Universität Wien

Bachelorarbeit

Design und Implementierung eines Scientific Dashboards für ein User Testing Framework

Autor:
Valon Lushaj

Betreuern:
Univ.-Prof. Dipl.-Math. Dr. Peter Reichl, M.A. St.
Univ.-Prof. Torsten Möller, PhD
Svenja Schröder, MSc

Cooperative Systems
Scientific Computing

30. 09. 2015

universität wien

Erklärung

Hiermit erkläre ich eidesstattlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die entnommenen Stellen als solche gekennzeichnet habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Wien, am 30. 09. 2015

Valon Lushaj

UNIVERSITÄT WIEN

Abstract

Fakultät für Informatik
Scientific Computing

Bachelor of Science

Design und Implementierung eines Scientific Dashboards für ein User Testing Framework

von Valon Lushaj

Im heutigen Prozess der Softwareentwicklung spielt Quality of Experience sowie Usability Testing eine große Rolle. Die Nachfrage nach Softwarepaketen, welche diese Bedürfnisse abdecken, wird immer großer. Aus einer vergangenen Bachelor-Arbeit existiert bereits ein Grundgerüst für ein derartiges Softwaresystem.

Das Ziel dieser Arbeit ist es, auf dieser Basis die Funktionalität zu erweitern und den Fokus auf die Implementierung eines des Operatoren-Dashboards zu legen. Das Dashboord wird für die Auswertung der Test-Ergebnisse benötigt, soll es aber auch ermöglichen, die Aktionen und Reaktionen des Probanden live zu verfolgen. In der vorliegenden Arbeit wird beschrieben wie das User Testing Framework entwickelt wurde. Angefangen von der Anforderungserhebung bis hin zur Evaluierung des kompletten Projekts. Die Evaluierung erfolgte mittels Usability Test.

Inhaltsverzeichnis

Erklärung i

Abstract ii

Inhaltsverzeichnis iii

Abbildungsverzeichnis v

Tabellenverzeichnis vi

Abkürzungen vii

Danksagung viii

1 Einleitung 1
- 1.1 Motivation . . . 1
- 1.2 Ziel der Arbeit . . . 1
- 1.3 Aufbau der Arbeit . . . 2

2 Grundlagen 3
- 2.1 Dashboard . . . 3
- 2.2 Quality of Experience . . . 5
- 2.3 Usability Testing . . . 6
- 2.4 Werkzeuge . . . 7
  - 2.4.1 Ruby . . . 7
  - 2.4.2 Rails . . . 7
  - 2.4.3 Bootstrap . . . 8
  - 2.4.4 Google Charts . . . 8
  - 2.4.5 The R Project for Statistical Computing . . . 9
  - 2.4.6 rinruby . . . 9

3 Entwurf 10
- 3.1 Anforderungsanalyse . . . 10
  - 3.1.1 Funktionale Anforderungen . . . 11
    - 3.1.1.1 Beschreibung der Funktionalität . . . 11
    - 3.1.1.2 Anwendungsfälle . . . 11
  - 3.1.2 Nichtfunktionale Anforderungen . . . 18

4 Realisierung **19**
4.1 Datenbankdesign . . . 19
4.2 Refactoring . . . 21
4.3 Operatoren-Verwaltung . . . 22
4.3.1 Funktionalität und das User Interface . . . 22
4.3.2 Source Code . . . 23
4.4 Dashboard Implementierung . . . 25
4.4.1 Funktionalität und UI . . . 25
4.4.2 Source Code . . . 27

5 Evaluierung **31**
5.1 Vorgang . . . 31
5.2 Aufgaben . . . 32
5.3 Ergebnisse . . . 33

6 Ähnliche Arbeiten **35**
6.1 Morae . . . 35

7 Conclusio **37**

Bibliography **38**

Abbildungsverzeichnis

3.1 Test-Ablauf Use case 12
3.2 Kampagne verwalten 13
3.3 Test-Ablauf überwachen 16

4.1 ER-Diagramm 20
4.2 Operator einlogen ansicht 22
4.3 Operator einladen ansicht 23
4.4 Operatoren ansicht 23
4.5 Dashboard 26
4.6 Test-Ablauf Ansicht 27

5.1 Demographische Daten 33
5.2 Operator einladen: Verbesserungen 34
5.3 Kampagnen Verwaltung: Verbesserungen 34

6.1 Morae Log und Chat 36
6.2 Morae Graph un Analyze 36

Tabellenverzeichnis

3.1 Use Case 1: Test-Ablauf 11
3.2 Use Case 2: Content betrachten 12
3.3 Use Case 3: Bewertung durchführen 12
3.4 Use Case 4: Kampagne aktiv setzen 13
3.5 Use Case 5: PDF Report herunterladen 13
3.6 Use Case 6: Daten exportieren 13
3.7 Use Case 7: Kampagne anlegen 14
3.8 Use Case 8: Content anlegen 14
3.9 Use Case 9: Assessment anlegen 14
3.10 Use Case 10: Kampagne bearbeiten 14
3.11 Use Case 11: Content bearbeiten 15
3.12 Use Case 12: Assessment bearbeiten 15
3.13 Use Case 13: Kampagne löschen 15
3.14 Use Case 14: Content löschen 15
3.15 Use Case 16: Test bestätigen 16
3.16 Use Case 17: Test wiederholen 16
3.17 Use Case 18: Test abbrechen 17
3.18 Use Case 19: Reihenfolge ändern 17
3.19 Use Case 20: Content überspringen 17
3.20 Use Case 21: Content wiederholen 17
3.21 Use Case 22: Proband löschen 18
3.22 Use Case 15: Assessment löschen 18

Abkürzungen

QoE	**Q**uality **o**f **E**xperience
QoS	**Q**uality **o**f **S**ervice
RoR	**R**uby **o**n **R**ails
UI	**U**ser **I**nterface
UTF	**U**ser **T**esting **F**ramework

Danksagung

Allen vorran möchte ich meinen Betreuern, Univ.-Prof. Dipl.-Math. Dr. Peter Reichl, M.A. St., Svenja Schröder, MSc sowie Dipl.-Ing. Patrick Zwickl, BSc der Forschungsgruppe Cooperative Systems für ihre Unterstützung bei diesem Projekt und für die Bereitstellung dieses Themas herzlich danken. Außerdem danke ich Thomas für die Korrektur dieser Arbeit.

Einleitung

Das erste Kapitel der Arbeit soll dem Leser die Möglichkeit geben, die Motivation der Arbeit und deren Ziele vorzustellen.

1.1 Motivation

Im heutigen Prozess der Softwareentwicklung spielt Quality of Experience sowie Usability Testing eine große Rolle. Die Nachfrage nach Softwarepaketen, welche diese Bedürfnisse abdecken wird immer großer. Es existieren bereits zahlreiche Tools für diesen Zweck am Markt, jedoch umfassen die meisten nur einen kleinen Teil des benötigten Spektrums. Viele bieten lediglich die Möglichkeit, Fragebogen zu Erstellen und diese anschließend auswertbar zu machen. Die wenigen, welche alle benötigten Funktionalitäten beinhalten sind kommerzielle Softwareprodukte, welche nicht nach Belieben erweitert und auf den Eigenbedarf angepasst werden können. Die wenigstens Produkte dieses Segments sind Open-Source, jedoch fällt es hier durch die nicht modular aufgebaute Softwarearchitektur schwer, Modifikationen vorzunehmen. Die Motivation ist klar - es muss ein modular konzipiertes Software-Framework entwickelt werden, welches out-of-the-box sämtliche benötigten Grundfunktionalitäten beinhaltet.

1.2 Ziel der Arbeit

Aus vergangenen Bachelor-Arbeit existiert bereits ein Grundgerüst für ein derartiges Softwaresystem. Die Aufgabe des Projekts und das Ziel dieser Arbeit war, auf dieser

Basis die Funktionalität zu erweitern. Außerdem muss sogenanntes Code-Refactoring durchgeführt werden, welches die Neustrukturierung von bereits vorhandenem Source-Code darstellt. Dies ist notwendig, um zukünftige Erweiterungen zu ermöglichen und zu vereinfachen. Außerdem wird hierdurch die Lesbarkeit und Verständlichkeit des Programmcodes und dessen Wartbarkeit verbessert. Eine weitere zu implementierende Funktion stellt das Dashboard dar, welches zum einen für die Auswertung der Testergebnisse benötigt wird aber auch die Möglichkeit bietet, die Aktionen und Reaktionen des Nutzers quasi in Echtzeit mitzuverfolgen und Einfluss darauf zu nehmen. Des Weiteren soll es möglich sein, die Resultate in verschiedene Dateiformate exportierbar zu machen.

1.3 Aufbau der Arbeit

Diese Arbeit ist folgendermaßen strukturiert:

- Abschnitt 2 beschäftigt sich mit den Grundlagen, welche für die Implementierung eines derartigen System benötigt werden.
- Kapitel 3 umfasst das Konzept und den Entwurf der Erweiterung der bestehenden Applikation.
- In Kapitel 4 wird die Realisierung des erweiterten Produkts erläutert.
- Abschnitt 5 beschäftigt sich mit der Evaluierung des Produkts und der Auswertung der Tests, welche zur Überprüfung der Usability durchgeführt wurden.
- Kapitel 6 beschreibt verwandte Arbeiten und stellt Produkte vor, welche über ähnliche Funktionalitäten wie das entwickelte Softwaresystem verfügen.
- Das letzte Kapitel, die Conclusio, beschreibt die durch die Entwicklung erlernten Fähigkeiten.

Grundlagen

In diesem Abschnitt werden die Grundlagen des entwickelten Dashboards erläutert und die Aspekte von Quality of Experience (QoE) beleuchtet. Außerdem folgt eine kurze Einführung in die für die Erstellung des Projekts verwendeten Werkzeugen.

2.1 Dashboard

> A dashboard is a visual display of the most important information needed to achieve one or more objectives; consolidated and arranged on a single screen so the information can be monitored at a glance. [1]

Ein Dashboard repräsentiert eine oft grafische Zusammenfassung an zahlreichen Informationen auf einem einzigen Screen (beispielsweise eine Unterseite einer Website), welches dem Nutzer beim Prozess-Monitoring unterstützen soll. Eine besondere Herausforderung bei der Konzeptionierung eines Dashboards ist es, die anzuzeigenden Informationen möglich klar und leicht interpretierbar darzustellen. Scheitert der Entwickler an dieser Aufgabe, so wird es dem User schwer fallen, die benötigten Informationen abzulesen. Im schlimmsten Fall kann es auch zur Missinterpretation der Daten kommen. Ein Dashboard ist ein Medium der Kommunikation und muss somit auf den Menschen ausgerichtet sein. Hierfür ist es wichtig zu verstehen, wie Menschen Informationen wahrnehmen und verarbeiten. Die Dashboards von vielen Produkten am Markt sind überladen und überfordern somit den Nutzer.Es ist wichtig, sich an die Design-Prinzipen der Mensch-Computer-Interaktion zu halten und bei der Planung zu berücksichtigen.

