

Bachelorarbeit 2017

Self- and Peer Assessment Tool

Die Entwicklung eines Webapplikationsprototypen auf dem Prinzip des Self- and Peer Assessments

SEPAT 

u^b

^b
UNIVERSITÄT
BERN

Student : Gerd Zurbriggen
Dozent : Prof. Dr. René Schumann
Abgegeben am : 2. August 2017

Gerd Zurbriggen
Litternaweg 16
3930 Visp
zurbriggen.gerd@outlook.com

Zusammenfassung

Das Ziel der vorliegenden Bachelorarbeit ist die Erstellung eines Prototyps für ein Self- and Peer Assessment. Der Prototyp ist in Zusammenarbeit mit Herrn Thomas Tribelhorn, Leiter des Bereiches Hochschuldidaktik und Lehrentwicklung der Universität Bern, realisiert worden. Von Seiten des Auftraggebers soll der Prototyp auf dem Prinzip des Self- and Peer Assessments beruhen. Das Prinzip des Self- and Peer Assessments findet heute Anwendung in Universitäten, Fachhochschulen oder in der Personalaus- und Weiterbildung. Die Namensgebung des Prototypen SEPAT setzt sich zusammen aus der abgekürzten Version von **Self- and Peer Assessment Tool**.

Die Universität Bern verfügt bereits über ein existierendes Plug-In namens Selevor. Die grundlegenden Funktionen des Selevor Plug-Ins werden in den SEPAT Prototypen übernommen, weshalb eine Analyse durchzuführen ist. Nach der Analyse des bestehenden Systems, wird der Kundenauftrag im Detail beschrieben. Im Kundenauftrag wird eine Priorisierungsliste der zu erledigenden Funktionen erstellt. Anhand der geforderten Erweiterungen gegenüber dem Selevor Plug-In werden Personas erstellt, die bei der GUI-Spezifikation massgebende Entscheidungskriterien für die grafische Ausrichtung des Prototyps bilden. Die GUI-Spezifikation beinhaltet die erstellten Mockups, die dem Auftraggeber einen ersten Einblick in die neue Software vermittelt.

Anschliessend werden die ausgewählten Technologien für die Implementation des SEPAT Prototypen besprochen. Anhand der gewählten Technologien wird die Softwarearchitektur inklusive der Datenbank Modellierung erklärt.

Die relevantesten Ergebnisse dieser Arbeit sind mittels Sourcecode Ausschnitten in dieser Bachelorarbeit erklärt und aufgezeigt. In einem letzten Schritt widmet sich die Bachelorarbeit den implementierten Resultaten. Nachfolgend wird eine Grobanalyse zwischen dem alten und neuen System durchgeführt. Die Grobanalyse zeigt die wichtigsten Differenzen der Systeme auf. Deshalb werden die definierten Erweiterungen des Kundenauftrages aufgezeigt.

Keywords: Self- and Peer Assessment, Selevor, ASP.NET MVC, Visual Studio, Softwareentwicklung
--

Vorwort und Danksagung

Die vorliegende Bachelorarbeit entstand von Mai bis Ende Juli 2017. Die Zeitplanung der Umsetzung teilte sich auf drei Hauptaufgaben auf. In einem ersten Schritt galt die Spezifikation und Eruiierung des Kundenauftrags. Nach einer zweiwöchigen Spezifikation startete ich mit der Implementation der Webapplikation. Die Implementation beanspruchte mit rund 270 Stunden die meiste Zeit dieser Bachelorarbeit. Die letzte Aufgabe lag in der Erarbeitung der wissenschaftlichen Arbeit. In der wissenschaftlichen Arbeit wurde die Ausgangslage, das Vorgehen und die Durchführung dokumentiert.

Aufgrund meines Interesses und der Motivation entschied ich mich, die Webapplikation mit der ASP.NET Technologie umzusetzen. Somit konnte ich meine Erfahrungen in dieser Technologie für meinen zukünftigen Werdegang erweitern.

Ebenfalls war es sehr aufschlussreich mit dem Auftraggeber in Kontakt zu treten, um Ideen, Anforderungen und Bedürfnisse in Kooperation in die Webapplikation einfließen zu lassen.

An dieser Stelle möchte ich mich bei allen Personen bedanken, die mich während der Realisierung und Umsetzung der Bachelorarbeit tatkräftig unterstützt haben. Einen speziellen Dank richte ich an meinen Dozenten, Herrn Prof. Dr. René Schumann. Herr Prof. Dr. Schumann hat mir wöchentlich hilfreiche Inputs und Feedbacks gegeben.

Des Weiteren bedanke ich mich bei dem Auftraggeber, Herrn Thomas Tribelhorn, für eine sehr gute und produktive Zusammenarbeit.

Inhaltsverzeichnis

Abbildungsverzeichnis	viii
Abkürzungsverzeichnis.....	xi
1. Einleitung.....	1
1.1. Methodisches Vorgehen.....	2
2. Allgemeine Informationen	3
2.1. Der Auftraggeber	3
2.2. Das ILIAS System	3
2.3. Selevor – das bestehende Plug-In.....	4
2.3.1. Prozessablauf einer Selbstevaluation in Selevor	5
2.3.2. Fragenblöcke.....	6
2.3.3. Analysefragen.....	6
2.3.4. Stammdatenfragen	8
2.3.5. Feedback.....	9
2.3.6. Einstellungen.....	10
2.3.7. Rechte	11
2.3.8. Alte Resultate	12
2.4. Der Begriff SEPAT.....	14
2.4.1. Self- Peer and Group Assessment.....	14
3. Ist-Analyse Selevor.....	15
3.1. User Experience	15
3.1.1. Umsetzung der UX in Selevor“	15
3.2. Usability.....	17
3.2.1. Umsetzung der Usability in Selevor	17
4. Der Kundenauftrag SEPAT	20
4.1. Wie kann SEPAT eingesetzt werden?	20
4.2. Grundlegende Funktionen	20
4.3. Erweiterungen	21
4.3.1. Grobarchitektur von SEPAT.....	21

4.3.2.	Optimierte UX und Usability	23
4.3.3.	Antwortmöglichkeiten der Stammdatenfragen.....	23
4.3.4.	Fragenbogen und Skalen-Management.....	24
4.3.5.	PDF Export.....	26
4.3.6.	Drittanbieter Authentifikation.....	26
4.3.7.	Selbstevaluation Management.....	26
4.3.8.	Gruppen Management.....	26
4.3.9.	Anonyme Durchführung.....	27
4.4.	Personas	27
4.4.1.	System Administrator: Ralph Dupont	28
4.4.2.	Site Administrator: Manfred Albrecht	29
4.4.3.	Master: Sophie Bracher	30
4.4.4.	User: Dennis Hofener	31
5.	Planungs- und Spezifikationsphase	32
5.1.	Priorisierung	32
5.2.	GUI-Spezifikation.....	33
5.2.1.	Mockups Aufbereitung	34
6.	Systemarchitektur.....	36
6.1.	Grundarchitektur der Software.....	37
6.2.	Frontend.....	38
6.2.1.	HTML5 & CSS3.....	38
6.2.2.	Bootstrap Framework	38
6.2.3.	JavaScript, JQuery & AJAX	39
6.3.	Backend	40
6.3.1.	Webserver	40
6.3.2.	ASP.NET MVC 5	40
6.3.3.	Entity Framework 6 mit MS SQL Server Express 2014.....	42
7.	Die Implementationen und Lösungsansätze	44
7.1.	Relationales Datenbank Modell	45

7.1.1.	Account-Management.....	45
7.1.2.	Analysefragen-Management.....	46
7.1.3.	Stammdatenfragen-Management	47
7.1.4.	Skalen-Management.....	48
7.1.5.	Analysis-Management	50
7.1.6.	Antworten-Management	51
7.2.	Google Authentifikation (OAuth 2)	52
7.2.1.	Anwendung von ASP.NET Identity	54
7.2.2.	Datenbankkontext.....	57
7.3.	Account-Management	58
7.3.1.	Account CRUD	60
7.4.	Stammdatenfragen Antwortoptionen.....	66
7.5.	Skalen-Management	68
7.5.1.	Fragen zur Skala zuordnen.....	71
7.6.	Segmentierung	72
7.7.	Sortierungsverfahren einer Selbstevaluation	76
7.7.1.	Manuelles Sortierungsverfahren	78
7.7.2.	Automatisiertes Sortierungsverfahren.....	78
7.7.3.	Speicherung des Sortierungsverfahren.....	79
7.8.	Selbstevaluation durchführen	82
7.9.	Auswertung anzeigen	85
7.10.	Export Gesamtausprägungen.....	86
7.11.	Responsive Design.....	88
8.	SEPAT versus Selevor	89
8.1.	Durchführung einer Selbstevaluation.....	89
8.2.	Anzeige der Auswertung.....	90
8.3.	User Experience	91
8.4.	Usability.....	93
8.4.1.	Wiederkehrende Prozessstrukturen.....	93

8.4.2. Wizard	94
8.5. Skalen-Management	95
8.6. Antwortausrichtungen der Stammdatenfragen	95
8.7. Einsetzbarkeit des Prototyps	96
Schlussfolgerung	97
Literaturverzeichnis	99
Anhang I: SEPAT Manual	103
Anhang II: Entwicklungshinweise	108
Anhang III: Rechtenmatrix	109
Anhang IV: Skalenlogik – Skizze	111
Anhang V: Erstellung eines Quiz/Selbstevaluation Projektes in eCoach	112
Anhang VI: Erstellung eines Selbstevaluation Projektes in Selevor	114
Anhang VII: Exposé Zeitplanung	116
Anhang VIII: Mockups	117
Anhang IX: Datenbankmodell SEPAT	121
Anhang X: Implementationsaufwand	122
Anhang XI: Grobkonzept von SEPAT	124
Anhang XII: Sourcecode Methode «SaveSelectedQuestionsInScale»	125
Anhang XIII: Sourcecode Methode «SetScalesAsBlocks»	126
Anhang XIV: Sourcecode Methode «PutSequence»	127
Anhang XV: Sourcecode Methode «Play»	129
Anhang XVI: Sourcecode Methode «Analysis»	130
Anhang XVII: Sourcecode Methode «SaveAnswers»	131
Anhang XVIII: Sourcecode Methode «End»	133
Selbstständigkeitserklärung des Verfassers	135

Abbildungsverzeichnis

Abbildung 1: Übersicht der Funktionalitäten einer Selbstevaluation in Selevor	5
Abbildung 2: Prozessablauf einer Selbstevaluation in Selevor	5
Abbildung 3: Erstellung eines Fragenblockes für Analysefragen	6
Abbildung 4: Erstellung einer Analysefrage.....	7
Abbildung 5: Visuelle Darstellung des Prinzips "Skala umkehren"	7
Abbildung 6: Erstellung einer Stammdatenfrage mit einer Auswahlliste von Antworten.....	8
Abbildung 7: Funktionsweise der spezifischen Segmentierung	9
Abbildung 8: Erstellung eines Feedbacks und dessen Segmenten	9
Abbildung 9: Erstellung eines neuen Segmentes	10
Abbildung 10: Definition der Likert Skala in den Einstellungen.....	11
Abbildung 11: Übersicht von lokalen Rollen mit deren Berechtigungen.....	11
Abbildung 12: Auszug aus der Resultaten einer Selbstevaluation.....	12
Abbildung 13: Balkendiagramm des Feedbacks zu einem Fragenblock.....	12
Abbildung 14: Spinnengrafik des Feedbacks zu einem Fragenblock.....	13
Abbildung 15: Balkendiagramm des Feedbacks aller Fragenblöcke	13
Abbildung 16: Fragenerstellung in Socrative	16
Abbildung 17: Fragenerstellung in Selevor.....	16
Abbildung 18: Aktivierung einer unvollständigen Selbstanalyse	17
Abbildung 19: Übersicht der Fragenblöcke mit fehlenden Daten.....	18
Abbildung 20: Erstellung eines Quiz und Department auf eCoach	19
Abbildung 21: SEPAT Grobarchitektur Modell.....	21
Abbildung 22: Erklärung der Sitemap mit spezifischen Beispielen	22
Abbildung 23: Diagramm des Skalen Management	24
Abbildung 24: Persona Ralph Dupont	28
Abbildung 25: Persona Manfred Albrecht.....	29
Abbildung 26: Persona Sophie Bracher	30
Abbildung 27: Persona Dennis Hofener	31
Abbildung 28: Priorisierung der Funktionalitäten	32
Abbildung 29: Mockup Account Management	33
Abbildung 30: Die modellierte Softwarearchitektur	36
Abbildung 31: Grundarchitektur und die wichtigsten Komponenten des ASP.NET MVC	38
Abbildung 32: Verwendung des Datenbankkontextes im Code First Approach	43
Abbildung 33: Vorgehensweise der Softwareimplementierung.....	44
Abbildung 34: Auszug des Datenbankmodells des Account-Managements	45
Abbildung 35: Auszug des Datenbankmodells des Analysefragen-Managements	46

Abbildung 36: Auszug des Datenbankmodells des Stammdatenfragen-Managements	48
Abbildung 37: Auszug des Datenbankmodells des Skalen-Managements	48
Abbildung 38: Auszug des Datenbankmodells des Feedbacks-Managements	49
Abbildung 39: Auszug des Datenbankmodells des Analysis-Managements	50
Abbildung 40: Auszug des Datenbankmodells des Antworten-Managements	51
Abbildung 41: Erstellung eines ASP.NET MVC Projektes mit ASP.NET Identity	52
Abbildung 42: Google Authentifikation und Zugriff in Startup.Auth.cs	54
Abbildung 43: Informationszugriff eines Google-Accounts in ExternalLoginCallback	54
Abbildung 44: BPMN-Model des Registrationsverfahrens	55
Abbildung 45: Authentifikation eines externen Kontos in ExternalLoginCallback	56
Abbildung 46: Objekterstellung und Speicherung eines neuen Gmail Anwender:	56
Abbildung 47: Datenbankkontext des SEPAT Prototyps	57
Abbildung 48: Die Verzeichnisstruktur der Views vom User Management	58
Abbildung 49: Account-Management aus SEPAT	59
Abbildung 50: Konfigurationen der DataTable für die Anzeige der Accounts	59
Abbildung 51: Razor View des Account-Managements	60
Abbildung 52: AddUser Bootstrap Modal Aufruf aus externem Script File	61
Abbildung 53: Bootstrap Modal für die Erstellung eines neuen Accounts	61
Abbildung 54: Sourcecode Teile eines ViewModels AddUser und einer Razor View	62
Abbildung 55: Razor PartialView Code-Ausschnitt der Funktionalität Add User	63
Abbildung 56: Implementierung der Autocompletion	64
Abbildung 57: HttpPost Methode AddUser	65
Abbildung 58: Definition der Antwortoptionen einer Stammdatenfrage	66
Abbildung 59: Pseudocode der dynamischen Erstellung von Antwortoptionen	67
Abbildung 60: Antwortoptionen aus der FormCollection lesen	68
Abbildung 61: Übersicht der Fragenzuordnung im Skalen-Management	69
Abbildung 62: JQuery Code der PartialViews vom Skalen-Management	70
Abbildung 63: Methode renderPaneSection für die Analysefragen	70
Abbildung 64: Linq Datenbankabfrage der Analysefragen in einer Skala	71
Abbildung 65: Speichern der neuen und Anpassung der bestehenden Analysefragen	72
Abbildung 66: Übersicht der Segmentierung einer Skala	72
Abbildung 67: AJAX Form in Razor View	73
Abbildung 68: Pseudocode der Einbindung von den Segmenten in den Multislider	74
Abbildung 69: Pseudocode der Generierung des Multisliders	75
Abbildung 70: Darstellung einer manuelle Sortierung	76
Abbildung 71: Verwaltung der Fragenblöcke für die Anzeige	77
Abbildung 72: Implementierung der Methode SetScalesAsBlocks	79

Abbildung 73: Methode zur Speicherung der Sequenzen	81
Abbildung 74: Workflow der Selbstevaluation	82
Abbildung 75: Implementation der Methode Play	82
Abbildung 76: Implementation der Methode Analysis.....	83
Abbildung 77: Implementation der Methode SaveAnswers	84
Abbildung 78: Implementation der Methode End.....	85
Abbildung 79: ChartJS Definition	86
Abbildung 80: Excel Export der Gesamtausprägungen einer Selbstevaluation	87
Abbildung 81: Einbindung der JS und CSS Klassen der DataTable	87
Abbildung 82: Definition der Export Buttons.....	88
Abbildung 83: Startoberfläche einer Selbstevaluation in SEPAT	89
Abbildung 84: Durchführung einer Selbstevaluation in SEPAT	90
Abbildung 85: Bar Chart Auswertung in SEPAT	90
Abbildung 86: Netzdiagramm Auswertung in SEPAT	91
Abbildung 87: Anzeigeblöcke für die Komponenten	92
Abbildung 88: Profil eines Accounts.....	92
Abbildung 89: Bearbeitung einer Analysefrage im Selevor Plug-In.....	93
Abbildung 90: Bearbeitung einer Analysefrage im SEPAT Prototypen	94
Abbildung 91: Progress Balken vom Wizard Manager	94

Abkürzungsverzeichnis

ASP.NET	Active Server Pages .NET
AJAX	Asynchronous JavaScript and XML
BPM	Business Process Modeling
BPMN	Business Process Model and Notation
BWL	Betriebswirtschaftslehre
CSS3	Cascading Style Sheets Version 3
CSV	Comma-separated values
DBMS	Datenbank-Management-System
DOM	Document Object Model
EF	Entity Framework
FIG	Filière Informatique de Gestion
GUI	Graphical User Interface
HES-SO	Haute école spécialisée de Suisse occidentale
HP	Hewlett-Packard Company
HTML5	Hypertext Markup Language Version 5
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IIS	Internet Information Services
ILIAS	Integriertes Lern-, Informations- und Arbeitskooperations-System
IP	Internet Protocol
JS	JavaScript
LINQ	Language Integrated Query
LMS	Learning Management System
MS	Microsoft
MVC	Model-View-Controller
MVVM	Model-View-ViewModel
OWIN	Open Web Interface for .NET

O/RM	object-relational-mapping
PC	Personal Computer
Prof.	Professor
Selevor	Selbstanalyse für Lehrvorträge und Vorlesungen
SEPAT	Self- and Peer Assessment Tool
SQL	Structured Query Language
UI	User Interface
UniBe	Universität Bern
URL	Uniform Resource Locator
Vkf.	Verkauf
VWL	Volkswirtschaftslehre
ZUW	Zentrums für universitäre Weiterbildung

1. Einleitung

Der Bereich Hochschuldidaktik & Lehrentwicklung der Universität Bern benutzt auf ihrem ILIAS System ein Plug-In namens Selevor. Dieses Plug-In dient zur Selbstevaluation von Studenten für Lehrvorträge und Vorlesungen. Die Selbstevaluationen werden zurzeit mehrheitlich durch die Dozenten erstellt und veröffentlicht. Nach der Durchführung eines spezifischen Fragebogens generiert Selevor für den Studenten eine grafische Auswertung in Kombination mit einem kurzen Feedbacktext. Die Selbstevaluationen werden teilweise als Befragungen eingesetzt, womit die Stammdaten der Befragten einen bedeutenden Stellenwert zu Forschungszwecken einnehmen.

