *Refactoring: Improving the Design of Existing Code* von Fowler und Beck (1999) entnommen und in Tabelle 4.1.2 beschrieben.

Gefundene Refactoring Vorschläge erkennt der Code-Editor (vgl. **Dientst2017**) bzw. das Entwickler-Tracking (vgl. Blume (2017)) der Plattform und zeigt diese direkt im Quellcode an. Dadurch sieht der Nutzer, an welchen Stellen Verbesserungsbedarf besteht und kann nachbessern.

Code-Smell	Beschreibung
Smell: Duplicate Code	Duplizierter Quellcode verringert die Wartbarkeit des Projekts und sollte möglichst durch Refactorings in Methoden und Funktionen ausgelagert werden.
Smell: Long-Method	Lange Methoden sind schwer zu überblicken und sollten aufgespaltet werden.
Smell: Long Parameter List	Zu viele Parameter sind schlecht zu überblicken und sollten durch Parameterobjekte ersetzt werden.
Smell: Comments	Kommentare an sich sind kein Smell, jedoch Kommentare, welche ausschließlich den Quellcode beschreiben. Bessere Kommentare beschreiben den Grund warum eine Zeile Code existiert.
Refactoring: Extract Method	Dieses Refactoring spaltet Quellcode auf und separiert ihn in eine abgegrenzte Funktion oder Methode. Dabei muss auf die korrekte Übergabe von Parametern und Rückgabewerten geachtet werden.
Refactoring: Introduce Explaining Variable	Oft ist es für die Lesbarkeit hilfreich einen Wert durch eine Variable mit einem erklärenden Namen zu ersetzen.
Refactoring: Remove Assignments to Parameteters	Methoden oder Funktionsparameter sollten nicht verändert werden, um die Übersichtlichkeit zu erhalten.
Refactoring: Replace Array with Object	Arrays sollten nicht verwendet werden um verschiedenartige Daten zu speichern. Stattdessen sollte ein Objekt eingeführt werden.
Refactoring: Replace Magic Number with Symbolic Constant	Statische Werte sollten in Konstanten ausgelagert werden.
Refactoring: Decompose Conditional	Komplexe `if`/`else` Verbindungen sollten augelöst werden und beispielsweise in eine Methode ausgelagert werden
Refactoring: Consolidate Duplicate Conditional Fragments	Wiederholte `if`/`else` Verbindungen sollten zusammengefasst werden und beispielsweise in eine Methode ausgelagert werden
Refactoring: Renaming Methods	Eine sinnvolle Umbenennung von Methoden und Funktionen sollte durchgeführt werden, um verständlichen Code zu erhalten. (Beispiel: `getinvcdtlmt` in `getInvoiceableCreditLimit`)

Tabelle 4.1.: Beispiele von Refactorings, die auf Quellcodeebene in kleinen Aufgaben erkannt werden können und vermieden werden sollten

Die Berechnung des Scores erfolgt anhand einer gewichteten Summe aller aufgezeichneten Smells und Refactorings. Nicht behobene *Smells* gehen negativ in die Bewertung ein, bereinigte werden positiv bewertet. Die Werte p_i und n_i stehen für positive bzw. negative Ereignisse und gehen mit ihrer jeweiligen Gewichtung w ein.

$$s_{refactoring} = \sum_{i=0}^{N} (w_p p_i + w_n n_i) \tag{4.5}$$

Da $s_{refactoring}$ je nach Code-Qualität beliebig hoch oder niedrig sein kann, wird der Score zusätzlich auf eine logarithmische Skala abgebildet.

$$refactoring\ score = \log\left(|s_{refactoring}| + 1\right) * sgn(s_{refactoring}) * 10 \tag{4.6}$$

Der Score soll möglichst auf Werte zwischen −2 und +2 abbilden und die Eigenschaften der Gaußschen Normalverteilung erfüllen. Um eine hinreichend gute Abbildung auf die Normalverteilung zu finden, müssen die Gewichte w_i angepasst werden. Die Bestimmung dieser Gewichtung ist ein erheblicher statistischer Aufwand und nicht Teil dieser Arbeit. Die Berechnung wird separat vorgenommen.
Um besser verständliche Zahlen (zwischen −20 und +20) zu erhalten, wird der Score abschließend mit 10 multipliziert.

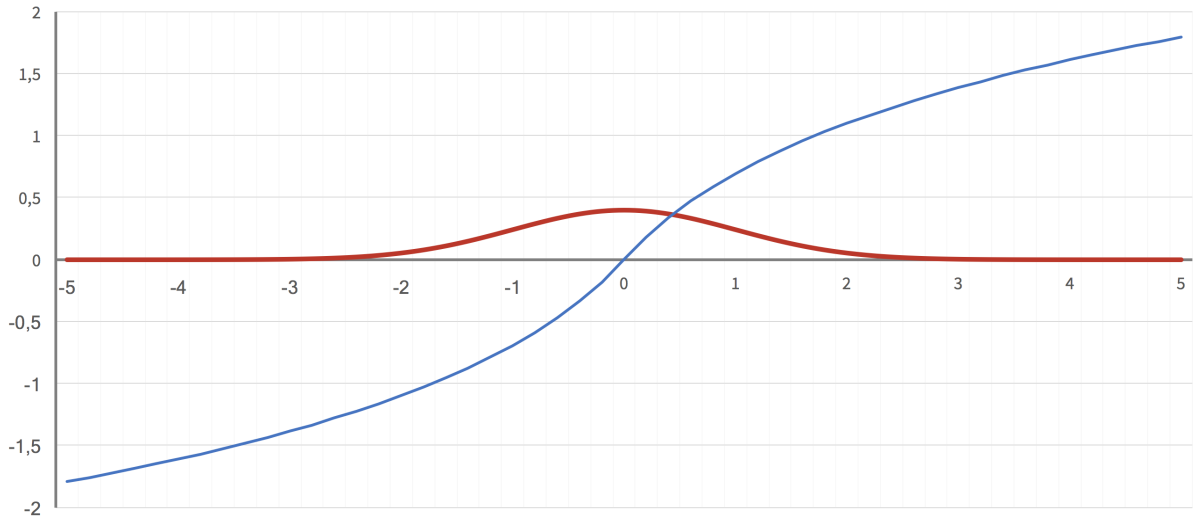


Abbildung 4.1.: Darstellung der Funktion des Refactoring Scores (im Bild ohne Multiplikation mit 10, um eine bessere Vergleichbarkeit der Funktionen zu erhalten) zusammen mit der Gaußschen Normalverteilung bei Standardabweichung $\sigma = 1$

CODE-LESBARKEIT UND WEITERE METRIKEN AUF QUELLCODEBASIS

Weitere Metriken wie Code-Lesbarkeit werden durch Blume (2017) entwickelt und beschrieben. Die Berechnung dieser Scores erfolgt beispielsweise durch Machine-Learning Algorithmen. Letztendlich werden diese ebenso auf einen numerischen Score abgebildet. Dieser kann dann analog zum *Refactoring Score* auf eine logarithmische Skala abgebildet und weiter genutzt werden.

4.1.3. ERFAHRUNGSPUNKTE UND LEVEL

Für viele Aktionen auf der Plattform erhält der Nutzer Erfahrungspunkte. Die Anzahl der gewährten Erfahrungspunkte orientiert sich an der Leistung des Nutzers in Bezug auf die in Abschnitt 4.1.2 genannten Metriken.
Erreicht der Spieler eine gewisse Anzahl Erfahrungspunkte, steigt er ein Level auf. Die Abbildung der Erfahrungspunkte auf Level wird durch die folgende Funktion angegeben:

$$Level = \lfloor \sqrt{XP} * 0.1 \rfloor \tag{4.7}$$

Die Funktion ist so gewählt, dass die Anzahl der nötigen Erfahrungspunkte mit jedem Level ansteigt. Diese Eigenschaft wird durch die Wurzelfunktion erreicht. Dadurch wird es zunehmend zeitaufwändiger, höhere Level zu erreichen. Der Anpassungsfaktor 0.1 erreicht, dass die Zahlen im gewünschten Zahlenbereich liegen. Da ein Level stets nur ganzzahlig sein kann, wird das berechnete Level durch Abrunden der Level-Funktion erreicht (Abbildung 4.2).

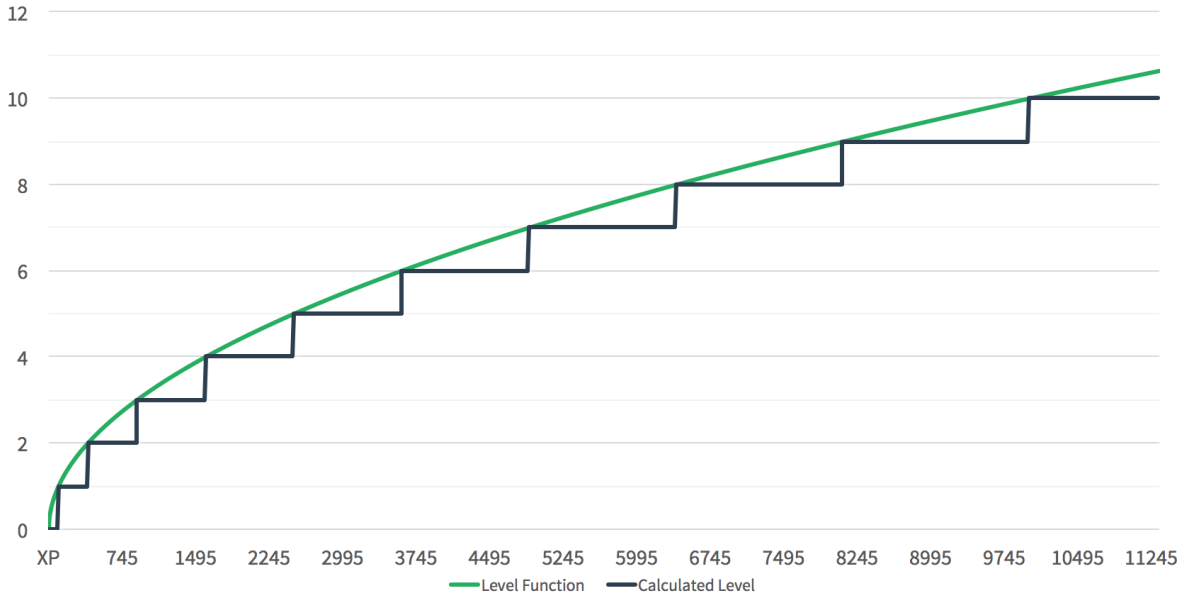


Abbildung 4.2.: Berechnung des Levels durch Abrunden der Level-Funktion

Die Erfahrungspunkte werden so modelliert, dass mit jedem Level der Fortschritt erneut bei 0 beginnt. Dadurch füllt der Nutzer mehrfach den Erfahrungsbalken komplett aus. Um diese Eigenschaft abzuleiten, werden zusätzlich zu den absoluten Erfahrungspunkten, relative Punkte bestimmt und stattdessen im Frontend verwendet:

$$EXP_{relativ} = EXP_{absolut} - EXP_{aktuellesLevel} \tag{4.8}$$

Um dem Nutzer das Level und die Erfahrungspunkte allgegenwärtig zu machen, sollen sie dauerhaft eingeblendet werden. Dafür wird innerhalb einer Seitenleiste ein Miniprofil mit einem Profilfoto, dem aktuellen Level, der Anzahl Erfolgspunkte und dem Fortschritt zum nächsten Level angezeigt. Zu diesem Zweck wurden mehrere Varianten des Mini-Profils entworfen (Abbildung 4.3). In den Varianten sind die Gamification-Elemente unterschiedlich präsent.

Die Erfahrungspunkte und Level sollen insbesondere die Langzeitmotivation der Nutzer erhöhen. Während Erfolge irgendwann erschöpft sein können, sind Erfahrungspunkte und Level endlos vorhanden. Dadurch werden die Nutzer weiterhin motiviert Aufgaben gut in Bezug auf die oben genannten Metriken zu erfüllen.
Nach der Abgabe jeder Aufgabe wird dem Nutzer eine Übersicht angezeigt, in der dargestellt wird, wie viele Erfahrungspunkte er für die abgeschlossene Aufgabe erhält. Zusätzlich zu einem Grundbetrag von 250 Erfahrungspunkten erhält er in Abhängigkeit der abgegebenen Code-Qualität und der Aufgabe, weitere Erfahrungspunkte. Hierbei werden die in Abschnitt 4.1.2 vorgestellten Metriken verwendet. Der Nutzer erhält zusätzliche Erfahrung für:

1. Abgaben mit höherer Schwierigkeit
2. Quellcode mit besserer Lesbarkeit (vgl. 4.1.2 *Code-Lesbarkeit*)
3. Durchgeführte Refactorings und behobene Code-Smells (vgl. 4.1.2 *Refactoring Score*)
4. Quellcode mit besserer Lesbarkeit (vgl. 4.1.2 *Aktivität*)
5. das Abschließen einer Aufgabe in einer bisher ungenutzten Programmiersprache
6. eine zeitnahe Abgabe, möglichst lange vor dem finalen Abgabedatum (vgl. 4.1.2 *Ausreizung der Bearbeitungszeit*)

Erfahrungspunkte können jede einzelne der Spielergruppen nach Bartle (1999) auf verschiedene Arten ansprechen. So sind *Achievers* und *Killers* offensichtlich daran interessiert Erfahrungspunkte zu sammeln und einen höheren Level zu erreichen als andere Nutzer. Insbesondere für diese Spielergruppen sind Erfahrungspunkte und Level interessant, da kein Maximum existiert. Gleichzeitig kann diese „ziellosigkeit" jedoch auch demotivierend sein. Dem wird mit der Implementierung von Ranglisten entgegengewirkt, sodass zumindest ein dauerhafter Vergleich möglich ist.
Explorers sind interessiert herauszufinden wofür sie (Bonus-)Erfahrung erhalten. Hier ist das Ziel bei der Weiterentwicklung des Gamification-Konzepts dauerhaft neue Anreize und Überraschungen zu bieten. Die Gruppe der *Socializers* können sich mit Spielern desselben Levels identifizieren und sich innerhalb der Ranglisten vergleichen.

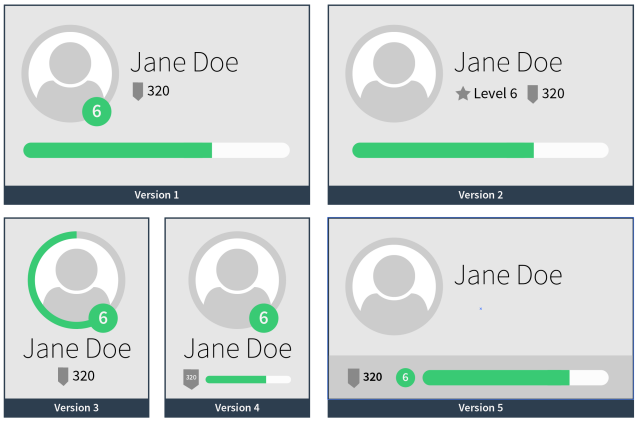


Abbildung 4.3.: Mockup-Varianten des Mini-Profils innerhalb der Plattform.

BONUSERFAHRUNG DURCH EREIGNISSE

Ist absehbar, dass in einem bestimmten Zeitraum besonders viele Implementierungsaufgaben anstehen, können die Erfahrungspunkte genutzt werden, um in diesen Zeiträumen zusätzliche Nutzer zu motivieren Aufgaben zu lösen. Viele Online-Spiele nutzen solche Bonus-Ereignisse bereits, um kurzzeitig zusätzliche aktive Spieler zu akquirieren.[1] [2] [3] Dies wird beispielsweise angeboten, um eine anstehende Erweiterung zu bewerben. In den Bonus-Ereignissen erhalten die Nutzer einen prozentualen-Bonus auf alle Erfahrungspunkte. Solche Ereignisse können genutzt werden, um die genannten Auftragsspitzen abzufangen und zusätzliche Nutzer auf die Plattform zu locken.

4.1.4. ERFOLGE

Die Erfolge auf der Plattform sind so konzipiert, dass sie hinreichend auf die Spielertypen nach Bartle (vgl. Kapitel 2.2.2) verteilt sind. Gleichzeitig sollen die Nutzer der Plattform ermutigt werden, die oben genannten Metriken zu erfüllen und Plattform-Funktionen auszuprobieren. Abbildung 4.4 stellt verschiedene Mockups zu Erfolgen innerhalb der Plattform da.

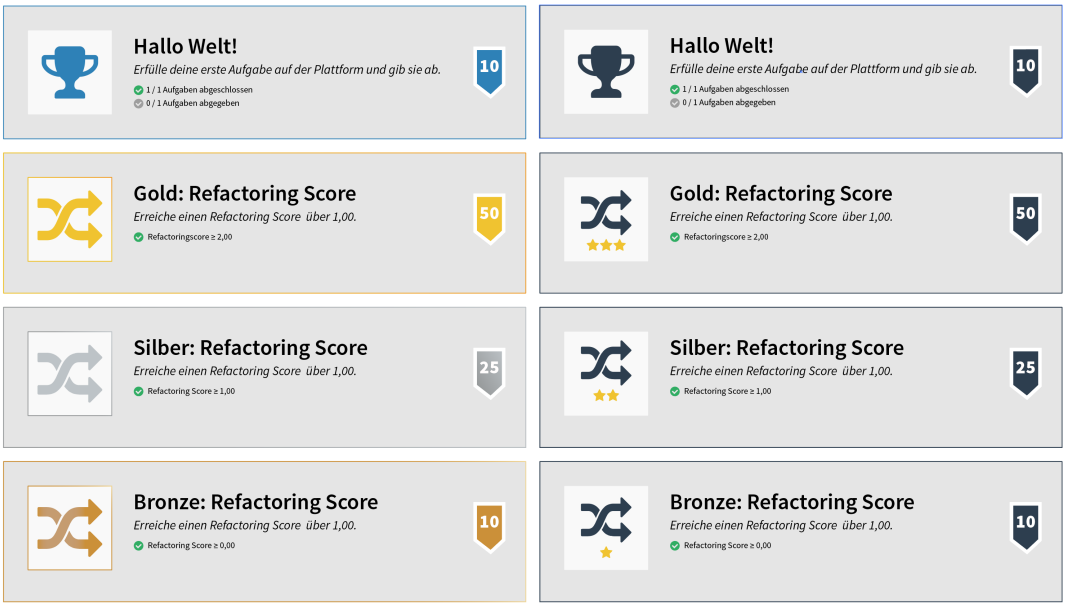


Abbildung 4.4.: Mockups eines Erfolgs für die polylith-Plattform

Der Entwurf des Erfolgsmodells beruht auf dem, in Kapitel 2.1.4 vorgestellten Framework von Hamari (2011). Das Framework beschreibt Erfolge als drei-geteilte Elemente aus einem *Signifier*, der *Completion Logic* und einem *Reward*. Details zum Implementierungskonzept zu Erfolgen werden in Kapitel 4.2 näher erläutert.

[1] Beispiel Overwatch: `https://playoverwatch.com/en-us/blog/20845154` (Letzter Abruf: 05.08.2017)
[2] Beispiel RuneScape: `http://runescape.wikia.com/wiki/Double_XP_Weekend` (Letzter Abruf: 05.08.2017)
[3] Beispiel Star Wars - The Old Republic: `https://www.reddit.com/r/swtor/comments/56c4ih/200_exp_now_active/` (Letzter Abruf: 05.08.2017)

Um eine Verteilung auf die verschiedenen Spielertypen zu erreichen werden die Erfolge mehrfach in Kategorien eingeteilt. Einerseits wird in – vor dem Erringen – sichtbare und unsichtbare Erfolge unterschieden. Unsichtbare Erfolge sind insbesondere für die Gruppe der *Explorers* spannend. Sie regen außerdem zum Ausprobieren unbekannter Funktionen an. Weiterhin bieten sie die Möglichkeit, Nutzer zu belohnen, Aktivitäten durchzuführen, die möglicherweise nicht für jeden Benutzer relevant sind. Ein Beispiel hierfür ist die *Fehler melden* Funktion. Nutzer sollen möglichst alle Probleme auf der Plattform mitteilen, jedoch sollen keine Meldungen erstellt werden, nur um Erfolge zu erreichen.
Andererseits werden Erfolge in einstufige und mehrstufige Erfolge unterteilt. Mehrstufige Erfolge sind beispielsweise Errungenschaften zum Level (Erreiche Level 1, Level 10, Level 20, etc.) oder analog dazu erfüllte Aufgaben auf der Plattform.

NUTZUNG DER NORMALVERTEILUNG

Um die Performance eines Nutzers in Bezug auf eine normalverteilte Metrik zu messen, können die Eigenschaften der Normalverteilungsfunktion $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}t^2} dt$. genutzt werden. Um die Steigerung der Performance in einer bestimmten Metrik stärker zu belohnen werden die drei Intervalle $[\mu, \mu + \sigma)$, $[\mu + \sigma, \mu + 2\sigma)$ und $[\mu + 2\sigma, \infty)$ genutzt und auf die drei Erfolgskategorien **Bronze**, **Silber** und **Gold** abgebildet. Dies hat den Vorteil, dass aufgrund der Beschaffenheit der zugrundeliegenden Metrik gleichzeitig eine höhere Schwierigkeit für Silber und Gold-Erfolge vorliegt (Abbildung 4.5). Weiterhin können sich die Nutzer somit besser selbstständig einschätzen.

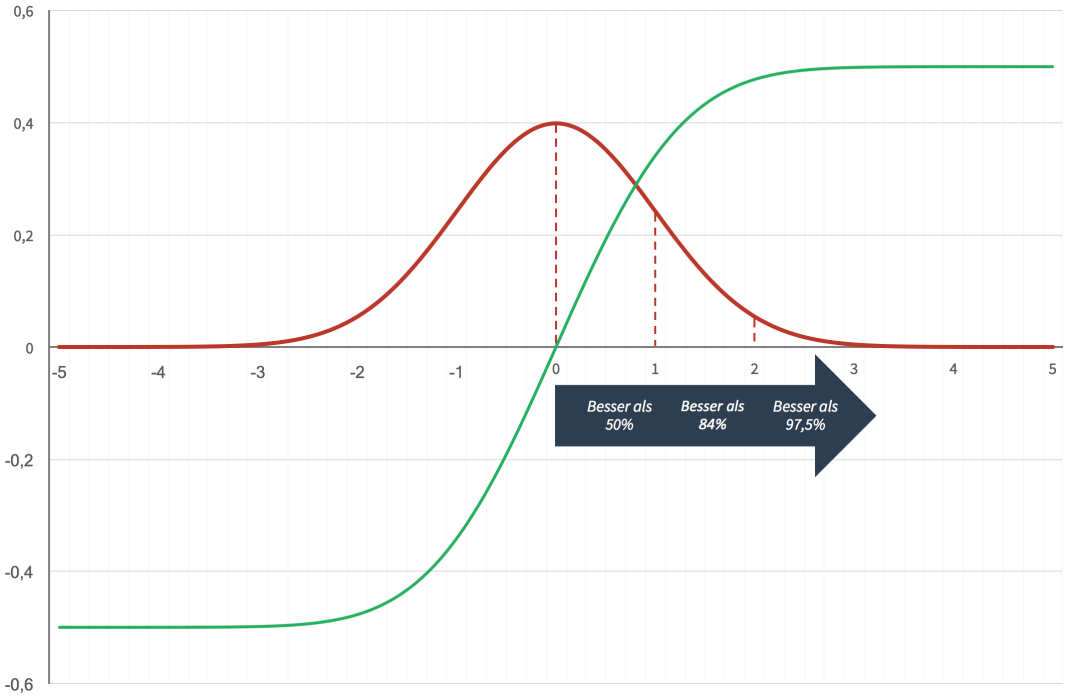
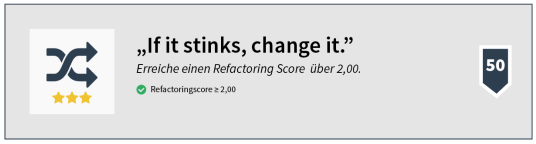
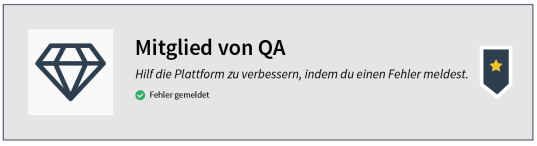


Abbildung 4.5.: Wertebereich der einzelnen Ränge für Erfolge innerhalb der Normalverteilung

BEISPIELE FÜR ERFOLGE

Die folgende Tabelle stellt eine Liste mit Erfolgen zusammen, die für die Plattform entworfen wurden in Verbindung mit der Metrik, die sie verbessern und dem Spielertyp nach Bartle, den sie ansprechen. Erfolge, die mit mehreren Rängen verbunden sind, wurden weggelassen, um die Übersichtlichkeit zu erhöhen.

Beispiel	Metrik	Spielertyp
	Aktivität	Explorer
	Aktivität	Achievers
	Abgabezeit	Achievers, Explorer
	Lesbarkeit	Achievers
	Aktivität	Achievers, Explorer
	Refactoring Score	Achievers
	Refactoring Score, Lesbarkeit	Achievers
	Aktivität	Explorer

Beispiel	Metrik	Spielertyp
	Aktivität, Refactoring Score	Achievers, Killers
	alle	Socializers, Killers
	alle	Socializers, Killers

4.1.5. RANGLISTEN

Um die Vergleichbarkeit der Nutzer untereinander zu erhöhen, werden öffentliche[4] Ranglisten implementiert (vgl. Abschnitt 2.1.3, Anforderung **S12**). Solche Listen sind eine einfache Möglichkeit, um die bereits vorhandenen Gamification-Elemente, wie Erfolgspunkte, Erfahrungspunkte und Level zu nutzen und zu verbinden, um zusätzliche Anreize zu schaffen. Zusätzlich werden die in Abschnitt 4.1.2 vorgestellten numerischen Metriken *Refactoring Score*, *Code-Lesbarkeit* und *Ausnutzung der Ausschreibungszeit* für Ranglisten verwendet, um Nutzer zusätzlich anzuspornen, gute Werte zu erreichen. Die Listen verbessern insbesondere das Nutzererlebnis der Spielertypen Socializers, aber auch Killers und Achievers.

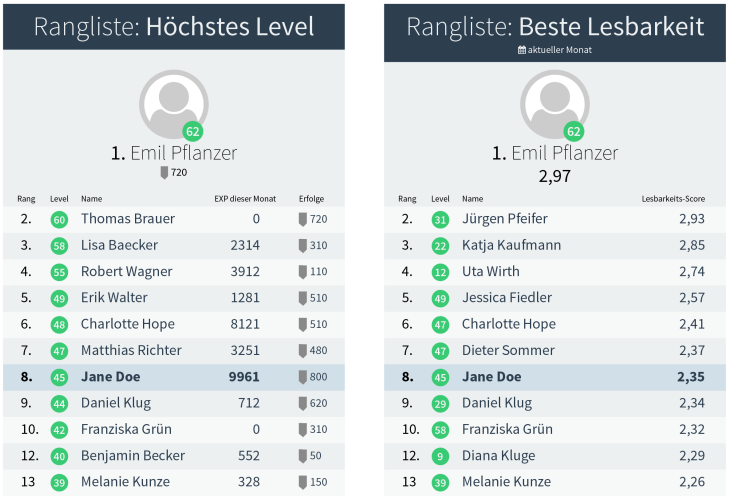


Abbildung 4.6.: Mock-up von Ranglisten der Nutzer mit dem höchsten Level und der besten Lesbarkeit des aktuellen Monats

[4] Für alle angemeldeten Nutzer auf der Plattform sichtbar.