[1] Stephen Few, "Dashboard Confusion," Intelligent Enterprise, March 20, 2004

Hierfür ist das Verhältnis der verwendeten Grafiken zu Informationstexten essenziell. Grafiken zeigen Daten, sind repräsentativ, haben einen hohen Identifikationswert und im Vergleich zu Texten eine weit höhere Informationsdichte. Folgende Grundlagen sind wichtig für die Entwicklung eines effizienten Dashboards:

- Update Frequency

  Die Frequenz der Aktualisierung der angezeigten Daten stellt für die Entwicklung eines Dashboard eine wichtige Rolle dar. Ist es aufgrund von internen Strukturen beispielsweise nur möglich, die Daten des Vortages zu ermitteln, muss dies auch für den Nutzer ersichtlich sein (Beispiel: Google Analytics). Andere Applikationen arbeiten mit Echtzeitdaten und erlauben es somit, die Aktualisierung der angezeigten Informationen in kurzen, regelmäßigen Abständen durchzuführen. Hierbei ist es wichtig darauf zu achten, wie viele verschiedene Informationstypen im Dashboard eingesetzt werden.

- Abstimmung auf die Bedürfnisse der Nutzer

  Die Abstimmung auf die Bedürfnisse des Nutzers oder der Nutzergruppe spielt eine entscheindende Rolle bei der Konzeptionierung und Entwicklung des Dashboards. Je nach Anzahl der Personen, welche das Produkt nutzen werden, fällt die Umsetzung des richtigen Designs leichter oder schwerer. Soll die Applikation für eine einzelne Person entwickelt werden und stammen die Informationen, welche der Nutzer benötigt, alle aus dem selben Datensegment, so kann man durch Befragung des Users schnell und einfach die gestellten Erwartungen und Anforderungen evaluieren. Soll das Produkt jedoch für die Verwendung für eine eine Vielzahl an Personen mit verschiedenen zu erreichenden Zielen und Kenntnissen implementiert werden, so fällt das Design deutlich schwerer. In diesem Fall wäre bespielsweise die Aufteilung der Informationen in mehrere Dashboards ein Ansatz, um die Bedürfnisse sämtlicher Nutzer zu befriedigen.

- Ausrichtung auf bestimmte Geräte und Bildschirmgrößen

  Je nach Wunsch des Kunden kann das Dashboard auf eine oder mehrere verschiedene Technologien bzw. Plattformen ausgerichtet werden. Viele Dashboards werden als Applikationen für normale Computer entwickelt oder als Web-Anwendung

implementiert. Durch den Hype der Smartphones und Tablets ist die Anpassung an kleine Bildschirmauflösungen und die alternative Art der Eingabe (Touch-Screen) ebenfalls oft eine Anforderung des Kunden.
[1]

2.2 Quality of Experience

Eine zentrales Element der Qualitätswahrnehmung durch Endkunden ist, dass die meisten Anwender entweder gar nicht wissen, wie die von ihnen empfundene Leistung eins Projekts zustande kommt. Wie ein Produkt oder eine Dienstleistung vom Kunden wahrgenommen wird, entscheidet über den wirtschaftlichen Erfolg der Arbeit. Der Anbieter muss also ein zentrales Interesse daran haben, seine Investitionsmittel und Ressourcen optimal in Hinsicht der Qualitätswahrnehmung durch den Kunden einzusetzen. Dazu ist es notwendig, die Wahrnehmung des Nutzers zu messen. Dies umfasst sowohl die Metriken selbst als auch die Vorgehensweisen der Erfassung. Quality of Service und Quality of Experience sind die Begriffe, welche diese Rahmen beschreiben. In der Literatur gibt es keine einheitliche Definition der Begriffe QoE und QoS. Im Gegenteil: Es bieten selbst verschiedene internationale Standards widersprüchliche Definitionen an. [2]

Im Folgenden sollen nun Definitionen von QoE vorgestellt werden, um einen Überblick zur Thematik zu erhalten. Ein Beispiel hierfür ist die Definition der International Telecommunication Union (ITU): “Die gesamte Akzeptanz einer Anwendung oder eines Dienstes, wird durch die subjektive Wahrnehmung des Endnutzers bestimmt”.[3] Li-Yuan definiert QoE wie folgt: “Die Aufgabe der Quality of Experience (QoE) Evaluation umfasst zwei Aspekte: Die Erfahrungen der Nutzer fortwährend zu überwachen, dann den Dienst, welcher auf QoE basiert, zu kontrollieren und zu korrigieren, um sicherzustellen, dass die Qualität der Dienstleistung höchstmöglich die Anforderungen des Anwenders sicherstellt.”[4] Laut Lopez wurde “QoE als eine Erweiterung des klassischen QoS-Ansatzes definiert”. Die hier aufgeführten Definitionen tendieren dazu, QoE als Schnittstelle zwischen Benutzer und Anwendung zu sehen, welche unsere bisherige Argumentation bestätigt.” [5]

Eine allgemeingültige Definition von Quality of Experience zu finden erweist sich als schwierig, da QoE je nach Anwendungsgebiet unterschiedliche Schwerpunkte setzt.

2.3 Usability Testing

Der Begriff Usability Testing beschreibt eine Technik zur Evaluierung von Produkten oder Systemen. Aus der Sicht von Unternehmen ist Usability Testing ein Teil eines größeren Verfahrens, um die Rentabilität eines Produkts zu verbessern. Der Ursprung des Usability Testings ist das Verfahren, welches bei klassischen kontrollierten Experimenten angewendet wird.
Für die Erstellung eines effizienten Test-Designs sind folgende Punkte essentiell:

- Formulieren einer Hypothese
  Eine Hypothese stellt die Erwartungen während des Tests dar und muss möglichst prezise formuliert und erfasst werden.

- Auswahl der Teilnehmer
  Nach einer Befragung, um passende Testprobanden zu ermitteln, sollte zusätzlich eine definierte Anzahl an Nutzern mit einem Zufallsalgorithmus ausgewählt werden, welche den Test durchführen sollen.

- Strenge Kontrollen
  Die Kontrolle des Experiments ist für die Bedeutung und Gültigkeit des Ergebnisses essentiell. Deshalb ist es bei der Selektion der Teilnehmer wichtig, die Vorerfahrungen der möglichen Kandidaten zu berücksichtigen. Die Auswahl sollte anschließend so erfolgen, dass alle Teilnehmer ungefähr am selben Wissensstand sind.

- Einsatz von Kontrollgruppen
  Die Validierung des Ergebisses erfolgt durch den Einsatz von Kontrollgruppen. Hierfür legt man das Augenmerk auf die Variation einzelner Variablen.

- Gruppengröße
  Die Gruppengröße stellt einen wichtigen Parameter dar, um wesentliche Unterschiede zwischen den Gruppenteilnehmern statistisch messen zu können. Hierfür ist es wichtig, eine ausreichend große Stichprobe zu wählen. Bei der Wahl einer zu

kleinen Stichprobe fällt das Zielen von Schlüssen schwerer oder ist teilweise auch gar nicht möglich.
[6]

2.4 Werkzeuge

Folgende Werkzeuge wurden bei der Implementierung dieses Projekts eingesetzt:

2.4.1 Ruby

Ruby ist eine höhere Programmiersprache, welche im Jahre 1995 von Yukihiro Matumoto entwickelt wurde. Als Inspiration dienten die Programmiersprachen Perl, Smalltalk, Eiffel, Ada und Lisp. Anders als in anderen höheren Programmiersprachen gibt es keine primitiven Datentypen, auch einfache Werte wie beispielsweise Zahlen werden als Objekte verwendet. Die Bestandteile der Programmiersprache Ruby sind beliebig erweitertbar, deshalb zählt Ruby zur Kategorie der flexiblen Programmiersprachen. Da der Programmierer die Bestendteile der Sprache beliebig verändern kann wird Ruby als flexible Sprache angesehen. Im Gegensatz zu vielen anderen objektorientierten Sprachen unterstützt Ruby nur Einfachvererbung, definiert jedoch das Konzept der Module, welche beliebig viele Klassen und Methoden umfassen können. [7]

2.4.2 Rails

Rails (Ruby on Rails) ist ein in der Programmiersprache Ruby entwickeltes Framework für die Entwicklung von Web Applikationen. Ruby prägte die Prinzipen "Don't repeat yourself!" sowie "Convention over Configuration" und implementiert wie die meisten Web-Frameworks das MVC-Paradigma:

- Model-Layer: Der Model-Layer bildet das Backend einer Rails-Applikation, welche im Normalfall Daten in relationalen Datenbanken persistiert. Hierfür erfolgt der Zugriff auf die Daten und Modelle über den Active-Record. Momentan werden folgende Datenbanken unterstützt: DB2, Informix, Firebird, MySql, Openbase, Oracle, PostgreSQL, SQLite, MS SQL Server sowie Sybase.

- Controller-Layer: Der Controller-Layer bildet die Steuerschicht der Geschäftslogik. Diese Klassen können über sogenannte Actions mit anderen Controllern kommunizieren und Daten austauschen.

- View-Layer: Der View-Layer ist für die Präsentation der Daten zuständig. Zum aktuellen Zeitpunkt existiert für folgende Ausgabeformate die native Unterstützung von Rails: HTML, XML, JavaScript, ERB, Builder für XML, Haml, Sass sowie CoffeeScript. Will der Entwickler ein anderes Ausgabeformat benutzen, geschieht dies über die Manipulation der HTTP-Header.
  [8]

2.4.3 Bootstrap

Bootstrap ist ein Open-Source CSS-Framework. Es enthält auf HTML und CSS basierte Gestaltungsvorlagen für Typografie, Formulare, Buttons, Tabellen, Grid-Systeme sowie Navigations- und Oberflächengestaltungselemente. Zusätzlich hierzu exisitieren zahlreiche JavaScript Elemente. Seit der 2. Version des Produkts wurde der Fokus auf die möglichst einfache Implementierung vom sogenannten "Responsive Webdesign" gelegt, welches das Anpassen des Layouts und der Elemente an die jeweilige Bildschirmauflösung beschreibt. Bootstrap ist modular aufgebaut und besteht im Kern aus Less-Stylesheets, welche die einzelnen Komponenten des Frameworks beinhalten. Die JavaScript-Komponenten von Bootstrap basieren auf dem JavaScript-Framework jQuery.[9]

2.4.4 Google Charts

Die Google Chart API ist ein im Jahre 2007 veröffentlichtes Werkzeug, welches die Erstellung von Diagrammen, welche mit eigenen Daten befüllt werden können, ermöglicht. Hierfür wird eine von Google angebotene Schnittstelle verwendet, welche je nach Anfragetyp Bilder der Grafiken im PNG-Format zurückgeben. Die Unterstützung für diverse Diagrammtypen ist zum aktuellen Zeitpunkt bereits stark ausgebaut. Es werden unter anderem Linien-, Balken-, Torten-, Geo-, sowie Bardiagramme unterstützt. Google Chart war ursprünglich ein internes Projekt der Firma, um eine schnelle Einbettung von Diagrammen innerhalb von Google Anwendungen zu ermöglichen. [10]

2.4.5 The R Project for Statistical Computing

R ist eine Programmiersprache, welche für die Erstellung von Statistiken und statistischen Grafiken entwickelt wurde. R ist Teil des GNU-Projekts und auf zahlreichen Plattformen verfügbar. R gilt als die Standardsprache für statistische Problemstellungen, sowohl in der Wirtschaft als auch in der Wissenschaft.[11] R kann Daten aus den Textdateiformaten CSV sowie TXT einlesen und Ergebnisse von Berechnungen ebenso in diverse Formate exportieren. Der Export von in R erzeugten Grafiken kann in die Formate JPEG, PNG, SVG, TIFF sowie Windows Bitmap exportiert werden. Die Unterstützung für Cairo-Grafiken, PDF- und Postscript-Dateien ist ebenso vorhanden. Der Funktionsumfang von R kann durch eine Vielzahl an Paketen erweitert und an spezifische statistische Problemstellungen aus diversen Anwendungsbereichen angepasst werden. Viele Pakete können dabei direkt aus einer über die R-Konsole abrufbaren Liste ausgewählt und automatisch installiert werden. Das zentrale Archiv für diese Pakete bildet das Comprehensive R Archive Network (CRAN).