Das bestehende Plug-In Selevor ist begrenzt in der User Experience und überdies ist die Usability in diversen Prozessen beschränkt. Des Weiteren deckt das Plug-In nicht die vollständigen Kundenanforderungen wie die Einsetzbarkeit in den verschiedenen Bereichen und Unternehmungen ab. Das Ziel dieser Bachelorthesis ist die Erstellung eines unabhängigen flexiblen Webapplikationsprototypen, die den Auftraggeber unterstützt, Selbstevaluationen in verschiedenen Bereichen und Unternehmungen durchzuführen. Die Einsatzgebiete der neuen Software sind beispielsweise in der Psychologie, Didaktik, Personal- und Organisationsentwicklung. Die neue Software wird in dieser Bachelorarbeit als SEPAT bezeichnet. SEPAT bedeutet **S**elf- and **P**eer **A**ssessment **T**ool.

Zu den Anwendern der Software zählen unterschiedliche Zielgruppen wie Studenten, Dozenten, Mitarbeiter und Abteilungsleiter, weshalb ein ansprechender Web-Content konzipiert werden muss, der gezielt auf die User Experience und Usability ausgerichtet ist. Die Applikation muss aus Sicht der Studenten Responsive sein. Das Responsive Design ermöglicht die Ausführung von einer Selbstevaluation auf verschiedenen Geräteauflösungen.

Hinsichtlich der multifunktionalen Einsetzbarkeit des Systems in verschiedenen Unternehmen muss eine strikte Berechtigungsstrategie umgesetzt werden. Die Berechtigungsstrategie trennt die Unternehmen voneinander ab.

Anhand der Datenverwaltung soll es dem Kunden möglich sein, bestehende Resultate der Selbstevaluationen und Datensätze der Stammdaten zu exportieren, um diese für Forschungszwecke einzusetzen. Vor allem bei der Erstellung der Selbstevaluation, der Stammdatenfragen und der Feedback Templates, müssen dem Administrator interaktive sowie adaptive Lösungen zur Verfügung gestellt werden. Ein wesentlicher Bestandteil aus Sicht des Auftraggebers sind die übersichtlichen Auswertungen, mit deren variablen Grafiken, am Ende einer Selbstevaluation.

1.1. Methodisches Vorgehen

Bevor mit der eigentlichen Implementierung begonnen werden kann, müssen die Kundenanforderungen analysiert und in verschiedenen Tasks strukturiert werden. Somit wird in einem ersten Schritt eine Projektskizze erstellt und modelliert. Ziel dieser Modellierung ist die visuelle Darstellung der Kundenanforderungen. Daraufhin können die verschiedenen Tasks der Projektskizze verfeinert werden. Angesichts der zeitlich engen Implementierungsphase müssen die Tasks mit dem Auftraggeber priorisiert werden, um die grundlegenden Funktionalitäten zu decken. Nach der Priorisierung der Funktionalitäten findet die Softwarespezifikation statt. Für die Spezifikation wird ein GUI-Spezifikationsverfahren angewendet. Aus diesem Grund werden Mockups der individuellen Oberflächen vorbereitet und mit dem Auftraggeber angepasst. Das GUI-Spezifikationsverfahren unterstützt ebenfalls die Ermittlung der Kundenbedürfnisse hinsichtlich der User Experience und Usability.

In einem nächsten Schritt werden die ausgewählten Technologien kurz erläutert, die zur Umsetzung des SEPAT Prototypen eingesetzt werden. Diese Definitionen der Technologien zeigen auf, wofür und warum diese Technologien gewählt werden.

Die Implementation wird mithilfe der Mockups und der Prioritätenliste durchgeführt. Als erster Schritt im Implementationsprozess wird die höchstpriorisierte Funktion aus der Prioritätenliste entnommen. Eine Funktion unterläuft die drei Standardprozesse des agilen Vorgehensmodells von Kanban (Getting Started with Kanban for Software Development, kein Datum).

Nach der Softwarespezifikation beginnt die Implementierung auf Seiten des Frontends. Für die Frontendentwicklung werden unterschiedliche User Interface (UI) Templates herausgefiltert und validiert. Nach der Festlegung des Templates werden iterativ die verschiedenen Tasks jeweils in die Webapplikation eingebaut. Aus diesem Grund wird der Code First Approach für die Generierung der Datenbank gewählt.

Um die Funktionen des Implementationsprozesses abzuschliessen, werden Software Validierungen stattfinden, damit die Qualität des Sources Codes und die Fehlervermeidung aufrechterhalten wird. Die Software Validierung umfasst das Testen der Applikation im Browser und die Überprüfung der Softwareausgaben in der Entwicklungsumgebung.

Nach der Dokumentation der Implementierungsphase wird eine kurze Gegenüberstellung der neuen und alten Software durchgeführt. Diese Analyse soll die Vorteile der neuen SEPAT Software aufzeigen.

2. Allgemeine Informationen

Als Einführung in die Bachelorarbeit werden zunächst generelle Informationen zum Auftraggeber sowie zum bestehendem ILIAS System und dem Selevor Plug-In erläutert. Ebenfalls wird das SEPAT System und dessen Bedeutung präzisiert.

Damit die Terminologie dieser Bachelorarbeit verständlich ist, sind im Anhang I: SEPAT Manual die Begriffserklärungen beigelegt worden. Diese wurden von Seiten des Auftraggebers Herr Thomas Tribelhorn definiert. Die vorhandenen Begriffserklärungen stammen vor allem aus dem aktuellen Plug-In und wurden im neuen System übernommen.

2.1. Der Auftraggeber

Das Thema dieser Bachelorarbeit wurde durch Herrn Thomas Tribelhorn von der Universität Bern vorgeschlagen. Herr Tribelhorn ist an der Universität Bern Leiter des Bereichs Hochschuldidaktik & Lehrentwicklung und zudem Leiter des Studienganges «CAS Hochschullehre». Herr Tribelhorn doziert zu Themen der Hochschuldidaktik und der Entwicklung von Studiengängen und innovativen Lehr- und Lernmethoden. Zusätzlich ist er in einer beratenden Funktion bei der Entwicklung von Weiterbildungsangeboten, Studiengängen und Lehrplänen tätig. Er ist Mitglied der Geschäftsleitung des **Z**entrums für universitäre **W**eiterbildung (ZUW) der Universität Bern. Im Weiteren realisierte er, in Zusammenarbeit mit Frau Lydia Rufer, das Plug-In namens Selevor für das ILIAS System für die Universität Bern.

2.2. Das ILIAS System

ILIAS ist eine Open-Source E-Learning Plattform und steht für Integriertes **L**ern, **I**nformations- und **A**rbeitskooperations-**S**ystem. ILIAS ist eine Lernplattform. Das Grundprinzip einer Lernplattform umfasst die Erstellung, den Austausch und den Gebrauch von Lern- sowie Lehrmaterialien auf elektronischer Basis (ILIAS open source e-Learning e.V., kein Datum).

Lernplattformen werden zudem **L**earning **M**anagement **S**ystems (LMS) genannt. Baumgartner et al. (2002, S. 24) definiert ein LMS wie folgt:

„Ein Lernmanagementsystem ist eine serverseitige installierte Software, die beliebige Lerninhalte über das Internet zu vermitteln hilft und die Organisation der dabei notwendigen Lernprozesse unterstützt.“

Zu den wichtigsten Funktionalitäten eines LMS zählen:

- Datei-Upload per Drag&Drop
- Kurs-Management
- Prüfungstools
- Erstellung von Lern- und Übungsmaterialien

Aktuell zählen über 50 Institutionen zu den Mitgliedern der ILIAS Community. Darunter befinden sich zahlreiche Universitäten sowie Fachhochschulen aus der Schweiz. Beispielsweise setzt die Lernplattform die Universität Bern, Basel und die ETH Zürich ein. Überwiegend benutzen deutsche Hochschulen und Universitäten die Open-Source Plattform (ILIAS open source e-Learning e.V., kein Datum).

2.3. Selevor – das bestehende Plug-In

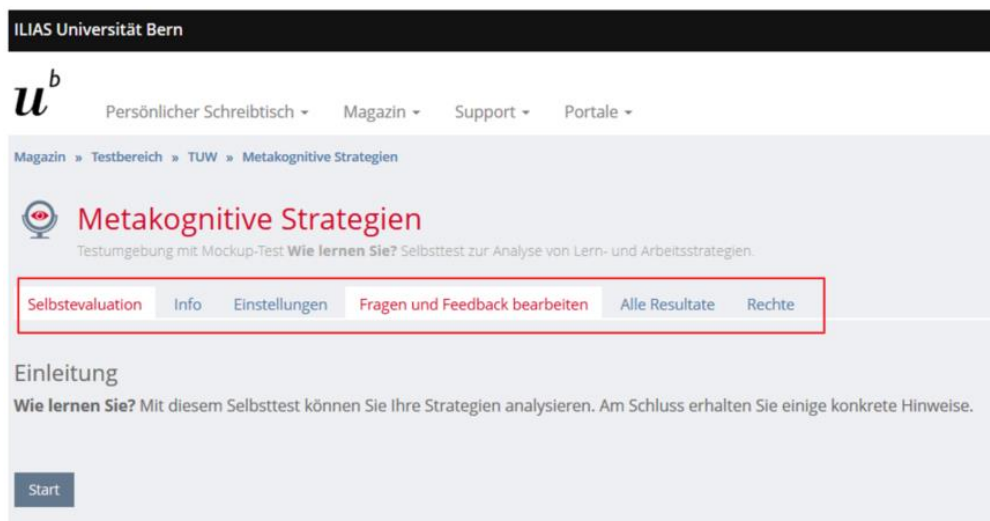
Selevor ist ein Plug-In, welches in das ILIAS System der Universität Bern eingebunden ist. Die technische Umsetzung sowie die Veröffentlichung von Selevor wurde von der «iLUB – Supportstelle für ICT-gestützte Lehre und Forschung» der Universität Bern durchgeführt.

Ziel der Verwendung des Selevor Plug-Ins ist die Ausbildung der Studierenden zu fördern. Mithilfe des Plug-Ins können Selbstevaluationen erstellt werden. Diese Selbstevaluationen beinhalten beispielsweise Fragen zu den Lehrvorträgen respektive den Vorlesungen. Aus den Selbstevaluationen resultieren Feedbacks, die den Studierenden eine Eigen- oder eine Fremdeinschätzung übermitteln.

Die Namensgebung wurde dem Einsatzgebiet angepasst. Aus diesem Grund steht Selevor für **S**elbstanalyse für **L**ehrvorträge und **V**orlesungen.

Das Selevor Plug-In umfasst diverse Komponenten wie die allgemeinen Informationen einer Selbstanalyse, Einstellungen, Fragen und Feedback, Resultate und die Zuweisungsrechte für Anwender. Diese Komponenten sind auf der Abbildung 1 ersichtlich.

Abbildung 1: Übersicht der Funktionalitäten einer Selbstevaluation in Selevor

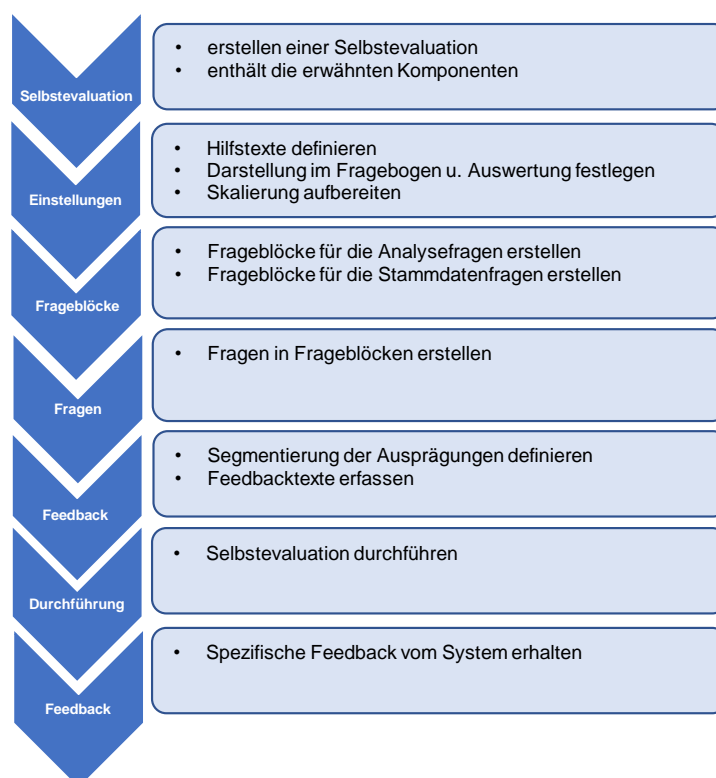


Quelle: <https://ilias.unibe.ch/>

2.3.1. Prozessablauf einer Selbstevaluation in Selevor

Der Prozessablauf beinhaltet die sukzessive Erstellung einer Selbstevaluation sowie deren Durchführung und Auswertung. Die Abbildung 2 visualisiert diesen Ablauf.

Abbildung 2: Prozessablauf einer Selbstevaluation in Selevor



In den folgenden Abschnitten werden die wichtigsten Funktionalitäten der einzelnen Komponenten von Selevor im Detail aufgezeigt. Hinsichtlich des neuen Systems sind diese Funktionalitäten relevant, weil die grundlegenden Funktionalitäten in das neue System übernommen werden sollen.

2.3.2. Fragenblöcke

Bei der Erstellung einer Selbstevaluation werden ein oder mehrere Fragenblöcke erstellt. An dieser Stelle muss zwischen zwei Arten von Fragetypen unterschieden werden. Es besteht die Möglichkeit, einerseits einen Fragenblock für die Analysefragen und andererseits einen Fragenblock für die Stammdatenfragen zu erstellen. Der wesentliche Unterschied zwischen den beiden Fragenblocktypen wird bei der Erstellung ersichtlich. In der Abbildung 3 ist die Erstellung eines Fragenblockes für Analysefragen dargestellt.

Abbildung 3: Erstellung eines Fragenblockes für Analysefragen

BLOCK ERSTELLEN

Titel *

Beschreibung

Abkürzung

Die Abkürzung wird für die Benennung der Feedback Grafiken verwendet (max. 8 Zeichen).

* Erforderliche Angabe

Quelle: <https://ilias.unibe.ch/>

Das Eingabefeld «Abkürzung» ist bei einem Fragenblock für die Stammdatenfragen nicht vorhanden, da die Stammdatenfragen nicht essentiell für das individuelle Feedback einbezogen werden. Es werden nur die Analysefragen evaluiert. Das genannte Eingabefeld dient lediglich zur Kennzeichnung des Fragenblocks in der Auswertung.

2.3.3. Analysefragen

Nach der Generierung der Fragenblöcke werden die Fragen erstellt. Um den gewünschten Fragetypen zu erstellen, in diesem Fall eine Analysefrage, muss ein Analysefragenblock ausgewählt werden. Es können keine Stammdatenfragen in einem Analysefragenblock generiert werden. Die Erstellung einer Analysefrage mit deren Eigenschaften ist in der Abbildung 4 dargestellt.