Ein Mockup einer solchen Rangliste wird in Abbildung 4.6 dargestellt. Der eingeloggte Anwender (im Beispiel *Jane Doe*) wird hervorgehoben. Der Nutzer auf dem ersten Platz wird besonders präsent dargestellt, um seine Position deutlicher zu machen. Dadurch wird der Anreiz zusätzlich erhöht, in einer der Kategorien den ersten Platz zu erreichen.
Die Ranglisten (bis auf *Höchstes Level*) sind zugeschnitten auf einen bestimmten Zeitraum.

4.1.6. ZUSAMMENFASSUNG

Im ersten Teil dieses Abschnitts wurden Metriken beschrieben, welche die Performance eines Crowd-Entwicklers abbilden und durch das Gamification-Konzepts der Plattform verbessert werden. Darauf folgend wurde die Nutzung von Spielelementen, welche in Abschnitt 2.1.3 beschrieben wurden, im Kontext der Plattform vorgestellt. Die einzelnen Elemente wurden in als Mock-Ups ausgearbeitet und beispielhaft dargestellt. Tabelle 4.1.6 beschreibt zusammenfassend, welche Metriken durch die Auswahl welcher Spielelemente verbessert wird.

Metrik	Spielelemente zur Verbesserung
Verlässlichkeit	**Erfahrungspunkte** für Abgabe von Aufgaben
Aktivität	**Erfahrungspunkte** für Aktionen auf der Plattform. **Erfolge** als Motivation für Exploration. Bei besonders hoher Aufgabenauslastung **Bonus-Ereignisse**
Ausreizung der Ausschreibungszeit	Bonus-**Erfahrungspunkte**, **Erfolge** für Überschreitung eines Grenzwertes und **Ranglisten** für das Erreichen von Best-Werten
Akzeptanz	Wird indirekt verbessert durch bessere Code-Lesbarkeit und besseren Refactoring-Score
Code-Lesbarkeit	Bonus-**Erfahrungspunkte**, **Erfolge** für Überschreitung eines Grenzwertes und **Ranglisten** für das Erreichen von Best-Werten
Refactoring-Score	Bonus-**Erfahrungspunkte**, **Erfolge** für Überschreitung eines Grenzwertes und **Ranglisten** für das Erreichen von Best-Werten

Tabelle 4.2.: Verbindung von Gamification Elementen und Metriken

4.2. IMPLEMENTIERUNGSKONZEPT

In diesem Abschnitt wird das Konzept der Implementierung beschrieben. Dabei werden alle Komponenten der entwickelten *polylith Gamification Engine* (**PGE**) anhand ihrer Rolle innerhalb der Crowdsourcing-Plattform erklärt.

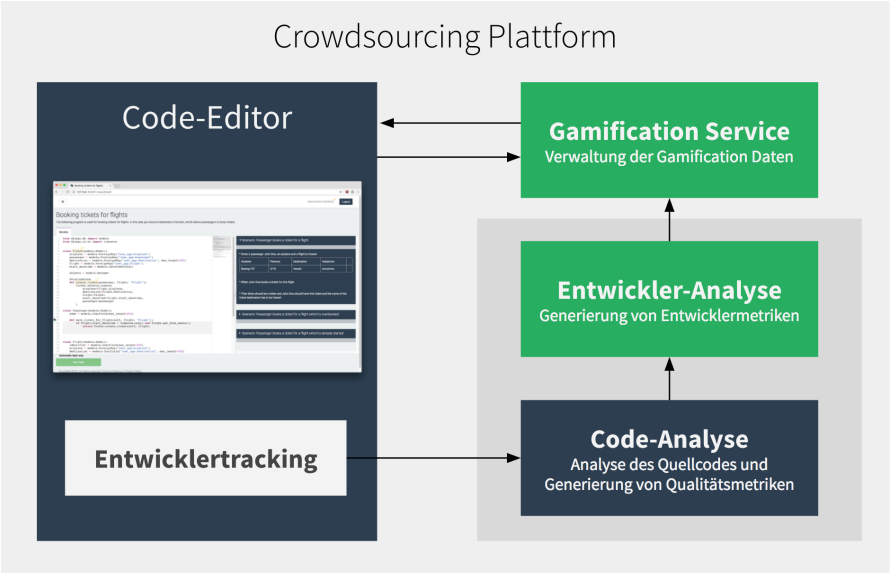


Abbildung 4.7.: Schematische Darstellung des Konzepts dieser Arbeit im Kontext der polylith-Plattform. Die in der Arbeit betrachteten Teile sind grün dargestellt.

Die in der Arbeit entwickelte Service-Komponente fügt sich in die Architektur der Crowdsourcing-Plattform (siehe Kapitel 3.1) ein und interagiert mit deren Komponenten. Die Grundlage für die oben beschriebenen Analysen sind die Daten, die der Code Editor (beschrieben durch Dienst (2017)) bereitstellt, sowie die darauf durchgeführten Code-Analysen (beschrieben durch Blume (2017)) (Abbildung 4.7). Gleichzeitig ist jedoch die Generizität der Lösung ein wichtiges Kriterium bei der Wahl der Architektur, sodass der Service problemlos innerhalb einer anderen Plattform verwendet werden kann. Die Analysen werden durch Funktionen (**Processing Functions**) durchgeführt, die ereignis- und daten-orientiert agieren. Sie werden bei Änderungen der Datengrundlage automatisch gestartet und aktualisieren selbstständig die Analysen anhand der neuen Datensätze. Somit arbeiten die Analysen asynchron und unabhängig voneinander (*loose coupling*). Die Daten auf denen die *Processing Functions* arbeiten werden in einer Rohdatenbank gehalten. Diese ist vor allem auf Schreibzugriffe, generische Datensätze und Skalierbarkeit ausgelegt.
Um die einzelnen *Processing Functions* zu verwalten wird ein Server (**PGE-Leader**) bereitgestellt. Dieser dient einerseits der Verwaltung der *Processing Functions* (siehe Abschnitt 4.2.2), andererseits der Speicherung der Meta-Daten und dem Auslesen der Ergebnisdaten. Weiterhin werden in diesem die Erfahrungspunkte und Level der Nutzer berechnet. Der PGE-Leader Server verwendet eine Ergebnisdatenbank, welche auf Aggregationen und häufige Lesezugriffe ausgelegt ist. Zusätzlich beinhaltet der Server eine **Live-Komponente**, welche für das Versenden von Benachrichtigungen an die Clients zuständig ist. Abbildung 4.8 stellt die einzelnen Komponenten und Datenbanken des Services schematisch dar.

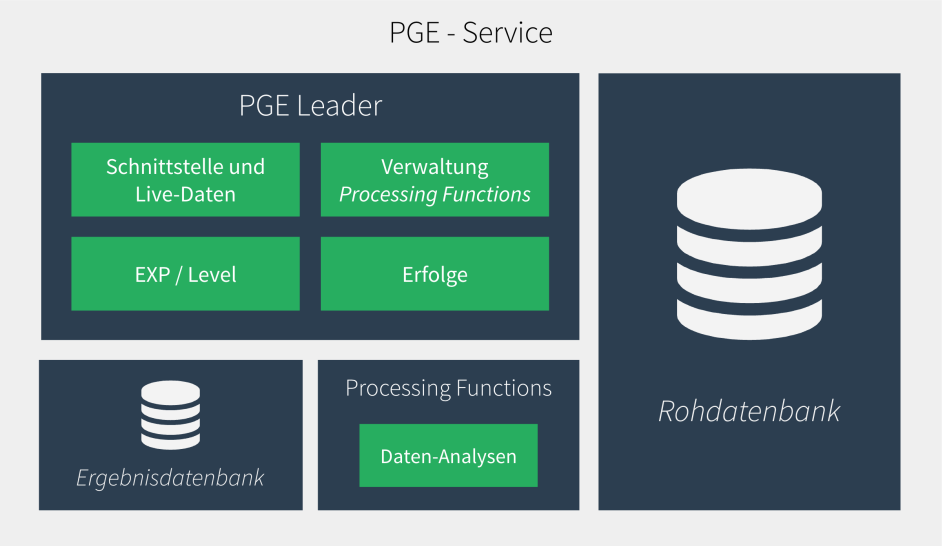


Abbildung 4.8.: Schematische Darstellung des der Komponenten des entworfenen Services

ERFAHRUNGSPUNKTE UND LEVEL

Zu den Ergebnisdaten gehören auch die in Abschnitt 4.1.3 beschriebenen Erfahrungspunkte und Level. Während Erfahrungspunkte zugeordnet zur Nutzer-ID in der Ergebnisdatenbank gespeichert werden, existieren Level nicht als Konzept in der Datenbank. Diese werden durch eine Helper-Klasse anhand der Erfahrungspunktefunktion bestimmt. Der Level-Helper kann zusätzlich Anzeigeinformationen, wie die aktuellen Erfahrungspunkte ($EXP_{relativ}$) und die für das nächste Level benötigten Erfahrungspunkte ($EXP_{benoetigt}$) berechnen.
Die Erfahrungspunkte werden zusammen mit ihrem Auslöser, einer Verbindung zum Nutzer und einem Zeitpunkt in der Datenbank gespeichert. Dies vereinfacht eine Anzeige in einer Aktivitätsanzeige (Anforderung **S13**).

ERGEBNISDATENBANK

In der Egebnisdatenbank werden die Metadaten und Ergebnisse für das Frontend gespeichert. Diese stehen im Gegensatz zu den Rohdaten, die in der Rohdatenbank (Abschnitt 4.2.1), welche die anfallenden Daten innerhalb einer NoSQL-Datenbank sichert. Eine zusätzliche und separate Ergebnisdatenbank wird benötigt, da SQL für strukturierte Daten geeigneter ist, als NoSQL-Datenbanken, die typischerweise Daten redundant Speichern um die kostenaufwändigen Aggregationen auszugleichen (Kleppmann 2017). Die Ergebnisdatenbank wird somit hauptsächlich für Lesezugriffe und Aggregationen verwendet.

4.2.1. ROHDATENBANK

IDEE

Die Rohdatenbank enthält die unverarbeiteten Informationen, die dem *PGE-Leader* verschickt werden. Im Kontext der Crowdsourcing Plattform bestehen die Rohdaten beispielsweise aus den Informationen des Code Editors (geschriebener Quellcode, erfüllte Tests, durchgeführte Refactorings, etc.), aber auch allgemeine Informationen, wie Nutzer-Aktionen (Hochladen eines Profilbildes, Annehmen einer Aufgabe, Teilen von Ereignissen auf Social Media Plattformen, Logins, etc.). Das Ziel der Rohdatenbank ist es, eine strukturierte, durchsuchbare Datenbank mit allen Informationen bereitzustellen, die für die Gamification oder die Entwickleranalyse genutzt werden könnten. Die Rohdatenbank muss daher eine hohe Schreibperformance bereitstellen. Lesezugriffe durch die Analysen sind seltener notwendig.
Mit den Daten der Rohdatenbank arbeiten außerdem die Code-Analyse und die Entwickler-Analyse. Änderungen an der Rohdatenbank können automatisch *Processing Functions* auslösen, welche Analysen anhand der Daten durchführen.

TECHNOLOGIE

Die Rohdatenbank sollte möglichst generisch aufgebaut sein und wenige Vorgaben an ein Schema vorgeben. Das ist insbesondere wichtig, da sich die Beschaffenheit der Rohdaten ändern kann und sehr unterschiedliche Datensätze möglich sein müssen, je nachdem, von welchem Service sie gesendet werden. Weiterhin müssen Schreibzugriffe stets möglich sein und nicht blockieren. Anhand dieser generischen Daten müssen komplexe Analysen durchgeführt werden, weshalb die Rohdaten trotz ihrer Generizität strukturiert und durchsuchbar sein müssen. Die Rohdatenbank sollte darauf ausgelegt sein, möglichst viele Schreibzugriffe zuzulassen. Aggregationen und Lesezugriffe werden seltener gebraucht.
Aufgrund dieser Anforderungen wurde die Nutzung einer NoSQL-Key-Value Technologie beschlossen, auch wenn die alleinigen Datenmengen diese nicht rechtfertigen würden: Dies zeigt die folgende Rechnung mit (optimistisch) geschätzten Werten:

Größe eines JSON-Samples	50 KB
Samples pro Stunde	80 h^{-1}
Crowdsourcing Stunden pro Monat	1000 h
Monate	12
$50KB * 80h^{-1} * 1000h * 12 =$	48.000 MB
Daten pro Jahr	**48 GB**

Die Kompromisse des fehlenden Schemas und der zeitweilig mangelnden Konsistenz können eingegangen werden, um eine höhere Performance bei Schreibprozessen zu erreichen. Weiterhin bieten NoSQL-Key-Value-Datenbanken eine gute Integration in komplexe Analyseprozesse wie *MapReduce* (Dean und Ghemawat 2004) an. Durch diese können zeitaufwändige Analysen parallel durchgeführt werden.

4.2.2. PROCESSING FUNCTION

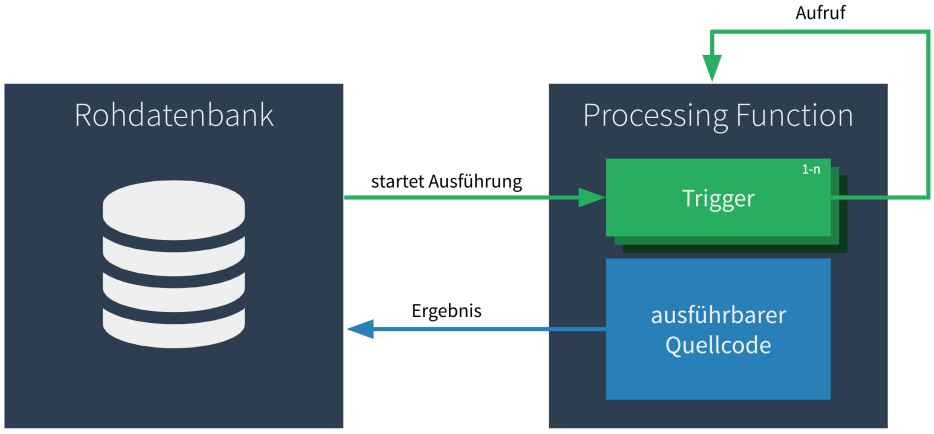


Abbildung 4.9.: Asynchroner Aufruf von *Processing Functions* durch die Rohdatenbank oder eine andere Funktion

Processing Functions berechnen aus den Werten der Rohdatenbank, benötigte Werte (Anzahl abgeschlossene Aufgaben, Anzahl erfüllte Tests, etc.) und Metriken (vgl Abschnitt 4.1.2). Die Funktionen sind mithilfe eines Function-as-a-Service Providers bereitgestellt und werden aufgerufen, wenn sich die zugrundeliegende Datenbasis verändert. Dadurch ist die Skalierung (Anforderung *NF20*, *NF30*) und Abgrenzbarkeit der einzelnen Funktionalitäten gegeben. Weiterhin bringt es den Vorteil einer freien Sprachwahl und damit eine einfache Möglichkeit einer polyglotten Anwendung.
Jede *Processing Function* besitzt einen oder mehrere *Trigger*, die festlegen, wann die Funktion ausgeführt wird. Typischerweise starten die Funktionen, wenn Änderungen an der Rohdatenbank durchgeführt werden. Jede Funktion kann als Ergebnis wiederum in die *Rohdatenbank*, in die *Ergebnisdatenbank* oder in einen *Live-Kanal* schreiben und damit wiederum andere *Processing Functions* auslösen (Abbildung 4.9).
Aufgrund ihrer Architektur sind die *Processing Functions*, bis auf die gemeinsam genutzte Datenbasis, unabhängig voneinander (*loose coupling*). Die geringe Kohärenz stellt insbesondere in Bezug auf die Erweiterbarkeit einen großen Vorteil dar, andererseits erhöhen sie die Komplexität der Architektur.

Der PGE-Leader speichert im **Function Registry** alle Meta-Daten der Processing Functions zusätzlich in der Ergebnisdatenbank. Dort sind Informationen über die, mit der Processing Function verbundenen *Trigger* der Erfolge, sowie ein eindeutiger Identifier zu jeder Funktion gespeichert. Da die Functions einzeln deployt werden, wäre weiterhin eine Verwaltung der aktuell verwendeten Funktionen hilfreich.

4.2.3. LIVE KOMPONENTE

Die Live-Komponente sendet dem Nutzer Mitteilungen. Diese können genutzt werden, um das Interface anhand der Live-Daten zu manipulieren, oder neue Informationen hinzuzufügen. (Anforderungen **C20**, **NF10**) Da jederzeit Benachrichtigungen auftreten können, ist es nötig die Live-Komponente asynchron und damit unabhängig von Nutzereingaben zu entwerfen.

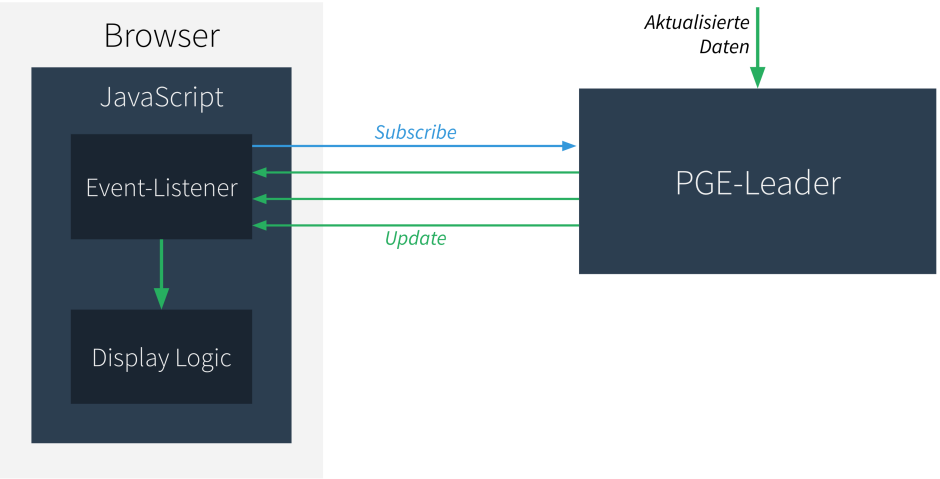


Abbildung 4.10.: Asynchrone Live-Komponente innerhalb der Plattform mittels *Publish/Subscribe*-Pattern

Die Live-Komponente arbeitet nach dem *Publish/Subscribe* Pattern und besteht aus drei Teilen: dem *Event-Listener* im Browser des Anwenders, einem offenen *Kanal*, über den mithilfe eines Real-Time Protokolls (z.B. Websocket-Protokoll) kommuniziert wird und der *Display-Logic* (Abbildung 4.10). Sobald sich ein Nutzer auf der Plattform einloggt, eröffnet der *Event-Lisener* eine real-time Verbindung zum *PGE-Leader* Server (*Subscribe*). In den geöffneten Kanal schreibt der *PGE-Leader* Aktualisierungen in Form von JSON-Objekten. Diese werden vom Event-Listener empfangen und an die entsprechende *Display-Logic* weitergeleitet und entsprechend verarbeitet. Als Beispiel erhält der Socket ein JSON-Objekt mit den Informationen, dass ein Erfolg angezeigt werden soll. Für die Aktion, die den Erfolg ausgelöst hat, erhält der Nutzer weiterhin Erfahrungspunkte. Der *Socket* empfängt somit vom *PGE-Leader* eine Benachrichtigung mit den Meta-Informationen (Titel, Beschreibung, Belohnung, Bild) des Erfolgs. Diese wird von der *Display-Logic* verarbeitet und in Form eines *Alerts* ausgegeben. Eine weitere Nachricht an den *Socket* beinhaltet den neuen Wert der Erfahrungspunkte. Auch diese wird an die *Display-Logic* übermittelt und der Erfahrungsbalken entsprechend aktualisiert und animiert.

Live-Nachrichten, die an einen Nutzer übermittelt werden sollen, der jedoch keinen geöffneten Kanal hat, können nicht angezeigt werden. Diese werden zwischengespeichert und entweder an den Nutzer übermittelt, sobald dieser einen Verbindung herstellt und über einen aktiven Kanal verfügt, oder werden verworfen.

4.2.4. ZUSAMMENFASSENDES BEISPIEL

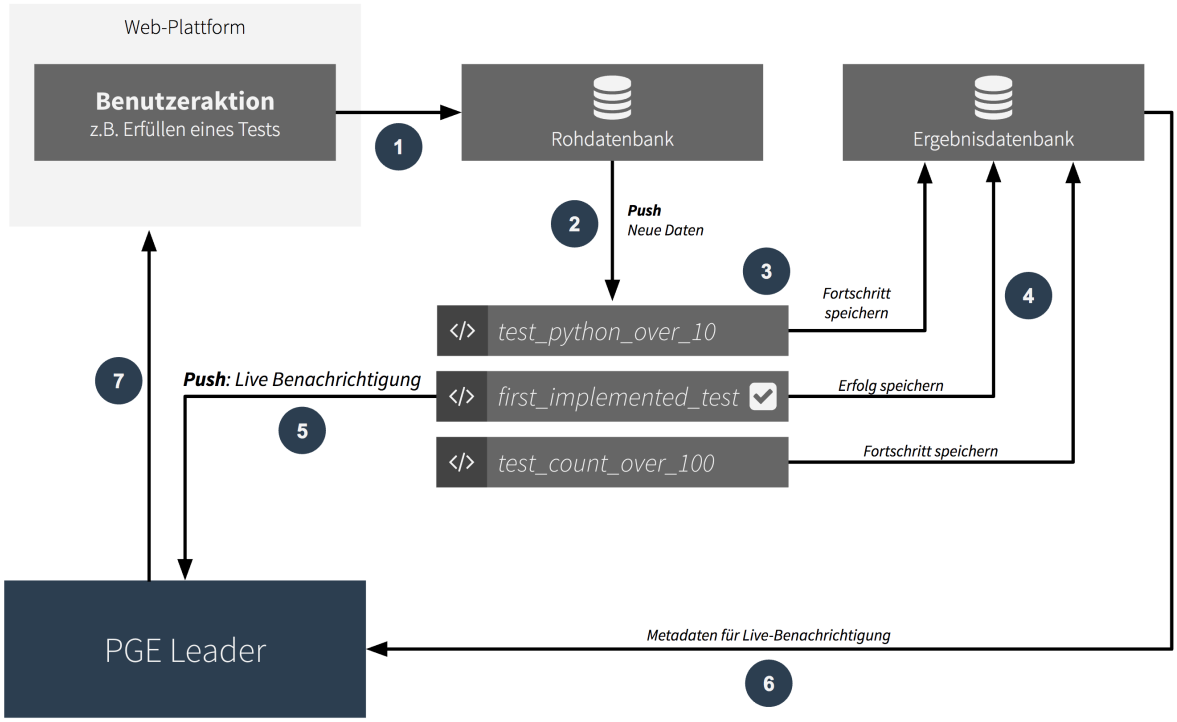


Abbildung 4.11.: Darstellung des Gesamtprozesses innerhalb der Plattform

In diesem Abschnitt wird das Zusammenspiel der oben genannten Komponenten beschrieben. Als Beispiel dient der Erfolg *Hallo Welt* für das erstmalige Abschließen eines Tests innerhalb der Plattform. Die Abbildungen 4.11 und 4.12 stellen den Ablauf des Prozesses dar. Beide verwenden jeweils dieselben Zahlen für die gleichen Prozesse.
Ausgelöst wird der Vorgang durch eine Benutzeraktion. In Beispiel schreibt der Nutzer Quellcode in den Code-Editor und erfüllt damit einen Test. Alle Benutzervorgänge werden durch den Code-Editor in die *Rohdatenbank* geschrieben (**1**). In dieser werden die Daten gespeichert und alle *Processing Functions* werden über die aktualisierten Daten informiert (**2**). Für drei der *Processing Functions* sind die aktualisierten Daten relevant. Diese berechnen den aktuellen Fortschritt des verbundenen Erfolgs. Die Voraussetzungen für den Erfolg der Funktion *first_implemented_test* wird erfüllt (**3**). Jede der drei Funktionen schreibt den aktualisierten Fortschritt in die Ergebnisdatenbank (**4**). Die Funktion des nun erfüllten „Hallo Welt" Erfolgs benachrichtigt weiterhin den *PGE-Leader*, dass dem Nutzer eine Live-Benachrichtigung angezeigt werden soll (**5**). Um die *Processing Functions* möglichst klein zu halten, ist der *PGE-Leader* für das Lesen der Meta-Daten und das Versenden der eigentlichen Benachrichtigungen zuständig. Dieser liest die Daten des Erfolgs aus der Ergebnisdatenbank (**6**) und sendet über den offenen *Live-Kanal* eine Benachrichtigung in Form eines JSON-Objekts an den Benutzer (**7**).

Die *Live-Komponente* empfängt das JSON-Objekt und stellt den Erfolg im Frontend dar. Der gesamte Prozess läuft asynchron ab, sodass beim Benutzer keine Ladezeiten entstehen können. Sollte das Netzwerk zwischen Nutzer und PGE-Leader unterbrochen sein, so erkennt dieser den geschlossenen Kanal und versendet keine Mitteilung. Hier kann im Einzelfall unterschieden werden, ob der Erfolg angezeigt werden soll, sobald der Nutzer wieder verbunden ist, oder ob die Mitteilung ausbleibt.

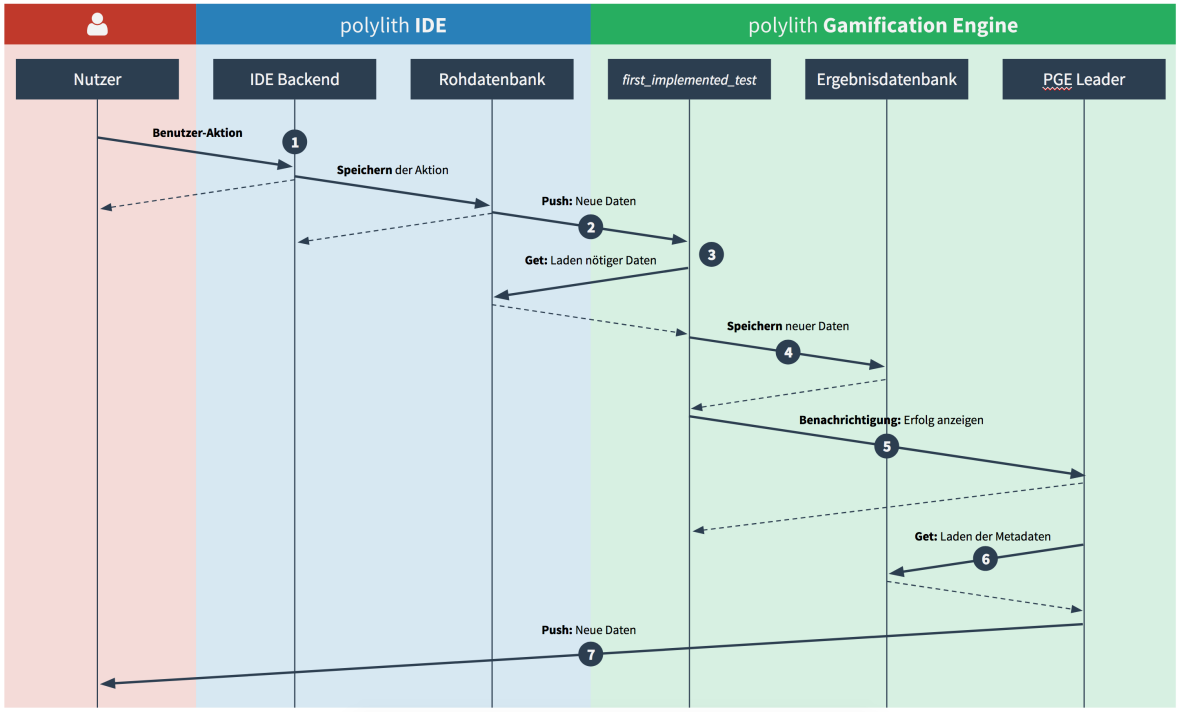


Abbildung 4.12.: Darstellung des Gesamtprozesses in Form eines Ablaufdiagramms. (Um die Übersichtlichkeit zu erhöhen wird nur die *Processing Function first_implemented_test* dargestellt.)