2.4.6 rinruby

RinRuby ist eine Ruby-Bibliothek, welche den R-Interpreter in Ruby integriert. Dies ermöglicht die Verwendung der Programmiersprache im eigentlichen Ruby Programmcode. Die Bibliothek umfasst lediglich ein einzelnes Ruby-Script, welches einfach zu verwenden ist und keine weiteren Anforderungen an das Programm oder den Server stellt. Die Bibliothek ist nativ in Ruby implementiert und unterstützt die Verwendung von zahlreichen Plattformen und unterschiedliche Versionen der Programmiersprache R. Auch bei der Benennung der Methoden des Pakets wurde darauf geachtet, sämtliche Funktionen so zu benennen, dass die Lesbarkeit des Programmcodes erhalten bleibt.[12]

Entwurf

In diesem Kapitel wird der Entwurf für die Realisierung der Ziele, welche in der Einleitung festgelegt wurden, näher beschrieben. Das Hauptziel dieser Arbeit ist die Implementierung einer modularen Anwendung, welche es erlaubt, Quality of Experience-Usability- sowie normale Fragebogenerhebungen durchzuführen. In Kapitel 2 wurden die wichtigsten Terminologien der Quality of Experience sowie Usability-Tests erläutert und eine Einführung in Dashboards geboten. In der Anforderungsanalyse 3.1 werden die Erfordernisse an das System zusammengefasst. Das Kapitel unterteilt sich in zwei Abschnitte, die funktionalen und nichtfunktionalen Anforderungen sowie die Anwendungsfälle und deren Beschreibung. Im Abschnitt "Daten und Privatsphäre" werden die von der Applikation gespeicherten Daten im Hinblick auf den Datenschutz genannt.

3.1 Anforderungsanalyse

Die Hauptaufgaben der zu entwickelnden User Testing Frameworks unterteilen sich in zwei größere Breiche:

- User-Bereich
  Über den User-Bereich sind die Testfragen erreichbar und können beantwortet werden
- Operator-Ansicht
  Die Operatoren-Ansicht ermöglicht das Überwachen des Tests während der Nutzer die Fragen beantwortet. Der Fokus der Arbeit gilt diesem Bereich, die Ansicht für die Nutzer sollen in einer anderen Arbeit behandelt werden.

3.1.1 Funktionale Anforderungen

Funktionale Anforderungen beschreiben, welche Vorraussetzungen das Produkt erfüllen muss und welche Aufgaben durch das Produkt behandelt werden sollen. Die Beschreibung der funktionalen Anforderungen kann entweder durch die Beschreibung der Funktionalität oder als Spezifikation der gewünschten Datenstruktur erfolgen und sind abhängig vom Produkt. [13–15]

3.1.1.1 Beschreibung der Funktionalität

Das Ziel der Arbeit ist die Implementierung eines Systems, welches die Durchführung von QoE- sowie Usablilty-Tests erlaubt. Zusätzlich soll es möglich sein, klassische Fragenbogen zu erstellen. Die Nutzer des Systems sind in zwei festgelegte Rollen unterteilt, die Test-Nutzer sowie die Operatoren. Test-Nutzer sollen Kampagnen durchführen, während Nutzer der Rolle des Operators den Bewertungsprozess überwachen sollen. Außerdem soll die Möglichkeit bestehen, gegebenenfalls während eines Tests einzugreifen. Hierfür muss es möglich sein, Inhalte umzusortieren, zu überspringen aber auch den kompletten Test abzubrechen. Das Dashboard, welches dem Operator bei der Auswertung der Ergebnisse unterstützen soll, soll diverse Evalivierungsmöglichkeiten anbieten. Die Rolle des Forschers erweitert die Möglichkeiten des Operators um die Fähigkeit, Kampagnen anzulegen, andere Operatoren einzuladen und die Rechte der Benutzer des Systems zu editieren.

3.1.1.2 Anwendungsfälle

Table 3.1: Use Case 1: Test-Ablauf

Akteur	Proband
Trigger	Formular mit demografischen Daten ausfüllen
Kurzbeschreibung	Der Anwendungsfall "Demografische Daten ausfüllen" macht es möglich, demografische Daten zu erfassen bevor Test-Phase beginnt.
Vorbedingung	Proband hat die Webapplikation gestartet.
Nachbedingung	Demografische Daten wurden erfasst und Benutzer wird zum Test weitergeleitet.

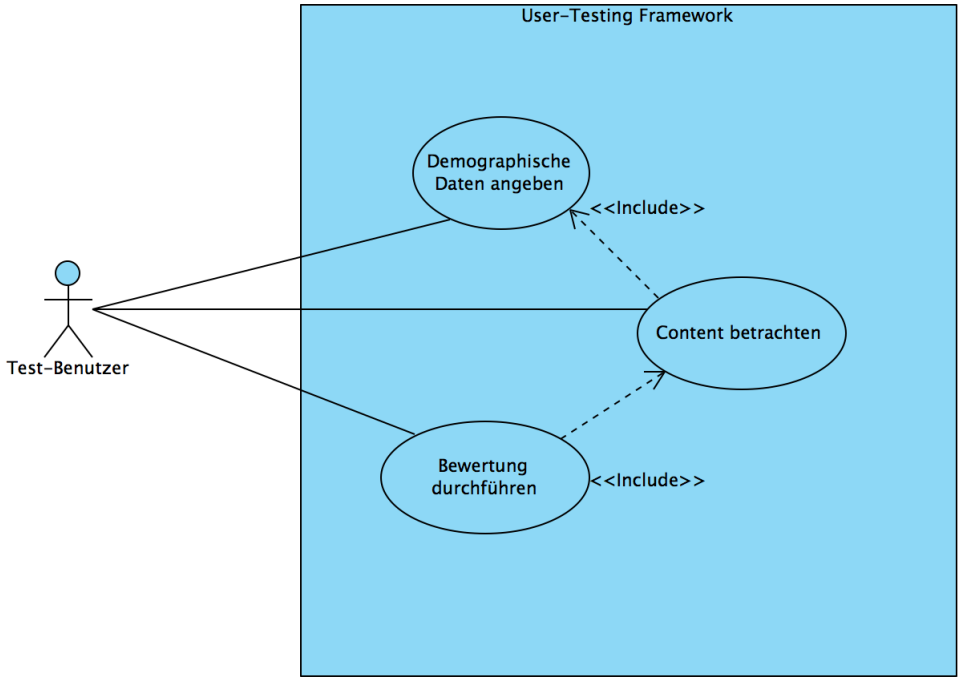

Abbildung 3.1: Test-Ablauf Use case.

Table 3.2: Use Case 2: Content betrachten

Akteur	Proband
Trigger	Demografische Daten wurden ausgefüllt und abgesendet.
Kurzbeschreibung	Der Anwendungsfall "Content betrachten" macht es möglich, Inhalte zu betrachten, bevor Fragen zu dem gezeigten Inhalt gestellt werden.
Vorbedingung	Proband hat die Test-Inhalte betrachtet. Jeder Inhalt verfügt über sogenannte Assessments, welche Aufgaben darstellen, die der Nutzer bearbeiten muss.
Nachbedingung	Der Proband hat die Inhalte betrachtet (beispielsweise in Textform oder als audiovisuelles Medium) und kann nun mit der Bearbeitung der Aufgabe fortfahren.

Table 3.3: Use Case 3: Bewertung durchführen

Akteur	Proband
Trigger	Assessments (Fragen bzw. Aufgaben) bewerten
Kurzbeschreibung	Der Anwendungsfall "Bewertung durchführen" macht es möglich, auf die gestellten Fragen zu antworten.
Vorbedingung	Der Proband hat das Fragenformular der Assessment-Bewertung betrachtet
Nachbedingung	Der Proband hat das Fragenformular der Assessments-Bewertung ausgefüllt und wird zum nächsten Inhalt weitergeleitet.

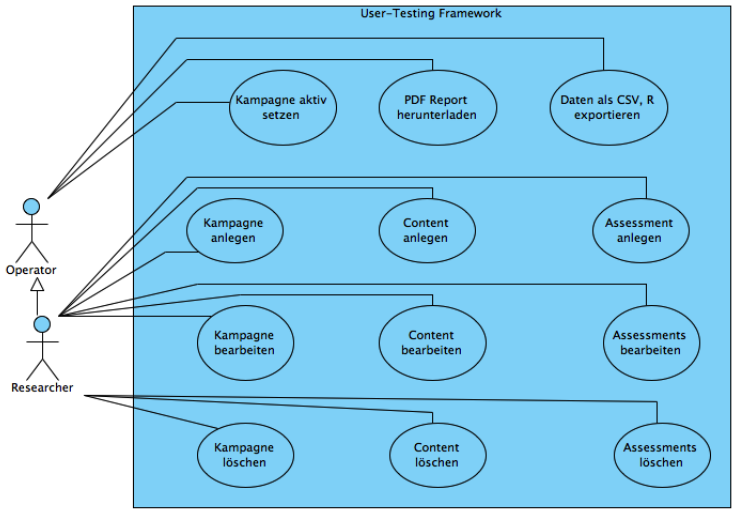

Abbildung 3.2: Kampagne verwalten.

Table 3.4: Use Case 4: Kampagne aktiv setzen

Akteur	Operator
Trigger	Klick auf Button "Kampagne aktiv setzen".
Kurzbeschreibung	Der Anwendungsfall "Kampagne aktiv setzen" macht es möglich, eine Kampagne für die Bewertung bereit zu stellen. Ruft nun ein Proband die Test-Seite auf, so erhält er die Inhalte der gerade aktiv gesetzten Kampagne. Zu einem Zeitpunkt kann immer nur eine einzige Kampagne aktiv gesetzt sein.
Vorbedingung	Der Operator muss eingeloggt sein und die zu aktivierende Kampagne muss noch inaktiv gesetzt sein.
Nachbedingung	Die Kampagne wurde aktiv gesetzt.

Table 3.5: Use Case 5: PDF Report herunterladen

Akteur	Operator
Trigger	Klick auf den Button "Report als PDF herunterladen".
Kurzbeschreibung	Der Anwendungsfall "PDF Report herunterladen" macht es möglich, einen Bericht für eine ausgewählte Kampagne zu generieren und im PDF-Format herunterzuladen.
Vorbedingung	Der Operator muss eingeloggt sein.