Abbildung 4: Erstellung einer Analysefrage

FRAGE ERSTELLEN

Frage *

Kurztitel

Die Abkürzung wird für die Benennung der Feedback Grafiken und den CSV-Export verwendet (max. 8 Zeichen).

Skala umkehren ☐

* Erforderliche Angabe

Quelle: <https://ilias.unibe.ch/>

Um ein übersichtlicheres Feedback zu erhalten, wird bei der Analysefrage, identisch wie bei der Erstellung eines Fragenblockes, einen Kurztitel vorgeschlagen. Mit der Aktivierung von «Skala umkehren» wird die Evaluierung beeinflusst. Die untenstehende Abbildung 5 soll dies visuell verdeutlichen.

Abbildung 5: Visuelle Darstellung des Prinzips "Skala umkehren"

	TRIFFT NICHT ZU			TRIFFT ZU
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirm? *	<input type="radio"/> 1	<input type="radio"/> 2	<input type="radio"/> 3	<input type="radio"/> 4
Lorem ipsum dolor sit amet, consetetur sadipscing elitr? *	<input type="radio"/> 4	<input type="radio"/> 3	<input type="radio"/> 2	<input type="radio"/> 1
Lorem ipsum ? *	<input type="radio"/> 1	<input type="radio"/> 2	<input type="radio"/> 3	<input type="radio"/> 4

Quelle: <https://ilias.unibe.ch/>

In der Abbildung 5 sind drei Fragen mit vier Antwortmöglichkeiten aufgezeigt. Zusätzlich sind jeweils neben den Radiobuttons die Werte der unterliegenden Likert Skala hervorgehoben (siehe Erstellung der Likert Skala in Kapitel 2.3.6: Einstellungen).

Die Likert Skala dient zur Messung der Einstellung von den Anwendern über ein spezifisches Thema oder Fach (Wübbenhorst, kein Datum). Die unterliegenden Werte ermöglichen die Berechnung der Mittelwerte von den einzelnen Fragenblöcken.

Die Umkehrung bewirkt widersprüchliche Antworten bei den Anwendern, welche die Selbstevaluation nicht sorgfältig ausfüllen. Dies hat zur Folge, dass die Mittelwerte der Fragenblöcke stark beeinflusst werden. Diese Auswirkungen sind für den Administrator, in der Übersicht der Resultate, erkenntlich. Um das Auswertungsverfahren der Fragenblöcke im Detail zu verstehen wird im Kapitel 2.3.5. Feedback eine Erklärung aufgeführt.

2.3.4. Stammdatenfragen

Generell gelten für die Stammdatenfragen, dass diese nicht in die Auswertung der Selbstevaluation miteinbezogen werden. Die Abbildung 6 zeigt die Erstellung einer Stammdatenfrage. Anhand dieser Abbildung ist der Unterschied zu einer Analysefrage ersichtlich. Aufgrund der Erstellung von vordefinierten Antworten können die Stammdatenfragen nicht mit einer Likert Skala gewichtet werden.

Abbildung 6: Erstellung einer Stammdatenfrage mit einer Auswahlliste von Antworten

The screenshot shows a web interface for creating a question. At the top, there is a header bar with the text 'HINZUFÜGEN'. Below this, the 'Name' field is labeled 'Welches Geschlecht?'. The 'Typ' field has two radio button options: 'Textfeld (beliebige Werte)' and 'Auswahlliste (feste Werte)', with the latter being selected. To the right of the 'Auswahlliste' option, there is a table with two rows: 'männlich' and 'weiblich'. Each row has a '+' button on the left and a '-' button on the right. At the bottom left, there is a checkbox labeled 'Erforderliche Angabe' which is checked.

Quelle: <https://ilias.unibe.ch/>

Aus der Abbildung 6 ist ersichtlich, dass der Administrator in der Lage ist, den Antworttyp zwischen einem Textfeld oder einer Auswahlliste zu wählen. Bei einem Textfeld werden keine weiteren Eingaben benötigt. Hingegen bei der Selektion einer Auswahlliste können mehrere Textfelder dynamisch generiert werden. Zusätzlich kann festgelegt werden, ob die Antwort ein Pflichtfeld ist oder nicht. Somit bietet das Selevor Plug-In dem Administrator die Single-Choice Option, in Form einer Dropdown-Liste, sowie eine Texteingabe als Antworttyp an.

Die Absicht der Stammdatenfragen ist die Sammlung von Informationen über den Studierenden, der die Selbstevaluation durchführt. Die gewonnenen Daten sind deshalb geeignet für Forschungszwecke. Die gesammelten Daten könnten beispielsweise interessant zur Feststellung von bestimmten Verhaltensmustern sein. Ein Beispiel, wie die Daten zu Forschungszwecken genutzt werden können, soll deren Wichtigkeit nachfolgend verdeutlichen:

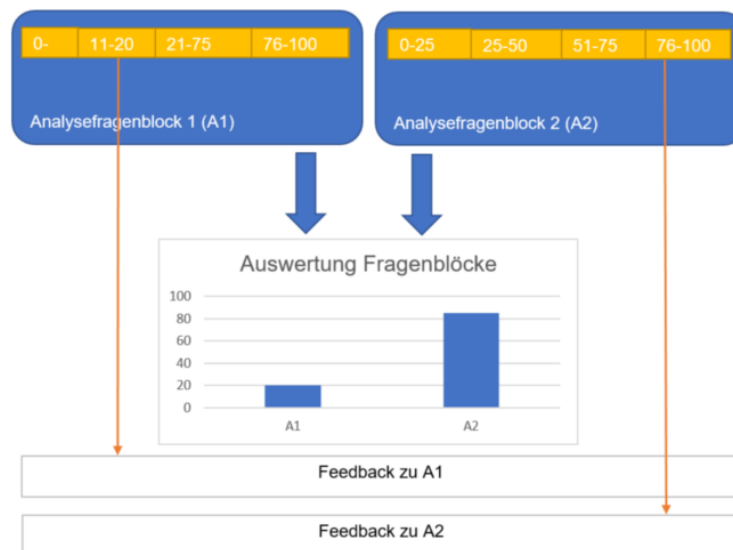
An der Universität in Bern im Studiengang Rechtswissenschaften wird eine Selbstevaluation durchgeführt. Diese bezieht sich auf die Volksabstimmungen vom 24. September 2017 bezüglich der Reform der Altersvorsorge 2020. Neben den Analysefragen, die sich auf die Reform beziehen, werden Stammdatenfragen erstellt. Diese sind auf die Studierenden abgestimmt. Einzelne Stammdatenfragen werden wie folgt definiert: «Wie alt sind Sie?», «Welche politische Ansichten teilen Sie?», «Sind Sie finanziell unabhängig?». Anhand der Resultate der Selbstevaluationen können die Antworten der Analysefragen mit den Stammdatenfragen analysiert werden, um mögliche Muster festzustellen.

2.3.5. Feedback

Ein Feedback wird jeweils einem Fragenblock zugeteilt. Das Feedback zählt mehrere Segmente, die erstellt werden können. Ziel dieser Segmente ist die Anzeige eines spezifischen Feedbacks am Ende einer Selbstevaluation.

Bei der Durchführung der Selbstevaluation beantwortet der Anwender die Analysefragen mittels der Likert Skala. Die unterliegenden numerischen Werte, wie diese in der Abbildung 5 hervorgehoben sind, werden pro Fragenblock zusammenaddiert. Aus dem gewonnen Resultat kann die Ausprägung berechnet werden. Wie die Ausprägung mit dem spezifischen Feedback zusammenhängt, ist in der Abbildung 7 visualisiert.

Abbildung 7: Funktionsweise der spezifischen Segmentierung



Die Segmente eines Fragenblockes können von 0 bis 100% im Sektor der Ausprägungen variieren. Beträgt die generierte Ausprägung eines Fragenblockes beispielsweise 20%, wird das dazugehörige Segment im Feedback in Form eines Feedbacktextes ersichtlich. In der Abbildung 8 ist eine Erstellung eines Feedbacks ersichtlich.

Abbildung 8: Erstellung eines Feedbacks und dessen Segmenten

5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
segment 1					Neu										segment 3				

FRAGEBLOCK: FEEDBACKS BEARBEITEN

(1 - 2 von 2)

Titel	Feedback	Ab	Bis	Aktionen
segment 1	Ihr Wissen ist leider nicht ausreichend. Lesen Sie das Kapitel 2.3 nochmals durch.	>= 0%	<= 27%	Aktionen ▾
segment 3	Sie sind bereit für die anstehende Prüfung.	> 72%	<= 100%	Aktionen ▾

Quelle: <https://ilias.unibe.ch/>

Es bestehen zwei Segmente mit einem jeweiligen Feedbacktext und deren Sektor des Prozentbereiches. Vor dem Start einer Selbstevaluation muss der Administrator den Balken auf 100% vervollständigen. So wird ein korrektes Feedback berücksichtigt.

Bei einem Klick auf «Neu» im freien Sektor des Prozentbereiches in der Abbildung 8, navigiert Selevor auf die Abbildung 9. Neben der Erfassung des Titels und der Beschreibung kann spezifisch ein neues Segment erstellt werden. Zwischen einer «Linearen Bereichszuordnung» und einer «Manuellen Zuordnung» kann ausgewählt werden. Auf der Abbildung 9 ist die zweite Möglichkeit ausgewählt worden, womit der Administrator die Wahl hat, wie er die Segmentierung durchführen will. Bei der ersten Möglichkeit führt das System für den Administrator die Segmentierung linear durch.

Abbildung 9: Erstellung eines neuen Segmentes

The screenshot shows a web form titled 'NEUES FEEDBACK ERFASEN'. At the top right are buttons 'Erstellen' and 'Abbrechen'. The form has several labeled sections: 'Block' with the value 'frageblock'; 'Titel' with an empty text input; 'Beschreibung' with an empty text input; 'Prozentbereich' with two radio button options: 'Lineare Bereichszuordnung' (unselected) and 'Manuelle Zuordnung' (selected). Below the selected option is a slider labeled 'Prozentbereich' with a range of '27% - 72%'. At the bottom is a 'Feedbacktext' section with a rich text editor toolbar and an empty text area.

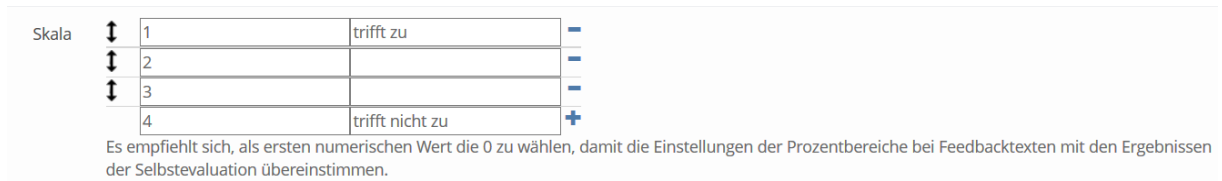
Quelle: <https://ilias.unibe.ch/>

2.3.6. Einstellungen

In den Einstellungen einer Selbstevaluation werden diverse Daten und Evaluationsverhalten festgelegt. Diese Einstellungen betreffen die Eigenschaften, die Darstellung, die Hilfstexte und die Likert Skala einer Selbstevaluation. Die Eigenschaften beinhalten den Titel und die Beschreibung. Zusätzlich kann die Selbstevaluation freigeschaltet oder blockiert werden. Die Darstellungskonfigurationen definieren das optische Verhalten während und nach der Durchführung. Die Hilfstexte umfassen einen Einleitungstext, abschliessenden Text und einen Standardtext für anonyme Anwender.

Die Likert Skala benötigt einen numerischen Wert. Die dazugehörige Bezeichnung ist optional. Mittels der Drag & Drop Funktion kann die Reihenfolge der Skala modifiziert werden. Die Abbildung 10 zeigt eine Erstellung einer Likert Skala.

Abbildung 10: Definition der Likert Skala in den Einstellungen



Skala

1	trifft zu
2	
3	
4	trifft nicht zu

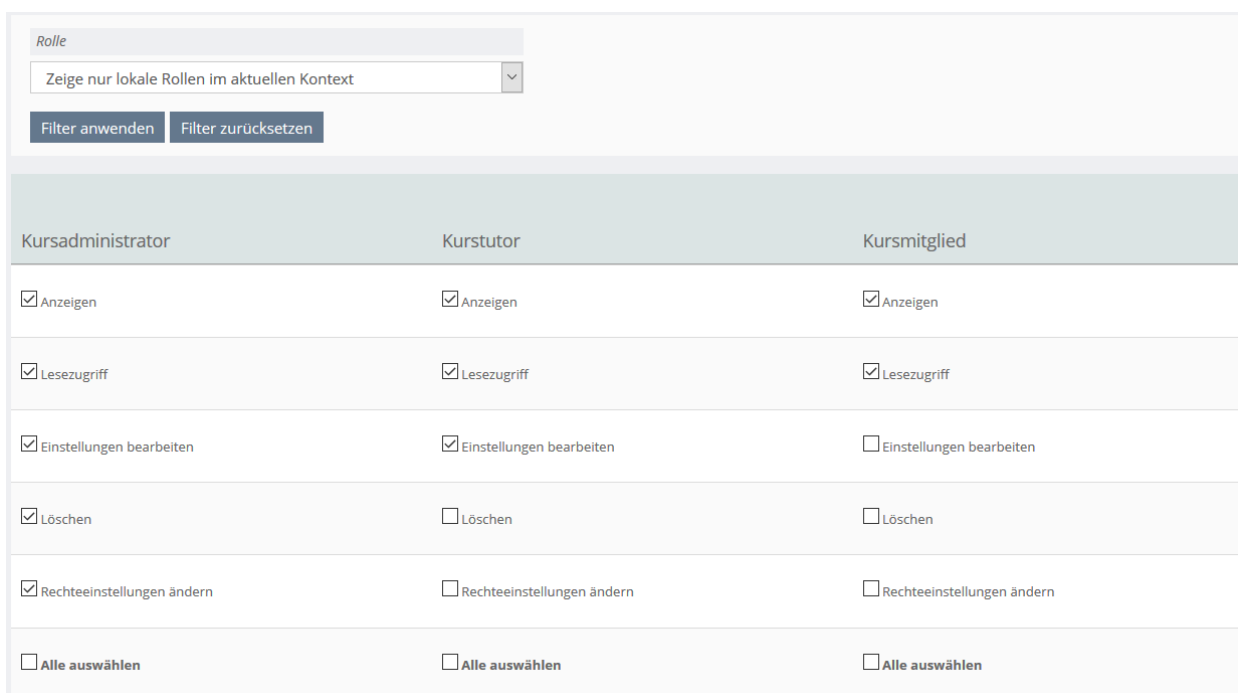
Es empfiehlt sich, als ersten numerischen Wert die 0 zu wählen, damit die Einstellungen der Prozentbereiche bei Feedbacktexten mit den Ergebnissen der Selbstevaluation übereinstimmen.

Quelle: <https://ilias.unibe.ch/>

2.3.7. Rechte

In der Komponente «Rechte» des Selevor Plug-Ins können Berechtigungen auf lokale als auch auf globale Rollen verteilt werden. In der Abbildung 11 sind lokale Rollen im aktuellen Kontext aufgelistet mit deren Berechtigungen. Es können zudem neu Rollen erstellt oder importiert werden.

Abbildung 11: Übersicht von lokalen Rollen mit deren Berechtigungen



Rolle Zeige nur lokale Rollen im aktuellen Kontext		
Filter anwenden Filter zurücksetzen		
Kursadministrator	Kurstutor	Kursmitglied
<input checked="" type="checkbox"/> Anzeigen	<input checked="" type="checkbox"/> Anzeigen	<input checked="" type="checkbox"/> Anzeigen
<input checked="" type="checkbox"/> Lesezugriff	<input checked="" type="checkbox"/> Lesezugriff	<input checked="" type="checkbox"/> Lesezugriff
<input checked="" type="checkbox"/> Einstellungen bearbeiten	<input checked="" type="checkbox"/> Einstellungen bearbeiten	<input type="checkbox"/> Einstellungen bearbeiten
<input checked="" type="checkbox"/> Löschen	<input type="checkbox"/> Löschen	<input type="checkbox"/> Löschen
<input checked="" type="checkbox"/> Rechteinstellungen ändern	<input type="checkbox"/> Rechteinstellungen ändern	<input type="checkbox"/> Rechteinstellungen ändern
<input type="checkbox"/> Alle auswählen	<input type="checkbox"/> Alle auswählen	<input type="checkbox"/> Alle auswählen

Quelle: <https://ilias.unibe.ch/>

2.3.8. Alte Resultate

Unter den Resultaten ist eine Übersicht der durchgeführten Selbstevaluationen sichtbar. An dieser Stelle ist das Startdatum, der Benutzer sowie dessen erreichten Ausprägung über allen Fragenblöcken ersichtlich. Dies wird in der Abbildung 12 gezeigt.