5. IMPLEMENTIERUNG

Dieses Kapitel beschreibt die Architektur des entstandenen Gamification-Services und dessen Implementierungskonzepte. Dabei wird einerseits auf den PGE-Leader Server eingegangen und die Komponenten beschrieben aus denen dieser aufgebaut ist. Weiterhin wird der grundlegende Aufbau der Processing Functions erläutert. Abschließend wird auf die Infrastruktur und das Deployment der Anwendung eingegangen.

5.1. VORGEHEN UND TECHNOLOGIEAUSWAHL

Um die Realisierbarkeit der verwendeten Technologien und Konzepte frühzeitig zu testen, wurde bei der Implementierung auf *Spikes* gesetzt. Diese sind kleine, als Experimente entwickelte Teillösungen des Gesamtproblems, mit deren Hilfe die konzipierte Lösung schnell und praxisnah getestet werden kann (Shore 2007). Dies hat sich insbesondere bei der Auswahl der Cloud-Technologien und dem Testen der Infrastruktur bewährt, da so schnell Probleme und Hindernisse erkannt werden konnten.
Bei der Implementierung des Services wurde nach den Grundsätzen der Test-getriebenen Programmierung (TDD) vorgegangen. Dies bedeutet, dass vor der Implementierung der eigentlichen Logik ein Softwaretest erstellt wird, der die Logik testet. Dies resultiert einerseits in einem strukturierten Vorgehen, andererseits in hinreichend getestetem Quellcode (vgl. Percival (2014)).
Für die Implementierung wurde die Programmiersprache Python[1] verwendet. Aufgrund von Inkompatibilität des Function as a Service Dienstes *AWS Lambda* wurde für die *Processing Functions* anfangs Python 2.7 genutzt. Seit April 2017 unterstützt AWS Lambda auch Python 3.6. [2] Seitdem wurden alle Funktionen auf Python Version 3.6 migriert, um eine zukunftssichere Lösung zu gewährleisten.[3]

[1] `http://python.org` (Letzter Aufruf: 11.06.2017)
[2] `https://aws.amazon.com/de/about-aws/whats-new/2017/04/aws-lambda-supports-python-3-6/` (Letzter Aufruf: 17.06.2017)
[3] Der Support für Python 2.7 läuft im Jahr 2020 aus. `https://www.python.org/dev/peps/pep-0373/` (Letzter Aufruf: 07.08.2017)

5.2. PGE-LEADER

Der Leader-Server wurde in Python 3.6 umgesetzt und verwendet das Webframework Django[4]. Ausgewählt wurde Django u.a. aufgrund des ausgereiften ORM (Object-relational mapping), sowie der schnellen Umsetzung einfacher Administrations-Oberflächen durch den automatisch generierten und einfach anzupassenden Django-Admin. Über dieses Admin-Interface können alle Einträge der Datenbank des PGE-Leaders händisch bearbeitet und Meta-Daten zu Erfolgen eingetragen werden.
Zur besseren Übersicht wurden die einzelnen Funktionalitäten des Leader-Servers in einzelne

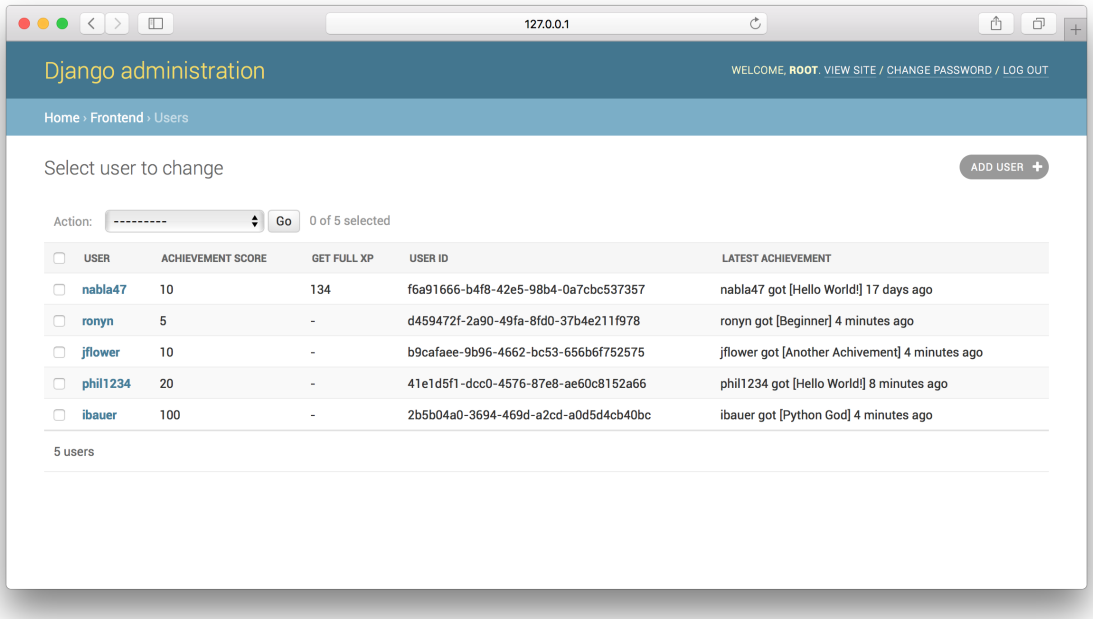


Abbildung 5.1.: Admin-Backend des PGE Leaders

Django-Pakete („*Apps*“) ausgelagert. Dies verbessert nicht nur die Wartbarkeit des Quellcodes aufgrund besserer Strukturierung, es trägt gleichzeitig auch zu einer besseren logischen Trennung zwischen den einzelnen Komponenten bei (*Separation of Concerns*). Weiterhin wird dadurch eine Wiederverwendung der einzelnen Unterkomponenten ermöglicht.

5.2.1. IMPLEMENTIERUNG DES PGE-LEADERS

Die einzelnen Funktionalitäten wurden gemäß des Paradigmas der objektorientierten Programmierung umgesetzt. Alle nicht-statischen Klassen werden durch das Django-Framework in einer *PostgreSQL*[5] Datenbank persistiert. Die entstandene Klassenstrukur der Komponenten werden in den jeweiligen Unterkapiteln dargestellt und erläutert.

[4] `https://www.djangoproject.com/` (Letzter Aufruf: 11.06.2017)
[5] `https://www.postgresql.org/` (Letzter Aufruf: 07.08.2017)

ACHIEVEMENTS APP

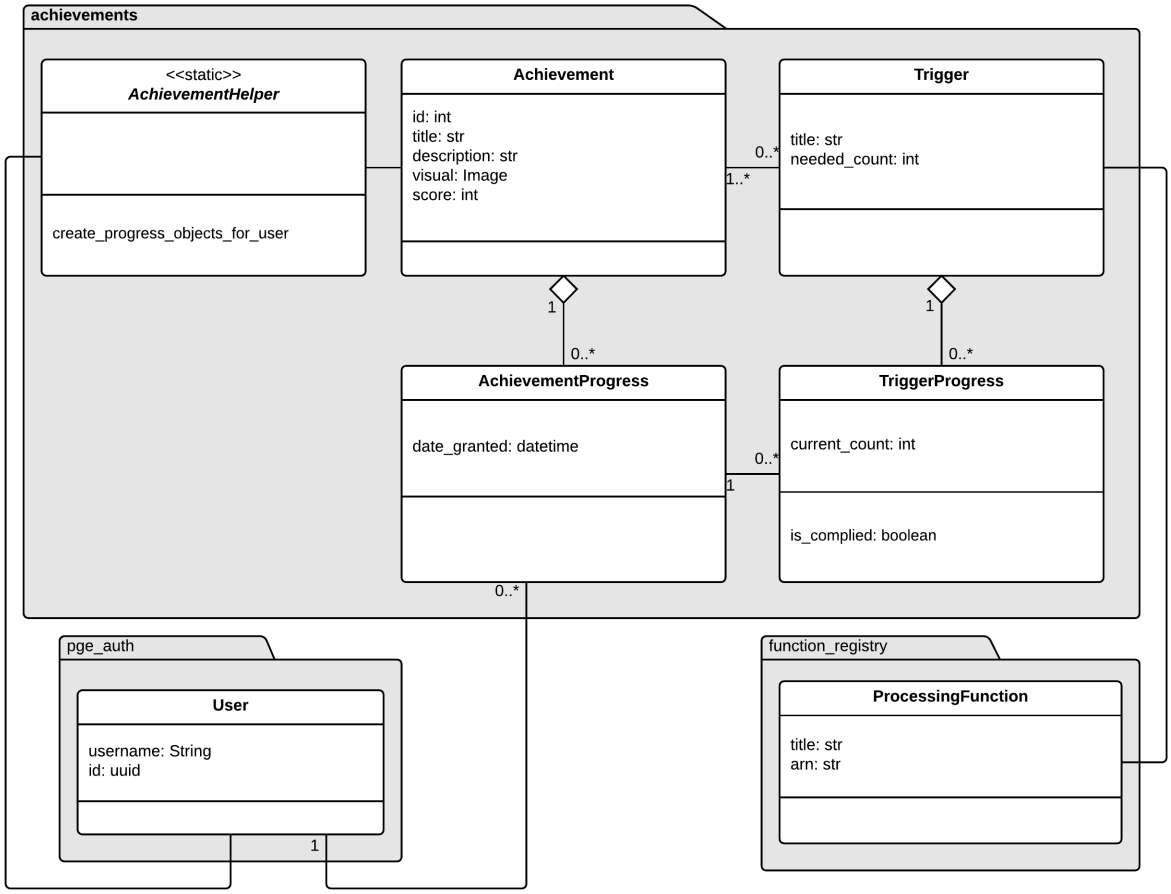


Abbildung 5.2.: Klassenstruktur der Achievements-App

Die Klassenstruktur der Achievements-App ist um die `Achievement`-Klasse aufgebaut. Diese speichert Meta-Daten (Bild, Beschreibung, Score) zu Erfolgen und verfügt über Relationen zur `Trigger`-Klasse, in der die Bedingungen für den Erfolg gespeichert werden. Weiterhin verfügt sie über eine Relation zur `AchievementProgress`-Klasse, in der zu der Fortschritt eines Erfolgs für einen Nutzer gespeichert ist. Welche Bedingungen erfüllt sein müssen, damit ein Erfolg erreicht wird, beschreibt eine Many-To-Many Relation zwischen `Achievement` und `Trigger`-Klasse. Jedes `Trigger`-Objekt wird durch einen Text und einen zu erreichenden Wert beschrieben. Um abzubilden, dass `Trigger` über denselben Datensatz verfügen können, verfügt jeder `Trigger` über eine Relation zur Klasse `ProcessingFunction` innerhalb der `function_registry` App (Beispiel: *Eine Aufgabe abgeschlossen*, *10 Aufgaben abgeschlossen*, *100 Augaben abgeschlossen*, etc.). Diese Relation vereinfacht das Speichern neuer Daten, da zu jeder `ProcessingFunction` die verbundenen `Trigger` bekannt sind. Die `Trigger` können jeweils mehreren Erfolgen zugeordnet sein, und damit wiederverwendet werden.

Um den Fortschritt eines Nutzers zu verfolgen und zu speichern, wird zusätzlich zu jedem Erfolg und für jeden Nutzer eine Instanz der Klasse `AchievementProgress` und zu jedem zum Erfolg gehörigen `Trigger` ein `TriggerProgress` Objekt angelegt. Dadurch können die Fortschrittswerte in der Datenbank gespeichert werden. Ein Erfolg wird als erfüllt markiert, sobald alle zugehörigen `TriggerProgress` Objekte durch die Methode `is_complied()` bestätigen, dass der jeweilige `Trigger` als erfüllt gilt. Errungene Erfolge werden dadurch unterschieden, dass das `date_granted`-Feld gesetzt ist.

AUTH APP

In der auth-App werden die Identifier (in Form von UUIDs) der Nutzer gespeichert und verwaltet. Obwohl das Django-Framework eine integrierte App für die Benutzerverwaltung und Authentifizierung anbietet, wurde davon abgesehen, diese zu erweitern und stattdessen eine eigene App für diese Zwecke implementiert. Dadurch bleiben die Nutzer des PGE-Leaders und die der polylith-Plattform dauerhaft getrennt und haben keine Relationen untereinander. Benutzerspezifischen Informationen können durch Helper-Funktionen des *User*-Modells in der auth-App gesammelt werden. Da die Mehrheit der abgerufenen Informationen nutzerspezifisch sind, ist die App somit neben der Live-App die wichtigste Komponente zur Kommunikation nach außen.

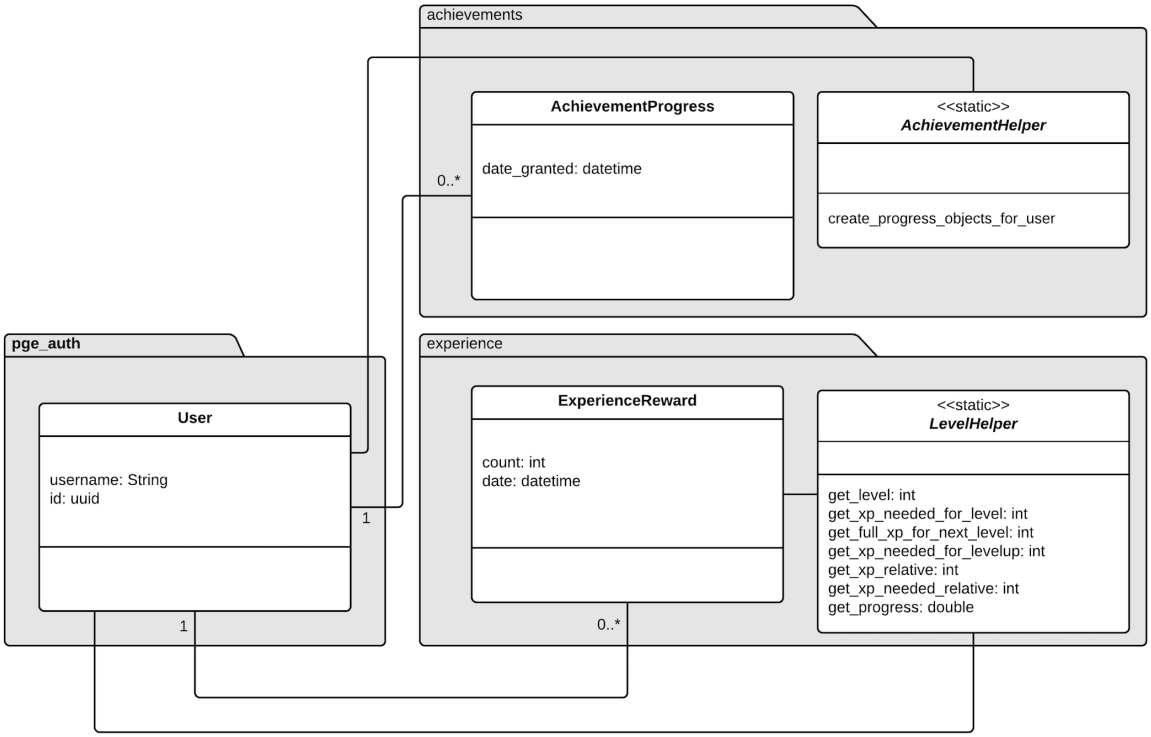


Abbildung 5.3.: Klassenstruktur der Auth-App

Die auth-App stellt JSON-Endpunkte bereit, die eine Liste von Nutzern zurückgeben. Dabei kann die Auswahl und Sortierung der angeforderten Liste in Form von GET-Parametern spezifiziert werden. Diese Funktion wird genutzt, um **Ranglisten** bereitzustellen.

Weiterhin wird ein JSON-Endpunkt bereitgestellt, der sämtliche Meta-Informationen zu einem Benutzer zurückgibt. Dadurch kann frontend-seitig ein **Nutzerprofil** realisiert werden.
Die Darstellung der Ranglisten und des Profils werden nicht durch den PGE-Leader vorgegeben. Die Darstellung dieser Elemente wird durch die Plattform vorgenommen, die an den PGE-Service angebunden wird.

EXPERIENCE APP

Die Experience App verwaltet Daten, die im Zusammenhang mit Erfahrungspunkten stehen. Die zugewiesenen Erfahrungspunktwerte werden durch `ExperienceReward` Objekte in der Datenbank gespeichert. Dadurch bleibt der Erfahrungspunkte-Verlauf eines Nutzers dauerhaft nachvollziehbar (vgl. *Event-Sourcing Pattern* nach Fowler 2005). Die Level werden von der statischen `LevelHelper`-Klasse berechnet. Diese bietet eine Vielzahl an Helper-Funktionen, welche für UI und interne Berechnungen mit Erfahrungspunkten hilfreich sind.
Nachteilig bei der gewählten Implementierungsart ist der höhere Rechenaufwand bei der Anzeige der Erfahrungspunkte und der Darstellung des Levels. Trotzdem wurde diese Variante gewählt, um stets den vollständigen Verlauf zu speichern. So kann der Service später einfach um eine Aktivitätsanzeige erweitert werden. Da die Objekte in einer relationalen PostgreSQL-Datenbank hinterlegt werden, können die Erfahrungspunkte mithilfe von SQL-Abfragen performant bestimmt werden. Sollte die Berechnungszeit zu stark wachsen (beispielsweise, weil zu viele `ExperienceReward`-Objekte gezählt werden müssen), kann an dieser Stelle eine Caching-Lösung eingeführt werden und somit die Berechnungszeit verringert werden.

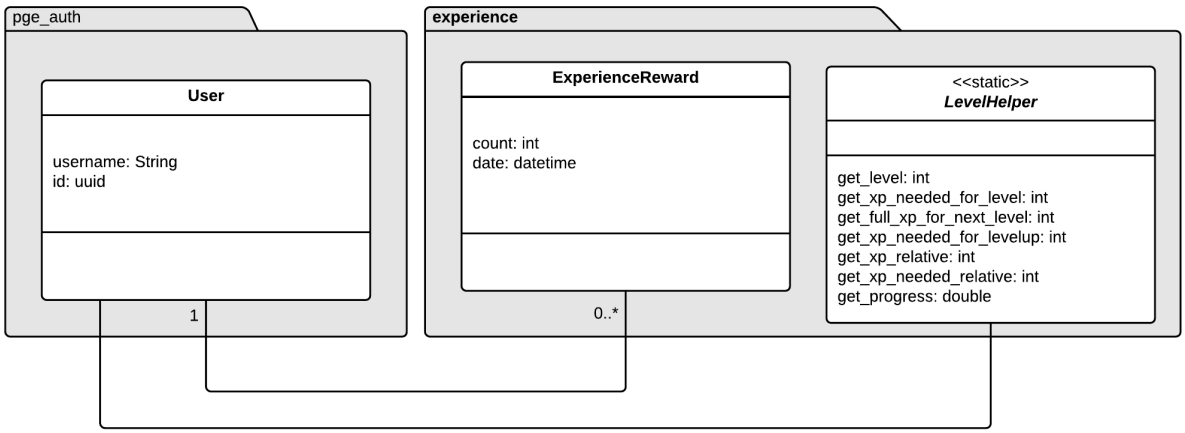


Abbildung 5.4.: Klassenstruktur der Experience-App

FUNCTION REGISTRY APP

In der Function Registry werden alle Processing Functions gespeichert. Die Verwaltung dieser Funktionen auf dem Leader Server ist notwendig, da `Trigger`, welche dieselben Processing Functions verwenden, auch dieselbe Datengrundlage besitzen und aus diesem Grund auch parallel aktualisiert werden müssen. Weiterhin ist es sinnvoll, eine Übersicht über alle vorhandenen Datenquellen für die Gamification anzubieten.
Jede Processing Function erhält vom *PGE-Leader* eine eindeutige Identifizierung in Form einer UUID. Für jede Funktion wird ein API-Endpunkt im Format

```
http://127.0.0.1/achievements/ba999d94-33c2-4cd4-88d7-f928a40a7c9c/update/
```

bereitgestellt, an den Daten durch einen HTTP-POST Request gesendet werden können. Dadurch wird sichergestellt, dass nur registrierte Processing Functions Daten an den *PGE-Leader* senden dürfen. Eine weitere Authentifizierung wurde für den Prototypen vorerst nicht implementiert.

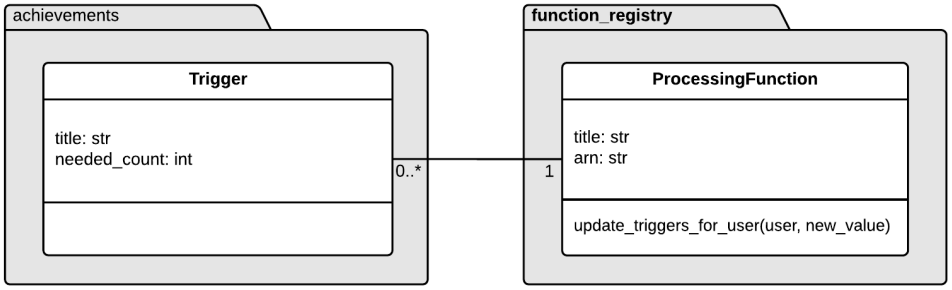


Abbildung 5.5.: Klassenstruktur der Function Registry-App

5.3. ROHDATENBANK

Für die Datenhaltung der Rohdaten wurde die Key-Value Datenbank *DynamoDB* gewählt. Die Datenmengen allein, die über die Plattform anfallen können, rechtfertigen keine verteilte NoSQL Datenbank. Das Konzept von DynamoDB wurde in einem White-Paper von DeCandia u. a. (2007) veröffentlicht und vorgestellt.

Insbesondere der *always-writable* Ansatz von DynamoDB (DeCandia u. a. 2007) und die NoSQL-typische freie Datenstruktur waren für die Wahl der Datenbank ausschlaggebend. Weiterhin bietet Dynamo eine umfassende Integration in weitere Clouddienste wie *AWS Lambda*, *AWS Elastic MapReduce* oder *Kinesis* an. Durch diese wird eine einfache und gehostete Erweiterung durch komplexe und infrastrukturlastige Services wie *MapReduce* möglich (Niranjanamurthy u. a. 2014). Für die Auswahl von DynamoDB war hauptsächlich diese Integration (insbesondere in AWS Lambda) entscheidend. Da keine Anforderungen an ein festes Schema vorhanden war, konnte DynamoDB für den Prototypen verwendet werden.

DynamoDB wurde von Amazon ursprünglich für die Realisierung des Warenkorbs innerhalb der amazon.com Plattform entwickelt. Hierbei war wichtig, dass einerseits jederzeit Daten geschrieben werden können und nie verloren gehen können. Andererseits war die Skalierung bei vielen gleichzeitigen Anfragen wichtig. Diese Anforderungen wurden zu Kosten der Konsistenz umgesetzt. DynamoDB gewährleistet, dass jeder Schreibprozess zwar einzeln atomar ist, jedoch nicht über mehrere Operationen. Dies stellt besondere Anforderungen, an Funktionen, die auf den DynamoDB-Daten arbeiten. Jede dieser Operationen sollte *idempotent* sein.

5.3.1. TRIGGER

Die Verwendung von DynamoDB ist nicht bindend für die Architektur des PGE-Services. Aus den vorher genannten Gründen wurde jedoch entschieden DynamoDB zu verwenden. Wichtig ist jedoch für die ereignisgesteuerte Architektur, dass Veränderungen der Rohdatenbank Processing Functions auslösen können. Für den, in dieser Arbeit entwickelten Prototypen wurde hierfür die von AWS angebotene Trigger-Funktionalität zwischen DynamoDB und Lambda verwendet. Bei der Verwendung einer anderen Rohdatenbank muss diese Funktionalität selbst implementiert werden.

5.4. PROCESSING FUNCTIONS

Die Processing Functions sind abgegrenzte Funktionen, welche die Daten der Rohdatenbank verarbeiten und die in Abschnitt 4.1.2 beschriebenen Metriken implementieren.

5.4.1. FUNKTIONSWEISE

AWS Lambda bietet verschiedene Möglichkeiten, Lambda-Funktionen auszulösen. Für den implementierten Service wurde ausschließlich die Aktivierung durch den DynamoDB Stream genutzt. Dieser führt die Lambda-Funktionen immer dann aus, wenn eine Aktualisierung des Datensatzes einer Tabelle der DynamoDB durchgeführt wird. Die veränderten Datensätze werden als Parameter in die Funktion gegeben, die von Lambda ausgeführt wird.

5.4.2. IMPLEMENTIERUNG

Die Processing Functions bestehen aus vergleichsweise wenigen Quellcode-Zeilen (ca. 50-300) und verfügen über wenige Abhängigkeiten zu Drittanbieter-Bibliotheken. Dadurch ist die Komplexität der Funktionen – abhängig vom implementierten Algorithmus – gering. Aufgrund der Komplexität der DynamoDB Datensätze wurde durch die `DynamoDOA` Klasse eine weitere Abstraktionsschicht für die Processing Functions eingeführt.
Bereits implementierte Processing Functions verwenden die Programmiersprache Python 3.6. Da die Funktionen vollständig unabhängig voneinander sind, werden an zukünftig entwickelte Analysefunktionen keine Anforderungen bezüglich der Programmiersprache gestellt.[6]
Jede Processing-Function verfügt über einen eindeutigen Identifier in Form einer (pseudo-)zufällig generierten UUID4[7]. Dieser wird verwendet, um die Funktionen seitens des PGE-Leaders zuordnen zu können. Der Identifier der Funktion muss dem PGE-Leader bekannt sein – andernfalls darf die Funktion keine Daten an diesen senden.

5.4.3. DEPLOYMENT

Im Konzept-Kapitel 1.3 wurde beschrieben, dass die Processing-Functions möglichst über den PGE-Leader deployt werden sollten. Dies wurde zum Zeitpunkt der Fertigstellung dieser Arbeit noch nicht umgesetzt.
Die einzelnen Funktionen werden (manuell) in Form eines ZIP-Archivs auf die AWS-Cloud Server hochgeladen. Die auszuführende Funktion muss hierbei direkt im ZIP-Archiv, ohne weitere Unterordner liegen. Drittanbieter Bibliotheken müssen sich ebenfalls direkt innerhalb des Verzeichnisses befinden.

5.4.4. KOMMUNIKATION

Die Kommunikation der Processing-Functions und dem PGE-Leader erfolgt ausschließlich über REST-Aufrufe. Daten werden über das JSON-Format versendet. Zwischen den einzelnen Processing Functions findet keine Kommunikation statt. Zu jeder *Processing Function* wird ein REST-Endpunkt bereitgestellt, an den die Funktion Daten übermitteln kann (siehe Kapitel 5.2.1 Abschnitt *Function Registry App*).

5.4.5. SICHERHEIT

Zum Zeitpunkt der Fertigstellung dieser Arbeit, sind die Processing Functions ausschließlich durch ihren eindeutigen Identifier abgesichert. Dies ist für die prototypische Umsetzung akzeptabel, für einen produktiven Einsatz jedoch unzureichend. Zusätzlich wurde für die Implementierung der JSON-API seitens des PGE-Leaders der CSRF-Schutz[8] ausgeschaltet. Dadurch ist die JSON-Schnittstelle nicht ausreichend vor Angriffen geschützt. Diese Absicherungen haben sehr hohe Priorität bei der Weiterentwicklung des Services.