Table 3.6: Use Case 6: Daten exportieren

Akteur	Operator
Trigger	Klick auf Button "Daten exportieren".
Kurzbeschreibung	Der Anwendungsfall "Daten Exportieren" macht es möglich, alle bisher erfassten Daten einer Kampagne im CSV-Format und zusätzlich als R-Datei zu exportieren.
Vorbedingung	Der Operator muss eingeloggt sein.

Table 3.7: Use Case 7: Kampagne anlegen

Akteur	Operator
Trigger	Klick auf Button "Neue Kampagne anlegen".
Kurzbeschreibung	Der Anwendungsfall "Kampagne anlegen" ermöglicht das Anlegen einer neuen Kampagne.
Vorbedingung	Der Operator muss eingeloggt sein und in der Nutzergruppe der Forscher eingetragen sein.
Nachbedingung	Die angelegte Kampagne wird angezeigt.

Table 3.8: Use Case 8: Content anlegen

Akteur	Operator
Trigger	Klick auf Button "Neuen Content anlegen".
Kurzbeschreibung	Der Anwendungsfall "Content anlegen" macht es möglich, neuene Inhalte einer Kampagne zu definieren.
Vorbedingung	Der Operator muss eingeloggt sein und in der Nutzergruppe der Forscher eingetragen sein.
Nachbedingung	Der angelegte Content wird angezeigt.

Table 3.9: Use Case 9: Assessment anlegen

Akteur	Operator
Trigger	Klick auf Button "Neues Assessment anlegen".
Kurzbeschreibung	Dieser Anwendungsfall ermöglicht das Hinzufügen von Fragen / Assessments, welche einem Content (Inhalt) zugeordnet sind.
Vorbedingung	Der Operator muss eingeloggt sein und in der Nutzergruppe Forscher der eingetragen sein.
Nachbedingung	Das angelegte Assessment wird angezeigt.

Table 3.10: Use Case 10: Kampagne bearbeiten

Akteur	Operator
Trigger	Klick auf Button "Bearbeiten".
Kurzbeschreibung	Ermöglicht das Editieren der Kampagnen-Einstellungen.
Vorbedingung	Der Operator muss eingeloggt sein und in der Nutzergruppe der Forscher eingetragen sein. Des Weiteren darf zum Zeitpunkt des Bearbeitens kein Nutzer die Kampagne durchführen.
Nachbedingung	Die bearbeitete Kampagne wird angezeigt.

Table 3.11: Use Case 11: Content bearbeiten

Akteur	Operator
Trigger	Klick auf Button "Bearbeiten".
Kurzbeschreibung	Der Anwendungsfall "Content bearbeiten" macht es möglich, einen Content zu bearbeiten. Wird der zu bearbeitende Content von einem Probanden gerade eingesehen, kann keine Bearbeitung erfolgen.
Vorbedingung	Der Operator muss eingeloggt sein und die Rechte des Forschers haben. Der Content darf bisher nicht bewertet sein.
Nachbedingung	Bearbeiteten Inhalt anzeigen.

Table 3.12: Use Case 12: Assessment bearbeiten

Akteur	Operator
Trigger	Klick auf Button "Bearbeiten".
Kurzbeschreibung	Der Anwendungsfall "Assessment bearbeiten" macht es möglich, ein Assessment zu bearbeiten. Nur bisher nicht beantwortete Assessements können editiert werden.
Vorbedingung	Der Operator muss eingeloggt sein und die Rechte des Forschers innehaben. Das Assessment darf bisher nicht beantwortet sein.
Nachbedingung	Bearbeitetes Assessment anzeigen.

Table 3.13: Use Case 13: Kampagne löschen

Akteur	Operator
Trigger	Klick auf Button "Löschen".
Kurzbeschreibung	Der Anwendungsfall "Kampagne Löschen" macht es möglich, eine Kampagne komplett aus dem System zu entfernen. Damit werden alle mit der Kampagne verbundenen Inhalte wie Nutzer und deren Ergebnisse ebenfalls gelöscht.
Vorbedingung	Der Operator muss eingeloggt sein und die Rechte des Forschers haben.
Nachbedingung	Alle zugehörige Probanden und deren Bewertungen werden gelöscht.

Table 3.14: Use Case 14: Content löschen

Akteur	Operator
Trigger	Klick auf Button "Löschen".
Kurzbeschreibung	Der Anwendungsfall "Content Löschen" macht es möglich, ein Content aus der Datenbank zu löschen. Die damit verbundenen Bewertungen werden anschließend ebenfalls entfernt.
Vorbedingung	Der Operator muss eingeloggt sein und die Rechte des Forschers haben.
Nachbedingung	Dazugehörige Bewertungen werden aus dem Datenbestand entfernt.

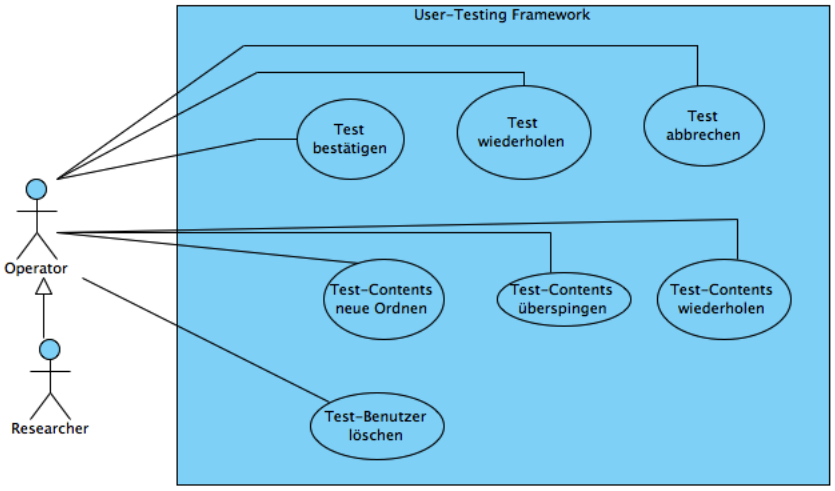

Abbildung 3.3: Test-Ablauf überwachen.

Table 3.15: Use Case 16: Test bestätigen

Akteur	Operator
Trigger	Klick auf den Button "Bestätigen".
Kurzbeschreibung	Der Anwendungsfall "Test bestätigen" macht es möglich, einen Test nach der Fertigstellung der Assessments durch den Probanden abzuschließen.
Vorbedingung	Der Operator muss eingeloggt sein, der Proband muss die Assessment-Bewertung abgeschlossen haben.
Nachbedingung	Der Proband bekommt eine Meldung, welche ihn über die Bestätigung des Tests informiert.

Table 3.16: Use Case 17: Test wiederholen

Akteur	Operator
Trigger	Klick auf Button "Wiederholen".
Kurzbeschreibung	Der Anwendungsfall "Test wiederholen" bietet dem Operator die Möglichkeit, einen Test zu wiederholen, falls es beim ersten Anlauf zu Unstimmungkeiten kam. Anschließend muss der Proband den Test wiederholen.
Vorbedingung	Der Operator muss eingeloggt sein und der Proband mit der Bewertung bereits angefangen haben.
Nachbedingung	Dazugehörige Bewertungen werden aus Datenbank entfernt.

Table 3.17: Use Case 18: Test abbrechen

Akteur	Operator
Trigger	Klick auf Button "Abbrechen".
Kurzbeschreibung	Der Anwendungsfall "Test abbrechen" bietet den Operator die Möglichkeit, einen Test abzubrechen. Hierbei werden alle Antworten des Tests gelöscht. Der Proband hat keine Möglichkeit, den Test zu wiederholen.
Vorbedingung	Der Operator muss eingeloggt sein.
Nachbedingung	Die dazugehörigen Bewertungen werden aus Datenbank entfernt.

Table 3.18: Use Case 19: Reihenfolge ändern

Akteur	Operator
Trigger	Klick auf Button "Up" oder "Down".
Kurzbeschreibung	Der Anwendungsfall "Reihenfolge ändern" macht es möglich, die Reihenfolge des Contents zu ändern. Diese Aktion kann nur von einem Operator durchgeführt werden.
Vorbedingung	Der Operator muss eingeloggt sein und der Proband soll die Bewertung bereits begonnen haben.

Table 3.19: Use Case 20: Content überspringen

Akteur	Operator
Trigger	Klick auf Button "Skip".
Kurzbeschreibung	Der Anwendungsfall "Content überspringen" macht es möglich, einen Content für einen bestimmten Probanden zu überspringen. Diese Aktion kann nur von einem Operator durchgeführt werden.
Vorbedingung	Der Operator muss eingeloggt sein und der Proband soll die Bewertung bereits begonnen haben.

Table 3.20: Use Case 21: Content wiederholen

Akteur	Operator
Trigger	Klick auf Button "Repeat".
Kurzbeschreibung	Der Anwendungsfall "Content wiederholen" macht es möglich, einen Content für einen bestimmten Probanden zu wiederholen. Um den Test abzuschließen, muss der Proband den das Assessment erneut beantworten.
Vorbedingung	Der Operator muss eingeloggt sein und der Proband soll die Bewertung bereits begonnen haben.
Nachbedingung	Die dazugehörigen Antworten werden aus dem Datenbestand entfernt.

TABLE 3.21: Use Case 22: Proband löschen

Akteur	Operator
Trigger	Klick auf Button "Löschen".
Kurzbeschreibung	Der Anwendungsfall "Proband löschen" macht es möglich, einen Proband aus der Datenbank zu entfernen und somit alle von ihm abgegebenen Bewertungen zu löschen.
Vorbedingung	Der Operator muss eingeloggt sein und der Proband soll die Bewertung bereits abgeschlossen haben.
Nachbedingung	Dazugehörigen Bewertungen aus Datenbank entfernen.

TABLE 3.22: Use Case 15: Assessment löschen

Akteur	Operator
Trigger	Klick auf Button "Löschen".
Kurzbeschreibung	Der Anwendungsfall "Assessment Löschen" macht es möglich, ein Assessment sowie die zugehörigen Bewertungen aus der Datenbank zu entfernen. Beim Durchführen dieser Aktion darf kein Test-Nutzer dieses Assessment bearbeiten.
Vorbedingung	Der Operator muss eingeloggt sein und die Rechte des Forschers haben.
Nachbedingung	Dazugehörige Bewertungen werden aus dem Datenbestand entfernt.

3.1.2 Nichtfunktionale Anforderungen

Es folgt eine kurze Definition der nichtfunktionalen Anforderungen:
Nichtfunktionale Anforderungen beschreiben die Qulität, welche die geforderte Funktionalität zusätzlich erbringen muss. Im ISO Standard 9126 wurden folgende Typen von nichtfunktionalen Anforderungen definiert: Performanz, Funktionalität, Usability, Portabilität und Sicherheit.[14, 16]
Der der Umsetzung des User Testing Frameworks lag der Fokus in der Realisierung in den beschriebenen funktionalen Anforderungen. Die Usability, Effizienz und Performance sind essentiell für den Erfolg des Produkts und als sehr wichtig einzustufen. Das Beachten sämtlicher nichtfunktionaler Anforderungen ist nicht geplant, da es nicht im Rahmen dieser Arbeit vorgesehen ist. Der Fokus soll auf die Erzeugung des Dashboards für die Operatoren liegen.