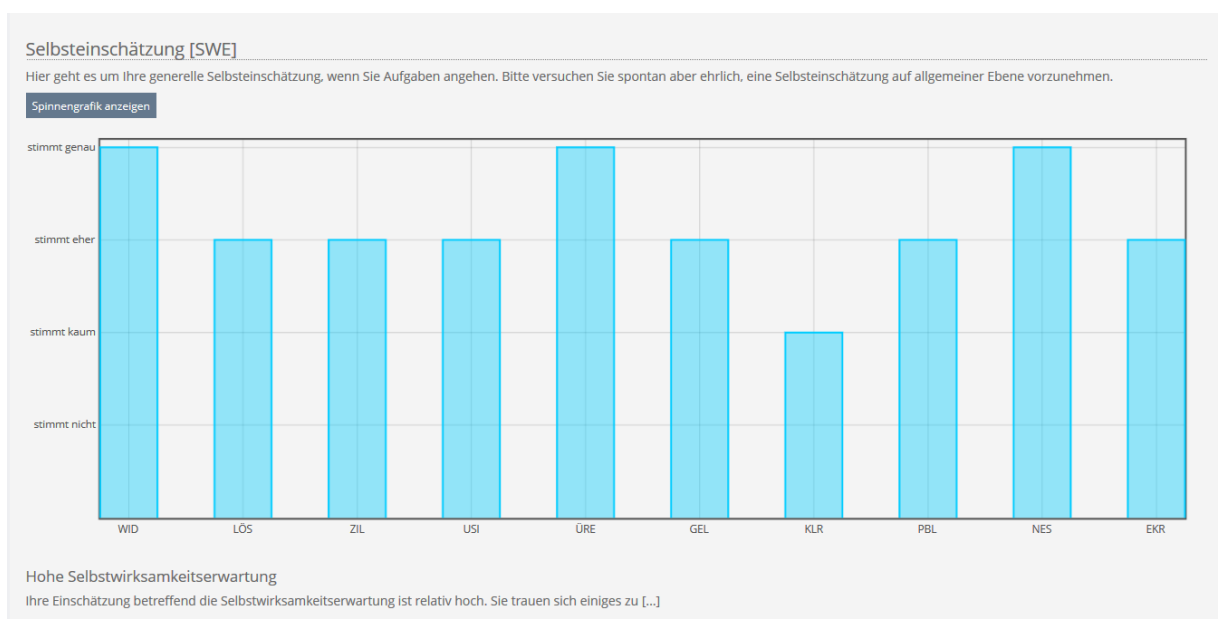
Abbildung 12: Auszug aus der Resultaten einer Selbstevaluation

DATENSÄTZE (1 - 9 von 9)				
Typ	Datum	Benutzer-Kennung	Prozent	Aktionen
ILIAS-Benutzer	28.06.2017 - 14:42:13	g.zurbriggen	19.17	Aktionen ▾
ILIAS-Benutzer	28.06.2017 - 14:49:09	g.zurbriggen	47.5	Aktionen ▾
ILIAS-Benutzer	28.06.2017 - 14:50:42	g.zurbriggen	62.5	Aktionen ▾

Quelle: <https://ilias.unibe.ch/>

Von dieser Oberfläche aus der Abbildung 12 kann auf ein spezifisches Feedback einer Selbstevaluation navigiert werden. Bei einem Befehl auf die Benutzer-Kennung oder unter dem Aktionsbutton «Feedback anzeigen» öffnet sich das Resultat des gewählten Datensatzes.

Abbildung 13: Balkendiagramm des Feedbacks zu einem Fragenblock

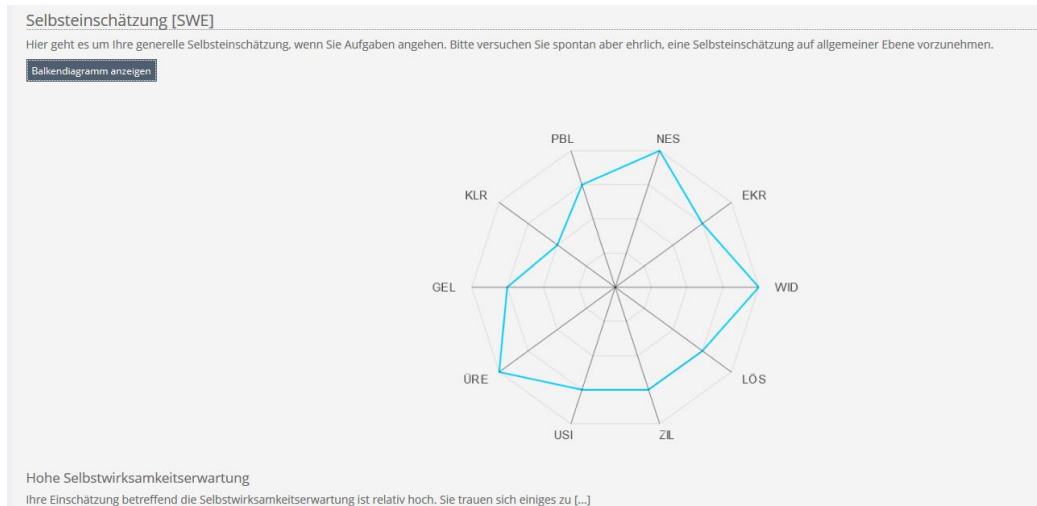


Quelle: <https://ilias.unibe.ch/>

Bei der Auswahl eines Feedbacks wird die Abbildung 13 geöffnet. Diese Abbildung zeigt ein konventionelles Feedback zu einem Fragenblock anhand eines Balkendiagrammes an. Diese Abbildung verdeutlicht die Wichtigkeit der bereits erwähnten Abkürzungen. Auf der X-Achse sind die Analysefragen mittels deren Abkürzungen dargestellt. Die Y-Achse orientiert sich nach der Likert Skala. Unterhalb des Balkendiagramms ist das Feedbacksegment der erlangten Ausprägung ersichtlich.

Bei der Anzeige des Feedbacks kann der Benutzer zwischen zwei Ansichtsoptionen auswählen. Standardmässig ist die Ansichtsoption eines Balkendiagrammes vordefiniert. Beim Klick auf den Button «Spinnengrafik anzeigen» wird die zweite Ansichtsoption generiert. Ein Beispiel ist auf der Abbildung 14 dargestellt.

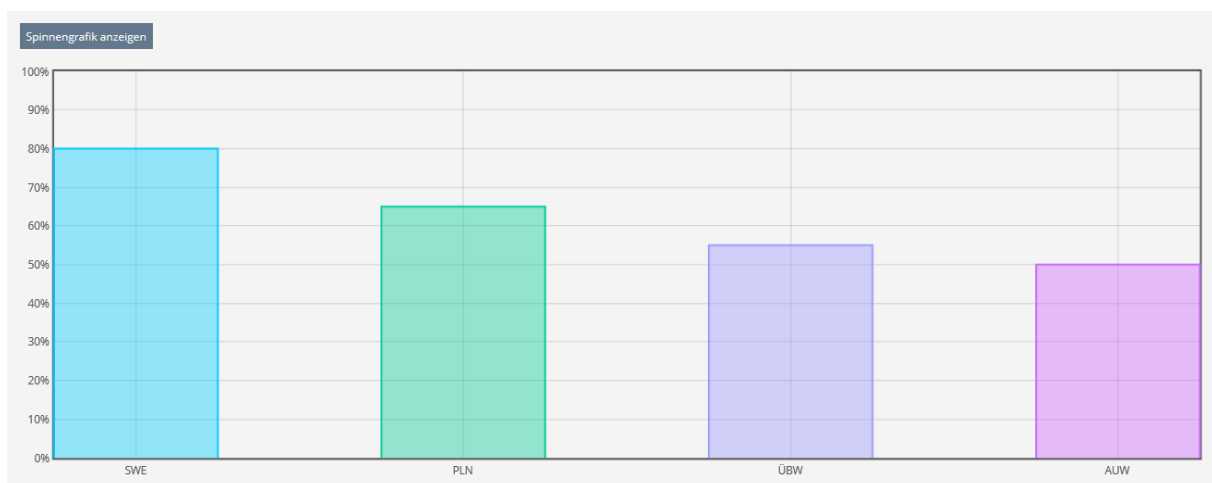
Abbildung 14: Spinnengrafik des Feedbacks zu einem Fragenblock



Quelle: <https://ilias.unibe.ch/>

Um die Ausprägungen über alle Fragenblöcke zu visualisieren, unterstützt dies die Abbildung 15. Diese Übersicht zeigt pro Fragenblock die Ausprägung in Prozent an. Aus den Ausprägungen der Fragenblöcke wird die Gesamtausprägung errechnet, die auf der Abbildung 12 in der Spalte «Prozent» abgebildet ist. Diese Grafik wird ebenfalls im Feedback aufgelistet.

Abbildung 15: Balkendiagramm des Feedbacks aller Fragenblöcke



Quelle: <https://ilias.unibe.ch/>

2.4. Der Begriff SEPAT

Im Rahmen dieser Bachelorarbeit wird ein Prototyp namens SEPAT implementiert. Der Begriff steht für **Self- and Peer Assessment Tool** was übersetzt bedeutet: «Eigen- und Fremdeinschätzungswerkzeug». Anhand des Begriffs ist die wesentliche Einsetzbarkeit der Software teilweise nachvollziehbar. Wie eine Eigen- und Fremdeinschätzung definiert wird, ist im nachfolgenden Kapitel 2.4.1. Self- Peer and Group Assessment erläutert. Mehr Details zur SEPAT Software sind im Kapitel 4 des Kundenauftrags zu finden.

Hinsichtlich des Self- and Peer Assessments handelt es sich um ein didaktisches Hilfsmittel für die Ausbildung von Studenten, Lernenden, etc. Der Begriff beinhaltet drei Gattungen von Assessments, die das Lernverhalten der Auszubildenden beeinflussen kann.

2.4.1. Self- Peer and Group Assessment

Anhand eines Self- oder Peer Assessment können die Resultate mit anderen Lernenden (Peers) ausgetauscht werden. Dies findet vor allem beim Peer Assessment statt. Beim Peer Assessment sind beispielsweise mehrere Peers in einer Gruppe. Der Leiter oder die Peers tauschen nach der Abschliessung des Assessments miteinander die Resultate aus. In einem nächsten Schritt analysieren die Peers ihre Ergebnisse untereinander. Der wesentliche Unterschied zwischen dem Group- und dem Peer Assessment hängt vom Zeitpunkt des Lernens ab. Mit anderen Worten, wenn das Lernen einzeln oder zumindest ausserhalb der Gruppe stattfand, handelt es sich um ein Peer Assessment. Ein Self Assessment hingegen bezieht sich primär um den Lernenden selbst. Mithilfe des spezifischen Feedbacks kann ein Lernender seine Resultate kritisch hinterfragen und zudem seinen Fortschritt beobachten. Es wurde bewiesen, dass kritische Reflexionen einen positiven Einfluss auf den Lernprozess zeigen. Die Aufzeichnungen der Resultate kann als Anreiz bezwecken das Lernverhalten anzupassen (Roberts, 2006, S. 3-9).

3. Ist-Analyse Selevor

Damit die Problemstellungen und deren Lösungsansätze für das neue System (SEPAT) optimal umgesetzt werden können, wird zuerst eine kompakte Analyse des aktuellen Selevor Plug-Ins durchgeführt. Ziel der Analyse ist die Ermittlung von möglichen Programmabläufen, die für SEPAT zu verbessern beziehungsweise zu vermeiden sind. Ebenfalls wird die **User Experience** (UX) und Usability des Selevor Plug-In im Allgemeinen analysiert. Die essentiellsten Ergebnisse werden ins neue System übernommen.

3.1. User Experience

Die UX ist ein wesentlicher Bestandteil von erfolgreichen Webseiten und Applikationen (Häusel, 2016, S. 249). Mithilfe von erstellten Zielgruppen oder Personas kann eine spezifizierte Benutzerfreundlichkeit für das **User Interface** (UI) generiert werden. Die Entwicklung wird somit an die Personas ausgerichtet.

Durch eine optimale UX wird der Benutzer in seinen Gefühlen und Wahrnehmungen indirekt beeinflusst. Dabei gewinnt der Benutzer an Erfahrungen für die Software. Dies bewirkt positive Effekte auf den generellen Gebrauch der Software (Büsching & Goderbauer-Marchner, 2014, S. 99-100).

3.1.1. Umsetzung der UX in Selevor

Das Selevor Plug-In kann sich nicht gegen deren Konkurrenten im Gebiete der E-Learningplattformen durchsetzen. Die Aspekte der UX sind beim Selevor Plug-In vorhanden, jedoch nicht weiterentwickelt worden. Die fehlenden Aspekte liegen vor allem in der attraktiven Darstellung und Präsentation für die bestimmten Zielgruppen. Im Vergleich zu anderen E-Learning Plattformen wie «eCoach»¹, «QuizLet»², «Socrative»³ oder «Kahoot»⁴ ist die UX von Selevor erheblich eingeschränkt.

Die folgenden Abbildung 16 und Abbildung 17 dienen als Vergleich zwischen der Lernplattform Socrative und dem Selevor Plug-In. Anhand der Abbildungen wird aufgezeigt, wie die Benutzerfreundlichkeit zwischen den beiden Systemen variiert. In den Abbildungen wird der Unterschied mithilfe vom Prozess der Fragenerstellung dargestellt. Die Generierung der Fragen in Socrative erfolgt dynamisch ohne neue Pageloads. Hingegen beim Selevor

¹ Quelle: <http://ecoach.com/>, 06.07.2017

² Quelle: <https://quizlet.com/de>, 06.07.2017

³ Quelle: <https://b.socrative.com/login/teacher/>, 06.07.2017

⁴ Quelle: <https://kahoot.it/>, 06.07.2017

Plug-In wird bei der Funktionsauswahl «Neues Datenfeld anlegen» und bei der Speicherung der Frage eine neue Seite geladen.

Abbildung 16: Fragenerstellung in Socrative

#1

Geben Sie Ihren aktuellen Beruf ein.

EDIT

#2

Formatting: ☐ SAVE

Geschlecht?

ANSWER CHOICE

A Weiblich

B Männlich

CORRECT? ☐

Quelle: <https://b.socrative.com/login/teacher/>,

Abbildung 17: Fragenerstellung in Selevor

Selbstevaluation Info Einstellungen Fragen und Feedback bearbeiten Alle Resultate Rechte

<< Zurück zu den Frageblöcken Neues Datenfeld anlegen 1

DIE GRUNDLAGEN: FRAGEN BEARBEITEN
(1 - 4 von 4)

Name	Typ	Erforderliche Angabe	Aktionen
↑ geschlecht	Auswahlliste (feste Werte)	<input type="checkbox"/>	Aktionen ▾
↑ zivilstand	Auswahlliste (feste Werte)	<input type="checkbox"/>	Aktionen ▾

Selbstevaluation Info Einstellungen Fragen und Feedback bearbeiten Alle Resultate Rechte

HINZUFÜGEN 2

Name * Geben Sie Ihren aktuellen Beruf ein.

Typ * ☒ Textfeld (beliebige Werte)
☐ Auswahlliste (feste Werte)

Erforderliche Angabe ☐

* Erforderliche Angabe Speichern Abbrechen

Quelle: <https://ilias.unibe.ch/>

Im Anhang V ist eine Erstellung eines neuen Quiz der Plattform eCoach zu finden. Um einen weiteren Vergleich durchzuführen, sind im Anhang VI ebenfalls Screenshots einer Erstellung einer neuen Selbstevaluation des Selevor Plug-Ins aufgelistet. Die Screenshots widerlegen die Tatsache der optimierbaren UX in Selevor. Bei eCoach wird die Übersicht durch Bilder und Animationen erheblich aufgewertet. Hingegen bei Selevor besteht das Problem, dass zu viele Informationen auf einem UI vorhanden sind. Dem Anwender fällt es schwer, diese Informationen zeiteffizient zu verarbeiten. Auswirkungen hat dies vor allem auf die Usability.

3.2. Usability

Die Usability betrifft vor allem die Bedienungsfreundlichkeit einer Software. Nennenswert ist, dass die Bedienungsfreundlichkeit ein Bereich der UX darstellt. Parallel zur Benutzerfreundlichkeit, wird eine rationale logische Bedienung bevorzugt (Häusel, 2016, S. 249).

Anhand der Personas oder Zielgruppen wird die Usability der Software während den Spezifikationen angepasst. Beispielsweise in Form von Mockups vordefiniert und in einem nächsten Schritt in die Software eingebaut. Die wesentlichen Qualitätskomponenten, die gezielt in den Fokus treten, sind nach Nielsen (2012):

- Wie Lernfähig ist ein Benutzer beim ersten Versuch?
- Wie effizient ist ein Benutzer?
- Wie vergesslich sind die Benutzer, wenn diese die Software für längere Zeit nicht mehr benutzt haben?
- Wie hoch ist die Fehleranfälligkeit eines Benutzers? Findet er sich zurecht?
- Ist der Benutzer mit dem Design zufrieden?

3.2.1. Umsetzung der Usability in Selevor

Der wesentliche Nachteil beim Selevor Plug-In bezieht sich auf die Erstellung eines Fragebogens, welches gleichzeitig eine Kernaufgabe des Tools ist. Keine bestimmte Navigation zur Erstellung eines Fragebogens, die den Administrator durch das Programm führt, wird gewährleistet. Der Vorteil beim Selevor Programm besteht in der Aktivierung des Fragebogens. Falls der Fragebogen unvollständig ist, wird dem Anwender mitgeteilt, dass die Durchführung blockiert ist. In der Abbildung 18 ist die Mitteilung dargestellt.