[6] AWS Lambda unterstützt derzeit (Stand Juni 2017) Python 2.7, Python 3.6, JavaScript(Node.js), C# und Java
[7] `https://tools.ietf.org/html/rfc4122` (Letzter Aufruf 12.08.2017)
[8] Cross-Site-Request-Forgery: `https://docs.djangoproject.com/en/1.11/ref/csrf/` (Letzter Abruf: 13.08.2017)

5.4.6. BEISPIEL

Listing 1 beschreibt eine komplett funktionstüchtige Lambda-Funktion, welche innerhalb des PGE-Service genutzt werden kann. Die Funktion wurde leicht vereinfacht um das Beispiel klein zu halten. So wurden unter anderem Importe und Logging-Ausgaben gelöscht, die jedoch an der Funktionalität des Skripts nichts verändern. Die `FUNCTION_ID` ist der Identifier der Funktion, ohne die sie nicht beim PGE-Leader registriert werden kann. Diese UUID wird benötigt um die Funktion im PGE-Leader zu registrieren. Wird der Datensatz der Rohdatenbank verändert, so wird die Funktion automatisch aufgerufen. Die veränderten Daten werden in einer Map-Struktur innerhalb des Parameters `event` an die Funktion übergeben. Die Funktion iteriert nun über alle neuen Records in der Rohdatenbank und führt eine *Scan-Query* durch die Klasse `DynamoDOA` durch und zählt dadurch alle Einträge, die mit dem Schlüssel `completed_tasks_<Nutzername>_` beginnen. Das ermittelte Ergebnis wird an den PGE-Leader Server mittels POST-Request versendet.

```
1  FUNCTION_ID = "13fb30c2-efc9-41ee-a577-4eb95c203d6d"
2
3  def lambda_handler(event, context):
4      records = event.get("Records")
5      for record in records:
6          record = record["dynamodb"]
7          username = record.get("NewImage").get("name").get("S", None)
8          if username is None:
9              continue
10         scan_query = "completed_tasks_{}_".format(username)
11         table_scan = DynamoDOA.scan(
12             FilterExpression=Attr('pk').begins_with(scan_query)
13         )
14         count = table_scan.get("Count")
15         response = requests.post(
16             "http://pge.polylith.de/achievements/{}/update/".format(FUNCTION_ID),
17             data={
18                 "user": username,
19                 "value": count,
20             }
21         )
```

Listing 1: Beispiel einer Lambda-Funktion, die eine Scan-Abfrage innerhalb der Rohdatenbank durchführt und den PGE-Leader Server über neue Daten informiert

Auffällig ist hierbei, dass die Funktion bei jedem Aufruf dem PGE-Leader einen Wert über die gelösten Aufgaben mitteilt – also auch, wenn dieser bereits den korrekten Wert gespeichert hat. Dies begründet sich darin, dass die Processing Function (bewusst) keine Informationen über den aktuellen Stand des PGE-Leaders hat. Weiterhin ist der Aufruf an den PGE-Leader idempotent bezüglich der Hintereinanderausführung gestaltet, um zu gewährleisten, dass keine inkonsistenten Daten existieren, falls die Lambda Funktion doppelt ausgeführt wird.

5.5. LIVE-KOMPONENTE UND FRONTEND

Die Live-Komponente dient zum Anzeigen von Live-Daten im Browser des Benutzers, ohne dass eine Interaktion von ihm notwendig ist. Der *PGE-Leader* muss in der Lage sein, Daten asynchron zu versenden. Um diese Funktionalität umzusetzen, wurde entschieden, das Websocket-Protokoll einzusetzen. Da Django keine Websockets unterstützt[9], wurde aufgrund der einfachen Anbindung und der automatischen Fallback-Lösungen das Socket.io Framework verwendet. Zu versendende Nachrichten werden in die Cache-Datenbank *Redis*[10] geschrieben und aus dieser von einem Message Broker, umgesetzt mit Node.js, mit dem Websocket Protokoll an den betreffenden Benutzer geschickt. Dieser Prozess wird durch Abbildung 5.6 dargestellt.

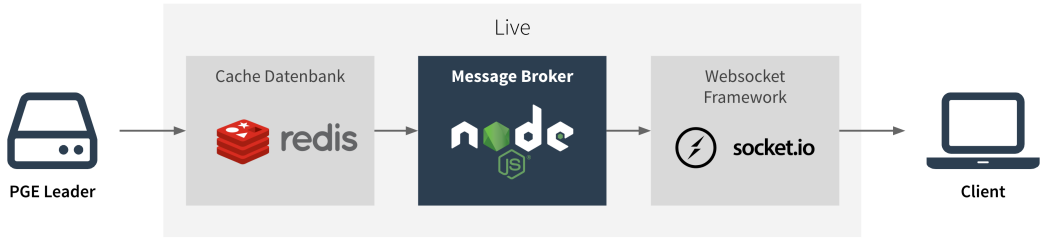


Abbildung 5.6.: Verwendete Technologien der Live-Komponente

5.5.1. SICHERHEIT

Die Sicherheit der Cache-Datenbank *redis* ist sichergestellt, da ausschließlich Anfragen vom selben Server akzeptiert werden. Dies bedeutet, dass PGE-Leader, Message-Broker und *redis* stets auf demselben Server deployt werden müssen. Falls dies – beispielsweise aufgrund zu hoher Auslastung – nicht mehr möglich ist, so muss *redis* zusätzlich abgesichert werden. Bis dies notwendig wird, ist die Absicherung der *redis*-Datenbank ausreichend.

Die Websocket-Verbindung ist derzeit nur durch die Kennung (UUID) des Nutzers geschützt. Nutzer können somit eine fremde UUID verwenden und erhalten damit Zugriff auf den Live-Kanal eines anderen Nutzers. Die Informationen sind zwar nicht kritisch, dennoch ist eine Authentifizierung und damit eine weitere Absicherung des Kanals notwendig. Dies hat vor Inbetriebnahme des PGE-Leaders im Produktivbetrieb hohe Priorität.

[9] Die Bibliothek *django-channels* `https://channels.readthedocs.io/en/stable/` (Letzter Aufruf 11.08.2017) bietet die Möglichkeit Websockets in Django zu verwenden. Da jedoch ausschließlich Daten gesendet, und nicht empfangen werden müssen, wurde für den Prototyp aufgrund der Komplexität und dem Einarbeitungsaufwand entschieden, *django-channels* nicht zu verwenden. Für eine weitere Entwicklung ist die Verwendung der Bibliothek jedoch sinnvoll.

[10] `https://redis.io/` (Letzter Abruf 21.07.2017)

5.5.2. FRONTEND

Für das Frontend wird die Bibliothek *Vue.js*[11] verwendet. Diese wurde vor allem wegen ihrer leichtgewichtigen Verwendung ausgewählt. *Vue.js* ermöglicht es *Apps* auf HTML-Elemente zu registrieren und diesen mit einer definierten Template-Sprache Platzhalter und Logik zuzuweisen. Diese *Apps* können ohne großen Aufwand aktualisiert und manipuliert werden. Dabei ist *Vue.js* einfach und klein gehalten und kombiniert sich gut mit anderen Frameworks.

BEISPIEL

Auf Client-Seite wird die Live-Komponente in JavaScript durch *Socket.io* und *Vue.js* angebunden. Listing 2 stellt diese Anbindung dar. Das Erfolgs-Element wird als SVG-Grafik vom PGE-Leader Server heruntergeladen, an das HTML-Dokument (Zeile 1) angehängt und durch Vue.js mit Daten befüllt (Zeilen 2 - 13).

```
1  $(".achievement").append(document.importNode(data.documentElement,true));
2  let achievement_app = new Vue({
3      el: '#achievement',
4      data: {
5          // initializing with dummy data
6          title: 'Title',
7          description: 'This is the description!',
8          trigger_1_text: null,
9          trigger_2_text: null,
10         trigger_3_text: null,
11         score: 10
12     }
13 });
```

Listing 2: Initialisierung der Live-Komponente mit Vue.js

War der Prozess erfolgreich, so verbindet sich der Client über *Socket.io* mit der Live-Komponente und registriert einen entsprechenden Handler (Listing 3). Sobald über den offenen Kanal Daten versandt werden, wird die Handler-Funktion aufgerufen (Zeile 3). In dieser Funktion werden die Daten als JSON ausgelesen und mithilfe von *Vue.js* in die geladene SVG-Grafik eingebunden. Die Funktion `show_achievement()` (Zeile 10) zeigt den Erfolg nun an, indem das vorher unsichtbare SVG sichtbar gemacht wird und mit der Bibliothek *animate.css*[12] animiert eingeblendet. Die Funktion fügt weiterhin Handler-Funktionen zum Erfolgs-Element hinzu, die diesen bei einem Klick animiert ausblenden.

[11] `https://vuejs.org/` (Letzter Aufruf: 18.08.2017)
[12] `https://daneden.github.io/animate.css/` (Letzter Aufruf: 18.08.2017)

```
// Handler for live data
let socket = io('http://pge.polylith.de:8080');
   socket.on('achievements', function (data) {
       let data = JSON.parse(data);
       Vue.set(achievement_app, "title", data.title);
       Vue.set(achievement_app, "description", data.description);
       update_triggers(data, achievement_app);

       // display achievement and add handlers
       show_achievement();
});
```

Listing 3: Anbindung der Live-Daten mit Socket.io und Vue.js

5.6. INFRASTRUKTUR UND DEPLOYMENT

Der implementierte Service verwendet die vollständig verwalteten Dienste *Lambda* und *DynamoDB* der AWS-Cloud. Diese werden automatisch gemäß ihrer Konfiguration bereitgestellt und skaliert.

Um den *PGE-Leader* Service bereitzustellen, wird eine Virtuelle Maschine oder ein Server verwendet, auf dem die Open-Source Software *Docker*[13] installiert ist. *Docker* ermöglicht es mithilfe von Betriebssystemvirtualisierungs-Mechaniken voneinander isolierte *Container* auszuführen. Die verwendeten Softwareartefakte (Docker-Container) werden von einem Continuous-Integration Tool[14] gebaut und innerhalb eines Docker-Repositories bereitgestellt. Dadurch müssen zur Installation des Services lediglich die fertig gebauten Docker-Container heruntergeladen und gestartet werden.

Der *PGE-Leader* besteht aus insgesamt vier Docker-Containern: *Django* (zur Bereitstellung der Software an sich), *node.js* (Message-Broker für die Live-Kommunikation), PostgreSQL (Als Datenbank für Django) und redis (als Cache Datenbank und für die Live-Kommunikation). Eine Übersicht der Container und Services wird in Abbildung 5.7 dargestellt.
Zur Installation des *PGE-Leaders* wurde eine Konfiguration für das Orchestrierungs-Tool Ansible[15] erstellt. Ansible führt mithilfe der Konfigurationsdatei alle zur Installation nötigen Schritte auf dem Zielsystem aus. Zu diesen gehören die Installation von Docker, das Herunterladen der vom CI-Tool gebauten Docker-Images (für PGE-Leader und Message Broker), der Download der verwendeten Drittanbieter-Images (PostgreSQL, Redis) und das Starten der Container.

[13] `https://www.docker.com/` (Letzter Aufruf 11.08.2017)
[14] Für den Prototyp der Arbeit wurde Teamcity (`https://www.jetbrains.com/teamcity/` (Letzter Aufruf 11.08.2017)) verwendet. Es besteht jedoch keine Abhängigkeit zu dem CI-Tool.
[15] `https://www.ansible.com/` (Letzter Aufruf 11.08.2017)

Abbildung 5.7.: Verwendete Technologien und Services zur Bereitstellung des PGE-Services

Die Verwendung des Orchestrierungs-Tools Ansible und der Container-Software Docker hat sich als äußerst hilfreich herausgestellt, da dadurch der Installationsaufwand sehr gering gehalten wird. Weiterhin wird durch das CI-Tool sichergestellt, dass ausschließlich Container gebaut werden, deren Quellcode valide ist und alle Tests erfüllt (*Quality Gates* vgl. Humble und Farley (2010)). Ein Deployment eines unfertigen Standes oder einer invaliden Version (z.B. entstanden durch das Kopieren von Quellcode-Dateien) ist somit nicht möglich. Die häufig auftretenden Probleme bei der Installation der Software wurden damit minimiert.

6. EVALUATION

In Abschnitt 2.5 wurden Anforderungen an einen Gamification-Service aufgestellt. Diese werden im folgenden evaluiert, indem sie mit dem aktuellen Stand verglichen werden. Diese technische Evaluation wird ergänzt durch eine kritische Betrachtung nicht-funktionaler Anforderungen wie der Performance des Systems und der Praxistauglichkeit des Services.
Weiterhin wird im zweiten Teil dieses Kapitels die durchgeführte Nutzerevaluation ausführlich beschrieben und ausgewertet. Diese wurde durchgeführt um das Gamification-Konzept unabhängig vom implementierten Gamification-Service zu überprüfen. Als Teil der Evaluation füllten die Teilnehmer der Nutzerevaluation einen Fragebogen zum Gamification-Konzept aus. In diesem sollten einerseits die einzelnen Gamification Elemente bewertet werden, die im Gamification-Konzept verwendet werden. Andererseits schätzten die Probanden das Gamification-Konzept nach den Kriterien von Octalysis (vgl. Abschnitt 2.2.1) einschätzen.

6.1. TECHNISCHE EVALUATION

In diesem Abschnitt werden technische Evaluationen vorgestellt. Zu diesen gehören Laufzeitmessungen und Round-trip Zeiten. Aufgrund des verteilten Systems wird weiterhin die Ausfallsicherheit der einzelnen Komponenten untersucht und bewertet.
Anhand der in der technischen Evaluation gefundenen Mängel werden Optimierungsvorschläge und Möglichkeiten zur Effizienzsteigerung dargelegt.

6.1.1. EVALUATION DER AUFGESTELLTEN ANFORDERUNGEN

In diesem Abschnitt werden die in Kapitel 2.5 aufgestellten Anforderungskriterien evaluiert. Dabei werden die aufgestellten Anforderungen mit dem zum Zeitpunkt der Fertigstellung dieser Arbeit implementierten Stand abgeglichen. Zur Übersicht werden die Tabellen aus Kapitel 2.5 verwendet und um eine Spalte erweitert, die den aktuellen Stand visualisieren. Dabei wird zwischen abgeschlossenen Kriterien (schwarzes Häkchen), erfüllten, jedoch nicht evaluierten Kriterien (graues Häkchen) und offenen Anforderungen (leerer Kreis) unterschieden.

MUST-HAVE KRITERIEN

Must-Have Kriterien wurden am höchsten priorisiert und sind deshalb vorrangig umgesetzt worden. Die Anforderungen bilden zusammen ein Minimum Viable Product (MVP) des Service. Die entstandene Implementierung erfüllt alle Anforderungen dieser Priorisierung. Tabelle 6.1.1 stellt den Stand dieser Anforderungen dar.

ID	Name	Beschreibung	
M10	Generische (Web-)Schnittstelle für den Empfang von Rohdaten	Um verschiedene Typen Rohdaten in Empfang zu nehmen, muss eine generische Schnittstelle angeboten werden.	✓
M20	Durchsuchbare und strukturierte Speicherung der Rohdaten	Die Rohdaten müssen durchsuchbar gespeichert werden, sodass diese auch bei großen Datenmengen effizient verarbeitet werden können.	✓
M30	Möglichkeit zur Berechnung von Metriken anhand strukturierter Rohdaten	Über die in *M10* definierte Schnittstelle werden generische strukturierte Daten (beispielsweise im JSON-Format) übertragen. Basierend auf den Rohdaten werden Analysen durchgeführt, die für die Gamification Elemente benötigt werden.	✓
M40	Speicherung berechneter Daten	Ebenso wie die Rohdaten, müssen auch die verarbeiteten Daten gespeichert werden. Die Menge der Daten ist hierbei signifikant kleiner als die der Rohdaten, jedoch ist die Zugriffszeit beim Laden der Daten von größerer Relevanz.	✓
M50	Abstraktion von Achievements/Badges	Als typisches Gamification-Element wurden Achievements bzw. Badges gewählt. Im Spielkontext beschreiben Achievements eine Errungenschaft, die separat vom Spielgeschehen ist (Hamari 2011). Diese können sehr generisch verwendet werden. Andere Konzepte (z.B. Level) könnten dieselbe Implementierung verwenden. Achievements bestehen aus einem Titel, einem beschreibenden Text, einem Bild und einem Zahlenwert. Dieser kann für Ranglisten verwendet werden.	✓
M51	Signifier	Der Signifier eines Achievements beschreibt die sichtbaren Elemente des Erfolgs. Er soll aus einem Titel, einem beschreibenden Text und einem zugehörigen Bild bestehen.	✓
M52	Trigger	Der Trigger ist die Logik, die erfüllt sein muss, um das Achievement freizuschalten.	✓

Tabelle 6.1.: Abgeschlossene (Kreis mit Häkchen) und offene (Kreis) Must-Have Kriterien

SHOULD-HAVE KRITERIEN

Die Should-Have Kriterien bilden die zweit-wichtigsten Kriterien. Aus der Liste der Should Have Kriterien wurde vorrangig **S20 Live-Benachrichtigungen** implementiert, da dieses Feature eine große Auswirkung auf die Benutzbarkeit und die User Expierience hat (Montola u. a. 2009; Herzig u. a. 2012). Weiterhin wurde **S11 Erfahrungspunkte und Level** und **S12 Ranglisten** umgesetzt, da die beiden Anforderungen eine größere Rolle im entwickelten Gamification Konzept besitzen.
Eine Aktivitätsanzeige (**S13**) wurde vorerst nicht implementiert, da die Live-Funktionen (**S20**) die geschätzte Komplexität überschritten haben. Eine Erweiterung um eine Aktivitätsanzeige ist jedoch unproblematisch. Diese kann unaufwändig durch eine JSON-Schnittstelle, mithilfe einer Datenbank-Abfrage nach den neusten Ereignissen in der Ergebnisdatenbank, implementiert werden. Die benötigten Daten-Attribute werden bereits in der Datenbank gespeichert. Tabelle 6.2 stellt den aktuellen Stand der Umsetzung dar.

ID	Name	Beschreibung	
S10	Abstraktion weiterer Gamification Elemente	Um die grundlegenden Gamification Features anbieten zu können, ist es nötig, übliche Spiel-Konzepte zu abstrahieren und anbieten zu können.	○
S11	Erfahrungspunkte und Level	Erfahrungspunkte sind ein Maß für den Langzeiterfolg des Spielers. (siehe Abschnitt 2.1.3)	⊘
S12	Ranglisten	Ranglisten sind ein gutes Mittel um Vergleichbarkeit zwischen den Spielern zu schaffen und den Ehrgeiz anzuregen (siehe Abschnitt 2.1.3; Xu (2011))	⊘
S13	Aktivitätsanzeige	Aktivitätsanzeigen bieten die Möglichkeit die letzten Errungenschaften des Nutzers darzustellen und verbessern damit das direkte Feedback (Xu 2011).	○
S20	Live-Benachrichtigungen	Direktes Nutzerfeedback ist insbesondere wichtig, um den Nutzer auf das Gamification System hinzuweisen und dessen Neugierde zu wecken (Montola u. a. 2009; Herzig u. a. 2012).	⊘

Tabelle 6.2.: Abgeschlossene (Kreis mit Häkchen) und offene (Kreis) Should-Have Kriterien

COULD-HAVE KRITERIEN

ID	Name	Beschreibung	
C10	Nutzerprofile	Nutzerprofile erlauben das öffentliche Einsehen der Nutzerdaten und sind sinnvoll um den Benutzern untereinander eine Vergleichbarkeit zu bieten (Xu 2011).	○
C20	Rechtesystem	Einige Nutzer möchten möglicherweise nicht, dass alle Informationen im Nutzerprofil freigegeben sind.	○

Tabelle 6.3.: Abgeschlossene (Kreis mit Häkchen) und offene (Kreis) Could-Have Kriterien

Von den aufgestellten Could-Have Kriterien konnten keine umgesetzt werden, da vorrangig Should-Have Kriterien ausgewählt wurden.

Offene funktionale Anforderungen sind damit die Could-Have-Kriterien **C10 Nutzerprofile** und **C20 Rechtesystem**. Die Kriterien wurden am niedrigsten priorisiert. Die Anforderungen werden in der nächsten Entwicklungsiteration implementiert. Sowohl das Rechtesystem, als auch Nutzerprofile waren Features, die von den Teilnehmern der Evaluation gefordert wurden (vgl. B.1).

6.1.2. NICHT-FUNKTIONALE ANFORDERUNGEN

ID	Name	Beschreibung	
NF10	Schnelles Feedback für den Nutzer	Schnelles Feedback ist für Gamification insbesondere wichtig, da dem Nutzer die Kausalität der Gamification-Elemente bewusst gemacht werden muss (Montola u. a. 2009; Herzig u. a. 2012).	✓
NF20	Skalierbarkeit bei vielen parallelen Anfragen	Um keine Beschränkung hinsichtlich der Nutzerzahl zu geben, ist es nötig, viele parallele Anfragen bearbeiten zu können.	✓
NF30	Robustheit bei häufigen Anfragen	Das System sollte gehäuften Anfragen standhalten, um zuzulassen, dass viele Daten aufgezeichnet und verarbeitet werden.	✓
NF40	Einfache Erweiterbarkeit	Auf einfache Erweiterbarkeit wird besonderen Wert gelegt, um den Service möglichst generisch und praxistauglich zu entwickeln.	✓
NF50	Schutz vor Datenverlust	Datenverlust ist kritisch, da eine Reihe von Gamificationelementen auf dem bisherigen Datenverlauf aufbaut.	✓

Tabelle 6.4.: Aktueller Stand der nicht-funktionalen Anforderungen. Graue Häkchen sind abgeschlossene, jedoch nicht evaluierte Features

Die nicht-funktionalen Kriterien benötigen einen höheren Evaluations-Aufwand. Die Anforderung **NF10** und zugehörige Schwächen der Implementierung werden in Abschnitt 6.1.3 diskutiert. Die Anforderungen **NF20**, **NF30** und **NF50** wurde nicht explizit evaluiert, sind allerdings seitens der AWS-Cloud abgesichert. Lasttests wurden nicht durchgeführt, da diese durch die Nutzung von Cloud-Diensten auch mit hohen Kosten verbunden sind. Die Erweiterbarkeit (Anforderung **NF40**) des Systems wird in Abschnitt 6.1.4 erörtert. Eine Übersicht des Standes der Nicht-Funktionalen Anforderungen wird in Tabelle 6.1.2 dargestellt.

6.1.3. LAUFZEITMESSUNGEN

Um die Performance des Systems zu evaluieren, wurde der PGE-Service an ein Testspiel angebunden und eine Messung der Dauer der einzelnen Anfragen vorgenommen. Dabei war insbesondere wichtig herauszufinden, ob die in Kapitel 5.5 vorgestellte Architektur-Erweiterung durch Redis und Websockets einen signifikanten Einfluss hat. Die Komponenten Rohdatenbank, sowie die Processing Functions basieren auf Technologien der AWS-Cloud und können gegen entsprechend höhere Gebühren skaliert werden.
Das Testsystem bestand aus einem quelloffenen Snake-Spiel[1], welches sämtliche Events[2] an den PGE-Service sendet.
Die Ergebnisse der Tests stellt Tabelle 6.5 dar. Insgesamt wurden zehn Stichproben aufgezeichnet. Durchschnittlich benötigt das System 1,2 Sekunden für die Analysen und die Darstellung eines Ergebnisses[3]. Einzelne Ausschläge (z.B. Stichprobe 6) lassen sich durch die Auslastung der Internetverbindung erklären.

Nr.	Gesamtdauer	Dauer Live-Komponente
1	1320 ms	208 ms
2	624 ms	306 ms
3	1046 ms	415 ms
4	847 ms	248 ms
5	808 ms	187 ms
6	2519 ms	170 ms
7	924 ms	185 ms
8	809 ms	275 ms
9	1001 ms	277 ms
10	1871 ms	258 ms
Ø	**1177 ms**	**253 ms**

Tabelle 6.5.: Ergebnisse der Zeitmessungen im Testsystem.

Die durchschnittliche Zeit, die für die Anzeige der Live-Benachrichtigung benötigt wird, beträgt ca. 250 Millisekunden. Da dies eine Übertragung vom Server zum Browser des Clients mit einschließt, ist diese Zeit vertretbar gering. Die Gesamtantwortzeit für Analysen und Darstellung von ca. einer Sekunde ist im Rahmen des Prototypen akzeptabel. Da die durchgeführten Analysen jedoch sehr einfach waren (Zählen von Datensätzen in der Rohdatenbank), sollten für die Weiterentwicklung des Systems, jedoch Performance-Optimierungen eingeplant werden.

[1] `https://github.com/patorjk/JavaScript-Snake` (Letzter Aufruf: 18.08.2017)
[2] Events im Snake-Spiel sind beispielsweise das Starten eines neuen Spiels, das „Essen", der Tod der Schlange etc.
[3] Alle Schritte, die gemessen werden, sind in Abschnitt 4.2.4 beschrieben.

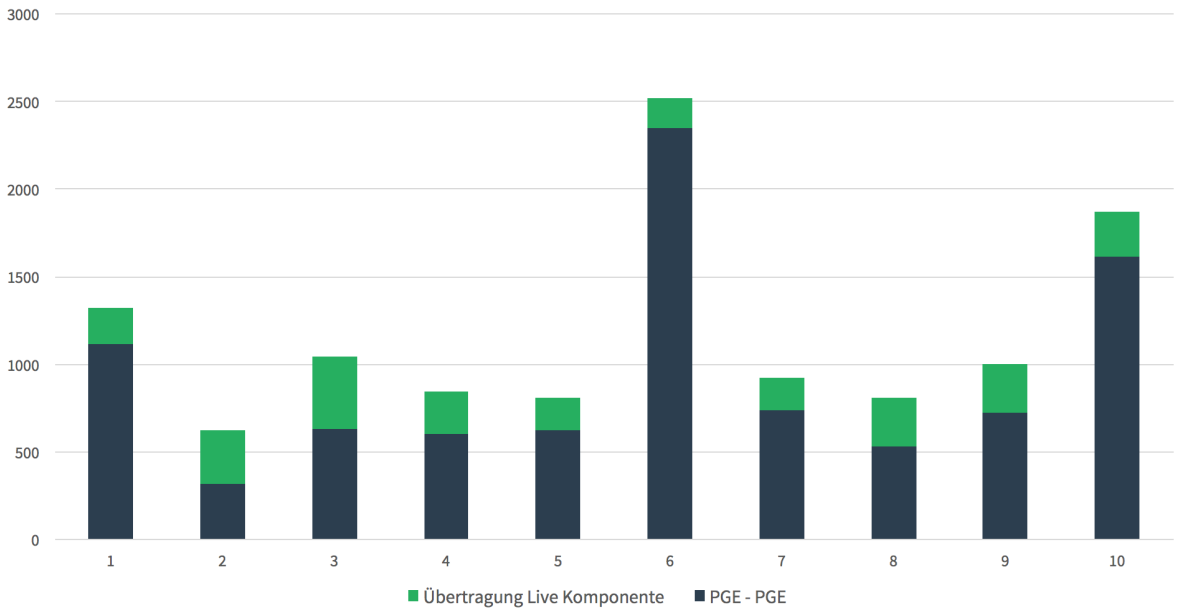


Abbildung 6.1.: Darstellung der Ergebnisse der Zeitmessungen im Testsystem. Die Dauer vom Auftreten des Events bis zum schreiben in den Kanal der Live-Komponente wird dunkelblau dargestellt, die Übertragungszeit zwischen PGE-Leader und Client über die Live-Komponente grün. Alle Werte sind in Millisekunden angegeben.