Realisierung

Das Kapitel Realisierung hat das Ziel, interessante Aspekte der Implementierung zu sammeln und diese in Kürze dem Leser zu präsentieren. Dabei sollen verschiedene Technologien, mit denen das System UTF entwickelt wurde, erläutert werden. Es wird neben den Technologien auch die Benutzeroberfläche des User Testing Framework vorgestellt und auf mögliche Benutzerinteraktionen eingegangen.

4.1 Datenbankdesign

Abbildung 4.1 zeigt ein Entity Relationship Modell, welches die Datenbankstruktur des User Testing Frameworks darstellt.

Tabelle Operator Die Entität "Operator" stellt einen User der Rolle Operator dar. Diese Tabelle beinhaltet Felder für persönliche Daten wie den Vornamen und den Nachnamen, Daten, welche für den Zugriff auf das Sytem verwendet werden (E-Mail Adresse und das Passwort in verschlüsselter Form) sowie Log-Einträge wie Informationen über dessen Einladung oder die IP-Adresse, von welcher auf den Account gerade zugegriffen wird. Außerdem existieren die Felder "created_at" sowie "updated_at", welche die Erzeugung und die letzte Aktualisierung des Eintrags darstellen und durch Rails generiert und verwaltet werden.

Tabelle Campaign Die Tabelle Campaign beinhaltet Entitäten, welche Kampagnen darstellen. Ein Element dieser Tabelle vefügt über einen Namen, eine Beschreibung, einen Zeitraum der Durchführung und eine Anzahl an Nutzern. Einer Kampagne ist eine beliebige Menge an Contents zugewiesen, welche der Kampagne angehörig sind.

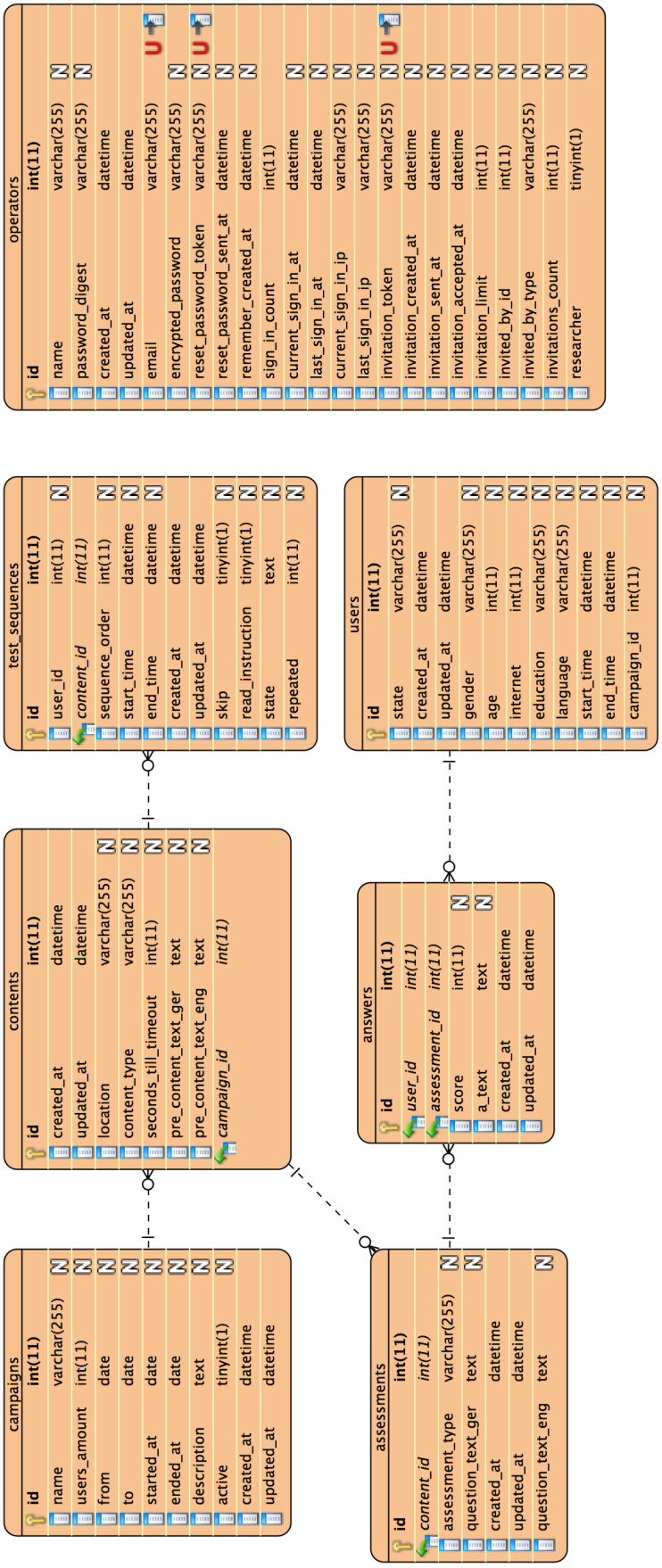

Abbildung 4.1: ER-Diagramm.

Tabelle Contents Ein Element der Tabelle Contents beinhaltet ebenfalls einen Namen. Das Feld Beschreibung existiert hier jedoch doppelt, um deutsche sowie englische Inhalte abbilden zu können. Das Feld "seconds_till_timeout" beschreibt die Laufzeit und "content_type" bezieht sich darauf, in welcher Art der Inhalt vorliegt. Einem Inhalt sind mehrere Assessments und genau eine Kampagne zugeordnet.

Tabelle Assessments Ein Assessment beinhaltet den Asessment-Typ ("assessment_type") sowie Felder für die Frage auf Deutsch sowie Englisch. Ein Asessment steht in einer Beziehnung mit einem Element der Content-Tabelle sowie mit mehreren Antworten.

Tabelle User Die Entität User stellt einen Probanden dar. Jeder Nutzer ist einer Kampagne zugeordnet und hat mehrere Tests ("test_sequences") sowie Antworten ("answers"). Diese Tabelle beinhaltet außerdem die vom User angegebenen demographischen Daten wie das Geschlecht und das Alter des Probanden. Führt der Nutzer einen Test durch, so wird die Startzeit ("start_time") sowie der Fertigstellungszeitpunkt ("end_time") ebenfalls mitgespeichert.

Tabelle TestSequence Die Entität Test-Sequence ("test_sequence") stellt einen Test dar. Jeder Test ist einem Nutzer sowie einem Content zugeordnet. Außerdem verfügt jeder Test im Feld "state" den Test-Zustand, die Aufgaben-Reihenfolge ("sequence_order") und die Start- und Abschlusszeit ("start_time", "end_time").

4.2 Refactoring

Das Refactoring ist ein wichtiger Bestandteil dieses Projekts. Hierbei wurde der komplette Source-Code neu strukturiert und das MVC-Pattern umgesetzt. MVC (Model View Controller) ist ein Entwurfsmuster mit dem Ziel eines flexiblen Programmentwurfs. Dies soll eine spätere Änderung oder Erweiterung stark vereinfachen. [17]
Andere Pattern, welche beim Refactoring des Projekts angewandt wurden, sind das "Tell Don't Ask" und das Gesetz von Demeter. Das "Tell Don't Ask" Prinzip soll daran erinnern, dass es von außerhalb eines Objekts nicht möglich ist, Fallunterscheidungen aufgrund dessen internen Zuständen zu treffen. [18] Das Gesetz von Demeter besagt, dass ein Objekt nur mit dem für das Objekt bekannten, anderen Objekten kommunizieren darf. Hierbei wird konkret definiert, welche Aktionen durch eine Methode vorgenommen werden darf. [19]

Auf der Ebene der Datenbank wurden alle Duplikate sowie nicht benötigte Spalten entfernt. Die Beziehungen zwischen Klassen wurde an die Convention von Rails angepasst. Anschließend war es möglich, die Funktionalität zu erweitern.

4.3 Operatoren-Verwaltung

Der erste Teil der Projekterweiterung beschäftigte sich mit der Entwicklung der Operatoren-Verwaltung, welche im Grund-Projekt bisher noch nicht implementiert wurde. Für die Programmierung der Module "Login" sowie "Registrierung" wurden die Rails Gems Devise sowie Devise Invitable herangezogen. Devise bietet dem Verwender eine flexible Authentifizierungslösung auf der Basis von Rails. [20] Die Erweiterung Devise Invitable erlaubt das Senden von Einladungen für die Plattform per E-Mail. [21]

4.3.1 Funktionalität und das User Interface

Zusätzlich zur Rolle des Operators existiert dessen Spezialisierung des Forschers. Forschern ist es möglich, sämtliche Funktionen der Applikation durchzuführen. Hierzu zählt das Einladen und Löschen von Operatoren sowie das Anlegen und Verwalten von Kampagnen.

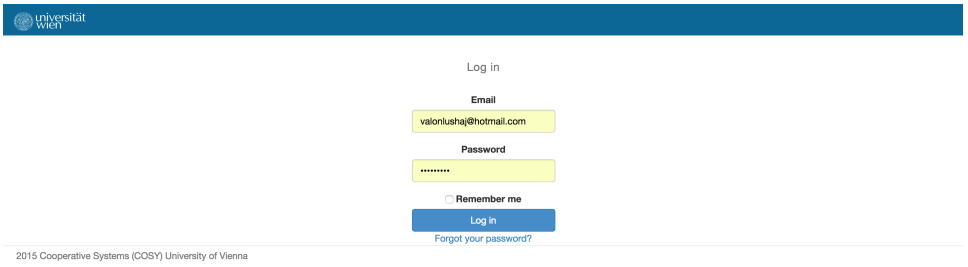

Abbildung 4.2: Operator einlogen ansicht.

Der Operator wird mit dessen E-Mail Adresse authentifiziert. Falls der Nutzer sein Passwort vergessen hat besteht die Möglichkeit, ein neues Password per E-Mail anzufordern. Dies wird von der "Passwort vergessen" Funktion durchgeführt. Hierfür gibt der Operator seine E-Mail Adresse an und erhält anschließend eine E-Mail mit einem Link zur Kennwortwiederherstellung.

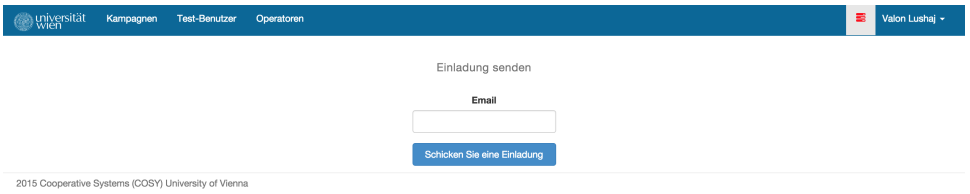

ABBILDUNG 4.3: Operator einladen ansicht.

Lediglich dem Forscher ist es möglich, andere Operatoren einzuladen. Das Anlegen von Operatoren-Accounts kann nur mittels Einladung durchgeführt werden. Lädt ein Operator einen anderen ein, so wird die E-Mail Adresse des Einzuladenden im Verwaltungspanel angegeben. Anschließend erhält auch in diesem Fall der Nutzer eine E-Mail mit einem Link, über welchen das Passwort des neuen Benutzers gesetzt werden kann. Zusätzlich hierzu wird das Angeben des Vor- sowie Nachnamens des Nutzers benötigt.