Abbildung 18: Aktivierung einer unvollständigen Selbstanalyse

EINSTELLUNGEN BEARBEITEN

Titel * Metakognitive Strategien

Beschreibung Testumgebung mit Mockup-Test
Wie lernen Sie? Selbsttest zur Analyse von Lern- und Arbeitsstrategien.

Online ☒

Selbstevaluation Info Einstellungen Fragen und Feedback bearbeiten Rechte

Die Selbstevaluation kann nicht gestartet werden, da sie noch keine Fragen oder Skala enthält, das Feedback unvollständig oder die Selbstevaluation offline ist.

Quelle: <https://ilias.unibe.ch/>

Selevor unterstützt den Administrator die fehlenden Daten der Selbstevaluation nachzutragen. Unter «Fragen und Feedback bearbeiten» wird dem Administrator eine Kurzfassung über alle Fragenblöcke geboten. Anhand der Spalte «Feedback-Block vollständig» wird dem Administrator visuell gezeigt, dass einige Daten fehlen. Welche Daten angepasst oder hinzugefügt werden müssen, wird mithilfe der Spalten «Anzahl Fragen» und «Anzahl Feedbacks» mitgeteilt. Ein Beispiel ist in der Abbildung 19 zu entnehmen.

Die Benachrichtigung ist hilfreich, jedoch nicht optimal umgesetzt worden. Eine Selbstevaluation kann nicht Online geschaltet werden, falls diese nicht durchführbar ist. Zusätzlich sollte die Benachrichtigung spezifiziert werden. Zum Beispiel könnte mitgeteilt werden, welche Daten fehlen oder in welchen Bereichen man mit Problemen konfrontiert wird.

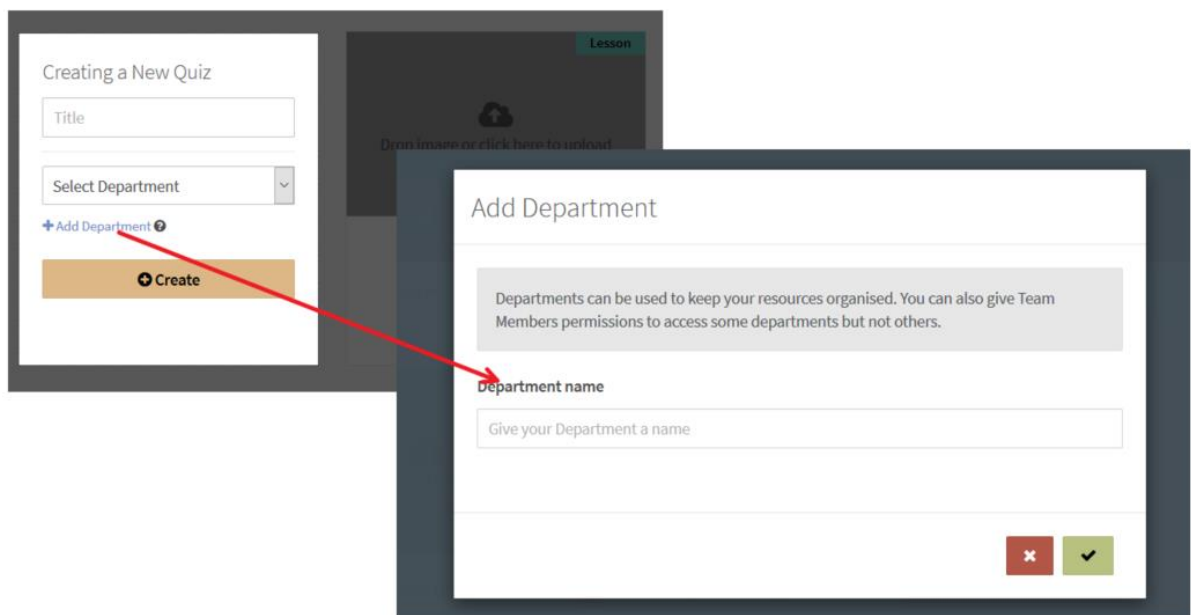
Abbildung 19: Übersicht der Fragenblöcke mit fehlenden Daten

FRAGEBLÖCKE (1 - 5 von 5)						
Reihenfolge speichern						
Titel	Abkürzung	Beschreibung	Anzahl Fragen	Anzahl Feedbacks	Feedback-Block vollständig	Aktionen
↑ Profildaten		Bitte geben Sie uns einige Hinweise. Ihre Daten werden vertraulich behandelt und unterliegen strengem Datenschutz	5	-	✓	Aktionen ▾
↑ Selbsteinschätzung	SWE	Hier geht es um Ihre generelle Selbsteinschätzung, wenn Sie Aufgaben angehen. Bitte versuchen Sie spontan aber ehrlich, eine Selbsteinschätzung auf allgemeiner Ebene vorzunehmen.	10	4	✓	Aktionen ▾
↑ Planen	PLN	Beantworten Sie jede Frage ehrlich aber spontan	5	4	✓	Aktionen ▾
↑ Überwachen	ÜBW	Beantworten Sie jede Frage ehrlich aber spontan	5	4	✓	Aktionen ▾
↑ Auswerten	AUW	Beantworten Sie jede Frage ehrlich aber spontan	5	4	✗	Aktionen ▾
Reihenfolge speichern						

Quelle: <https://ilias.unibe.ch/>

Ein weiterer Faktor in der Umsetzung von Selevor ist das allgemeine Seitenaufrufverhalten. Ein auffallender Faktor ist, dass fast keine Pop-Ups oder Dialogboxen existieren. Das Plug-In lädt bei jedem Aufruf eine neue Oberfläche. Selevor wirkt aus diesem Grund schwerfällig. Damit die Performance gesteigert werden kann und die Website dynamischer wirkt, helfen Pop-Ups. Diese können bei der Erstellung einer Selbstevaluation oder diversen Fragen eingesetzt werden. Für die visuelle Erklärung unterstützt die Abbildung 20, die sich auf die Erstellung eines neuen Quiz und Department der Seite eCoach bezieht.

Abbildung 20: Erstellung eines Quiz und Department auf eCoach



Quelle: <http://ecoach.com/>

Bei einem Klick auf «New Quiz» lädt ausschliesslich ein Pop-Up. Es wird nicht die gesamte Oberfläche aktualisiert. Falls ein neues Department hinzugefügt werden möchte, wird wiederum lediglich ein Pop-Up geladen. Die Performance ist dadurch optimiert und wirkt generell angenehmer auf den Anwender.

4. Der Kundenauftrag SEPAT

Im folgenden Kapitel wird der Kundenauftrag im Detail aufgezeigt. In einem ersten Schritt wird der Einsatzbereich der SEPAT Software definiert. Danach werden die grundlegenden Funktionen aufgelistet, die von Selevor übernommen werden sollen. Anschliessend wird die Grobarchitektur der Software illustriert, die Zielgruppen anhand fiktiven Personas definiert und die Problemstellungen des Kunden erläutert.

4.1. Wie kann SEPAT eingesetzt werden?

Der Auftraggeber Herr Tribelhorn definiert den Einsatzbereich der neuen Software wie folgt (siehe Anhang I):

«Stellen Sie einen Online-Selbsttest auf Ihrer Webseite zur Verfügung, der durch Ihre Zielgruppe ausgefüllt wird. Dazu erfassen Sie vorgängig einen Online-Fragebogen mit Hilfe entsprechender Analysefragen. Sie können der Klientel direktes Online-Feedback zur Verfügung stellen, nachdem der Onlinetest ausgefüllt wurde, inklusive einer grafischen Darstellung der Ergebnisse. Diese können durch die Klientel auch als PDF exportiert werden.»

SEPAT kann als multifunktionale Softwarelösung als Hilfsmittel für Schulungen eingesetzt werden. Hinsichtlich der Unabhängigkeit der SEPAT Software bezüglich anderen LMS Plattformen, ermöglicht die Software den Einsatz in diversen Bereichen. Beispielsweise kann die Schulung im Personalwesen von verschiedenen Unternehmungen erfolgen. Wie das genannte Beispiel in der Praxis umgesetzt wird, ist im Kapitel 4.3.1. Grobarchitektur von SEPAT detailliert erklärt.

4.2. Grundlegende Funktionen

Laut dem Kundenauftrag sollen die Komponenten des Selevor Plug-Ins übernommen und weiterentwickelt werden. Aus diesem Grund werden in den Prototyp folgende Funktionalitäten einbezogen:

- Fragenblock Management inklusive Analyse- und Stammdatenfragen
- Feedback Management
- Resultat Export in CSV
- Fragebogen Einstellungen
- Benutzermanagement

4.3. Erweiterungen

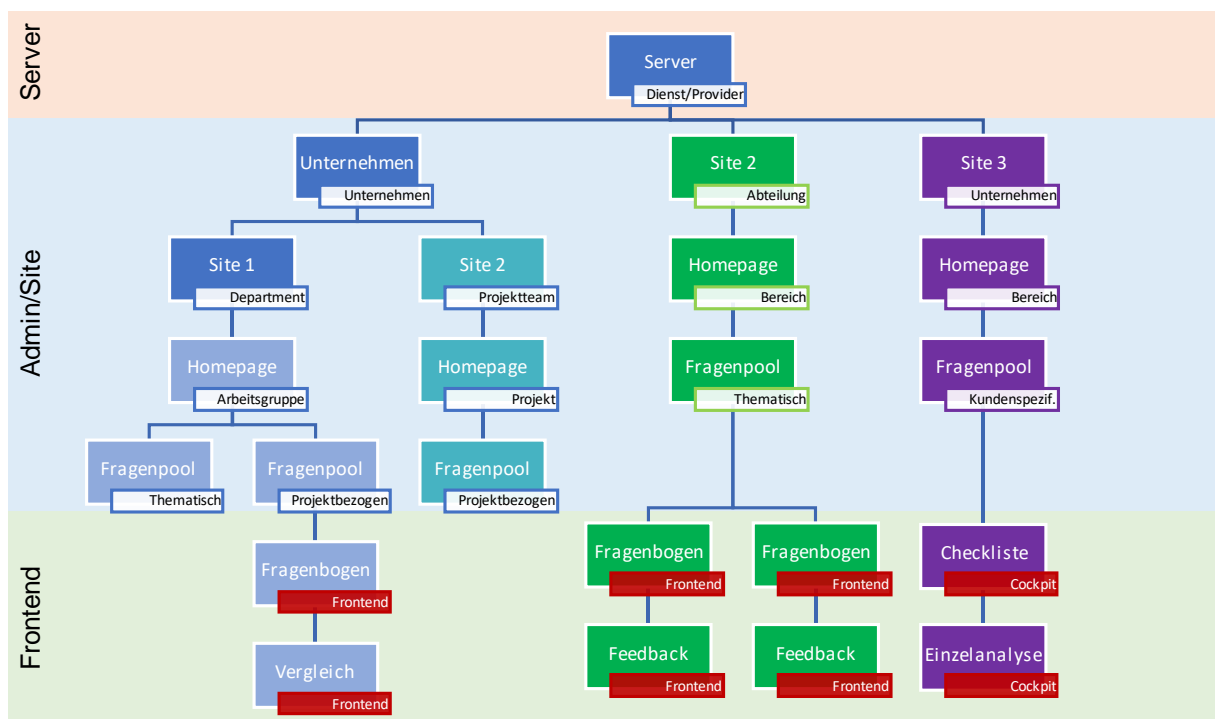
Die Erweiterungen des SEPAT Prototypen verändern die Architektur und einige grundlegenden Funktionen in deren Handlungen. Folglich werden die Erweiterungen und Modifizierungen, die für SEPAT zu übernehmen sind, in den nächsten Kapiteln erläutert.

4.3.1. Grobarchitektur von SEPAT

Der Einsatzbereich der SEPAT Software ist breiter ausgerichtet als diejenige des Selevor Plug-Ins. Erkennbar ist diese Tatsache aufgrund der Zielgruppen. Die Selevor Software ist im ILIAS System eingebunden, weshalb die Zielgruppen auf Dozenten und Studenten/Auszubildende der Universität Bern beschränkt sind. Demzufolge sind ausschliesslich Anwender des ILIAS Systems der Universität Bern berechtigt, den Zugriff auf das komplette Plug-In auszuüben. Anwender, die nicht im ILIAS System registriert sind, können lediglich eine Selbstevaluation durchführen.

Die SEPAT Software ist eine unabhängige Webapplikation. Dementsprechend sind keine Verbindungen zu einem LMS notwendig. Um die Funktionalität und Aufbau expliziter aufzuzeigen, unterstützt die Abbildung 21, welche die Grobarchitektur darstellt.

Abbildung 21: SEPAT Grobarchitektur Modell



Quelle: in Anlehnung an Anhang XI: Grobkonzept von SEPAT

SEPAT ist in drei Ebenen horizontal unterteilt. Es existieren eine Server-, Administrations- und Frontendebene. Wie diese Ebenen zusammengesetzt sind und miteinander interagieren, wird in den nachfolgenden Unterkapiteln erklärt.

Serverebene

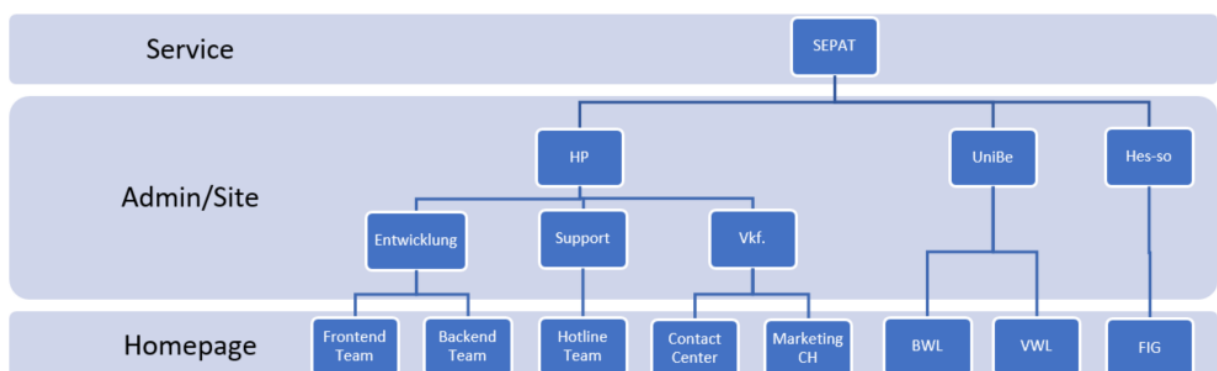
Die Serverebene gilt als Dienstprovider und verwaltet das System. In der Verwaltung sind Aktivitäten wie die Weiterentwicklungen, die Pflege und das Hosting einbezogen. Demzufolge sind auf der Serverebene keine potenziellen Kunden von SEPAT, sondern die Entwickler oder Supporter des Softwarebetreibers. Die Entwickler und Supporter auf der Serverebene werden Server-Administratoren genannt. Ein Server-Administrator verwaltet die vorhandenen und erstellt die neuen Sites. Es können beliebig viele Sites oder Unternehmen vom Server Administrator erstellt werden, weshalb die vertikalen Spalten auf der Sitemap in Abbildung 21 dynamisch ansteigen. Zusätzlich verfügt ein Server Administrator über eine Rolle mit Berechtigungen, die Zugriffe auf das gesamte System haben.

Site- / Administrationsebene

Auf der Administrationsebene befinden sich die erstellten Sites oder Unternehmen. Eine Site stellt eine unabhängige Umgebung dar. Daher enthält jede Site ihre eigene anpassungsfähige Oberfläche. Hingegen ein Unternehmen kann mehrere Sites umfassen. Die Erstellung eines Unternehmens ermöglicht bei grösseren Kunden die Strukturierung.

Die anpassungsfähigen Oberflächen sind in der Abbildung 21 als «Homepage» benannt worden. Um das Prinzip der Sites und Unternehmen verständlicher darzustellen, hilft die Abbildung 22 mit potentiellen Kunden des SEPAT Systems.

Abbildung 22: Erklärung der Sitemap mit spezifischen Beispielen



Quelle: eigene Darstellung in Anlehnung an Anhang XI: Grobkonzept von SEPAT

Eine Site besteht im produktiv System aus einer Organisation oder Gruppe von nicht juristischen Personen. Hinsichtlich bei einer Registration eines Unternehmens, soll die SEPAT Software die zugehörigen Sites strukturieren können.

Eine Homepage ist ein Bereich innerhalb einer Site. In der Abbildung 21 sind diese als Departments, Abteilungen oder Projekte abgebildet. Demzufolge hat jede Homepage ihre eigene Schulungsumgebung. Diese Schulungsumgebung beinhalten Fragenpools. Ein Fragenpool setzt sich aus mehreren Fragen innerhalb einer Homepage zusammen. Mit den erstellten Fragen im Fragenpool werden die Selbstevaluationen zusammengestellt. Zweck des Fragenpools ist die Modularität der Fragen in einer Site zu gewährleisten.

Frontendebene

Auf der Frontendebene findet die Interaktion zwischen dem Anwender und der durchgeführten Selbstevaluation statt. Das System lädt die generierten Fragebögen einer Selbstevaluation sowie die darauffolgenden Feedbacks auf die Frontendebene. Diese können per PDF Format exportiert werden.