6.1.4. ERWEITERBARKEIT

Aufgrund der objektorientierten Herangehensweise ist das System gut erweiterbar. Einschränkung ist, dass die Erweiterung des PGE-Leaders in der Programmiersprache Python (Version 3) durchgeführt werden muss. Der PGE-Leader ist weiterhin stark an das Django-Framework gebunden.
Die Processing Functions sind hingegen von keiner Programmiersprache und keinem Framework abhängig. Momentan setzen alle Processing Functions auf Python (Version 3.6) und werden auf AWS-Lambda ausgeführt. Dies ist jedoch keine Voraussetzung für die Entwicklung weiterer Processing Functions. Einerseits können zusätzliche Processing Functions in anderen Programmiersprachen für AWS-Lambda entwickelt werden, andererseits ist auch die Verwendung von AWS-Lambda nicht bindend. Die Processing Functions müssen lediglich im PGE-Leader registriert werden und diesen mittels REST-Aufruf über neue Daten benachrichtigen.
Dadurch ist auch die Nutzung von DynamoDB keine Voraussetzung für die Verwendung des Services. Die Kombination wurde in dieser Arbeit verwendet, da DynamoDB und AWS Lambda gut zusammen verwendbar sind und bereits viele Integrationen[4] anbieten.

[4] Automatische Ausführung von Lambda Funktionen bei neuen Datensätzen, einfacher Zugriff von Lambda auf Dynamo ohne zusätzliche Bibliothek

6.1.5. SICHERHEIT

Durch die prototypische Implementierung wurde an vielen Stellen die Sicherheit des Systems vernachlässigt. Viele kritische Lücken werden zwar durch Mechanismen von Django oder der AWS-Cloud bereitgestellt, insbesondere die Nachrichtenkanäle und JSON-APIs sind jedoch für einen Produktivbetrieb nicht ausreichend geschützt. Diese offenen, im Kapitel 5 beschriebenen, Sicherheitslücken zu schließen, hat für die Weiterentwicklung des Systems hohe Priorität.

6.2. NUTZEREVALUATION

Die Nutzerevaluation wurde zur Beurteilung des Gamification-Kozepts, vorgestellt in Kapitel 4.1, durchgeführt. In diesem Abschnitt werden die Durchführung und die Ergebnisse der Nutzerevaluation beschrieben. Abschließend wird das Resultat bewertet und ein abschließendes Fazit gezogen.

6.2.1. DURCHFÜHRUNG

Die Nutzerevaluation wurde mit der Evaluation des Code-Editors verbunden, um den Suchaufwand nach geeigneten Probanden zu minimieren. Anforderung an die Teilnehmer der Evaluation waren mindestens rudimentäre Programmierkenntnisse. Diese sind notwendig, da die Probanden Qualitätsmetriken zu Quellcode einschätzen und finden sollten.
Die Teilnehmer der Evaluation waren sowohl ausgebildete Informatiker, teils mit längerer Berufserfahrung, als auch fachfremde Programmieranfänger. Die Personengruppe wurde bewusst heterogen gewählt, um ein möglichst ambivalentes Testergebnis zu erreichen.

Insgesamt wurden sieben Teilnehmer befragt. Drei der Probanden haben jeweils mehr als sieben Jahre Berufserfahrung und sind in etablierten Softwareentwicklungs-Firmen tätig. Zwei Teilnehmer haben je zwei bis fünf Jahre Erfahrung in Firmenprojekten, einer der Probanden hat zwar einen Bachelorabschluss im Fach Informatik, jedoch weniger als ein Jahr Berufserfahrung. Ein weiterer Proband ist Quereinsteiger (abgeschlossenes Biologie-Studium), keine Berufserfahrung und wenig Programmiererfahrung (weniger als ein Jahr). Somit waren unter den Probanden sowohl Einsteiger (für eine Bewertung aus Perspektive der Crowd-Entwickler), als auch erfahrene Software-Architekten für eine fundierte Bewertung aus Sicht des Plattformbetreibers. Die Bearbeitung der Implementierungsaufgaben und die Beantwortung des Fragebogens erfolgte auf einem bereitgestellten Laptop. Die Blank Paper Evaluation wurde auf Klebezetteln bzw. A7-Karteikarten durchgeführt. Es wurde Wert darauf gelegt, die Befragung an, für den Probanden angenehmen aber kontrollierten Orten, ohne äußerlichen Einfluss, durchzuführen. Die Uhrzeit der Befragung wurde durch die Teilnehmer bestimmt.

BLANK PAPER EVALUATION

Im ersten Schritt wurde den Probanden das Konzept der crowd-basierten Outsourcing-Plattform erklärt. Die Teilnehmer sollten sich hierbei sowohl in die Rolle des Plattform-Betreibers, als auch in die eines Crowd-Entwicklers hineindenken. Aus diesen Perspektiven entwarfen sie, ohne vorher das in dieser Arbeit entstandene Konzept kennengelernt zu haben, ein eigenes Konzept. Die einzigen Vorgaben dieses kreativen Prozesses sind die drei Kernpunkte, mit denen sich die Probanden auseinandersetzen sollten. Speziell sollten sich diese mit folgenden Kernproblemen befassen:

1. Performance-Metriken zur Bewertung von Crowd-Entwicklern
2. Gamification-Elemente mit denen diese Metriken verbessert werden und
3. Features, die für das entstandene Gamification-Konzept essentiell sind.

Um die Gedanken der Teilnehmer zu strukturieren, sollten diese ihre Ideen auf Klebezetteln vermerken. Gleichzeitig diskutierten sie ihre eigenen Ideen, erkannten damit selbstständig Probleme bei der Durchführung und entwarfen Lösungsvorschläge.
Ziel dieses Evaluationsschrittes war das Finden einer Schnittmenge der Ideen der Teilnehmer und damit eines naheliegenden Konzepts. Dieses Konzept wurde nachfolgend mit dem Ansatz dieser Arbeit verglichen. Auf dieser Basis konnten entsprechende Erweiterungen bzw. Veränderungen geplant werden.

DEMONSTRATOR UND NUTZERBEFRAGUNG

Nach der Blank-Paper Evaluation implementierten die Teilnehmer drei Entwicklungsaufgaben in drei unterschiedlichen Code-Editoren. Einer der Editoren war der Prototyp des Editors auf der Plattform. Dieser ist mit einfachen Gamification-Konzepten angereichert: einerseits erlangen die Nutzer einen Erfolg für die erste absolvierte Aufgabe, andererseits erhalten sie bei Abgabe Erfahrungspunkte. Ein Fenster, welches bei der Abgabe erscheint, zeigt die gewährten Erfahrungspunkte an und deutet die Bewertung der Code-Qualität an (Abbildung 6.2). Dadurch wurde den Befragten ein erster Eindruck der Entwicklerbewertung mit Gamification auf der Plattform vermittelt.
Nachdem die Test-Aufgaben implementiert wurden, beantworteten die Teilnehmer einen Fragebogen zum vorgestellten Konzept. In diesem sollten diese den Entwurf der polylith-Plattform ohne das Gamification-Konzept anhand der Kriterien des *Octalysis*-Frameworks (vorgestellt in Abschnitt 2.2.1) beurteilen. Dazu erhielten sie eine kurze Beschreibung der Plattform, anhand der sie die acht Core-Drives von *Octalysis* bewerten sollten. Die Bewertung erfolgte durch jeweils einen Schieberegler, welcher auf die Zahlen von 1-10 abbildet. Um zusätzliches, nicht-statistisches Feedback zuzulassen, hatten die Probanden die Möglichkeit, zusätzlich einen Text in ein Kommentarfeld einzutragen.
Nachfolgend beantworteten sie Fragen zu den einzelnen Elementen *Erfolge*, *Ranglisten*, *Erfahrungspunkte* und *Social Media Integrationen*. Die Fragen zielten hierbei auf das Verständnis, die Optik und die persönliche Meinung zu diesem Konzept. Zusätzlich wurden Fragen zur Nutzung der Gamification Elemente beantwortet. Weiterhin konnten die Nutzer zu jedem der Konzepte einen Verbesserungsvorschlag oder einen Kommentar in einem Freitextfeld angeben. So setzten sich die Probanden gleichzeitig detaillierter mit der Gamification auf der Plattform auseinander und hatten eine bessere Vorstellung der Integration.

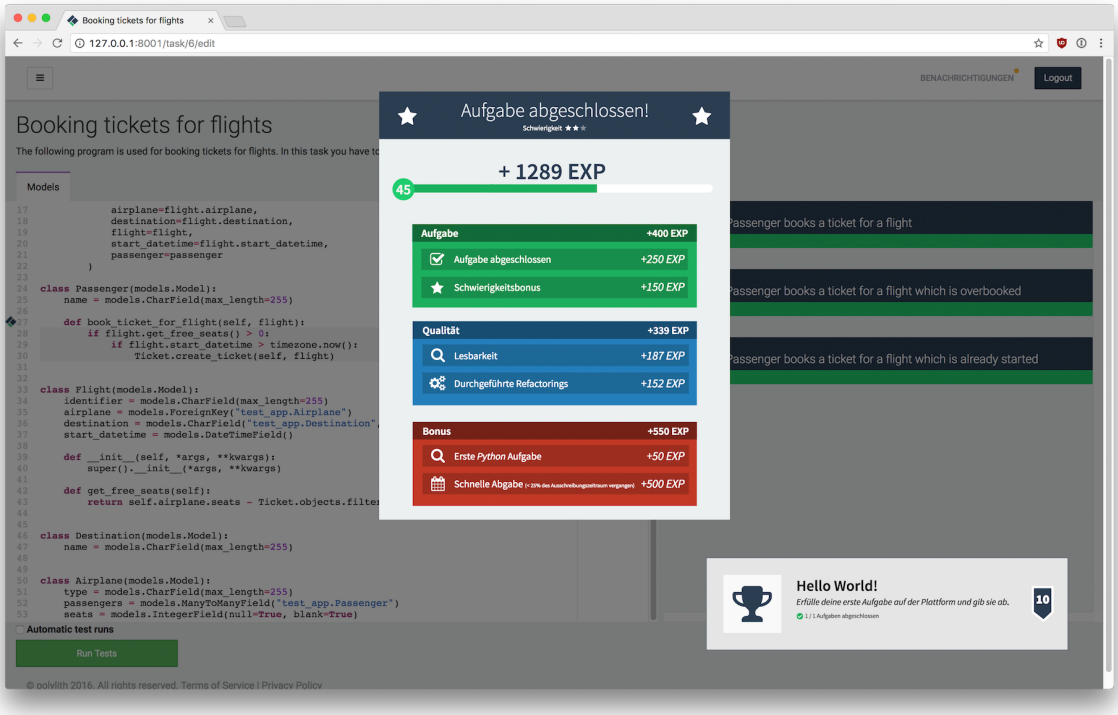


Abbildung 6.2.: Screenshot des Demonstrator mit Bewertungsfenster, welches erhaltene Erfahrungspunkte in Abhängigkeit der Codequalität und einen Erfolg zeigt

Im nächsten Schritt wurde die Plattform **mit** Gamification-Konzept nach den Kriterien von *Octalysis* beurteilt. Die Plattform und das Konzept wurden im Fragebogen kurz beschrieben, um den Teilnehmern einen klareren Eindruck zu vermitteln, wie die Entwicklerbewertung und Gamification in die Plattform integriert ist. Die Einschätzung der Core-Drives erfolgte erneut durch Schieberegler mit Werten von 1-10 und einem Freitextfeld. Auf Basis der Einzelbewertungen kann der *Octalysis*-Score berechnet werden und die beiden Stände verglichen werden. Anhand der Beurteilungen der Core-Drives wird das *Octalysis*-Diagramm gezeichnet, an welchem man die Ausprägung der einzelnen Core-Drives und die Art der Motivation erkennen kann. Die Einschätzungen der Probanden sind hierbei stark subjektiv und dementsprechend vielfältig ausgeprägt. Damit kann die Verbesserung des Gamification-Konzepts beurteilt werden. Abschließend geben die Befragten statistische Informationen zu Programmierkenntnissen, Geschlecht und Alter an.

6.2.2. ERGEBNISSE

In diesem Abschnitt werden die Ergebnisse der Evaluation zusammengefasst vorgestellt. Die Resultate jedes einzelnen Probanden sind in Anhang Abschnitt B.1 zu finden.

BLANK PAPER EVALUATION

Hinsichtlich der genutzten **Gamification Elemente** entwarf die Mehrheit der Probanden eine Lösung ähnlich zum Konzept dieser Arbeit (Abbildung 6.6). Die meisten Probanden würden die Plattform mit Erfolgen und Erfahrungspunkten erweitern, um die Motivation zu steigern und die Performance der Crowd-Entwickler zu optimieren. Viele Teilnehmer empfanden die Geldbelohnung als sehr motivierend und schlugen teils Konzepte vor, die diese nutzen und erweitern. So wurden beispielsweise durch Level freischaltbare Stundensatzerhöhungen oder Statistiken über bisher verdientes Geld vorgeschlagen.
Von einzelnen Probanden wurden erwähnenswerte ungewöhnliche Elemente entwickelt. So schlug ein Proband vor, für besondere Errungenschaften reale Badges an die Crowd-Entwickler zu versenden. Ein weiterer Proband entwarf das Konzept von einfachen *„grinding“*[5] Aufgaben, welche zur Erholung erfüllt werden können. Diese erbringen nur wenige Belohnungen, können aber beliebig oft wiederholt werden. Als Beispiel nannte der Proband Refactorings, um Quellcode von anderen Crowd-Entwicklern zu verbessern.
Ein weiterer Proband schlug vor, zu jedem Account eine Spielfigur (Avatar) vom Crowd-Entwickler erstellen zu lassen. Für diesen können Ausrüstungsgegenstände durch Quests freigeschaltet werden oder in einem Item-Shop erworben werden.

	1	2	3	4	5	6	7	Anteil
Erfahrungspunkte / Level	×	×	×	×			×	71%
Erfolge	×	×	×		×	×		71%
Ranglisten	×	×				×	×	57%
Motivation Coins	×							14%
Geldbelohnungen			×	×	×	×		57%
Avatar mit Ausrüstung			×					14%
Item Shop			×					14%
Quests			×					14%
Titel				×				14%
Wettkämpfe				×				14%
Tägliche / Wöchentliche Aufgaben				×				14%
"Grinding" Aufgaben				×				14%
Statistiken					×	×		29%
Öffentliches Profil	×		×	×	×			57%
Freischaltungen	×				×			29%

Tabelle 6.6.: Verteilung der genutzten Gamification-Elemente innerhalb der Blank Paper Evaluation. Im Gamification-Konzept enthaltene Elemente sind grün markiert.

[5]Das Konzept von Grinding kommt aus Online-Rollenspielen, in denen durch häufiges Wiederholen derselben Aufgabe (z.B. das Töten eines Gegners) ein Fortschritt erzielt werden kann. `https://en.wikipedia.org/wiki/Grinding_(gaming)` (Letzter Aufruf 13.08.2017)

Die Tendenz der entworfenen **Features** war weniger eindeutig. So wollten ebenso viele Nutzer die vorerst nicht implementierten Privatsphären-Einstellungen, wie die angedachte Social-Media Integration. Die bei der Blank Paper Evaluation entworfenen Features sind in Tabelle 6.7 dargestellt. Viele Versuchspersonen hatten Schwierigkeiten Features und Gamification Elemente zu trennen – Viele benötigten eine zusätzliche Erklärung um eine Vorstellung zu bekommen was gemeint ist. Aus diesem Grund waren die Teilnehmer der Evaluation in diesem Schritt weniger kreativ.

	1	2	3	4	5	6	7	Anteil
Social Media Integration	×	×						29%
Schnelles Feedback	×		×		×			43%
Privatsphären-Einstellungen		×		×			×	43%
Forum			×					14%
Timeline				×				14%
Gimmicks						×		14%

Tabelle 6.7.: Verteilung der genutzten Gamification-Features innerhalb der Blank Paper Evaluation. Angedachte Features sind grün markiert.

Die, von den Nutzern entwickelten **Metriken** sind sehr unterschiedlich. Mehrere Probanden nannten die in dieser Arbeit beschrieben Metriken in Bezug auf Refactorings und Umstrukturierungen, sowie Lesbarkeit. Mehrere Probanden wünschten sich Metriken zur Code-Qualität, konnten aber nicht benennen wonach diese Bewertet wird. Code-Performance (Ausführungsdauer) wurde von ebenso mehreren Probanden genannt.

ERGEBNISSE DES FRAGEBOGENS

Frage	1	2	3	4	5	6	7	ø
Ich verstehe, was dieses Element aussagt.	100%	100%	100%	80%	100%	100%	100%	97%
Ich finde dieses Element optisch ansprechend.	100%	100%	80%	100%	100%	80%	100%	94%
Ich würde versuchen herauszufinden, welche Erfolge es zusätzlich gibt.	100%	100%	80%	80%	40%	100%	100%	86%
Ich hätte Interesse daran, welche Erfolge andere Benutzer errungen haben.	80%	60%	20%	80%	20%	40%	80%	54%
Ich wäre motiviert weitere Erfolge zu erringen.	100%	80%	100%	100%	60%	100%	100%	91%
Ich hätte Interesse daran, Erfolge zu erringen, die keiner meiner Freunde hat.	100%	100%	20%	100%	20%	40%	80%	66%
Es würde mich demotivieren, wenn ich sehen würde, dass meine Freunde Erfolge haben, die ich nicht habe.	40%	40%	20%	100%	20%	20%	40%	40%
Es würde mich besonders anspornen, wenn ich sehen würde, dass meine Freunde Erfolge haben, die ich nicht habe.	100%	80%	20%	100%	20%	20%	60%	57%
Mich würde es stören, wenn ich für zu viele Aktionen Erfolge bekomme.	80%	20%	80%	60%	80%	20%	60%	57%
Es würde mich stören, wenn meine errungenen Erfolge öffentlich wären.	20%	60%	80%	20%	20%	40%	40%	40%
Es würde mich stören, wenn meine Freunde meine errungenen Erfolge sehen könnten.	20%	60%	60%	20%	20%	20%	40%	34%
Ich finde es spannend, herauszufinden welche Erfolge es noch gibt, indem ich Dinge ausprobiere.	100%	80%	80%	60%	20%	100%	80%	74%

Tabelle 6.8.: Detaillierte Fragebogenergebnisse zu Erfolgen

Der Abschnitt des Fragebogens zu **Erfolgen** verdeutlicht, dass die Probanden das Konzept der Erfolge verstanden haben und diese optisch ansprechend fanden. Es würde sie motivieren, herauszufinden welche Erfolge auf der Plattform erreichbar sind und die Mehrheit der Probanden würde besonderen Wert darauf legen, sich mit Freunden und anderen Crowd-Entwicklern zu vergleichen. Die Teilnehmer der Befragung hatten dabei jedoch auch mäßig die Befürchtung, mit Erfolgen überhäuft zu werden. Eine Veröffentlichung der Erfolge (z.B. auf einem Nutzerprofil) würde die meisten Teilnehmer nicht stören. Alle Ergebnisse werden in der folgenden Tabelle zusammengefasst dargestellt:
Das Konzept der **Ranglisten** auf der Plattform wurden von den Befragten ebenfalls positiv aufgenommen. Die Mehrheit der Befragten verstand das Konzept und die Darstellung – ein Proband hatte Probleme, die Sortierung nach nur einem Wert, anstatt der Gesamtleistung zu erkennen. Die Frage nach einer potentiellen Demotivation durch Ranglisten verneinten die meisten Teilnehmer. Die Ergebnisse der einzelnen Befragungen werden in Tabelle 6.2.2 zusammengefasst:

Frage	1	2	3	4	5	6	7	ø
Ich verstehe, was dieses Element aussagt.	100%	100%	100%	100%	100%	80%	100%	97%
Ich finde dieses Element optisch ansprechend.	100%	100%	80%	100%	80%	80%	100%	91%
Ich würde versuchen in der Rangliste aufzusteigen.	100%	80%	20%	60%	60%	60%	100%	69%
Es würde mich anspornen einen höheren Rang als meine Freunde zu haben.	100%	100%	20%	80%	20%	20%	80%	60%
Es würde mich demotivieren, wenn ich sehen würde, dass meine Freunde einen höheren Rang haben als ich.	20%	60%	20%	80%	20%	20%	40%	37%
Es würde mich stören, in der Rangliste aufzutauchen.	20%	20%	60%	40%	20%	20%	20%	29%

Tabelle 6.9.: Detaillierte Fragebogenergebnisse zu Ranglisten

Mit den **Erfahrungspunkten** kamen die Probanden schon nach der Implementierung der Aufgaben in Kontakt. Im Fragebogen wurden Fragen zu diesem Konzept beantwortet. Schon nach Abschluss der Implementierungsaufgaben konnte den Versuchspersonen durch das animierte Erfahrungspunktefenster sehr positives Feedback und großes Interesse an der dahinter liegenden Bewertung entnommen werden. Entgegen der Erwartung[6] wurde die Frage nach der ungerechten Bewertung nur von zwei Entwicklern als zutreffend (über 50 %) angesehen.

Frage	1	2	3	4	5	6	7	ø
Ich verstehe, was dieses Element aussagt.	100%	100%	60%	100%	100%	80%	100%	91%
Ich finde dieses Element optisch ansprechend.	100%	100%	80%	100%	60%	100%	100%	91%
Ich würde versuchen beim Entwickeln größeren Wert auf die Bewertungskriterien zu legen um mehr Erfahrungspunkte zu erreichen.	100%	60%	100%	80%	80%	100%	80%	86%
Es würde mich demotivieren, wenn meine Freunde mehr Erfahrungspunkte hätten, als ich.	20%	40%	20%	40%	20%	20%	40%	29%
Es würde mich anspornen mehr Erfahrungspunkte zu sammeln als meine Freunde.	100%	60%	20%	40%	20%	20%	60%	46%
Ich habe das Gefühl, dass ich ungerecht bewertet werden könnte.	20%	100%	40%	40%	20%	60%	20%	43%

Tabelle 6.10.: Detaillierte Fragebogenergebnisse zu Erfahrungspunkten

[6] Entwicklerbewertungen sind ein stark diskutiertes Thema, da die wirkliche Performance nur schwer von außen bewertet werden kann und meist ein Vergleichswert fehlt. (vgl. York (o.D.), Baggelaar (2008))

6.2.3. OCTALYSIS ERGEBNISSE

Um das Konzept der Gamification zu Evaluieren wurde eine Einschätzung der einzelnen Core-Drives des Octalysis Frameworks (vgl. Kapitel 2.2.1) durchgeführt. Das durchschnittliche Ergebnis der Befragung stellt Abbildung 6.3 in der typischen Achteck-Darstellung dar. Tabelle 6.11 schlüsselt die Einzelergebnisse und deren Veränderung auf.

Aus dem Vergleich der Octalysis-Scores ohne Gamification (Gesamtscore: 95) und mit Gamification Konzept (Gesamtscore: 351) geht eine deutliche Verbesserung hervor. Insbesondere die Core-Drives *Epic meaning and calling*, *Social influence and relatedness* und *Scarcity and impatience* haben durch das angewendete Konzept eine starke Verbesserung erhalten.
Das Endergebnis der Octalysis-Analyse ist sehr ausgewogen, jedoch sind die oberen Core-Drives (*Meaning*, *Accomplishment*, *Empowerment*) etwas stärker ausgeprägt. Da das Gamification-Konzept bewusst positiv gestaltet ist, zeigt das Ergebnis, dass die Nutzer die Motivatoren der Gamification ebenso eher positiv wahrgenommen haben.

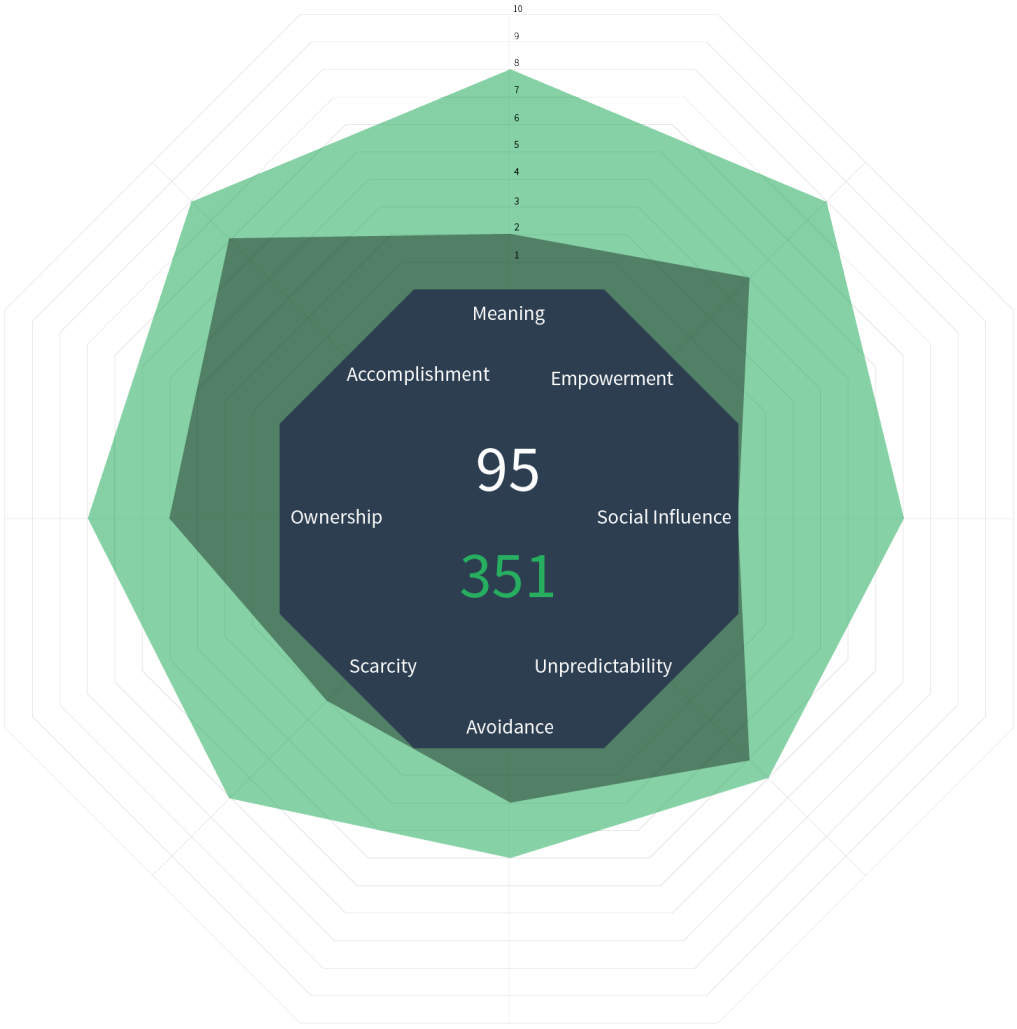


Abbildung 6.3.: Durchschnittliche Bewertung nach den Core-Drives von Octalysis **ohne** Gamification (*grau/dunkelgrün*) und **mit** Gamification (*grün*)

	ohne Gamification	mit Gamification	Veränderung
Epic meaning and calling	2,43	7,57	+ 5,14
Development and accomplishment	5,86	7,71	+ 1,85
Empowerment of creativity and feedback	4,43	8,29	+ 3,86
Ownership and possession	4,14	7,43	+ 3,29
Social influence and relatedness	0,43	5,86	+ 5,43
Scarcity and impatience	0,57	6,29	+ 5,72
Unpredictability and curiosity	3,86	4,71	+ 0,85
Loss and avoidance	1,57	3,71	+ 2,14
Gesamt	**94,88**	**350,53**	+ 255,65

Tabelle 6.11.: Ergebnisse der einzelnen Core-Drives des Octalysis

6.2.4. BEWERTUNG UND FAZIT

Die Blank Paper Evaluation ergab im Wesentlichen ein ähnliches Gamification Konzept, wie das in dieser Arbeit beschriebene. Dennoch hatte jeder Proband originelle Einfälle, für eine Erweiterung des Konzeptes. Rückblickend konnten die Teilnehmer kreativ mit den beiden vorgegebenen Punkten *Metriken* und *Gamification Elemente* arbeiten. Die Kategorie *Features* war für viele Teilnehmer nicht greifbar, die Trennung zwischen Gamification Elementen und Features fiel vielen schwer. Teils konnten sich die Versuchspersonen erst etwas unter dem Punkt vorstellen, nachdem ihnen ein Beispiel genannt wurde.
Die Optik und das Verständnis der Elemente haben durchweg die höchste Zustimmung erhalten. Dies ist insbesondere deshalb ein positives Ergebnis, da die Ästhetik für die User Experience eines Systems eine große Rolle spielt (Sonderegger und Sauer 2010) und das Verständnis der Gamification Elemente bereits in anderen Studien als mögliches Problem dargestellt wird (Montola u. a. 2009). Die Demotivation durch eine bessere Bewertung anderer Crowd-Entwickler war bei keinem der Elemente höher als 40% (Erfolge). Als höchsten anspornenden Anreiz gaben diesbezüglich die Versuchspersonen die Ranglisten (60%), dicht gefolgt von Erfolgen (57%) an. Die höheren Werte zeigen, dass sich die Teilnehmer von anderen Nutzern eher motivieren, als demotivieren lassen. Die Frage zur Nutzung der Elemente wurde ebenfalls durchgängig positiv bewertet. Das größte Interesse hatten hierbei die Nutzer an Erfolgen (91% wären motiviert, weitere Erfolge zu erringen). Bei diesen wurde besonders der Erkundungsfaktor hervorgehoben (86% wären motiviert, herauszufinden, wofür es Erfolge zu erringen gibt). Die Umfrageergebnisse verdeutlichen, dass die gewählten Gamification-Elemente einerseits die Erwartungen der Nutzer erfüllen (gezeigt durch Blank Paper Evaluation), und andererseits die Nutzergruppen nach Bartle abgebildet werden konnten. Die Nutzergruppen lassen sich teils aus den Umfrageergebnissen ableiten. So gab Proband 3 beispielsweise in Fragen, die auf die Gruppe der *Socializers* abzielen unterdurchschnittliche Bewertungen, Fragen im Kontext der *Achievers* erhielten jedoch höhere Punktzahlen. Proband 6 kann anhand der Ergebnisse stark den *Explorers* zugeordnet werden. Trotz der deutlich unterschiedlichen Vorlieben der Versuchspersonen ist die durchschnittliche Gesamtbewertung positiv.