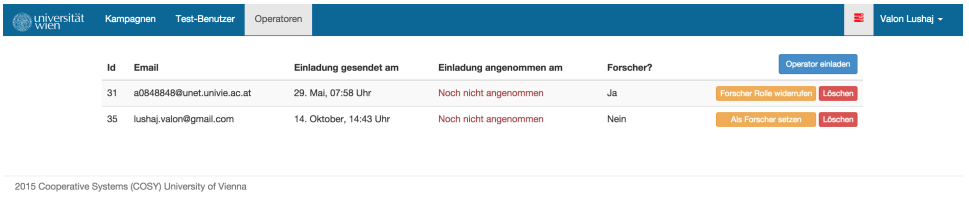

ABBILDUNG 4.4: Operatoren ansicht.

Dem Forscher ist es nach der Einladung jeder Zeit möglich zu prüfen, ob die Einladung bereits bestätigt wurde. Des Weiteren besteht die Möglichkeit, den gerade eingeladenen Nutzer bereits als Forscher festzulegen, selbst wenn dieser die Einladung noch nicht akzeptiert hat.

Um die einem Operatoren-Account zugeordnete Daten zu ändern oder den Nutzer ganz zu löschen, muss für die Durchführung das Kennwort des Nutzers eingegeben werden. Dies soll verhindern, dass es zu unabsichtlichen Änderungen oder Löschung kommt.

4.3.2 Source Code

Model:

```
class Operator < ActiveRecord::Base

  devise :invitable, :database_authenticatable, :registerable,
```

```
        :recoverable, :rememberable, :trackable, :validatable
end
```

In der Modell Klasse des Operators wurde das Ruby Gem "Devise" eingebunden. Das Gem bietet zahlreiche Funktionen an, von denen nur diejenigen verwendet werden, welche für den Use-Case relevant sind.

- Database Authenticatable: Das verschlüsselte Passwort wird in der Datenbank gespeichert.
- Registerable: Steuert die Anmeldungen des Operators sowie die Funktionen für das Bearbeiten und Löschen des Kontos.
- Recoverable: Ermöglicht das Zurücksetzen des Passworts indem es dem Nutzer eine E-Mail zur Passwortwiederherstellung zusendet.
- Validatable: Bietet die Funktionalität für das Validieren der Benutzer E-Mail Adresse sowie des Passworts.
- Trackable: Speichert für jeden Anmeldeversuch die Uhrzeit und die Zugriiffsadresse (IP-Adresse).
- Invitable: Macht es möglicht, durch das "DeviseInvitable" Gem andere Nutzer für das System einzuladen. Hierfür erhält der Nutzer per E-Mail einen Registrierungslink.

Controller: Devise besteht aus mehreren Controller-Klassen. Jede Funktionalität wie der Login oder beispielsweise die Registrierung wurde in einem eigenen Controller implementiert. Eine Veränderung für das UTF Projekt am "RegistrationsController" war notwendig, da es nur der speziellen Rolle des Forschers möglich sein soll, neue Nutzer einzuladen. Außerdem wurde die Funktion der offenen Registrierung deaktiviert, welche das Registrieren ohne Einladung ermöglichen würde. Hierfür wurde die "new" Methode angepasst und die neue Methode "is_researcher?" programmiert, welche überprüft, ob der eingeloggte Nutzer der Gruppe der Forschern angehört.

```
class Operators::RegistrationsController <
Devise::RegistrationsController
  before_filter :is_researcher?, :only => [:new, :create]

   private
    def after_update_path_for(_)
      dashboard_campaigns_path
    end

   def is_researcher?
     unless current_operator.try(:researcher?)
       flash[:error] = t('operator_invite_error')
       redirect_to dashboard_index_path
     end
   end
end
```

4.4 Dashboard Implementierung

Das Konzept sowie die Implementierung eines Scientific Dashboards war das Hauptziel dieser Arbeit. Das Dashboard soll einen Überblick über die durchgeführten Tests geben, den Test-Prozess steuerbar machen sowie den Export von Daten ermöglichen.

4.4.1 Funktionalität und UI

Authentifiziert sich ein Operator mit dem System, erfolgt die Weiterleitung auf das Dashboard der aktuellen Kampagne. Hier soll es dem Nutzer möglich sein, sich einen Überblick über den aktuellen Stand der Kampagne zu verschaffen. Beispielsweise die Information, wie viele Contents, Assessments und User der Kampagne zugeordnet sind, sind hier auf einen Blick ablesbar. Des Weiteren gibt es eine Statistik, welche Aufschluss über die demographischen Daten des Nutzers gibt. Diese Statistik wurde mit Hilfe von

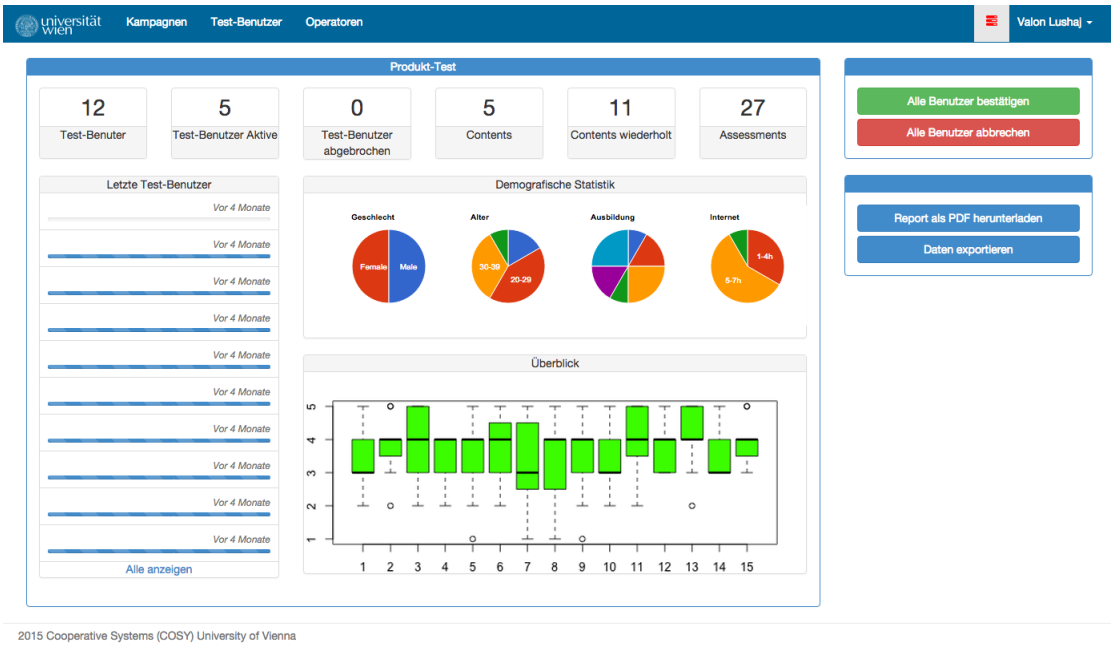

ABBILDUNG 4.5: Dashboard.

Google Charts (siehe 2.4.4) implementiert. Im Konzept der Arbeit musste die Entscheidung getroffen werden, ob eine Pie- oder eine Barchart für Daten dieser Art geeigneter wären. Die Entscheidung viel auf die Pie-Chart, da dies besser zum Gesamtkonzept und zum optischen Eindruck beiträgt.

Die Evaluierung der Kampagne ist ebenfalls im Dashboard integriert. Hierfür wurden Box-Plot Diagramme als Darstellungsart herangezogen. Der Box-Plot ist der ideale Diagrammtyp für Daten dieses Typs, da es dem Nutzer schnell und einfach ermöglicht wird, die Verteilung der Test-Daten festzustellen. [22] Da die Box-Plot Funktion, welche von Google Charts angeboten wird, ausschließlich bei einer gleichen Anzahl an Daten für ein Assessment verwendbar ist, wurde hierfür R (siehe 2.4.5) herangezogen. R selbst ist nicht web-fähig, jedoch existiert ein Ruby Gem mit dem Namen "Rinruby" (siehe 2.4.6), welche den Einsatz von R in Ruby on Rails ermöglicht. Des Weiteren gibt es eine Liste, welche die letzten Test-Nutzer chronologisch anführt. Die Information, wann der Nutzer den Test begonnen hat, ist hier ebenfalls enthalten und wird als Fortschrittsbalken zusätzlich grafisch dargestellt. Der Operator hat hier die Möglichkeit, einen gewünschten Test-Nutzer auszuwählen, um nähere Informationen über den Ablauf und Status des Testdurchlaufs zu erhalten.

Bei der Test-Ablauf-Ansicht (Abbildung 4.6) sieht der Operator, welche Frage der Nutzer

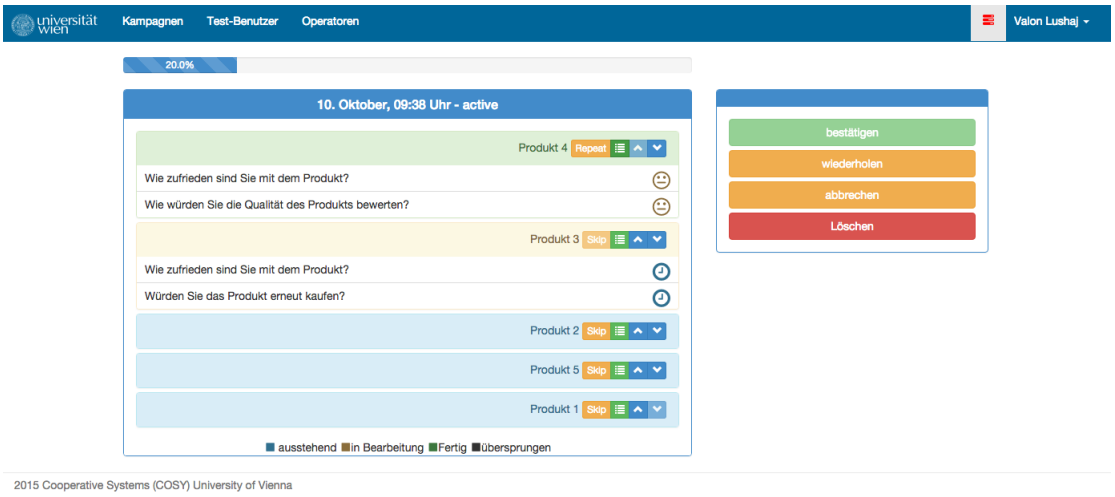

ABBILDUNG 4.6: Test-Ablauf Ansicht.

gerade bearbeitet. Je nach Farbe des Panels kann unterschieden werden, welche Fragen schon beantwortet wurden und welche Fragen noch offen sind. Die Antwort auf jede Frage kann in Form von Smilies vom Operator eingesehen werden. Das betrachten der genauen Antwort ist nicht möglich um zu vermeiden, dass der Operator auf den Test-Ablauf Einfluss nimmt. Weitere Funktionen ermöglichen das Ändern der Reihenfolge und Überspringen von Fragen. Falls der Operator feststellt, dass es beim Beantworten einer Frage Probleme oder offensichtliche Missverständisse gab, so besteht die Möglichkeit, Fragen erneut für den Nutzer freizuschalten. Das Löschen oder Abbrechen von Tests wurde ebenfalls umgesetzt. Am Ende jedes Tests muss der Operator das Test-Ergebnis validieren und bestätigen. Eine andere essentielle Funktionalität des Frameworks ist der Export von Daten. Hierfür erfolgt der Export der Daten in CSV und nach belieben zusätzlich in R. Das Generieren eines Test-Reports im PDF-Format soll es dem Operator möglich machen, genaue Analysen durchzuführen.