4.3.2. Optimierte UX und Usability

Eine grundlegende Kundenanforderung ist die Umsetzung einer attraktiver UX und einer fortgeschrittenen bedienungsfreundlichen Usability des SEPAT Systems. Der Inhalt soll interaktiv gestaltet werden um möglichst viele Anwender auf die Webapplikation zu führen. Von Seiten des Kunden wurden Beispiele wie «playbuzz»⁵, «h5p»⁶ und «socrative» vorgeschlagen. Diese Beispiele sollen helfen, den Kundenauftrag in Bezug auf das UX der UI besser zu verstehen.

Die Mehrheit der Anwender sind Studenten oder Lernende, die Selbstevaluationen durchführen. Aus diesem Grund steht die Umsetzung eines Responsive Designs auf Seiten der Studenten und Lernenden im Vordergrund. Somit wird den Anwendern die Durchführung einer Selbstevaluation auf verschiedenen Geräteauflösungen ermöglicht.

4.3.3. Antwortmöglichkeiten der Stammdatenfragen

Die Antwortausrichtung bezieht sich auf die Auswahl der Antwortmöglichkeiten der Stammdatenfragen in einer Selbstevaluation. Bei den Analysefragen wird die Antwortausrichtung der vordefinierten Likert Skala bestehen bleiben. SEPAT soll für die Stammdatenfragen mehrere Antwortmöglichkeiten zur Verfügung stellen. Diejenigen Antwortausrichtungen aus dem Selevor Plug-In werden übernommen. Diese betreffen die

⁵ Quelle: <http://www.playbuzz.com/>, 08.07.2017

⁶ Quelle: <https://h5p.org/>, 08.07.2017

Textfeldeingabe und die Single Choice Option. Das SEPAT System wird mit zwei Antwortmöglichkeiten erweitert. Ein Dropdown-Feld, das auf dem Single Choice Prinzip beruht, und eine Multiple Choice Auswahl werden hinzugefügt. Die erwähnten Antwortmöglichkeiten sollen dynamisch durch den Administrator erstellt werden können.

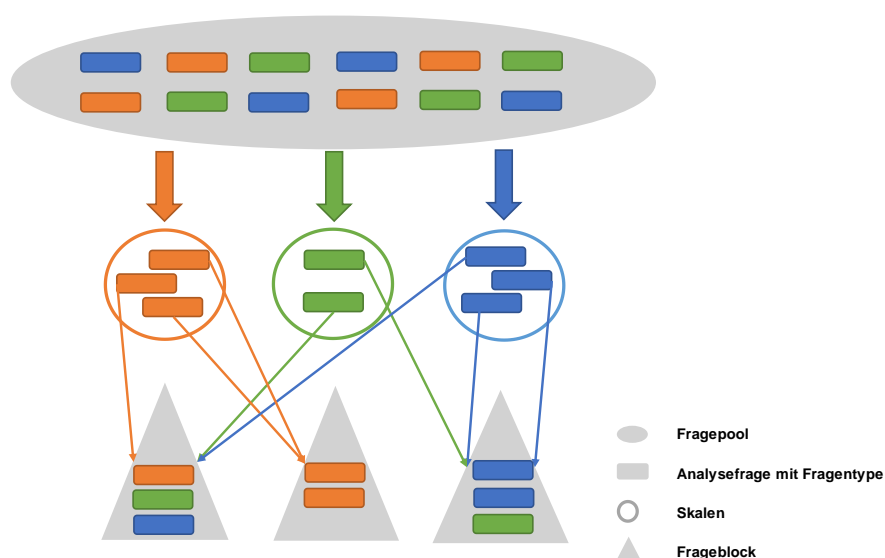
4.3.4. Fragenbogen und Skalen-Management

Im Vergleich zum Fragenbogen Management im Selevor Plug-In soll die Funktionalität im SEPAT System erweitert werden. Primär handelt es sich um die Präsentation auf dem Frontend während der Durchführung einer Selbstevaluation. Sekundär sollen die Erstellung und die Berechnung der Ausprägungen vom aktuellen Fragenbogen Management auf das Skalen-Management erweitert werden.

Beim Fragenbogen Management in Selevor können mittels der Drag & Drop Funktion die Reihenfolge der Fragenblöcke definiert werden. Die Berechnungen der einzelnen Ausprägungen werden dadurch nicht beeinträchtigt und die Fragen sind statisch einem Fragenblock zugeteilt (siehe Kapitel 2.3.5. Feedback).

Das SEPAT System soll ein Skalen-Management beinhalten, welches das Fragenbogen Management des Selevor Plug-In in dessen Funktionalität erweitert. Das Skalen-Management betrifft die Präsentation der Fragenblöcke und das Management der Analysefragen im SEPAT System. Die Abbildung 23 hilft die erweiterte Methodik des Skalen-Management zu verstehen.

Abbildung 23: Diagramm des Skalen Management



Alle Analysefragen, die von den Administratoren generiert werden, sind im Fragenpool abgespeichert. Nach der Erstellung einer Analysefrage können diese einer oder mehreren Skalen zugeteilt werden. Anhand der gekennzeichneten Farben einer Analysefrage sowie Skala, können diese auf die spezifischen Fragentypen verwiesen werden. Ein Fragentyp

bezieht sich auf ein individuelles Thema. Beispielsweise enthält der Fragenpool mehrere Fragentypen. Diese Fragentypen ermöglichen die Unterteilung von thematisch zusammengehörigen Fragen. Ein Fragenpool einer Marketing Site könnte Fragentypen wie Marketing Konzepte, Konkurrenzanalyse oder Grundlagen Marketing umfassen.

Eine Skala besteht somit aus mehreren Fragen und einem Fragentypen. Diese Skalen beinhalten deren eigene Likert Skala und bewirken eine Modularität der Analysefragen für zukünftige Selbstevaluationen. Des Weiteren sind die Frageblöcke unabhängig der Skalen.

Es soll eine Skala direkt als Fragenblock genutzt oder die Analysefragen von verschiedenen Skalen auf diverse Fragenblöcke aufgeteilt werden können. Die Abbildung 23 zeigt die Zuteilung der Analysefragen zu den Fragenblöcken. Dies soll einerseits manuell mittels Drag & Drop durchgeführt werden können. Andererseits soll das System eine automatische Generierung durchführen können. Aus diesem Grund soll eine direkte Erstellung einer Skala als Fragenblock gewährleistet werden. Diese Kriterien müssen bei der Erstellung einer Selbstevaluation in den Vordergrund gestellt werden

Der Vorteil der Wiederverwendbarkeit liegt in der Verteilung von gebündelten Analysefragen aus Skalen über diverse Fragenblöcke. Als Folge davon können dem Anwender in einem Fragenblock abwechslungsreiche Analysefragen von grundverschiedenen Fragentypen geboten werden. Dies hat einen enormen Einfluss auf das Durchführungsverhalten eines Anwenders.

In der Abbildung 23 sind die Stammdatenfragen nicht miteinbezogen. Im neuen SEPAT System wird exakt ein Stammdatenblock der Selbstevaluation zugeteilt. Der Stammdatenblock kann als Fragenblock dargestellt werden. Ebenfalls ermöglicht das Skalen-Management die Kombination von Stammdatenfragen mit unterschiedlichen Analysefragen in einem Fragenblock.

4.3.5. PDF Export

Spezifische Daten des SEPAT Systems sollen als PDF und CSV exportierbar sein. Primär betrifft der PDF Export die Auswertung, die am Ende einer individuell durchgeführten Selbstevaluation erscheint. Nach der Durchführung soll dem Anwender der PDF Export einer Auswertung ermöglicht werden, der die generierten Graphen und Diagramme exportiert.

Ein Administrator einer Selbstevaluation soll alle Resultate von durchgeführten Selbstevaluation im CSV Format exportieren können. Dieser CSV Export umfasst sämtliche Informationen vom Anwender und der erlangten Daten der Selbstevaluation. Die Daten der Selbstevaluation umfassen die Antworten der Analysefragen und der Stammdatenfragen. Informationen der einzelnen Selbstevaluation sind ebenfalls von Bedeutung, wie beispielsweise ein Identifikator des Anwenders sowie der Zeitpunkt der Ausführung und des Abschlusses. Zusätzlich können Server- oder Site Administratoren Listen der Anwender innerhalb deren Sites als PDF oder CSV exportieren.

4.3.6. Drittanbieter Authentifikation

Die Authentifikation auf SEPAT soll anhand von Drittanbieter Konten optimiert werden. Drittanbieter Konten sind zum Beispiel «Google»⁷, «Facebook»⁸, «Twitter»⁹ oder «LinkedIn»¹⁰. Mithilfe einer registrierten Email Adresse der erwähnten Social Media Konten, soll der Anwender sich auf SEPAT registrieren können. Für den Prototyp SEPAT ist die Umsetzung einer Google Authentifikation priorisiert.

4.3.7. Selbstevaluation Management

Eine bedienerfreundliche Lösung zur Erstellung und der Pflege einer Selbstevaluation ist beauftragt worden. Aus diesem Grund wird ein Autorenwerkzeug konzipiert, das die Zielgruppen weitgehend bei der Verwaltung der Selbstevaluationen unterstützt. Primär ist ein Lösungsvorschlag zur Usability der Erstellung einer Selbstevaluation zu berücksichtigen (Tribelhorn, Erstes Meeting: Kundenauftrag spezifizieren, 2017).

4.3.8. Gruppen Management

Das Gruppen Management gewährleistet eine erweiterte Nutzung der Auswertung bezüglich der Peer- und Group Assessments. Deswegen sollen Gruppen von Anwendern, durch einen Site-Administrator oder Master aus der Administrationsebene, gebildet werden können. Diese Gruppen bezwecken den Auswertungsvergleich der Peers am Ende einer

⁷ Quelle: <https://myaccount.google.com/intro>, 09.07.2017

⁸ Quelle: <https://www.facebook.com/login>, 09.07.2017

⁹ Quelle: <https://twitter.com/login>, 09.07.2017

¹⁰ Quelle: <https://www.linkedin.com/>, 09.07.2017

Selbstevaluation. Die verschiedenen Auswertungen sollen in einer zusammengefassten Version für den Gruppenleiter ersichtlich sein.

4.3.9. Anonyme Durchführung

Die anonyme Durchführung bezieht sich auf die potenziellen Anwender, die an Selbstevaluationen, wie beispielsweise Strassenbefragungen, teilnehmen. Diese bleiben anonym, allerdings sind die erhaltenen Daten von Wichtigkeit.

4.4. Personas

Im folgendem Abschnitt werden vier Personas im Detail beschrieben. Diese Personas bilden die potentiellen Zielgruppen ab und sollen die Verständlichkeit der Problemstellung und deren Lösungsansätze unterstützen. Zudem werden diese die Kommunikation zwischen den einzelnen Parteien in einem IT-Projektteam sowie gegenüber den Stakeholdern aufwerten. (Cooper et al. 2014, S. 64).

«Eine Persona ist ein hypothetischer Nutzer mit konkret ausgeprägten Eigenschaften und Vorlieben sowie einem konkreten Nutzungsverhalten. Dabei steht eine Persona repräsentativ für eine reale Benutzergruppe, die ein System nutzt, beispielsweise eine Software, App oder Website. Persona sind ein Konzept aus der Informatik, genauer gesagt aus dem Teilgebiet Mensch-Computer-Interaktion, und kommen daher vor allem in der Softwareentwicklung zum Einsatz.» (Jendryschik, 2010)

Die nachfolgenden Personas lehnen sich an die Sitemap in der Abbildung 21 an. Auf der Sitemap sind drei Ebenen vorhanden. In der Ebene der Administration/Site sind wiederum zwei Unterebenen inbegriffen. Infolgedessen werden vier Personas erstellt. Für jede Ebene und Unterebene jeweils eine Persona, die ebenfalls als eine Anwenderrolle fungiert. Aus diesem Grund existieren auf der Administrations-/Siteebene die Rollen des Site-Administrators und Masters, auf der Serverebene der Server-Administrator und auf der Frontendebene der User.

Es existieren ein Server Administrator, Site Administrator, Master und User. Die Personas wurden in Anlehnung des Buches von Kuhn Christian (2013) generiert.

4.4.1. System Administrator: Ralph Dupont

Abbildung 24: Persona Ralph Dupont



Geburtsdatum:	28. Januar 1980
Wohnort:	Olten, Schweiz
Hobbys:	Netflix, Programmieren, Mit Freunden ausgehen
Zivilstand:	ledig
Stärken:	Abstraktes logisches Denkvermögen, Softwareentwicklung

Sein Statement: «Don't trust anyone who can't code!»

Leben & Wohnen

Herr Dupont lebt im Zentrum von Olten in einer 2.5-Zimmer Wohnung. Seine Wohnung ist unmittelbar neben dem Bahnhof. Sein berufliches Leben nimmt ihm sehr viel Zeit wodurch er abends meistens zu Hause bleibt und Fernsehsendungen und Dokumentation schaut. Während den Wochenenden wird ihm nie langweilig. Wenn er nicht an eigenen Softwareprojekten arbeitet, unternimmt er mit seinen engsten Freunden Aktivitäten.

Beruf & Karriere

Herr Dupont absolvierte einen Bachelor in Informatik und einen darauffolgenden Master in Wirtschaftsinformatik an der Fachhochschule Nordwestschweiz in Brugg. Nach seinem erfolgreichen Studium wurde er von der Unic AG in Bern als Junior Software Entwickler eingestellt. Bei der Unic AG wurde er zum IT-Projekt Manager befördert. Nach fünf Jahren wollte er neue Erfahrungen sammeln, weshalb er zur SEPAT AG als System Administrator wechselte. Er verwaltet somit die gesamte SEPAT Software.

Technologie & Medien

Herr Dupont ist bestens mit den Technologien vertraut und nutzt täglich mehrere Geräte.

4.4.2. Site Administrator: Manfred Albrecht

Abbildung 25: Persona Manfred Albrecht



Geburtsdatum:	2. Mai 1968
Wohnort:	Uster, Schweiz
Hobbys:	Lesen, Radfahren, Reisen
Zivilstand:	verheiratet, 3 Kinder
Stärken:	Geduldig, Organisationstalent
Schwächen:	IT Kenntnisse

Sein Statement: «Vertrauen und Geduld sind die grundlegendsten Faktoren für eine gesunde Entwicklung!»

Leben & Wohnen

Herr Albrecht wohnt zusammen mit seiner Ehefrau Benita und seinen drei Kindern in Uster im Kanton Zürich. Ihr Einfamilienhaus liegt in einem ruhigen Wohngebiet nahe der Waldgrenze. Seine Ehefrau arbeitet mit einem Arbeitspensum von 20% als Sachbearbeiterin in der Abteilung der Administration der Stadtverwaltung Zürich. Zwei seiner Kinder besuchen zurzeit das Gymnasium und das älteste Kind studiert Psychologie an der Universität Zürich. Herr Albrecht unternimmt in seiner Freizeit viel mit seinen Kindern, da ihm deren Entwicklung und die Familie sehr am Herzen liegen.

Beruf & Karriere

Manfred Albrecht arbeitet als Abteilungsleiter für die Weiterbildung des internen Personals der SEPAT AG Schweiz in Zürich. Er ist bereits seit über zehn Jahren bei diesem Softwareentwicklungsunternehmen angestellt. Ihm gefällt die Zusammenarbeit mit Jung und Alt. Neue Herausforderungen kann er gut im Team erarbeiten und lösen.

Herr Albrecht studierte an der Universität Bern Wirtschaftswissenschaften mit Vertiefung in Psychologie. Nach dem Bachelor Zertifikat absolvierte er erfolgreich das Masterstudium in Psychologie. Vor der Anstellung bei der SEPAT AG arbeitete Herr Albrecht bei der UBS in Zürich als Abteilungsleiter und Koordinator des Human Ressource Departement.

Technologie & Medien

Während seinem Arbeitstag benutzt Herr Albrecht einen PC, den ihm die Firma zur Verfügung gestellt hat. Aktuelle Informationen holt sich Herr Albrecht über Fachzeitschriften und Wirtschaftszeitungen. Er ist bereit, neue Programme und Geräte zu verwenden.

4.4.3. Master: Sophie Bracher

Abbildung 26: Persona Sophie Bracher



Geburtsdatum:	29. Juli 1987
Wohnort:	Zürich, Schweiz
Hobbys:	Reisen, Yoga, Lifestyle
Zivilstand:	ledig
Stärken:	Networking, Planung, Belastbar, Extrovertiert
Schwächen:	IT-Kenntnisse, ungeduldig

Ihr Statement: «Kommunikation ist das Wichtigste für ein angenehmes Arbeitsklima!»

Leben & Wohnen

Frau Bracher lebt seit ihrer Kindheit in der Stadt Zürich. Zurzeit wohnt sie mit Ihrem Freund Gabriel Winkelmann in einer 3.5-Zimmer Wohnung direkt an der Küste am Zürcher See. In der Freizeit unternimmt das Paar diverse Aktivitäten wie Shoppen, Schwimmen und Ausgehen.