Die Analyse mit Octalysis ergab ein äußerst positives Ergebnis. Dabei muss jedoch beachtet werden, dass einige Probanden Schwierigkeiten hatten, die einzelnen Core-Drives mit der Crowdsourcing Plattform in Verbindung zu setzen. Dadurch bewerteten einige Versuchspersonen den Vorher-Stand (ohne Gamification) eher negativer. Dennoch zeigt das Ergebnis, dass die Versuchspersonen das Gamification-Konzept gut aufgenommen haben. Insbesondere die Ausprägung auf positive Motivatoren spiegelt sich im Ergebnis der Octalysis-Analyse wider.

Abschließend kann festgestellt werden, dass das vorgestellte Gamification-Konzept positiv von den Probanden aufgenommen wurde. Die Evaluation des implementierten Konzepts auf der Crowd-Sourcing Plattform muss jedoch aufgrund der kleinen Anzahl Versuchspersonen (7) mit einer größeren Menge Probanden erneut durchgeführt werden.

6.2.5. VERBESSERTE FARBWAHL

Das Bewertungsfenster bei Abgabe der Aufgabe (Abbildung 6.2) wurde von mehreren Probanden aufgrund der Farbwahl kritisiert. Einerseits wurde bemängelt, dass die rote Farbe im Bonus-Bereich als Warnfarbe wahrgenommen wird und dadurch zu stark ablenkt, beziehungsweise wie ein Fehler wirkt. Dies tritt insbesondere dadurch auf, dass fehlgeschlagene Tests und die zugehörigen Fehler im Code-Editor in derselben Farbe dargestellt werden. Andererseits wurde die Farbwahl des Fensters für Personen mit Rot/Grün-Schwäche kritisiert. Da dieser Sehstörung relativ häufig auftritt (ca. 5% bei Männern, 0,4% bei Frauen; Kolb H (1999)) wurde der Mockup entsprechend angepasst und beide Verbesserungsvorschläge angenommen. Dieser Effekt wird (simuliert) in Abbildung 6.4 dargestellt. Der Rotton wurde durch einen Blau- / Cyan-Ton ersetzt.

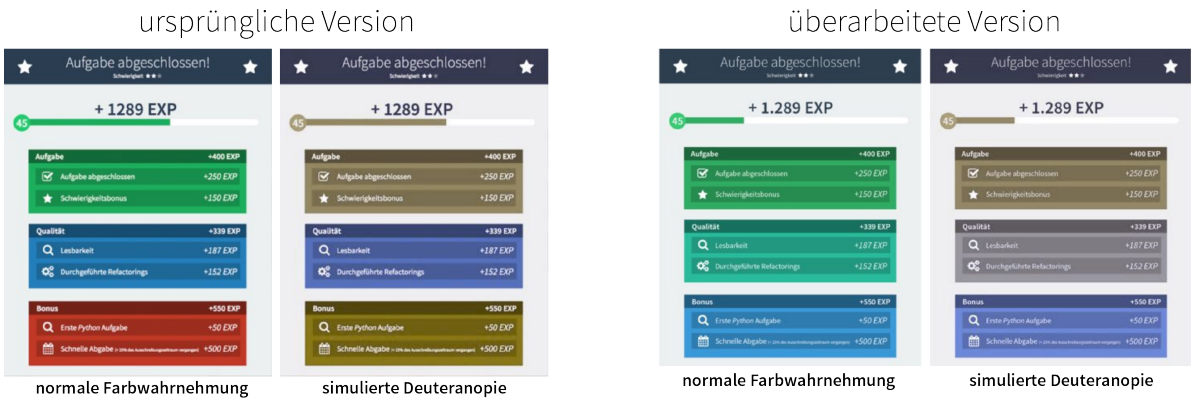


Abbildung 6.4.: Farbwahrnehmung des Erfahrungspunkte-Dialogs einer Person mit Rot/Grün-Schwäche und dessen Verbesserung simuliert durch `http://www.vischeck.com/vischeck/vischeckImage.php` (Letzter Aufruf 10.08.2017)

7. ZUSAMMENFASSUNG UND AUSBLICK

Das Ziel dieser Arbeit ist es einerseits, ein Gamification-Konzept für eine Crowd-basierte Outsourcing-Plattform zu entwerfen, mit dem aufgestellte Performance-Metriken für diese optimiert werden. Andererseits sollte ein Service gefunden oder implementiert werden, mit dessen Hilfe die Umsetzung des Konzepts möglich ist. In diesem Kapitel werden die Vorgehensweise und die Resultate dieser Arbeit zusammengefasst und bewertet. Abschließend wird ein Ausblick in die zukünftige Entwicklung des Konzepts und des Gamification-Services gegeben.

7.1. ZUSAMMENFASSUNG

7.1.1. GAMIFICATION-KONZEPT

Für das Gamification-Konzept der Plattform sollten zunächst Metriken aufgestellt werden, mit deren Hilfe man die Performance der Crowd-Entwickler einschätzen kann. Anhand dieser Metriken wurde mit den, im Kapitel State-of-the-Art beschriebenen, Gamification-Elementen ein Konzept ausgearbeitet, welches einerseits die entworfenen Metriken verbessert und andererseits die User Experience des Crowd-Entwicklers aufwertet. Die entworfenen Metriken beziehen sich hierbei einerseits auf die Nutzung der Crowdsourcing-Plattform an sich, andererseits bauen sie auf die Code-Qualitätsmetriken von Blume (2017) und eigenen Metriken auf. Die entstandenen Metriken sollen die Performance der Crowd-Entwickler aus Sicht des Plattformbetreiber in beiden Bereichen beschreiben können.

Das Gamification-Konzept baut auf den verbreiteten Spielelementen *Erfolge*, *Erfahrungspunkte und Levels* sowie *Ranglisten* auf. Diese Spielelemente wurden angepasst auf die verschiedenen Spielergruppen nach Bartle (1999) angewendet. Dadurch soll eine möglichst gute Abdeckung des Gamification-Konzepts auf die ambivalenten Crowd-Entwickler erreicht werden. Zusätzlich wurden zum Konzept Mockups zu allen Gamification-Elementen erstellt. Dadurch konnte ein besserer Eindruck der fertigen Elemente gewonnen werden. Zusätzlich konnten die Mockups für den Entwurf und die Implementierung des Gamification-Service *PGE* genutzt werden.

Das entstandene Gamification-Konzept wurde abschließend evaluiert. Die Evaluation soll hierbei sowohl die Sichtweise des Crowd-Entwicklers, als auch die des Plattformbetreibers bewerten. Um das (abstrakte) Konzept zu evaluieren, wurden Teile des Konzepts (Erfahrungspunkte und Erfolge) in die Evaluation des Code-Editors von Dienst (2017) eingebunden. Anschließend bewerteten die Versuchsteilnehmer das Konzept anhand eines Fragebogens. Die Evaluation ergab, dass das Gamification-Konzept im Allgemeinen positiv aufgenommen wurde, es wurden jedoch auch fehlende Features wie Privatsphären-Einstellungen deutlich. Die aufgefallenen Mängel werden in die weiteren Entwicklungs-Iterationen des Konzepts aufgenommen.

7.1.2. GAMIFICATION SERVICE

Das Ziel beim Entwurf und der Implementierung des Gamification-Services war es, einen Prototypen zu entwickeln, der möglichst zeitnah in den produktiven Betrieb überführt werden kann. Um zu evaluieren, ob ein vorhandener Service genutzt werden kann, wurden fertige Services anhand selbst aufgestellter Anforderungen untersucht. Keiner der Services erfüllte diese hinreichend. Daher wurde entschieden, eine eigene Implementierung vorzunehmen. Für diese wurden die Anforderungen in Kapitel 2.5 aufgestellt und anhand der MoSCoW-Methode priorisiert. Diese Anforderungen bildeten die Grundlage für das Implementierungskonzept. Bei der Implementierung wurde besonderer Wert auf Generizität gelegt und gleichzeitig sichergestellt, dass das entworfene Gamification-Konzept mit dem Service umsetzbar ist.

Ziel des Gamification-Services ist es, das oben genannte Konzept umsetzen zu können. Dafür ist einerseits die Abstraktion von Gamification Elementen wie Erfolgen und Erfahrungspunkten notwendig. Andererseits wird, um die teils komplexen Metriken implementieren zu können eine Technologie benötigt, die auf eine ereignis-getriebene Art Berechnungen ausführen kann. Dies wurde mithilfe des Function-as-a-Service Dienstes AWS Lambda und dort deployten *Processing Functions* ermöglicht. Die berechneten Metriken müssen dem Nutzer in Form einer Live-Benachrichtigung dargestellt werden können, weshalb der Service um eine Live-Komponente auf Basis von Websockets und der Key-Value-Datenbank *redis* erweitert wurde. Um die vielfältigen Rohdaten des Services performant speichern zu können, wurde die Big-Data Datenbank DynamoDB angebunden. Diese ermöglicht das Speichern von beliebig vielen Rohdaten und gleichzeitig das automatische Ausführen von Processing Functions. Dadurch setzt die entstandene Infrastruktur bereits viele der notwendigen Anforderungen um. Die Implementierung der Gamification-Konzepte und die Verwaltung der Meta-Daten übernimmt der eigens entwickelte PGE-Leader Server.

Die Evaluation des Services ergab, dass alle grundlegenden Features umgesetzt sind. Gleichzeitig sind jedoch auch Schwächen im Bereich der Performance und Sicherheit deutlich geworden. Da der Service in einer Produktivumgebung eingesetzt werden soll, müssen diese Mängel bei der zukünftigen Weiterentwicklung des Services hoch priorisiert und entsprechend ausgebessert werden. Fehlende funktionale Features wie Aktivitätsanzeigen, Nutzerprofile und ein Rechtesystem werden in der zukünftigen Weiterentwicklung eingeplant. Perspektivisch muss auch die Evaluation der bisher außer Acht gelassenen nicht-funktionalen Kriterien durch Lasttests nachgeholt werden.

7.2. AUSBLICK

Der entstandene Service und das entwickelte Gamification-Konzept soll innerhalb der Crowdsourcing-Plattform verwendet werden, um Crowd-Entwickler zu motivieren. Dafür sollen die von Blume (2017) entwickelten Code-Metriken und der von Dienst (2017) entworfene und implementierte Code-Editor verwendet werden. Der Gamification-Service soll als eine der maßgeblichen Komponenten zum Start der Crowdsourcing-Plattform eingesetzt werden. Dafür müssen vorrangig die Metriken evaluiert und implementiert werden. Zusätzlich ist es essentiell, die Sicherheit und die Performance des Systems zu verbessern.

Die Weiterentwicklung schließt zusätzlich die nicht-, oder nicht vollständig umgesetzten Features Aktivitätsanzeige, Nutzerprofile und ein Rechtesystem ein. Insbesondere das Rechtesystem ist zwar aufgrund der objektorientierten Implementierung leicht programmatisch umsetzbar, benötigt jedoch zusätzliche Konfigurations-Screens und APIs zur Umsetzung.

Als mögliche Erweiterungen des Services und des Konzepts können einige der Vorschläge der Nutzerevaluation implementiert werden. So erweitern zusätzliche Gamification-Elemente wie Titel oder ein Aktivitätsfeed das Konzept gut und fügen sich tadellos in das bestehende Konzept ein. Die in der Evaluation bereits erwähnte Social-Media-Integration in den Service bietet ebenfalls eine sinnvolle Erweiterung. So können mehr Nutzer erreicht werden und insbesondere die Nutzergruppe der *Socializers* erhält dadurch einen weiteren Motivationsfaktor. Es ist jedoch darauf zu achten, dass Social-Media-Integrationen Nutzer auch stören können.

Um die Definition von *Processing Functions* zu vereinfachen kann der Service um die Möglichkeit einer natürlichsprachlichen Definition der Funktionen erweitert werden. Um diese natürlichsprachliche Definition zu erstellen können einerseits Tools zur Generierung von Quellcode eingesetzt werden (z.B. NatSpec[1]) oder andererseits Tools zur Abbildung von natürlicher Sprache auf Quellcode (wie z.B. behave[2]). Auf dieser Basis kann die Definition der Processing Functions weiter abstrahiert und das Hinzufügen neuer Funktionen erleichtert werden.

[1] `http://www.nat-spec.com/` (Letzter Aufruf: 18.08.2017)
[2] `https://github.com/behave/behave(LetzterAufruf:18.08.2017)`

LITERATUR

Baggelaar, H. (2008). "Evaluating Programmer Performance". In: August.

Bartle, R. a. (1999). "Players Who Suit MUDs". In: *Mud*, S. 1. URL: http://mud.co.uk/richard/hcds.htm.

Betts, B. W., J. Bal und A. W. Betts (2013). "Gamification as a tool for increasing the depth of student understanding using a collaborative e-learning environment". In: *International Journal of Continuing Engineering Education and Life Long Learning* 23.3-4, S. 213–228.

Blume, J. (2017). "Bewertung der Kodequalität in einer testgetriebenen Crowdsourcing-Plattform". unveröffentlicht.

Chou, Y.-K. (2016). "Actionable gamification: Beyond points, badges, and leaderboards". In: *Octalysis Media*, S. 1–151. URL: https://leanpub.com/actionable-gamification-beyond-points-badges-leaderboards/read.

Csikszentmihalyi, M. (2000). *Beyond boredom and anxiety.* Jossey-Bass.

Dean, J. und S. Ghemawat (2004). "MapReduce: Simplified Data Processing on Large Clusters". In: *Proceedings of 6th Symposium on Operating Systems Design and Implementation*, S. 137–149.

DeCandia, G., D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall und W. Vogels (2007). "Dynamo: Amazon's Highly Available Key-value Store". In: *Proceedings of the Symposium on Operating Systems Principles*, S. 205–220. URL: http://dl.acm.org/citation.cfm?id=1323293.1294281.

Deterding, S. (2015). "The Lens of Intrinsic Skill Atoms: A Method for Gameful Design". In: *Human-Computer Interaction* 30.3-4, S. 294–335. URL: http://dx.doi.org/10.1080/07370024.2014.993471.

Deterding, S., D. Dixon, R. Khaled und L. Nacke (2011a). "From game design elements to gamefulness". In: *Proceedings of the 15th International Academic MindTrek Conference on Envisioning Future Media Environments - MindTrek '11*, S. 9–11. URL: http://doi.acm.org/10.1145/2181037.2181040%7B%%7D5Cnhttp://dl.acm.org/citation.cfm?doid=2181037.2181040.

— (2011b). "From Game Design Elements to Gamefulness: Defining Gamification". In: *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems - CHI EA '11*, S. 2425. URL: http://portal.acm.org/citation.cfm?doid=1979742.1979575.

Dienst, P. (2017). "Konzept zur Unterstützung von Entwicklern bei der Aufgabenbearbeitung in einer testgetriebenen Crowdsourcingplattform für Softwareentwicklung". unveröffentlicht. TU Dresden.

Domínguez, A., J. Saenz-De-Navarrete, L. De-Marcos, L. Fernández-Sanz, C. Pagés und J.-J. Martínez-Herrálz (2013). "Gamifying learning experiences: Practical implications and outcomes". In: *Computers & Education* 63, S. 380–392.

Ed, F. F.-h. N. und D. Hutchison (2014). "Gamification of Education: A Review of Literature". In: *1st International Conference on HCI in Business (HCIB 2014)* 8527.February, S. –8. URL: http://link.springer.com/10.1007/978-3-319-07293-7.

Fowler, M. (2005). *Event Sourcing - Capture all changes to an application state as a sequence of events.* URL: https://martinfowler.com/eaaDev/EventSourcing.html.

Fowler, M. und K. Beck (1999). *Refactoring: improving the design of existing code*. Addison-Wesley Professional.

Goligoski, E. (2012). "Motivating the Learner : Mozilla ' s Open Badges Program". In: *Access to Knowledge* 4.1, S. 1–8.

Hamari, J. (2011). "Framework for Designing and Evaluating Game Achievements". In: *Proceedings of DiGRA 2011 Conference: Think Design Play*, S. 20. URL: http://www.mendeley.com/catalog/framework-designing-evaluating-game-achievements/.

Hamari, J. und J. Koivisto (2015). "Why do people use gamification services?" In: *International Journal of Information Management* 35.4, S. 419–431. URL: http://dx.doi.org/10.1016/j.ijinfomgt.2015.04.006.

Hamari, J., J. Koivisto und H. Sarsa (2014). "Does gamification work? - A literature review of empirical studies on gamification". In: *Proceedings of the Annual Hawaii International Conference on System Sciences*, S. 3025–3034.

Herzig, P. (2014). "Gamification as a Service - Conceptualization of a Generic Enterprise Gamification Platform". In: S. 244.

Herzig, P., M. Ameling und A. Schill (2012). "A generic platform for enterprise gamification". In: *Proceedings of the 2012 Joint Working Conference on Software Architecture and 6th European Conference on Software Architecture, WICSA/ECSA 2012*, S. 219–223.

Humble, J. und D. Farley (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation (Adobe Reader)*. Pearson Education.

Jetter, C. und J. Gerken (2006). "A simplified model of user experience for practical application". In: *The 2nd COST294-MAUSE international open workshop üser experience - towards a unified view."*, S. 106–111.

Jones, B. A., G. J. Madden und H. J. Wengreen (2014). "The FIT Game: Preliminary evaluation of a gamification approach to increasing fruit and vegetable consumption in school". In: *Preventive Medicine* 68, S. 76–79. URL: http://dx.doi.org/10.1016/j.ypmed.2014.04.015.

Kapp, K. M. (2012). "Games, gamification, and the quest for learner engagement". In: *T+ D* 66.6, S. 64–68.

Kleppmann, M. (2017). *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*. Ö'Reilly Media, Inc.".

Kolb H Fernandez E, N. R. (1999). *Webvision: The Organization of the Retina and Visual System [Internet]*. University of Utah Health Sciences Center. URL: https://www.ncbi.nlm.nih.gov/books/NBK11538/.

Miranda, E. (2011). "Time boxing planning: Buffered Moscow rules". In: *ACM SIGSOFT Software Engineering Notes* 36.6, S. 1–5.

Montola, M., T. Nummenmaa, A. Lucero, M. Boberg und H. Korhonen (2009). "Applying game achievement systems to enhance user experience in a photo sharing service". In: *Proceedings of the 13th International MindTrek Conference: Everyday Life in the Ubiquitous Era* 4.46, S. 94–97. URL: http://portal.acm.org/citation.cfm?id=1621841.1621859%7B%5C&%7Dcoll=GUIDE%7B%5C&%7Ddl=GUIDE%7B%5C&%7DCFID=103096974%7B%5C&%7DCFTOKEN=12484716.

Morschheuser, B., K. Werder, J. Hamari und J. Abe (2017). "How to gamify? A method for designing gamification". In: *Proceedings of the 50th Annual Hawaii International Conference on System Sciences (HICSS), Hawaii, USA, January 4-7, 2017* January, S. 1–10.

Niranjanamurthy, M., U. L. Archana, K. T. Niveditha, A. J. S und N. S. Shravan (2014). "The Research Study on DynamoDB - NoSQL Database Service". In: 3.10, S. 268–279.

Percival, H. (2014). *Test-Driven Development with Python: Obey the Testing Goat: Using Django, Selenium, and JavaScript*. Ö'Reilly Media, Inc.".

Schulten, M. (2014). "Gamification in der Unternehmenspraxis: Status quo und Perspektiven". In: *Dialogmarketing Perspektiven 2013/2014: Tagungsband 8. wissenschaftlicher interdisziplinärer Kongress für Dialogmarketing*. Wiesbaden: Springer Fachmedien Wiesbaden, S. 261–274. URL: https://doi.org/10.1007/978-3-658-05232-4_13.

Shore, J. u. a. (2007). *The art of agile development*. Ö'Reilly Media, Inc.".

Sonderegger, A. und J. Sauer (2010). "The influence of design aesthetics in usability testing: Effects on user performance and perceived usability". In: *Applied Ergonomics* 41.3, S. 403–410. URL: http://dx.doi.org/10.1016/j.apergo.2009.09.002.

Xu, Y. (2011). "Literature Review on Web Application Gamification and Analytics". In: *CSDL Technical Report 11-05* August, S. 1–40. URL: http://csdl-techreports.googlecode.com/svn-history/r674/trunk/techreports/11-05/11-05.pdf.

York, B. *The Best Developer Performance Metrics - What Are They?* URL: https://medium.com/@yupyork/the-best-developer-performance-metrics-6295ea8d87c0.

Zichermann, G. und C. Cunningham (2011). *Gamification by design: Implementing game mechanics in web and mobile apps*. Ö'Reilly Media, Inc.".

ABBILDUNGSVERZEICHNIS

1.1. Die Darstellung von Contributions auf Github ist ebenfalls eine Art der Gamification. Visualisiert wird die taggenaue Häufigkeit der Beiträge in Github-Projekten. 7
1.2. Die Frage- und Antwortplattform StackOverFlow bietet ebenfalls Badges/Achievements und andere Gamification-Mechaniken zur Motivation ihrer Nutzer an. 8
1.3. Relationen zwischen Gamification, Frameworks, Services, sowie Motivation und Metriken. 9

2.1. Google Suchen nach Stichwort *Gamification* der letzten 10 Jahre Quelle: `https://trends.google.de/trends/explore?date=2007-01-01%202017-08-20&q=Gamification` (Abgerufen am 20.08.2017). 11
2.2. Positionierung von Gamification zwischen Serious Games und Playful Design nach Deterding u. a. (2011b) 12
2.3. Einfluss durch Gamification (Hamari u. a. 2014) 13
2.4. Achievement-Modell nach Hamari (2011) 15
2.5. Beispielerfolg im Online-Rollenspiel *World of Warcraft* 16
2.6. Anordnung der acht Core-Drives des Octalysis Frameworks 18
2.7. Funktionsweise von OpenBadges (Bildquelle: Goligoski 2012) 21
2.8. Interface zur Regelerstellung in Bunchball Nitro (Bildquelle: `http://www.bunchball.com/products/nitro` (Abgerufen: 18.08.2017)) 22
2.9. Beispiel eines Badgeville Profils (Bildquelle: `https://badgeville.com/` (Letzter Aufruf: 15.08.2017)) 23
2.10. Eine Quest in einem Softwareprojekt auf der getbadges Plattform (Bildquelle: `https://getbadges.io/` (Letzter Aufruf: 15.08.2017)) 24

3.1. Schematische Darstellung des test-getriebenen Crowdsourcing Ansatzes der polylith-Plattform als Ergänzung des Software-Entwicklungszyklus beim polylith-Kunden 30
3.2. Prototyp des Code-Editors der Plattform mit integrierten Hilfen, wie dem Ausblenden von (für diese Aufgabe) unnötigem Quellcode Bildquelle: Dienst (2017) 31
3.3. Durch den Workflow in der polylith Plattform können Daten über den genauen Ablauf der Softwareentwicklung aufgezeichnet und verarbeitet werden. Diese werden ausgewertet und als zusätzliche Gamification-Grundlage verwendet. Die Entwicklermetriken bilden gleichzeitig eine interne Bewertungsgrundlage der Crowd-Entwickler 32

4.1. Darstellung der Funktion des Refactoring Scores (im Bild ohne Multiplikation mit 10, um eine bessere Vergleichbarkeit der Funktionen zu erhalten) zusammen mit der Gaußschen Normalverteilung bei Standardabweichung $\sigma = 1$ . . . 37
4.2. Berechnung des Levels durch Abrunden der Level-Funktion 38
4.3. Mockup-Varianten des Mini-Profils innerhalb der Plattform. 39
4.4. Mockups eines Erfolgs für die polylith-Plattform 40
4.5. Wertebereich der einzelnen Ränge für Erfolge innerhalb der Normalverteilung . 41
4.6. Mock-up von Ranglisten der Nutzer mit dem höchsten Level und der besten Lesbarkeit des aktuellen Monats 43
4.7. Schematische Darstellung des Konzepts dieser Arbeit im Kontext der polylith-Plattform. Die in der Arbeit betrachteten Teile sind grün dargestellt. 45
4.8. Schematische Darstellung des der Komponenten des entworfenen Services . 46
4.9. Asynchroner Aufruf von *Processing Functions* durch die Rohdatenbank oder eine andere Funktion 48
4.10. Asynchrone Live-Komponente innerhalb der Plattform mittels *Publish*/*Subscribe*-Pattern 49
4.11. Darstellung des Gesamtprozesses innerhalb der Plattform 50
4.12. Darstellung des Gesamtprozesses in Form eines Ablaufdiagramms. (Um die Übersichtlichkeit zu erhöhen wird nur die *Processing Function first_implemented_test* dargestellt.) 51

5.1. Admin-Backend des PGE Leaders 54
5.2. Klassenstruktur der Achievements-App 55
5.3. Klassenstruktur der Auth-App 56
5.4. Klassenstruktur der Experience-App 57
5.5. Klassenstruktur der Function Registry-App 58
5.6. Verwendete Technologien der Live-Komponente 62
5.7. Verwendete Technologien und Services zur Bereitstellung des PGE-Services . 65