4.4.2 Source Code

```
$(document).ready(function () {
    refreshPartialBoxPlot()
    // will call refreshPartial every second
    setInterval(refreshPartialBoxPlot, 1000)
  });
  // calls action refreshing the partial
```

```
function refreshPartialBoxPlot() {
  $.ajax({
    url: "<%= dashboard_box_plot_path(@campaign.id) %>"
  })
}
```

Um die Aktualisierung der Daten in Echtzeit zu ermöglichen, wurde das Javascript Framework JQuery herangezogen. Jede Sekunde erfolgt die Abfrage und erneute Anzeige bestimmter Module. Im Kontext des UTF ist eine derartige Lösung valide, da zu jedem Zeitpunkt nur eine sehr beschränkte Anzahl an Nutzern das Dashboard nutzen.

```
def answers_to_csv
    @users_ids = @campaign.users.each do |user|
      user.id
    end

    @answers = Answer.where(user_id: @users_ids)

    @daten = CSV.generate do |csv|
      csv << ["user_id", "campaign_id",
                "content_id", "assessment_id", "score"]
      @answers.each do |answer|
        csv << [answer.user_id, answer.user.campaign_id,
                answer.assessment.content.id,
                answer.assessment_id, answer.score]
      end
    end

    respond_to do |format|
      format.csv {
        if params[:r]
          send_file (zip @daten, file_name.gsub(/\s+/, '_')),
          :type => 'application/zip',
```

```
                    :disposition => 'attachment',
                    :filename => "#{file_name.gsub(/\s+/, '_')}.zip"
        else
          send_data @daten, filename: "#{file_name}.csv"
        end
      }
    end
  end
```

Mit Hilfe der Ruby CSV Klasse werden die CSV Dateien generiert, welche anschließend heruntergeladen werden können. Entscheidet sich der Nutzer zusätzlich, die Daten im R Format anzufordern, so werden beide Dateien in einer ZIP-Datei gebündelt und komprimiert. Dies wurde durch die Verwendung des "Rubyzip" Gems ermöglicht.

```
# encoding: UTF-8
class CampaignReport < Prawn::Document
  include ActionView::Helpers::DateHelper

  def initialize(campaign, author)
    @campaign = campaign
    @author   = author
    super(:page_size => 'A4', :page_layout => :portrait)
  end

  def to_pdf
    first_page
    start_new_page
    header
    text I18n.t('overview'), align: :center, size: 15
    move_down 30
    text I18n.t('test_users'), size: 10
    move_down 10
    formatted_text [
```

```
          { text: "#{I18n.t('test_users')}: ", size: 8 },
          { text: " #{@campaign.users.count}", size: 12 },
          { text: "/", size: 12 },
          { text: " #{@campaign.users_amount}", size: 12 },
                    ]
    ...
    start_new_page
  ...
end
```

Der PDF-Export für den Kampagnen-Report wurde mit dem "Prawn" Gem implementiert. Prawn ist eine komplett in Ruby geschriebene Bibliothek, welche das Erzeugen von PDF Dateien ermöglicht. Andere Funktionen der Erweiterung sind das Zeichnen von Vektoren, das Einfügen von Grafiken sowie zahlreiche Funktionen im Bereich des Text-Renderings.[23]

Evaluierung

In diesem Abschnitt erfolgt eine Evaluierung des UTF Tools. Die Ergebnisse der Beurteilung sollen als Analysekriterien für die weitere Entwicklung und Verbesserung des Frameworks dienen. Hierfür mussten fünf Probanden vorgegebene Aufgaben mit Hilfe des User Testing Frameworks lösen. Bewertet wurde die Nutzerfreundlichkeit, die Effektivität und die Effizienz des Produkts. Im darauf folgenden Abschnitt wird der genaue Vorgang der Evaluierung erläutertn die zu lösenden Aufgaben vorgestellt und abschließend die Ergebnisse der Evaluierung analysiert.

5.1 Vorgang

Der Prozess der Evaluierung erfolgte in drei Phasen. Angefangen wurde mit einer Einleitung, gefolgt von der Bearbeitung der Aufgaben und schließlich das Feedback der Teilnehmer. In der ersten Phase erfolgte die Vorstellung des User Testing Frameworks und der Ablauf der Evaluierung erläutert. Bevor die Testpersonen in die nächste Phase übergingen war es gefordert, demographische Daten anzugeben.

Die zweite Phase des Tests beschäftigte sich mit der Lösung der an die Teilnehmer gestellten Aufgaben. Die Probanden wurden während des Tests beobachtet und der Click-Path sowie Auffälligkeiten wurden notiert. Von den fünf Probanden waren zwei weiblich und drei männlich. Drei Teilnehmer gaben als höchste abgeschlossene Ausbildung Matura an und zwei den Abschluss einer Hochschule.

5.2 Aufgaben

Aus dem Bereich der Operatoren-Verwaltung wurden drei Aufgaben gewählt, die restlichen Elf befassten sich mit dem Dashboard. Alle Teilnehmer bekamen die selben Fragen.

1. Laden Sie einen Operator ein.
2. Legen Sie den eingeladenen Operator als Forscher fest.
3. Löschen Sie alle bisher eingeladenen Operatoren, welche die Einladung bisher noch nicht akzeptiert habem.
4. Evaluieren Sie, welche Kampagne aktuell aktiv gesetzt ist.
5. Finden Sie heraus, wie viel Prozent der Test-Nutzer der aktuellen Kampagne weiblich sind.
6. Erzeugen Sie einen Report für die aktuell aktiv gesetzte Kampagne und laden Sie anschließend die PDF-Datei des Reports herunter.
7. Hat der letzte User, welcher die aktive Kampagne gestartet hat, bereits alle Fragen beantwortet?
8. Identifizieren Sie alle Frage Gruppen, welche sich beim Durchlauf des letzten Nutzers im Status "In Bearbeitung" oder "Übersprungen" befinden.
9. Sortieren Sie die Frage-Gruppen so, dass das aktuell letzte Element die nächste Frage ist, welche dem Nutzer gestellt wird.
10. Stellen Sie die Frage-Gruppen so ein, dass der vorletzte Task für den letzten Test-Nutzer übersprungen wird.
11. Konfigurieren Sie den Ablauf des Tests für den letzten Test-User so, dass der erste Task (der erste Content) wiederholt werden muss.
12. Bestätigen Sie das Ergebnis eines Nutzers, welcher bereits alle Fragegruppen beantwortet hat.
13. Führen Sie einen Export aller Daten einer beliebiger Kampagne im CSV sowie R-Format durch.

14. Aktivieren Sie eine beliebige Kampagne.

15. Erzeuge eine neue Kampagne und nenne sie "User-Test 1".

16. Löschen Sie einen beliebigen Test-Nutzer.

17. Melden Sie sich vom Operatoren-Bereich ab.

5.3 Ergebnisse

In Abbildung 5.1 sind die demographischen Daten zu sehen, welche von den Nutzern vor dem Test angegeben wurden.

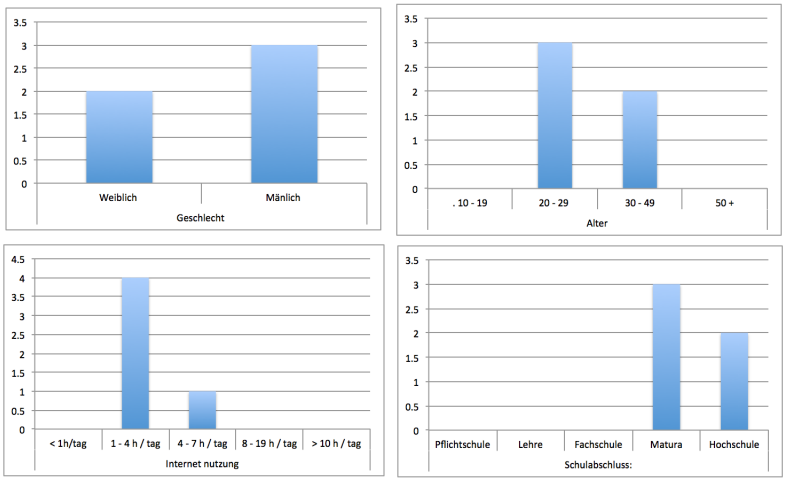

ABBILDUNG 5.1: Demographische Daten.

Von den 17 zu lösenden Aufgaben traten lediglich bei den Tasks "Operator einladen", "Kampagne erstellen" und "Kampagne aktiv setzen" Probleme für die Probanden auf. Beim Einladen des Operators wussten die Nutzer nicht, in welchem der zwei in der Applikation vorhandenen Kontext-Menüs der Funktionsaufruf zu finden ist. Auch beim Erstellen und Verwalten der Kampagnen wurde der eigentliche Sinn der Trennung in zwei Navigationen durch die Test-Nutzer missverstanden. Folgende Verbesserungen wurden nach der Analyse des Click-Paths durchgeführt:

Die Funktionen "Operator einladen" sowie "Kampagne verwalten" wurde in das andere der zwei vorhandenen Kontextmenüs, Kampagne verschoben. Um in der Ansicht der

im System vorhandenen Kampagnen besser aktive von nicht aktiven Elementen unterscheiden kann, wurde eine neue Spalte in die Tabelle eingefügt, welche die Erkennung deutlich erleichern sollte.