Beruf & Karriere

Ihr Bachelor Studium in Wirtschaftswissenschaften mit Vertiefung in BWL absolvierte die gelernte Kauffrau EFZ an der Universität Bern. Nach Beendigung des Studiums reiste sie ein Jahr lang in Begleitung mit ihrem Freund um die Welt. Nach der Reiserückkehr fand sie bei der SEPAT AG in Zürich eine Arbeitsstelle. Ihre Arbeit besteht darin, gezielte Weiterbildungen für das Personal der Buchhaltungsabteilung durchzuführen.

Technologie & Medien

Frau Bracher setzt täglich einen Laptop für ihre Arbeiten ein. Gelegentlich benutzt sie den Laptop für private Aktivitäten wie die Pflege ihres Facebook Accounts. Als Informationsquelle benutzt sie das Internet, Zeitschriften und Zeitungen.

4.4.4. User: Dennis Hofener

Abbildung 27: Persona Dennis Hofener



Geburtsdatum:	12. November 1992
Wohnort:	Bern, Schweiz
Hobbys:	Joggen, Skifahren, Ausgehen mit Freunden
Zivilstand:	ledig
Stärken:	Lernfähig, Geduldig
Schwächen:	Schüchtern, Nervös

Sein Statement: «Von Nichts kommt Nichts!»

Leben & Wohnen

Herr Hofener wohnt seit kurzem in einem kleinen Appartement in Bern, da er nach seiner Ausbildung unabhängig von seinen Eltern sein wollte. Bis zum Studium wohnte er in einem kleinen Dorf im Emmental. Er geht in seiner Freizeit viel aus, damit er neue Leute kennen lernt.

Beruf & Karriere

Nach seiner zwei jährigen Bankenlehre bei der Credit Suisse in Burgdorf wurde er von der SEPAT AG als Kreditor-Buchhalter eingestellt.

Technologie & Medien

Herr Hofener ist neugierig und offen für neue Geräte wie Smartphones und Tablets. Für private Zwecke benutzt Herr Hofener ein Galaxy S8 und einen Windows Laptop um sich auf dem aktuellsten Stand zu halten. Er liest Zeitungen in Form von E-Books und tauscht sich täglich mit Freunden auf diversen Social Media Plattformen wie Facebook oder Twitter aus.

5. Planungs- und Spezifikationsphase

In der Planungs- und Spezifikationsphase werden die umzusetzenden Funktionalitäten festgelegt und priorisiert. Nach der Priorisierung werden die Softwarespezifikationen unter Verwendung von Mockups mit Herrn Tribelhorn spezifiziert.

5.1. Priorisierung

Die Festlegung der Funktionalitäten wurden vom Auftraggeber bereitgestellt. Die Liste der Funktionalitäten, ist im Anhang III ersichtlich. Ausserdem zeigt die Liste die Berechtigungen der einzelnen Anwenderrollen auf.

Nach der Besprechung dieser Liste, ist eine Priorisierung durchgeführt worden. Die Priorisierung bezweckt bei der Implementation die schrittweise Abarbeitung der wichtigsten Funktionen. Dies war eine notwendige Massnahme, da sich die Implementationsphase auf den Zeitraum vom 22.05.2017 bis 26.06.2017 beschränkte.

Die Priorisierung wurde ebenfalls mit dem Auftraggeber abgestimmt. Diese wurde mithilfe der Funktionenliste vorgenommen. Der Fokus für die Bestimmung der Prioritäten wurde auf die Bedürfnisse des Kunden sowie dem ungefähren Entwicklungsaufwand pro Funktionalität gelegt. Aus dieser Zusammenarbeit resultierte eine Liste der Prioritäten, die in der Abbildung 28 illustriert ist.

Abbildung 28: Priorisierung der Funktionalitäten

Priorisierung der Funktionalitäten	
1	User Management
1.1	Login
1.2	Registration
1.3	Google Authentifikation
2	Fragenpool Konfigurationen
2.1	Fragenpools
2.2	Fragetypen
3	Komponenten der Selbstevaluation
3.1	Analysefragen
3.2	Stammdatenfragen
3.3	Skalen Management
3.4	Stammdatenblöcke
4	Selbstevaluation erstellen
4.1	Sequenzen
4.2	Frageblöcke
4.3	Evaluationssessionen
4.4	Evaluationseinstellungen
5	Auswertungen generieren
5.1	Auswertungen speichern
5.2	Auswertungen anzeigen
6	PDF/CSV Export
6.1	CSV Export aller Daten einer Selbstevaluation
6.2	PDF Export einer Auswertung
7	Gruppen Management
7.1	Gruppenverwaltung
7.2	Gruppenauswertung
8	Anonyme Durchführung
9	Site UI anpassen
10	Homepage UI anpassen

Die Datei der Funktionenliste enthält Register mit den jeweiligen Funktionalitäten. Diese Register wurden vorwiegend übernommen und somit als Blöcke mit deren Unterfunktionen gruppiert. Im Detail bedeutet dies, dass beispielsweise eine Analysefrage die **Create-Read-Update-Delete** (CRUD) Funktionen beinhalten soll. Dies sind die Basisoperationen für die persistente Datenspeicherung (Stringfellow, 2017).

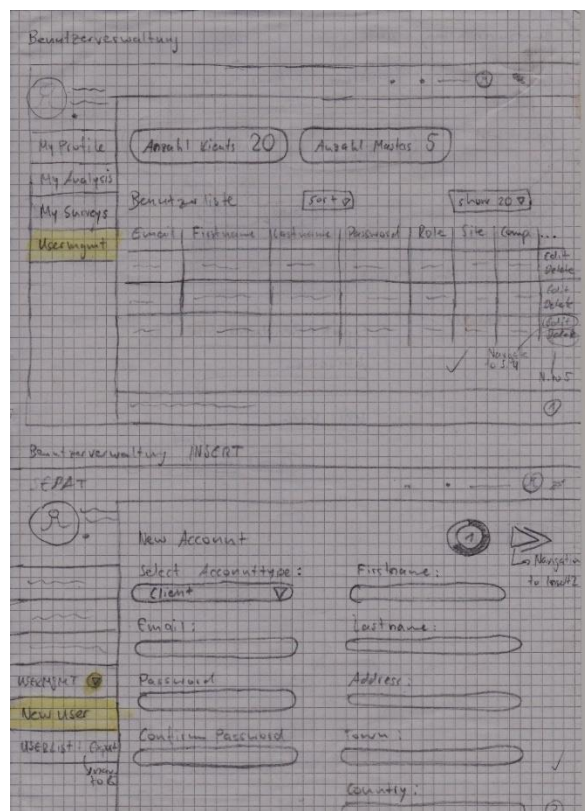
In einem nächsten Schritt werden die Priorisierungen mit der Liste der Funktionalitäten zusammengeführt um eine Softwarespezifikation durchzuführen.

5.2. GUI-Spezifikation

Als Softwarespezifikation wird die Methode der GUI-Spezifikation gewählt. Das Graphical User Interface (GUI) bedeutet in das Deutsche übersetzt «Grafische Benutzeroberfläche». Ein GUI besteht aus grafischen Elementen einer Software, die mit dem System interagieren mittels der Eingaben des Anwenders. Elemente einer GUI sind beispielsweise Fenster, Icons, Buttons oder Textfelder (Schultens, 2010).

Die GUI-Spezifikation wird gewählt, damit dem Kunden ein erster visueller Einblick in die Umsetzung garantiert werden kann. Aus diesem Grund werden Mockups angelegt, die den Softwareverlauf simulieren sollen. In der Abbildung 29 ist ein Beispiel eines erstellten Mockups dargestellt. Im Anhang VIII sind alle erarbeiteten Mockups ersichtlich.

Abbildung 29: Mockup Account Management



Ein anderer Faktor, der zur Wahl der GUI-Spezifikation beiträgt, ist die Unterstützung während der Implementierungsphase. Schrittweise kann die Software erstellt und getestet werden. Bei einem erfolgreichen Abschluss eines Tests wird die nächst höhere Priorisierte Funktion im GUI implementiert. Die Vorgehensweise der Entwicklung ist somit nach der agilen Softwareentwicklungsmethode Kanban ausgerichtet worden.

Nach der Fertigstellung der Mockups wurden diese zusammen mit dem Auftraggeber analysiert. Der Auftraggeber konnte seine Ansichten und Bedürfnisse in die Gestaltung der Mockups einfließen lassen. Die vorgeschlagenen Veränderungen konnten besprochen und in Kooperation in die Mockups einbezogen werden.

5.2.1. Mockups Aufbereitung

Die Mockups werden unter Berücksichtigung der vordefinierten Personas und der Prioritätenliste erstellt. In der Prioritätenliste sind überwiegend Funktionalitäten eines Site Administrators zu erkennen. Demzufolge basieren die Mockups grösstenteils auf den Funktionalitäten eines Site Administrators.

Aus den Mockups im Anhang VIII oder der Abbildung 29 ist ersichtlich, dass in den GUI's durchgehend das Dashboard eingeblendet ist. Das Dashboard dient zur einheitlichen Übersicht der Funktionalität des Server-, Site Administrators und des Masters. Für einen User bezweckt das Dashboard keinen vorteilhaften Nutzen, weil dieser lediglich Selbstevaluationen starten, Auswertungen analysieren und seine persönlichen Daten pflegen kann. Die drei anderen Personas können wiederum die Einstellungen der Selbstevaluationen beziehungsweise Sites und Homepages verwalten.

Von den drei Personas ist lediglich der Server Administrator ein IT-Fachmann. Die Altersgruppen zwischen 25 bis 30 Jahren der Personas sind mit den neuesten Geräten und Technologien bekannt und sind ausserdem lernfähig. Der Site Administrator ist jedoch eingeschränkt in der Nutzung und im Wissen der IT und bevorzugt deswegen eine einfache und unkomplizierte UX.

Für die Personas Users und Masters ist die Gewichtung der UX von Bedeutung. Eine benutzerfreundliche Oberfläche hilft ihnen, Prozesse besser wahrzunehmen. Diese Tatsache kann ebenfalls für den Server-Administrator mit einem besseren IT-Wissen betreffen. Im Kontrast zu den Altersgruppen von 25 bis 30, die eine animationsreiche Oberfläche bevorzugen, stehen die Personas der Serverebene und der Siteebene. Vielmehr stellen diese Personas die Usability in den Vordergrund. Bei den unerfahreneren, beispielsweise dem Site Administrator, kann eine zu spielerische Oberfläche verwirrend wirken. Ebenfalls bei

professionellen Anwendern kann dieser Zustand negative Auswirkungen auf deren Bedienung haben. Es besteht die Gefahr der Übervereinfachung für einen Server-Administrator.

Aufgrund dieser Merkmale, die aus den Personas evaluiert werden konnten, musste eine Kombination einer angenehmen UX und einer fortgeschrittenen Usability ausgearbeitet werden. Somit wurden bei der Erstellung der Mockups folgende Umsetzungsmethoden berücksichtigt:

1. Ein Mockup auf die wichtigsten Daten begrenzen.
2. Ein Mockup gezielt mit Icons und Bilder zu vereinfachen.
3. Ein Mockup mit einer wiedererkennbaren Prozessstruktur auszustatten.

Die erste Umsetzungsmethode soll den unerfahreneren Anwendern die wichtigsten Informationen auf jeder Oberfläche bieten. Folglich müssen diese nicht nach den gewünschten Daten und Funktionen suchen, weshalb die Anzahl der Klicks minimal gehalten werden.

Die CRUD Funktionen beinhalten vier Aktionen. Aus der Funktionenliste ist ersichtlich, dass diese wiederholt auftreten. Damit die ineffiziente Ausführung dieser Funktionen vermieden wird, sind Icons unumgänglich. Die Vermeidung von Untermenüs steht im Vordergrund aufgrund der enormen Anzahl der Aktionen für alle Komponenten. Die Komponenten wie zum Beispiel die Analyse-, Stammdatenfragen, Stammdatenblock oder Skala beinhalten jeweils die CRUD Funktionen. Eine Auflistung all dieser CRUD Funktionen verursacht ein unübersichtliches Dashboard.

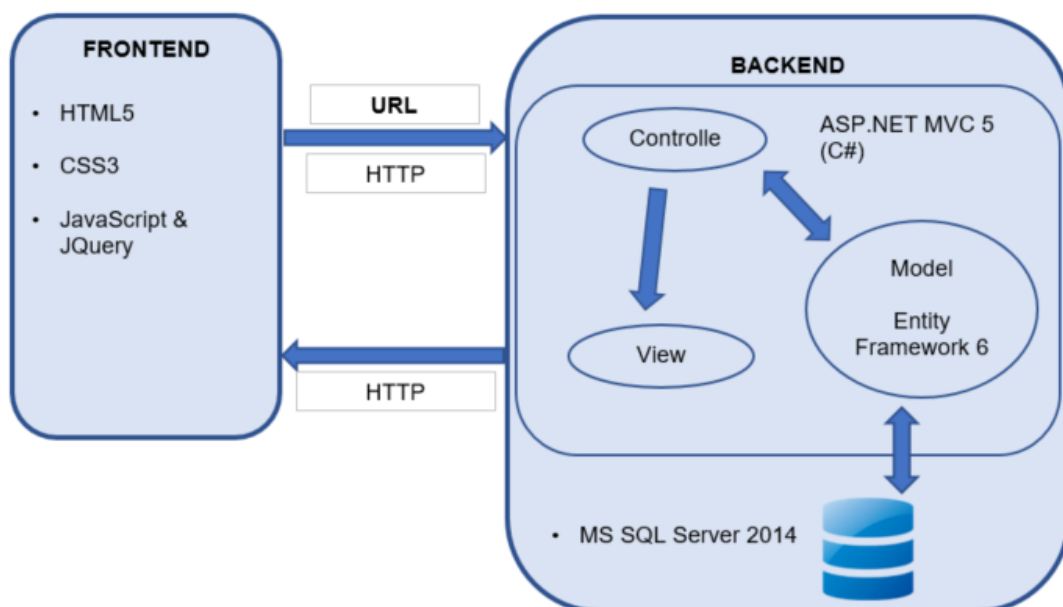
Die Platzierung der CRUD Funktionen auf den Oberflächen soll einheitlich über die Software dargestellt werden. Infolgedessen können die Anwender Prozessschritte in anderen Einsatzgebieten anwenden. Die Verwaltung der Stammdatenfragen wird ähnlich ausgerichtet sein wie diejenige der Analysefragen.

6. Systemarchitektur

In diesem Kapitel wird die Systemarchitektur mit den technischen Umsetzungen erläutert. Mit den technischen Umsetzungen werden die gewählten Technologien in ihrem Einsatzbereich aufgezeigt und weshalb diese gewählt worden sind.

Die Architektur der Software besteht aus zwei Schichten. Das Frontend ist die Programmschicht, die für den Anwender sichtbar ist und mit der er interagiert. Das Backend ist für die Programmlogik zuständig. Diese zwei Anwendungsschichten sind in der Abbildung 30 mit deren Technologien dargestellt. Die Abbildung wurde in Anlehnung der Microsoft Dokumentationen von ASP.NET MVC erstellt (Microsoft Corporation, kein Datum).

Abbildung 30: Die modellierte Softwarearchitektur



Die SEPAT Software ist eine Client-Server Architektur und besteht aus zwei Komponenten. Die Client Komponente ist in der Abbildung 30 als das Frontend und die Server Komponente als Backend zu verstehen. Mehrere Clients können mittels einen HTTP (Hypertext Transfer Protocol) Request über das Netzwerk auf die Programmlogik und Datenverarbeitung des Webserver zugreifen. Dieser nimmt die Anfragen entgegen, verarbeitet diese und sendet mittels einem HTTP Response die Antwort zu dem anfragenden Client gegebenenfalls zurück (Rouse & Ligan, 2015). Das Backend stellt im SEPAT System den Web- und Datenbankserver zur Verfügung.

Der Webserver kommuniziert mit dem **Datenbank-Management-System (DBMS)** um Daten zu lesen oder zu speichern. Das DBMS verwaltet die Anfragen der Anwender und bezweckt die Integrität der Daten (Rouse & Christiansen, 2014).

Der Thin Client ist vollumfänglich abhängig von der Serverlogik, weil die ganzen Verarbeitungen und Berechnungen auf dem Server stattfinden. Thin Clients sind auf eine stabile Netzwerkverbindung zum Server angewiesen. Diese Clients kommunizieren anhand eines Windows Terminal, das beispielsweise für Remote-Applikationen erstellt wurde (Thin-Client, 2013).

Im Gegensatz zum Thin Client existiert der Thick Client. Bei einem Thick Client wird die Programmlogik sowie die Datenhaltung auf dem Client ausgeführt. Thick Clients sind häufig in Windows Applikationen anzutreffen. Diese Applikationen werden meistens auf einem PC vorinstalliert. Zusätzlich weisen Thick Clients einen höheren Auslastungsgrad der lokalen Ressourcen auf als die Thin Clients. Das Hauptunterschiedsmerkmal ist erkennbar an der Anwendungsschicht, auf dem der Datenprozess durchgeführt wird (Welk, 2009, S. 35-37).

Der Client des SEPAT System enthält eine minimale Programmlogik, die Daten verarbeiten muss. Demzufolge handelt es sich um einen Hybrid Client, der ausschliesslich zur Anzeige und Validation auf der Frontendebene zuständig ist. Die Anzeige und Validation wird mittels eines lokalen installierten Browsers, wo die Scripts ausgeführt werden, gewährleistet. Der Hybrid Client ist eine Zwischenstufe des Thin und Thick Clients.