6.1. Darstellung der Ergebnisse der Zeitmessungen im Testsystem. Die Dauer vom Auftreten des Events bis zum schreiben in den Kanal der Live-Komponente wird dunkelblau dargestellt, die Übertragungszeit zwischen PGE-Leader und Client über die Live-Komponente grün. Alle Werte sind in Millisekunden angegeben. 72
6.2. Screenshot des Demonstrator mit Bewertungsfenster, welches erhaltene Erfahrungspunkte in Abhängigkeit der Codequalität und einen Erfolg zeigt 75
6.3. Durchschnittliche Bewertung nach den Core-Drives von Octalysis **ohne** Gamification (*grau/dunkelgrün*) und **mit** Gamification (*grün*) 79
6.4. Farbwahrnehmung des Erfahrungspunkte-Dialogs einer Person mit Rot/Grün-Schwäche und dessen Verbesserung simuliert durch `http://www.vischeck.com/vischeck/vischeckImage.php` (Letzter Aufruf 10.08.2017) 81

TABELLENVERZEICHNIS

2.1. Vergleich von etablierten Gamification-Services anhand der aufgestellten Kriterien 25

4.1. Beispiele von Refactorings, die auf Quellcodeebene in kleinen Aufgaben erkannt werden können und vermieden werden sollten 36
4.2. Verbindung von Gamification Elementen und Metriken 44

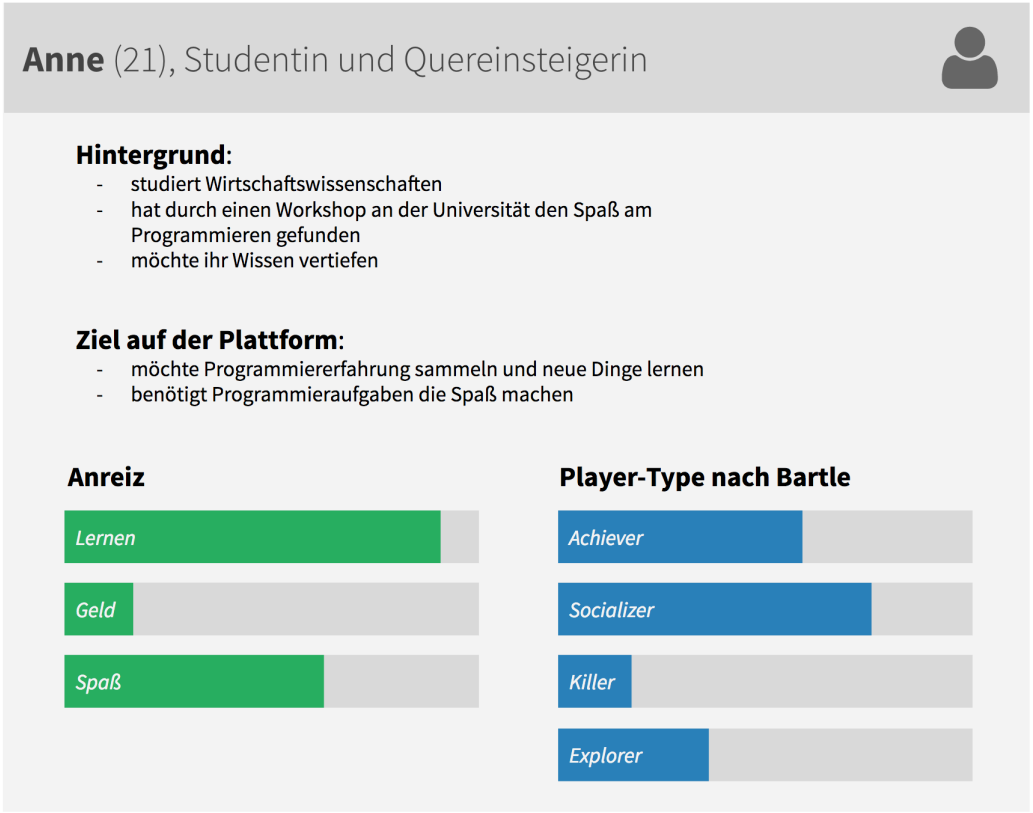
6.1. Abgeschlossene (Kreis mit Häkchen) und offene (Kreis) Must-Have Kriterien . . 68
6.2. Abgeschlossene (Kreis mit Häkchen) und offene (Kreis) Should-Have Kriterien . 69
6.3. Abgeschlossene (Kreis mit Häkchen) und offene (Kreis) Could-Have Kriterien . 70
6.4. Aktueller Stand der nicht-funktionalen Anforderungen. Graue Häkchen sind abgeschlossene, jedoch nicht evaluierte Features 70
6.5. Ergebnisse der Zeitmessungen im Testsystem. 71
6.6. Verteilung der genutzten Gamification-Elemente innerhalb der Blank Paper Evaluation. Im Gamification-Konzept enthaltene Elemente sind grün markiert. . . . 76
6.7. Verteilung der genutzten Gamification-Features innerhalb der Blank Paper Evaluation. Angedachte Features sind grün markiert. 77
6.8. Detaillierte Fragebogenergebnisse zu Erfolgen 77
6.9. Detaillierte Fragebogenergebnisse zu Ranglisten 78
6.10. Detaillierte Fragebogenergebnisse zu Erfahrungspunkten 78
6.11. Ergebnisse der einzelnen Core-Drives des Octalysis 80

QUELLCODEVERZEICHNIS

1. Beispiel einer Lambda-Funktion, die eine Scan-Abfrage innerhalb der Rohdatenbank durchführt und den PGE-Leader Server über neue Daten informiert . . 61
2. Initialisierung der Live-Komponente mit Vue.js 63
3. Anbindung der Live-Daten mit Socket.io und Vue.js 64

A. PERSONAS

Zum besseren Verständnis der Zielgruppe wurden Personas für die einzelnen Nutzerarten angefertigt. Diese sind im Folgenden dargestellt.

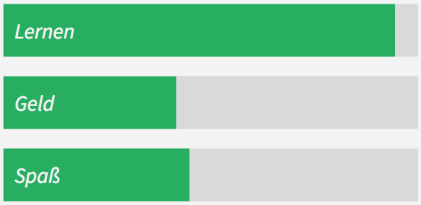


Sebastian (19), Student

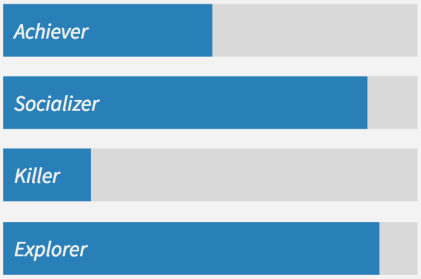
Hintergrund:
- ist im ersten Informatik-Semester
- hat sehr wenig Programmiererfahrung

Ziel auf der Plattform:
- praktische Kenntnisse sammeln

Anreiz



Player-Type nach Bartle

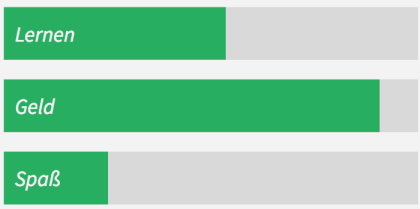


Tom (26), Student

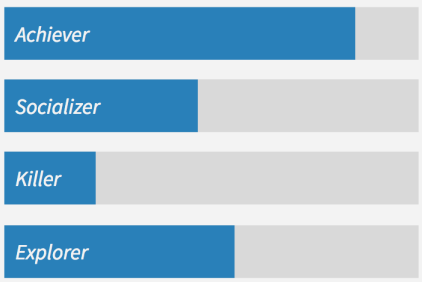
Hintergrund:
- studiert Informatik
- möchte nebenbei Geld verdienen und aus dem Studium gelerntes Wissen anwenden

Ziel auf der Plattform:
- Geld verdienen
- an echten Industrie-Aufgaben arbeiten

Anreiz



Player-Type nach Bartle



Yvonne (31), Freelancer

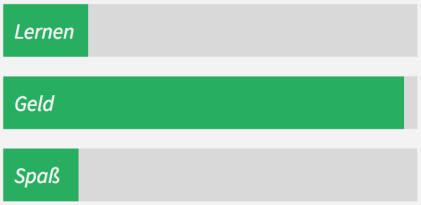
Hintergrund:

- arbeitet seit 5 Jahren freiberuflich als Entwicklerin
- möchte aufgrund geringer Auftragslage zusätzlich Geld verdienen

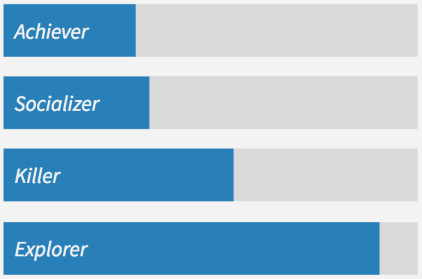
Ziel auf der Plattform:

- Geld verdienen

Anreiz



Player-Type nach Bartle



B. EVALUATIONSERGEBNISSE

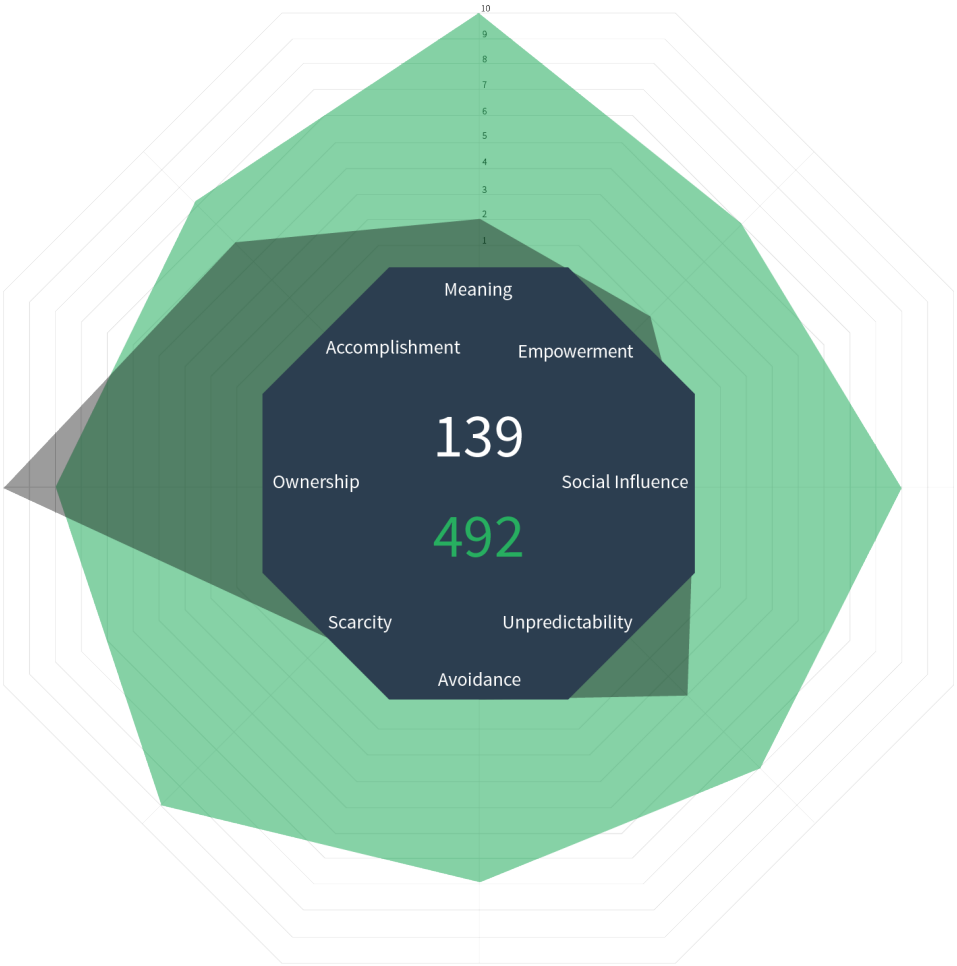
B.1. ERGEBNISSE

B.1.1. PROBAND 1

BLANK PAPER EVALUATION

Features	Metriken	Gamification Elemente
Benutzerprofil - öffentliches Profil - 3 Erfolge sollen auswählbar präsentiert werden **Social Media Integration** - Teilen von Erfolgen auf Twitter/Facebook **Kurze Feedback Zyklen** - Benutzer soll sofort über Erfolge/Erfahrungspunkte informiert werden	**sichere Abgabe wenn Aufgabe angenommen** **Kein Code Smell** - z.B. kein Switch/Case, keine nutzlosen Kommentare **zusätzliches Refactoring obwohl alle Tests grün sind** **Wartbarkeit** - Lesbarkeit (Variablenbenennung) - kleine Verschachtelungstiefe - Idiomatic Code (z.B. "*Pythonic Code*") **Erkennung von Stack-Overflow-Code**	**Erfahrungspunkte** - Umtausch gegen Gutscheine **Level** - Freischaltung von höherem Stundensatz, schwierigeren Aufgaben **Erfolge** - Benennung humorvoll **Ranglisten** - gestaffelte Liste **reale Motivation Coins für besondere Verdienste**

OCTALYSIS

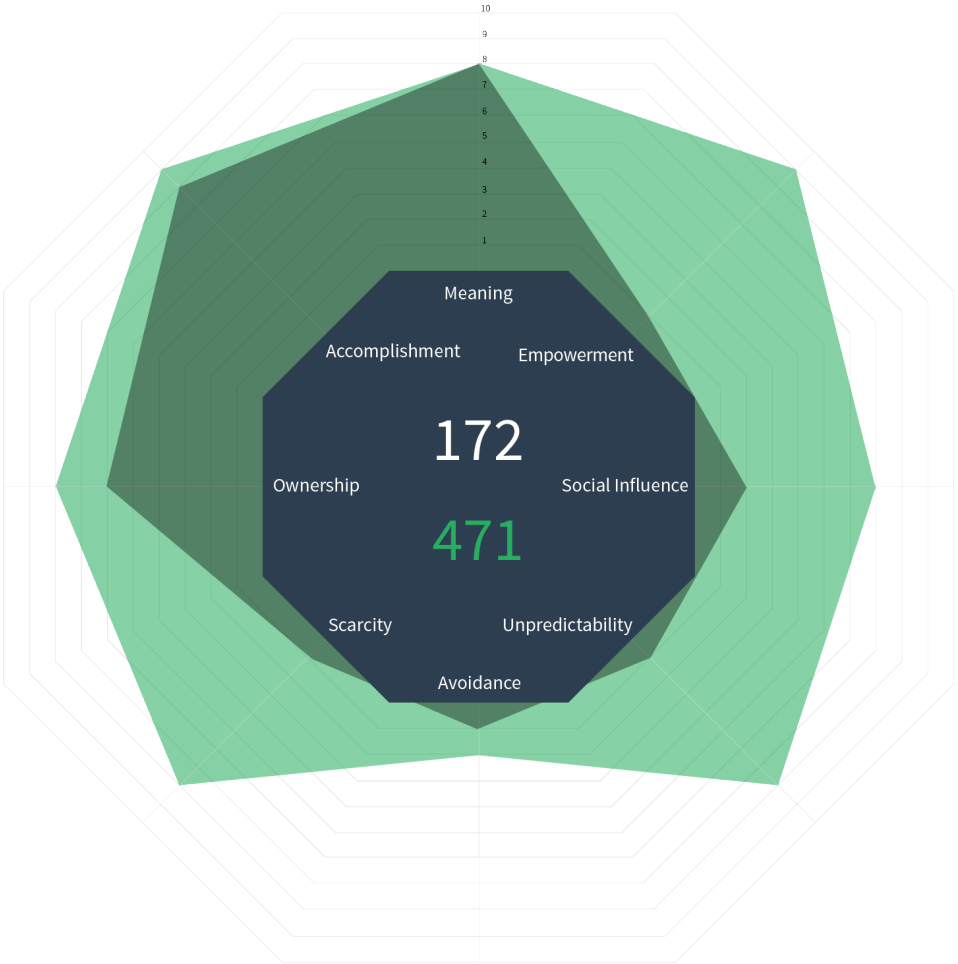


B.1.2. PROBAND 2

BLANK PAPER EVALUATION

Features	Metriken	Gamification Elemente
Privatsphären-Einstellung - Gamification, aber nicht öffentlich / nur Freunde **Social Media Integration**	**Anzahl Code-Umstrukturierung** **Schnelligkeit** **Erfüllrate** **Anzahl erfüllte Aufgaben** **Bearbeitete Aufgabentypen (Programmiersprache, Framework, ...)**	**Erfahrungspunkte** **Erfolge** **Ranglisten** - Vergleich nur innerhalb des gleichen Levels - Vergleich untereinander bei gleicher bearbeiteter Aufgabe

OCTALYSIS

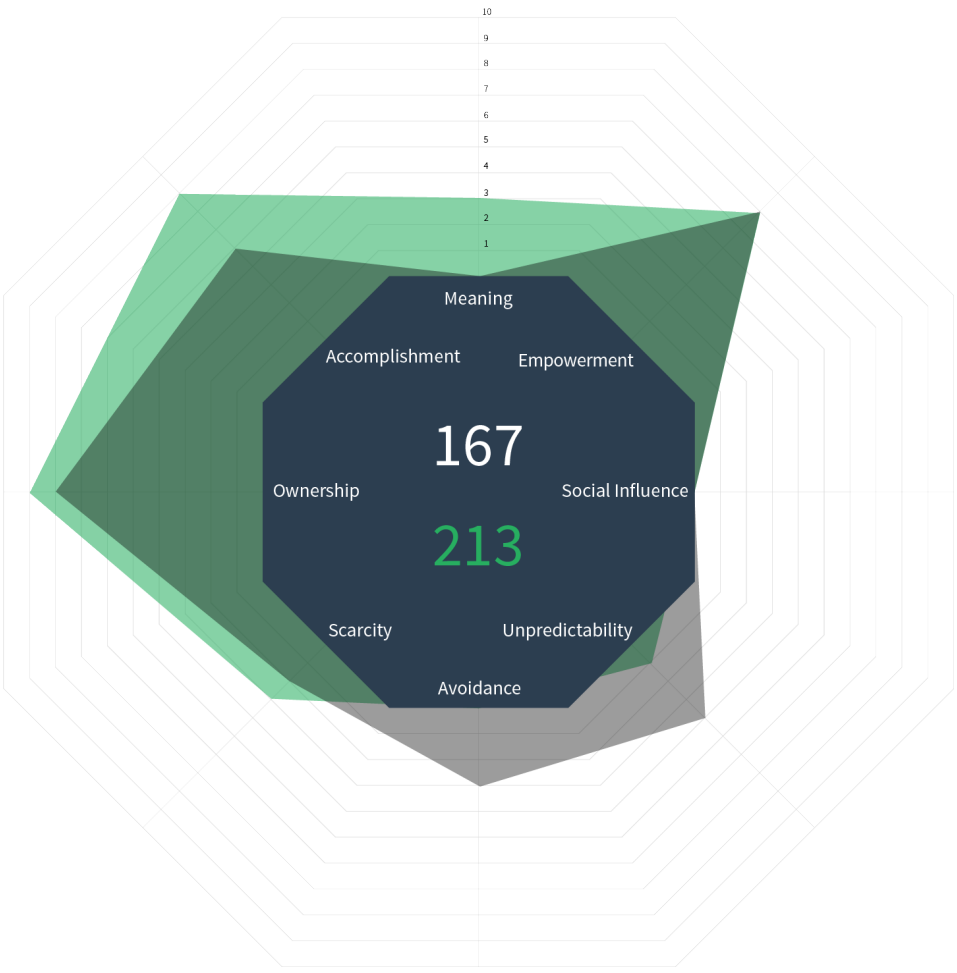


B.1.3. PROBAND 3

BLANK PAPER EVALUATION

Features	Metriken	Gamification Elemente
Schnelles Feedback	**Anzahl richtig gelöste Aufgaben**	**Geldbelohnung**
Vorhersehbare Ereignisse	**Zeit pro Aufgabe**	**Erfahrungspunkte und Level**
Gamification steuerbar - Ein/Aus/sichtbar/unsichtbar	**Code Qualität**	**Badges**
Forum für Austausch mit anderen Entwicklern	**erhaltenes Geld**	**Profil / Account**
Technischer Support und Hilfen		**Avatar mit Ausrüstung**
Tutorial-Quests		**Item Shop**
		Quests - Belohnung für Avatar

OCTALYSIS

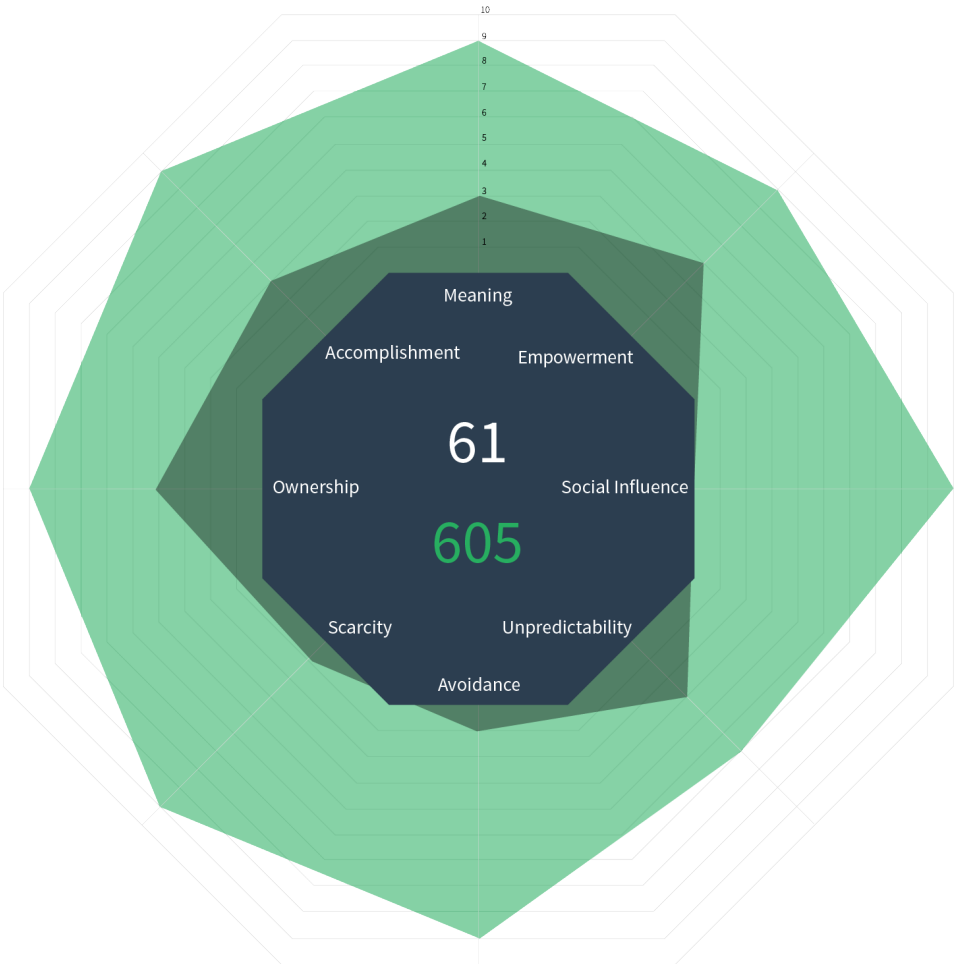


B.1.4. PROBAND 4

BLANK PAPER EVALUATION

Features	Metriken	Gamification Elemente
Timeline	**Anzahl gelöste Aufgaben**	**Anzeige für verdientes Geld**
Nutzerprofil	**Anzahl abgebrochene Aufgaben**	**Level**
Einstellung was privat, was öffentlich angezeigt wird	**Zeit/Abgabe**	**Wettkämpfe**
	Aktionen pro Minute, Code-Zeilen pro Minute	**Tägliche / wöchentliche Aufgaben**
	Qualitäts-Score	**Titel** - können auch verloren werden - Historie zeigt an welche Titel mal errungen wurden
		einfache Aufgaben mit wenig Anspruch und geringer Belohnung zum Abschalten

OCTALYSIS

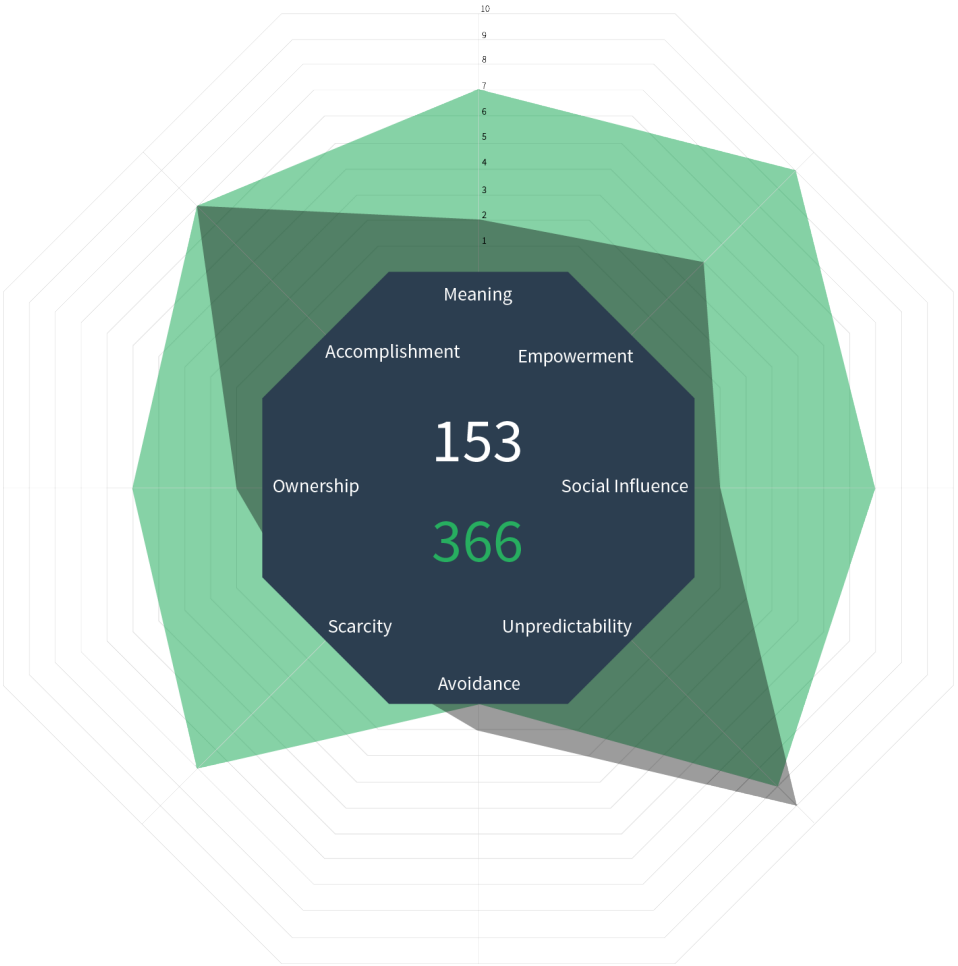


B.1.5. PROBAND 5

BLANK PAPER EVALUATION

Features	Metriken	Gamification Elemente
öffentliches Entwicklerprofil **schnelles Feedback** **Statistiken über sich selbst**	**Score aus Aktivität und abgeschlossenen Aufgaben** **Score aus Länge und Lesbarkeit des Codes** **Effizienz des Codes (z.B. Ausführungszeit)** **Gewichtete Abstimmung von anderen Entwicklern** - Gewichtung z.B. Python Entwickler haben bei Python Code höheres Gewicht **Abbrecherquote**	**Geld** **Bewertungssystem** -> Scores aus Bewertungen einzelner abgegebener Aufgaben **öffentliche Referenzen** (*Person X* hat bereits für *Unternehmen Y* gearbeitet) **themenbezogene Erfolge** (z.B. bestimmte Programmiersprache, Framework, ...) **Erstellung von Tutorial-Aufgaben als freischaltbare Belohnung**