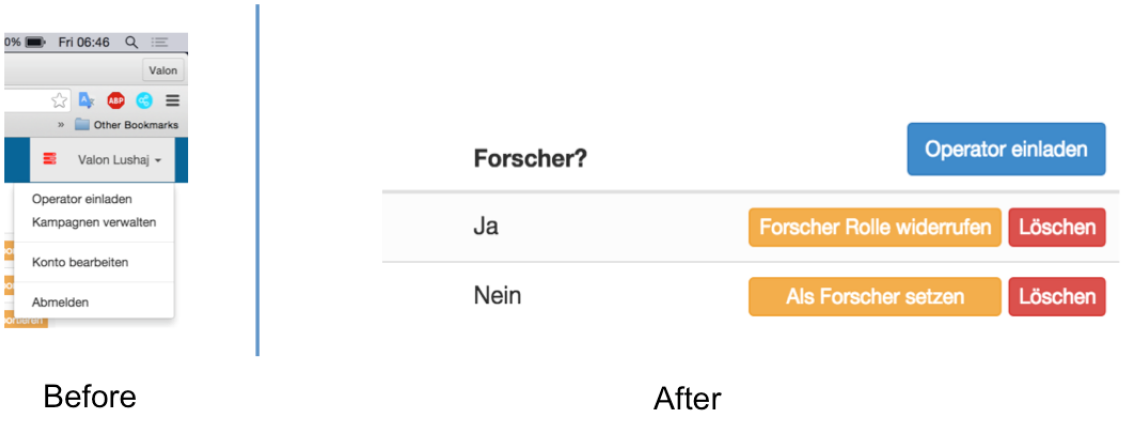

ABBILDUNG 5.2: Operator einladen: Verbesserungen

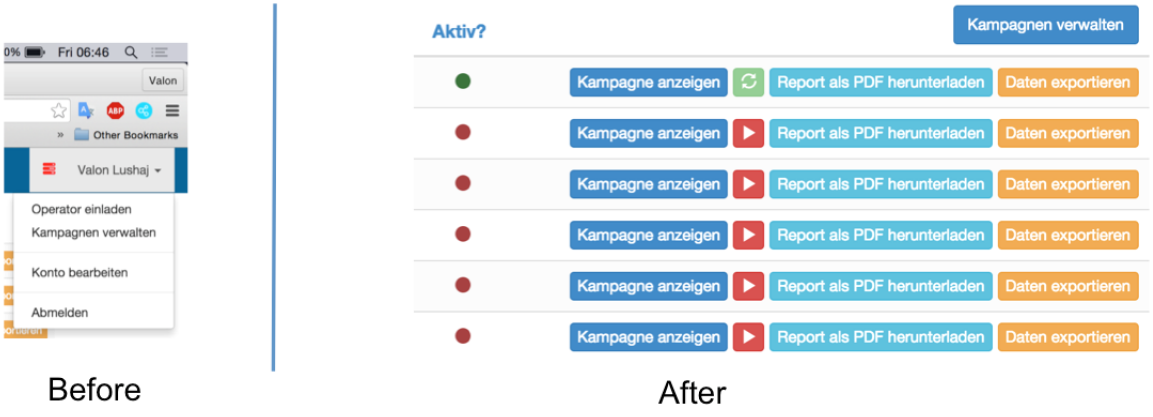

ABBILDUNG 5.3: Kampagnen Verwaltung: Verbesserungen.

Ähnliche Arbeiten

Der Abschnitt "Ähnliche Arbeiten" stellt eine Auswahl an Arbeiten vor und beschreibt die Zusammenhänge und Unterschiede zum Usability Testing Framework.

6.1 Morae

Morae ist eines der markführenden Usability-Tools für Marktforschung. Hierbei besteht die Möglichkeit, die Probanden beim Test aufzunehmen und zu beobachten. Des Weiteren ist es möglich, die durchgeführten Usability-Studien im Detail zu analysieren und Fokusgruppen, Feldstudien sowie Produkttests durchzuführen. Die Beobachtung ist vor Ort oder online möglich und es existieren Funktionen zum Nehmen von Notizen und um Präsentationen zu erstellen.

Die Funktionen in Morae sind in drei Kategorien unterteilt:
Recorder: Die Funktion des Recoders dient zum Aufzeichnen der User Tätigkeiten. Hier besteht die Möglichkeit, Audio- sowie Video Aufzeichnungen anzufertigen. Auch die Mausbewegungen sowie Tastatureingaben des Probanden können erfasst werden. Morae ist ebenso auf Kamera-Aufnahmen ausgerichtet und kann Interaktionen aufzeichnen, die nicht auf dem Bildschirm passieren. Beispiele hierfür sind Fokusgruppen, Interviews sowie Hardware-Tests, für welche Webcams oder normale Digitalkameras an den Computer angeschlossen werden. **Observer:** Die Observer Funktionalität macht es möglich, die Aktionen des Test-Nutzers quasi live zu verfolgen, Notizen anzulegen und Tasks mit Markern zu versehen. Die integrierte Chat Funktion erlaubt die Kommunikation mit dem Probanden während des Tests. Hierbei unterstützt Morae den Operator

ABBILDUNG 6.1: Morae Log und Chat

dabei, strukturiertes sowie detailliertes Feedback zu erzeugen, welches zu konfigurierbaren Stellen beispielsweise in einem Video angezeigt werden können.

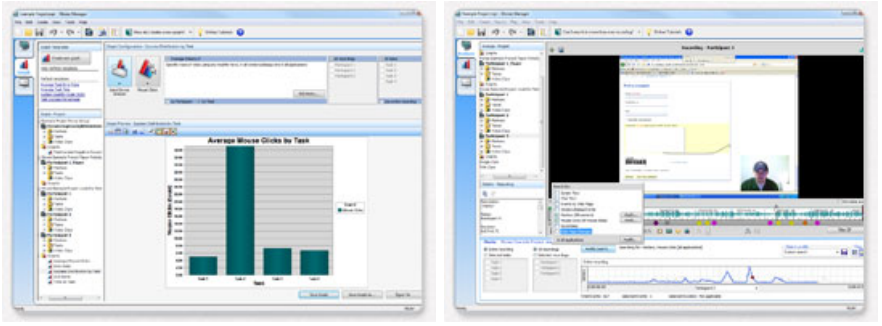

ABBILDUNG 6.2: Morae Graph und Analyze

Manager-Modul: Das Manager-Modul, welches zur Analyse und Auswertung der Test-Daten dient, erlaubt das automatische Erzeugen von Tabellen sowie Grafiken. Standardwerte wie Effektivität, Wirkungsgrad sowie die gesamte Kundenzufriendenheit werden ohne Zutun des Operators berechnet. Es existiert die Unterstützung für die Generierung von eigenen Grafiken und Statistiken, wofür beispielsweise die Mausbewegungen, Klicks, der Zeitaufwand, Markierungen oder die Ergebnisse von Umfragen als Datenbasis herangezogen werden. Das Einbetten von Videos, Grafiken sowie Titelclips in PowerPoint Präsentationen wird nativ von Morae angeboten. Außerdem existiert ein Add-In für Word, welches das Übernehmen von Grafiken und Screenshots, Datentabellen und Fragen aus Umfragen erlaubt. [24]

Conclusio

In dieser Bachelorarbeit wurde erfolgreich ein User Testing Framework weiterentwickelt, wobei der Fokus auf das Design und der Implmentierung eines Scientific Dashboards gelegt wurde. Nach einer kurzen Einleitung und den Grundlagen der Arbeit wurden die Anforderungen im Kapitel Entwurf analysiert und erhoben. Die detaillierte Aufstellung der benötigten Funktionen sind in Abschnitt 3.1.1.2 in Form von Use-Case Diagrammen aufgelistet. Im vierten Kapitel, Realisierung, wurde sämtliche interesannte Stellen im programmierten Sourcecode erläutert. Um das erstellte Produkt zu testen, wurde in Kapitel 5 eine Evaluierung durchgeführt. Hierzu erhielten fünf Probanden Aufgaben und Fragen, welche neue Erkenntnisse für die Verbesserung der Benutzerfreundlichkeit, Effektiviät und Effizient der Software liefern sollten und anschließend umgesetzt wurden. Nach der Erweiterung und Adaptierung des Projekts wurde ein erneuter Test durchgeführt, um die getätigten Änderung messbar zu machen. Auch wenn sich die beiden Ergebnisse nur geringfügig von einander unterscheiden wurde hiermit gezeigt, dass die Arbeit als funktional und ausbaufähig bezeichnet werden kann.

Bibliography

[1] Stephen Few. *Information Dashboard Design: The Effective Visual Communication of Data*. O'Reilly & Associates, Beijing ; Cambride MA, 1 edition, February 2006. ISBN 9780596100162.

[2] Wolfgang Balzer. *Quality of Experience und Quality of Service im Mobilkommunikationsbereich: Von den grundlegenden Konzepten zur praktischen Umsetzung*. Springer-Verlag, May 2015. ISBN 978-3-642-55348-6.

[3] Definition of quality of experience (qoe).

[4] Liu Li-yuan, Zhou Wen-an, and Song Jun-de. The research of quality of experience evaluation method in pervasive computing environment. In *Pervasive Computing and Applications, 2006 1st International Symposium on*, pages 178–182, Aug 2006. doi: 10.1109/SPCA.2006.297564.

[5] D. Lopez, F. Gonzalez, L. Bellido, and A. Alonso. Adaptive multimedia streaming over ip based on customer oriented metrics. In *Computer Networks, 2006 International Symposium on*, pages 185–191, 2006. doi: 10.1109/ISCN.2006.1662531.

[6] Jeffrey Rubin and Dana Chisnell. *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*. Wiley, 2 edition, August 2008.

[7] Über Ruby. URL `https://www.ruby-lang.org/de/about/`.

[8] Ruby on Rails | ZIDline | Zentraler Informatikdienst | TU Wien. URL `http://www.zid.tuwien.ac.at/zidline/zl18/ror/`.

[9] Bootstrap (Framework), August 2015. URL `https://de.wikipedia.org/w/index.php?title=Bootstrap_(Framework)&oldid=145130109`. Page Version ID: 145130109.

[10] Charts. URL https://developers.google.com/chart/.

[11] heise Developer. Mario Inchiosa und Bill Jacobs zum Potenzial von R. URL http://www.heise.de/developer/artikel/Mario-Inchiosa-und-Bill-Jacobs-zum-Potenzial-von-R-2572305.html.

[12] clbustos/rinruby. URL https://github.com/clbustos/rinruby.

[13] Funktionale Anforderungen erheben. http://www.re-wissen.de/Wissen/Anforderungserhebung/Praktiken/Funktionale_Anforderungen_erheben.html, . Accessed: 2015-06-10.

[14] Funktionale, nicht funktionale Anforderungen. http://www.anforderungsmanagement.ch/in_depth_vertiefung/funktionale_nicht_funktionale_anforderungen/. Accessed: 2015-06-10.

[15] Beate Hartmann, Andree Teusch, and Matthias Wolf. Spezifikation von funktionalen und nichtfunktionalen Systemanforderungen auf Basis von Geschäftsprozessmodellen. *Wirtschaftsinformatik Proceedings 2013*, January 2013. URL http://aisel.aisnet.org/wi2013/81.

[16] Nichtfunktionale Anforderungen erheben. http://www.re-wissen.de/Wissen/Anforderungserhebung/Praktiken/Nichtfunktionale_Anforderungen_erheben.html, . Accessed: 2015-06-10.

[17] Erich Gamma, Richard Helm, and Ralph Johnson. *Entwurfsmuster . Elemente wiederverwendbarer objektorientierter Software*. Addison-Wesley, München, 2. aufl. edition, 2001. ISBN 978-3-8273-1862-6.

[18] bliki: TellDontAsk. URL http://martinfowler.com/bliki/TellDontAsk.html.

[19] Andreas Lennartz. *Mehr als Clean Code: Gedanken zur Softwareentwicklung*. BoD – Books on Demand, September 2014. ISBN 978-3-7357-3225-5.

[20] plataformatec/devise, . URL https://github.com/plataformatec/devise.

[21] scambra/devise_invitable, . URL https://github.com/scambra/devise_invitable.

[22] John W. Tukey. *Exploratory Data Analysis*. Addison Wesley Pub Co Inc, Reading, Mass, June 1977. ISBN 978-0-201-07616-5.

[23] prawnpdf/prawn. URL `https://github.com/prawnpdf/prawn`.

[24] Morae Funktionen. URL `http://www.techsmith.de/morae-funktionen.html`.