OCTALYSIS

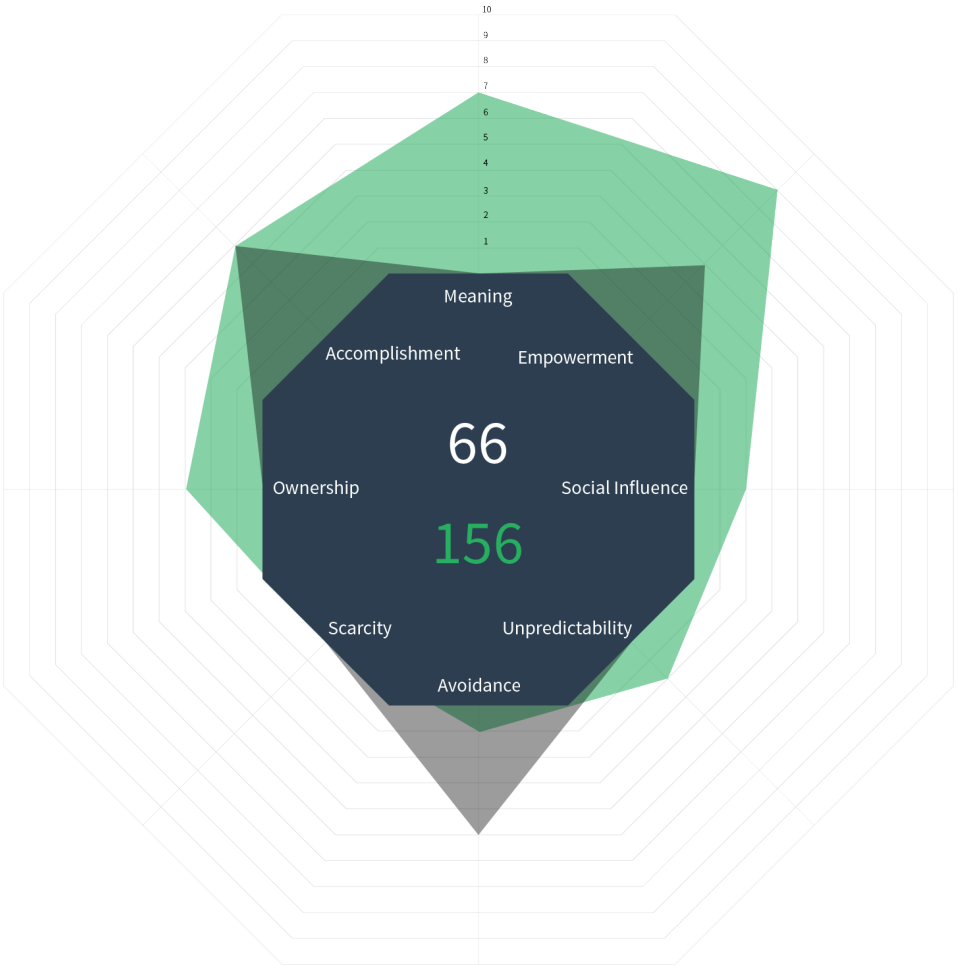


B.1.6. PROBAND 6

BLANK PAPER EVALUATION

Features	Metriken	Gamification Elemente
Transparenz - Warum welcher Platz; Welche Daten? **Statistiken** **Gimmicks (z.B. Minispiele)**	**Anzahl Versuche bis alle Tests erfüllt sind** **Meldungen von Lintern** **Lines of Code** **Lesbarkeit** **Kommentare** **Performance**	**Ranglisten** **Orden** (Erfolge) **Geldbelohnung**

OCTALYSIS

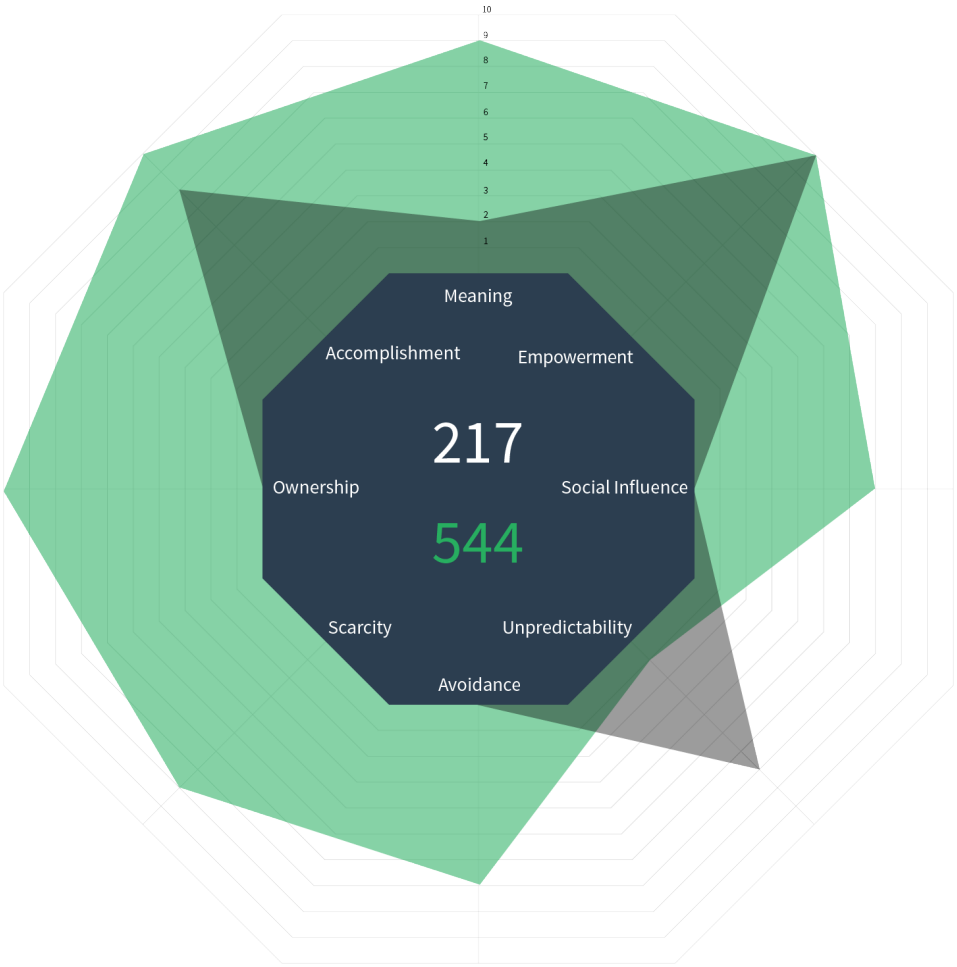


B.1.7. PROBAND 7

BLANK PAPER EVALUATION

Features	Metriken	Gamification Elemente
Statuslevel: z.B. Mentor	**Erfüllte Tests**	**Level**
Freischaltbare Boni	**Lines of Code**	**Punkte**
Privatsphären-Einstellungen	**Angenommene Lösungen/ eingereichte Lösungen**	**Ranglisten**

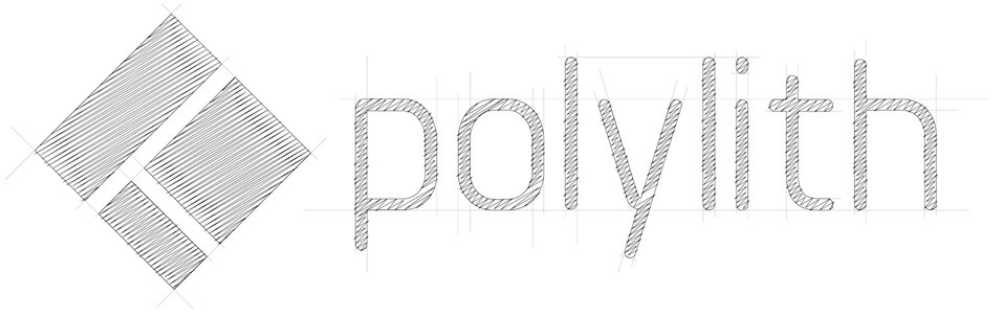
OCTALYSIS



C. VOLLSTÄNDIGER FRAGEBOGEN DER EVALUATION

C.1. ONLINE-FRAGEBOGEN

Abbildung C.1.: Erste Seite des Online-Fragebogens: Einschätzung des Editors mit Hilfen



polylith IDE & Gamification Evaluation

Seite 1

Bewertung des Editors mit Hilfen *

	stimme ich gar nicht zu				stimme ich voll und ganz zu
Ich verstehe wie der Editor funktioniert	◯	◯	◯	◯	◯
Ich fand es hilfreich, dass unrelevante Klassen ausgeblendet werden.	◯	◯	◯	◯	◯
Der Editor hat mir die Arbeit erleichtert.	◯	◯	◯	◯	◯
Ich würde den Editor nutzen um kleine Aufgaben zu entwickeln.	◯	◯	◯	◯	◯

Was ist Dein Verbesserungsvorschlag?

Bewertung der Plattform ohne Gamification

Versuche die Plattform, ohne die Erweiterung durch Gamification-Elemente (Erfolge, Erfahrungspunkte, Ranglisten, usw) aus der Sicht des Crowd-Entwicklers anhand der Kriterien des Octalysis-Systems einzuschätzen. Versuche anzugeben wie stark die einzelnen Kriterien ausgeprägt sind.

Die Plattform besteht aus einer Liste von verfügbaren, annehmbaren Aufgaben (je nach Verfügbarkeit mehr oder weniger Aufgaben). Ich kann immer jeweils eine dieser Aufgaben annehmen. Die Aufgaben sind echte Implementierungsaufgaben von Unternehmen. Für jede Aufgabe erhalte ich eine feste und sichere Bezahlung, wenn ich Quellcode schreibe, der alle Kriterien (Tests) der Aufgabe erfüllt und diesen vor dem Ende der Ausschreibungszeit abgebe. Jede Aufgabe besitzt eine Schwierigkeitsstufe (1-3 Sterne), die angibt wie kompliziert, die zu implementierende Aufgabe ist. Jede Aufgabe kann von einer bestimmten Anzahl Crowd-Entwickler angenommen werden. Die Implementierung der Aufgabe erfolgt komplett im Web-Code-Editor innerhalb der Plattform.

Epische Bedeutung und Berufung *

Es vermittelt dem Nutzer, dass er an etwas größerem teilnimmt oder das er berufen ist zu einer Sache, die nur er lösen kann.

wenig ausgeprägt stark ausgeprägt

Entwicklung und Leistung *

Dies ist der innere Antrieb, den jeder Nutzer hat, um voran zu kommen und sich zu entwickeln.

wenig ausgeprägt stark ausgeprägt

Anregung von Kreativität und Feedback *

Wird der Nutzer angeregt seine Kreativität auszuleben, neues auszuprobieren und erhält er dazu sinnvolles Feedback?

wenig ausgeprägt stark ausgeprägt

Eigentum und Besitz *

Der Nutzer möchte gerne Dinge haben und darüber entscheiden können. Das kann sowohl reale Dinge betreffen wie Geld oder Gegenstände, aber auch virtuelle Güter wie Währungen, Punkte, etc.

wenig ausgeprägt stark ausgeprägt

Sozialer Einfluss und Verbundenheit mit anderen Nutzern *

Hierbei geht es darum wie sehr ein Nutzer mit anderen interagiert. Beispielsweise, dass soziale Elemente hergestellt werden, wie. soziale Akzeptanz oder Rückmeldung, aber auch Wettbewerbe.

wenig ausgeprägt stark ausgeprägt

Abbildung C.2.: Zweite Seite des Online-Fragebogens: Bewertung der Plattform ohne Gamification

Knappheit und Ungeduld *

Ist der Antrieb etwas haben zu wollen, weil es sehr selten, exklusiv oder gegenwärtig nicht verfügbar ist.

Bsp. einen Gegenstand, einen Status

wenig ausgeprägt stark ausgeprägt

Unberechenbarkeit und Neugier *

Unverhersehbarkeit von Ereignissen, welche den Nutzer betreffen und die Neugierde beim Versuch genau solche Ereignisse auszulösen.

wenig ausgeprägt stark ausgeprägt

Verlust und Vermeidung *

Die Nutzer wollen ungern etwas verlieren, daher werden sie angetrieben dies zu vermeiden und agieren daher.

Bsp. der Verlust eines Gegenstandes, eines Status oder die Vermeidung von anderen negativen Elementen

wenig ausgeprägt stark ausgeprägt

Seite 3

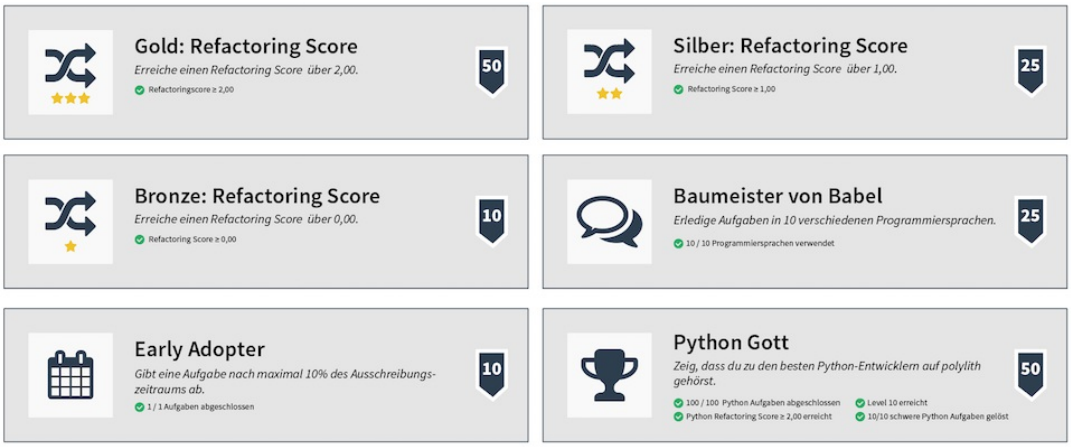
Ein Einsteiger-Erfolg auf der Plattform

Für einige Aktionen auf der Plattform erhält der Nutzer (einmalig) Erfolge. Diese drücken bestimmte Errungenschaften auf der Plattform aus. Einige Erfolge sind sichtbar, bevor man sie errungen hat, für einige müssen die Erfolgskriterien selbst gefunden werden. Für jeden Erfolg erhält der Nutzer Erfolgspunkte (im Beispiel 10). Die Nutzer mit den meisten Erfolgspunkten werden in einer Rangliste dargestellt.

Errungene Erfolge können über Soziale Medien geteilt werden. Weiterhin hat jeder Nutzer ein Profil mit allen errungenen Erfolge.

Abbildung C.3.: Dritte Seite des Online-Fragebogens: Bewertung der Plattform ohne Gamification und Vorstellung eines Erfolges

Weitere Beispiele für Erfolge



Was ist deine Meinung zu Erfolgen auf der Plattform? *

	stimme ich gar nicht zu				stimme ich voll und ganz zu
Ich verstehe was dieses Element aussagt.	◯	◯	◯	◯	◯
Ich finde dieses Element optisch ansprechend.	◯	◯	◯	◯	◯
Ich würde versuchen herauszufinden welche Erfolge es zusätzlich gibt.	◯	◯	◯	◯	◯
Ich hätte Interesse daran welche Erfolge andere Benutzer errungen haben.	◯	◯	◯	◯	◯
Ich wäre motiviert weitere Erfolge zu erringen.	◯	◯	◯	◯	◯
Ich hätte Interesse daran, Erfolge zu erringen, die keiner meiner Freunde hat.	◯	◯	◯	◯	◯
Es würde mich demotivieren, wenn ich sehen würde, dass meine Freunde Erfolge haben, die ich nicht habe.	◯	◯	◯	◯	◯

Abbildung C.4.: Vierte Seite des Fragebogens: Evaluation der Erfolge

Es würde mich besonders anspornen, wenn ich sehen würde, dass meine Freunde Erfolge haben, die ich nicht habe.	◯	◯	◯	◯	◯
Mich würde es stören, wenn ich für zu viele Aktionen Erfolge bekomme.	◯	◯	◯	◯	◯
Es würde mich stören, wenn meine errungenen Erfolge öffentlich wären.	◯	◯	◯	◯	◯
Es würde mich stören, wenn meine Freunde meine errungenen Erfolge sehen könnten.	◯	◯	◯	◯	◯
Ich finde es spannend, herauszufinden welche Erfolge es noch gibt, indem ich Dinge ausprobiere.	◯	◯	◯	◯	◯

Kommentar

Abbildung C.5.: Fünfte Seite des Fragebogens: Evaluation der Erfolge

Seite 4

Entwickler-Ranking anhand des Levels

Für viele Aktionen auf der Plattform erhält der Nutzer Erfahrungspunkte, mit durch die er ein Level aufsteigen kann. Die Nutzer mit den höchsten Leveln, den meisten Erfolgspunkten und den meisten Erfahrungspunkten innerhalb der letzten Woche/Monat/Jahr können in der Rangliste angezeigt werden.

Abbildung C.6.: Sechste Seite des Fragebogens: Evaluation der Rangliste des Gamification Konzepts

Was ist deine Meinung zu Ranglisten auf der Plattform *

	stimme ich gar nicht zu				stimme ich voll und ganz zu
Ich verstehe was dieses Element aussagt.	◯	◯	◯	◯	◯
Ich finde dieses Element optisch ansprechend.	◯	◯	◯	◯	◯
Ich würde versuchen in der Rangliste aufzusteigen.	◯	◯	◯	◯	◯
Es würde mich anspornen einen höheren Rang als meine Freunde zu haben.	◯	◯	◯	◯	◯
Es würde mich demotivieren, wenn ich sehen würde, dass meine Freunde einen höheren Rang haben als ich.	◯	◯	◯	◯	◯
Es würde mich stören, in der Rangliste aufzutauchen.	◯	◯	◯	◯	◯

Abbildung C.7.: Siebte Seite des Fragebogens: Evaluation der Rangliste des Gamification Konzepts

Seite 5

Bewertung der abgegebenen Aufgabe anhand der Code-Qualität

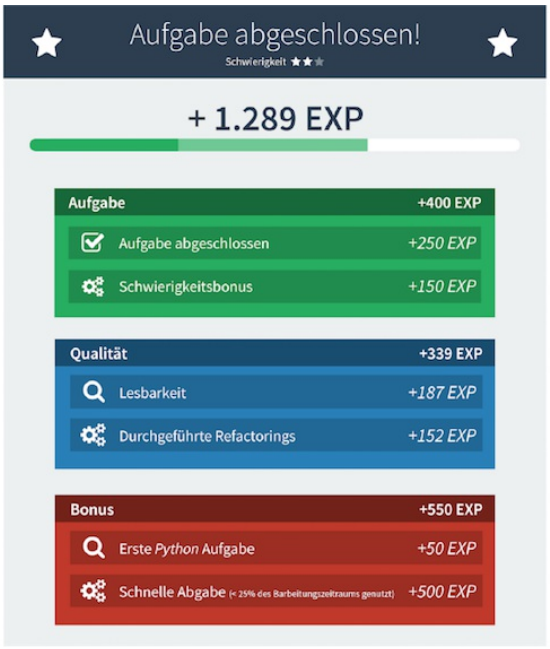


Abbildung C.8.: Achte Seite des Fragebogens: Evaluation der Erfahrungspunkte und Level

Was ist deine Meinung zu Erfahrungspunkten und der Bewertung der Aufgabe *

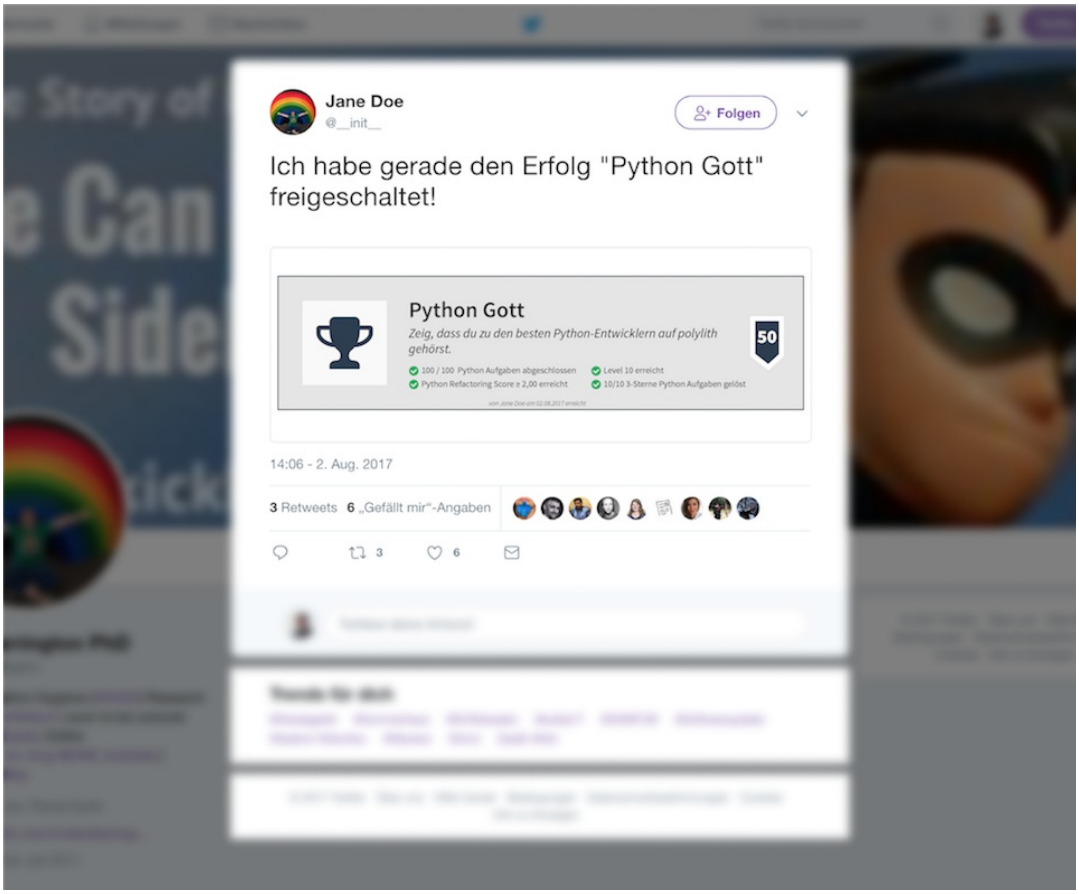
	stimme ich gar nicht zu				stimme ich voll und ganz zu
Ich verstehe was dieses Element aussagt.	◯	◯	◯	◯	◯
Ich finde dieses Element optisch ansprechend.	◯	◯	◯	◯	◯
Ich würde versuchen beim Entwickeln größeren Wert auf die Bewertungskriterien zu legen um mehr Erfahrungspunkte zu erreichen.	◯	◯	◯	◯	◯
Es würde mich demotivieren, wenn meine Freunde mehr Erfahrungspunkte hätten, als ich.	◯	◯	◯	◯	◯
Es würde mich anspornen mehr Erfahrungspunkte zu sammeln als meine Freunde.	◯	◯	◯	◯	◯
Ich habe das Gefühl, dass ich ungerecht bewertet werden könnte.	◯	◯	◯	◯	◯

Kommentar

Abbildung C.9.: Neunte Seite des Fragebogens: Evaluation der Social-Media Integration

Seite 6

Social-Media Integration



Erfolge können öffentlich über Social Media Plattformen geteilt werden.

Abbildung C.10.: Zehnte Seite des Fragebogens: Evaluation der Social-Media Integration und Bewertung der Plattform mit Gamification

Was ist deine Meinung zu Social Media Integrationen? *

	stimme ich gar nicht zu				stimme ich voll und ganz zu
Ich verstehe was dieser Tweet aussagt.	◯	◯	◯	◯	◯
Es würde mich motivieren denselben Erfolg zu erringen.	◯	◯	◯	◯	◯
Es würde mein Interesse wecken die Plattform zu nutzen.	◯	◯	◯	◯	◯
Es würde mich stören diesen Tweet zu sehen.	◯	◯	◯	◯	◯

Kommentar

Bewertung der Plattform mit den gezeigten Gamification-Elementen

Versuche die Plattform, mit der Erweiterung durch Gamification-Elemente (Erfolge, Erfahrungspunkte, Ranglisten, usw) aus der Sicht des Crowd-Entwicklers anhand der Kriterien des Octalysis-Systems einzuschätzen. Versuche anzugeben wie stark die einzelnen Kriterien ausgeprägt sind.

Zusätzlicher Kontext:
Zusätzlich zur vorher genannten Beschreibung der Plattform, erhalte ich für jede abgeschlossene Aufgabe Erfahrungspunkte und steige Level auf. Am Ende der Aufgabe sehe ich die Übersicht zur Qualität der abgegebenen Aufgabe, und erhalte abhängig davon zusätzliche Erfahrung. (wie im Beispiel Seite 5) Für einige besondere Leistungen (Leistung über einem Grenzwert z.B. besonders gute Qualität, erste Aufgabe in einer Programmiersprache oder Framework, etc.) erhalte ich Erfolge. (wie im Beispiel Seite 3) Diese Erfolge kann ich auf Social-Media Plattformen wie Twitter oder Facebook teilen. Zusätzlich kann ich mich in einer Rangliste mit anderen Nutzern auf der Plattform vergleichen, Ränge aufsteigen, aber auch wieder absteigen.

Epische Bedeutung und Berufung *

Es vermittelt dem Nutzer, dass er an etwas größerem teilnimmt oder das er berufen ist zu einer Sache, die nur er lösen kann.

wenig ausgeprägt stark ausgeprägt

Abbildung C.11.: Elfte Seite des Fragebogens: Bewertung der Plattform mit Gamification

Entwicklung und Leistung *

Dies ist der innere Antrieb, den jeder Nutzer hat, um voran zu kommen und sich zu entwickeln.

wenig ausgeprägt stark ausgeprägt

Anregung von Kreativität und Feedback *

Wird der Nutzer angeregt seine Kreativität auszuleben, neues auszuprobieren und erhält er dazu sinnvolles Feedback?

wenig ausgeprägt stark ausgeprägt

Eigentum und Besitz *

Der Nutzer möchte gerne Dinge haben und darüber entscheiden können. Das kann sowohl reale Dinge betreffen wie Geld oder Gegenstände, aber auch virtuelle Güter wie Währungen, Punkte, etc.

wenig ausgeprägt stark ausgeprägt

Sozialer Einfluss und Verbundenheit mit anderen Nutzern *

Hierbei geht es darum wie sehr ein Nutzer mit anderen interagiert. Beispielsweise, dass soziale Elemente hergestellt werden, wie. soziale Akzeptanz oder Rückmeldung, aber auch Wettbewerbe.

wenig ausgeprägt stark ausgeprägt

Knappheit und Ungeduld *

Ist der Antrieb etwas haben zu wollen, weil es sehr selten, exklusiv oder gegenwärtig nicht verfügbar ist.

Bsp. einen Gegenstand, einen Status

wenig ausgeprägt stark ausgeprägt

Unberechenbarkeit und Neugier *

Unverhersehbarkeit von Ereignissen, welche den Nutzer betreffen und die Neugierde beim Versuch genau solche Ereignisse auszulösen.

wenig ausgeprägt stark ausgeprägt

Abbildung C.12.: Zwölfte Seite des Fragebogens: Bewertung der Plattform mit Gamification

Verlust und Vermeidung *

Die Nutzer wollen ungern etwas verlieren, daher werden sie angetrieben dies zu vermeiden und agieren daher.

Bsp. der Verlust eines Gegenstandes, eines Status oder die Vermeidung von anderen negativen Elementen

wenig ausgeprägt stark ausgeprägt

Kommentar

Seite 8

Alter

Geschlecht

◯ Männlich ◯ Weiblich

Erfahrungen

	keine	wenig	mittel	gute	sehr gute
Programmiererfahrung	◯	◯	◯	◯	◯
Erfahrung mit Entwicklungsumgebungen	◯	◯	◯	◯	◯

Probandennummer *

» **Umleitung auf Schlussseite von Umfrage Online** (ändern)

Abbildung C.13.: Dreizehnte Seite des Fragebogens: Allgemeine Daten