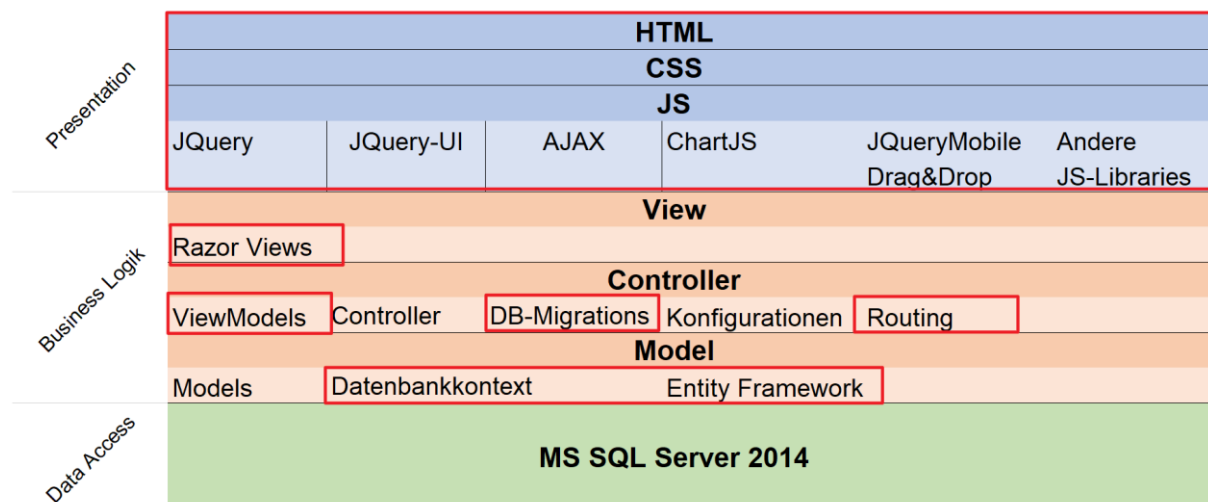
Aus der Abbildung 30 sind die Eigenschaften des Hybrid Clients anhand der Verteilung der Technologien klar verdeutlicht worden. Die Technologien wie JavaScript (JS), HTML5 und CSS3 sind für die Anzeige und Validationen der Eingaben zuständig. Diese interagieren mittels dem URL Routing mit dem Backend, wo die Daten verarbeitet und bezogen werden.

6.1. Grundarchitektur der Software

Im Kapitel der Grundarchitektur wird die Systemarchitektur detaillierter beschrieben. Das Ziel von diesem Kapitel bezieht sich auf die Visualisierung der Grundlagen und deren Einsetzung in den Technologien.

Die Abbildung 31 verdeutlicht ein klassisches Schichtenmodell, das auf einer Drei-Schichten-Architektur beruht. Es besteht eine Präsentations-, Business (Logik) und Datenschicht. Diese Schichten tauschen miteinander Daten aus. Beispielsweise werden die Benutzerinteraktionen auf der Präsentationsschicht aufgenommen und weiter zur Businessschicht geleitet. Auf der Businessschicht werden die Daten validiert. Nach der Datenmanipulation können diese wiederum an die Datenschicht versendet werden, welche die physische Datenbank anpasst (Mrknowing, 2013).

Abbildung 31: Grundarchitektur und die wichtigsten Komponenten des ASP.NET MVC



In der Grundarchitektur der Abbildung 31 werden die wichtigsten Komponenten der Technologien verdeutlicht. Im Weiteren werden die Grundlagen der gewählten Technologien erklärt. Die Abbildung soll das Verständnis der eingesetzten Technologien unterstützen.

6.2. Frontend

6.2.1. HTML5 & CSS3

Die englische Abkürzung von HTML bedeutet «**H**ypertext **M**arkup **L**anguage» und die Fünf steht für die aktuellste Version. HTML ist eine textbasierte Auszeichnungssprache. Eine Auszeichnungssprache definiert die Struktur einer Webseite. Dementsprechend können Textstrukturierungen durchgeführt werden, die über das Netzwerk erreichbar sind und mithilfe eines Internet Browser angezeigt werden können (Lackers & Siepermann, kein Datum).

Die **C**ascading **S**tyle **S**heets (CSS) beschreiben und definieren die Darstellungen der HTML Dokumente. Die Stylesheet Sprache gilt deshalb als Formatierungs- und Gestaltungssprache. Zusammen mit der HTML Auszeichnungssprache bilden diese die meistgenutzten Technologien im World Wide Web (Kirchbergerknorr, kein Datum).

Für die Darstellung der Oberflächen des Frontend wird bewusst die Versionen HTML5 und CSS3 gewählt. HTML5 und CSS3 sind für die Umsetzung der Auswertungen von Bedeutung, da HTML5 Tags eingesetzt werden müssen. Ebenfalls unterstützt die HTML5 und CSS3 die Entwicklung mit Bootstrap (siehe Kapitel 6.2.2. Bootstrap Framework).

6.2.2. Bootstrap Framework

Das Bootstrap Framework ermöglicht die Umsetzung des Responsive Designs. Der Vorteil von diesem Framework liegt in den vordefinierten Spaltendefinitionen. Die Spaltendefinitionen

können durch den Entwickler individuell in eine Webseite eingesetzt werden. Bootstrap ist ein Open-Source JavaScript Framework, welches von einem Entwicklerteam von Twitter¹¹ implementiert worden ist. Das Framework optimiert die UI Implementierung, weshalb HTML und CSS eingebunden werden können (Markle, 2013).

6.2.3. JavaScript, JQuery & AJAX

Die plattformunabhängige Skriptsprache **JavaScript** (JS) ist eine Erweiterungssprache für HTML in Webseiten. Diese dient grundsätzlich zur Auswertung von Benutzerinteraktionen und um dynamische Inhalte zu generieren. Die JavaScript-Befehle werden vom ausführenden Browser, respektive dem Clientrechner, ausgeführt (Flanagan, 2007, S. 1-10). JavaScript ist eine weitverbreitete Technologie. Der TIOBE Programming Index ermittelt monatlich die populärsten Programmiersprachen im World Wide Web. Im Monat Juli des Jahres 2017 lag JavaScript auf dem Rang 8 (TIOBE Software BV, 2017).

JQuery ist eine frei verfügbare JavaScript-Library. Diese Library unterstützt die DOM (**D**ocument **O**bject **M**odel) Manipulation. DOM ist eine Schnittstelle zwischen HTML- oder XML-Dokumenten. Neben der DOM Manipulation wird JQuery für die Client-Server Kommunikation eingesetzt. Dazu werden AJAX Applikationen auf der Clientseite erstellt. Wie JavaScript ist JQuery eine sehr weit verbreitete Technologie (JQuery, 2012).

AJAX steht für «**A**synchronous **J**avascript and **X**ML» und bietet den Datenaustausch über das Netzwerk. Die in der Abbildung 29 erkennbaren HTTP Requests und Responses werden dann durchgeführt, falls der Client eine Anfrage versendet und wenn nötig, eine Antwort erwartet. Ein weiterer positiver Aspekt AJAX einzusetzen ist das Prinzip der asynchronen Datenübertragung. Asynchron bedeutet, dass der Client nicht auf die Antwort des Servers abhängig ist. Somit kann der Anwender beispielsweise eine Datensicherung durchführen und ist nicht auf die Antwort des Webserver angewiesen (El Moussaoui & Zeppenfeld, 2008, S. 25-33).

JavaScript wird aus diesen Gründen gewählt, um eine dynamische Webseite zu garantieren und mit dem Anwender zu interagieren. Ein weiterer Grund ist die Einbettung der nötigen JavaScript-Libraries wie JQuery, JQuery-UI¹² und AJAX, welche die Interaktion und die Dynamik einer Website erheblich verbessern können. Um die DOM Veränderung mit zusätzlichen Funktionen auszustatten, wird die Kernbibliothek JQuery und die UI-Erweiterung JQuery-UI eingefügt. Ausserdem sind für bestimmte Funktionalitäten und graphische Elemente mehrere JavaScript-Libraries vorgesehen. In die SEPAT Software werden

¹¹ Quelle: <https://twitter.com/?lang=de>, 27.07.2017

¹² Quelle: <https://jqueryui.com/>, 11.07.2017

beispielsweise ChartJS und DataTables Libraries einbezogen. Für die Interaktion wird teilweise AJAX eingesetzt, um asynchrone Zugriffe durchzuführen. AJAX wird gezielt bei dynamischen UI-Veränderungen des DOMs oder der Erstellung von neuen Datensätzen auf der Datenbank eingesetzt. Des Weiteren beinhaltet das Bootstrap Framework ebenfalls JavaScript Klassen die ausgeführt werden müssen, um das Responsive Design und die vereinfachte UI Erstellung zu ermöglichen.

6.3. Backend

In diesem Kapitel werden die Backend Technologien beschrieben. Das Backend umfasst den Webserver und das DBMS. Zuerst wird eine kurze Einführung zur Nutzung des Webserver während der Implementation durchgeführt. Anschliessend werden die Technologien erklärt.

6.3.1. Webserver

Die SEPAT Software wird auf einem lokalen Internet Information Services (IIS) Webserver umgesetzt. Es können jedoch diverse Webserver wie IIS vom Microsoft Stack¹³ oder Apache¹⁴, einer der am meist verbreitenden Webservern, dazu genutzt werden (Gelbmann, 2017). Um SEPAT auszuführen, benötigt diese eine IP (Internet Protocol) Adresse, die auf dem Webserver gehostet wird. In diesem Kontext wird der Localhost verwendet. Dem lokalen Host, beziehungsweise dem Webserver, wird standardmässig die IP Adresse 127.0.0.1 zugewiesen. Bei einem gestarteten Webserver kann dieser mittels der Eingabe der IP Adresse 127.0.0.1 oder «localhost» im Browser aufgerufen werden. Grund der Erreichbarkeit des Webserver über «localhost» ist der Domänenname, der auf die lokal reservierte IP Adresse zugewiesen ist (Spona, 2007, S. 53,54).

Bei der Übertragung der SEPAT Software auf einen Webserver, der an einem Netzwerk angebunden ist, müssen diverse Änderungen in den Konfigurationspfaden unternommen werden. Die Programmlogik kann jedoch unverändert hochgeladen werden aufgrund der Implementierung auf einem Localhost, der einen Webserver nachbildet.

6.3.2. ASP.NET MVC 5

ASP.NET MVC 5 bedeutet ausgeschrieben «Active Server Page .NET» und ist ein Webapplikationsframework von Microsoft. MVC steht für die Softwarearchitektur, die sich nach dem Model-View-Controller-Prinzip orientiert. In der Abbildung 30 ist die Umsetzung des MVCs Prinzip auf Seiten des Backend ersichtlich. Ein besseres Sourcecode Management wird

¹³ Quelle: <https://www.quora.com/What-is-the-Microsoft-technology-stack-all-about>, 11.07.2017

¹⁴ Quelle: <https://httpd.apache.org/>, 11.07.2017

durch die Anwendung des MVC Prinzip gewährleistet. Die Fünf bezeichnet die gewählte Framework Version von ASP.NET MVC. ASP.NET wird generell in der IDE (Integrated Development Environment) Visual Studio von Microsoft entwickelt. Die Entwicklung von ASP.NET Projekten in Visual Studio bietet Vorteile. Beispielsweise kann beim Installationsprozess von Visual Studio der Webserver IIS integriert werden. Somit fallen die Konfigurationen des lokalen Webserver aus. Ein anderer wichtiger Faktor ist die Softwareintegration in den Webserver.

ASP.NET Anwendungen werden grundsätzlich mit der Programmiersprache C# geschrieben, kann aber auch mit Visual Basic¹⁵ umgesetzt werden.

Das Webapplikationsframework wird vor allem auf dem Backend eingesetzt. Auf dem Frontend kann die Technologie ebenfalls angewendet werden, denn ASP.NET MVC entwickelte für die Anwender der Technologie HTML Helpers und die ViewModels (**Model-View-ViewModel** kurz MVVM). Die HTML Helpers sind lediglich Methoden, welche in die View eingesetzt werden können. Diese Methoden generieren auf der View die standardmässigen HTML Tags, sind jedoch für komplexeren Inhalt ideal geeignet. Für die Anwendung dieser HTML Helper Methoden sind jedoch spezielle Views einzusetzen. Aus diesem Grund sind die Razor Views für ein ASP.NET MVC Projekt von Bedeutung. Die Razor View ermöglicht die Integration von serverbasiertem Sourcecode in der View. Dieser serverbasierte Code wird auf dem Server ausgeführt und in die Razor View eingefügt. Das Format «.cshtml» verdeutlicht, dass C# Code mit HTML Tags zusammengeführt werden können (Mullen & Anderson, 2017).

¹⁵ Quelle: <https://docs.microsoft.com/en-us/dotnet/visual-basic/>, 27.07.2017

Die ViewModels sind für die direkte Einbindung von Backend Models auf der View geeignet. Das MVVM Prinzip bezweckt, dass für eine oder mehrere spezifische Websites ein individuelles ViewModel erstellt wird. Ein ViewModel wird auf dem Backend mit Daten gefüllt und in der View eingebunden. Die Eigenschaften des ViewModels werden anhand der View festgelegt (Steyer & Softic, 2017, S. 15-17).

Um die zukünftige Entwicklung der SEPAT Software zu optimieren, unterstützt das Framework das Agile-Test-Driven-Development aufgrund des MVC Prinzips und des komponenten-orientierten Designs vom ASP.NET Framework. Dies sind massgebende Veranlassungen ASP.NET MVC für die SEPAT Software einzusetzen.

6.3.3. Entity Framework 6 mit MS SQL Server Express 2014

Entity Framework (EF) ist von Microsoft entwickeltes Framework, um den Zugriff auf eine Datenbank zu erleichtern. Das Framework basiert auf der O/RM (**O**bject-**R**elational **M**apping) Technik. Diese Technik unterstützt die objektorientierte Programmierung in Bezug auf das Datenmanagement. Deswegen wird die Datenbankstruktur nicht auf der Datenbank definiert, sondern von den Model Klassen des Webserver abgeleitet. Anhand dieser Definitionen in den Model Klassen wird die Datenbank erzeugt. Somit können die Daten anhand der Models strukturiert aus der Datenbank gelesen und weiterverarbeitet werden. Der Datenaustausch erfolgt durch die Query Language LINQ (**L**anguage **I**ntegrated **Q**uery). Die explizite Erstellung einer Multi-tier Architektur kann deswegen umgangen werden. Die Data Access Ebene und die Data Transfer Objekte werden vom Entity Framework bereitgestellt. Bei der Implementierung wird dementsprechend der Datenzugriff und -speicherung vom Entity Framework verwaltet (What is Entity Framework?, kein Datum).

Der Microsoft SQL (**S**tructured **Q**uery **L**anguage) Server wurde im SEPAT System als physikalische Datenbank eingesetzt. Die Vorgehensweise, wie mit der Datenbank interagiert wird, ist bei der Verwendung von Entity Framework vorgelegt. Es gibt drei verschiedene Vorgehensweisen, die umgesetzt werden können. Nachfolgend werden diese in Kombination mit Kriterien (Database First development with Entity Framework, kein Datum) aufgelistet. Die Kriterien unterstützen eine geeignete Wahl der Vorgehensweise auszuwählen:

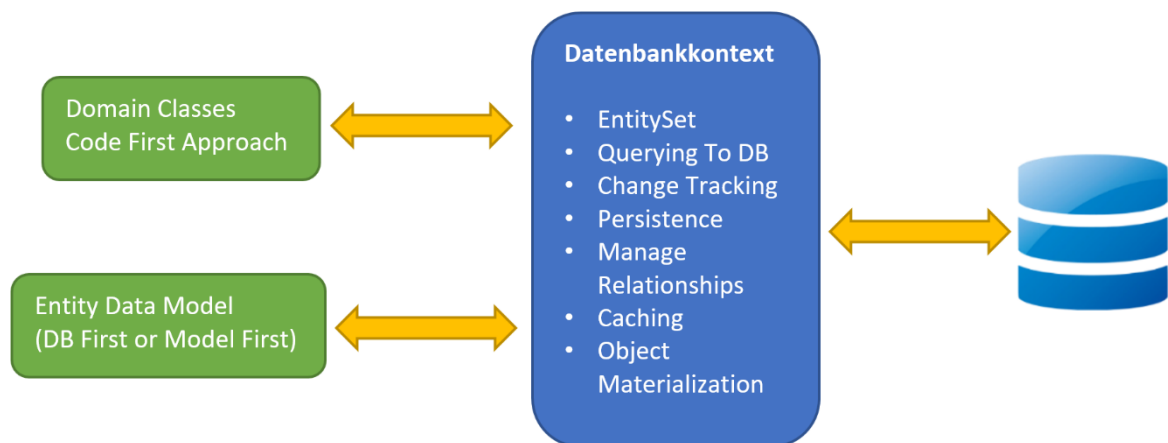
1. Database First Approach – Wenn eine Datenbank bereits besteht.
2. Model First Approach – Wenn der Visual Designer eingesetzt werden möchte.
3. Code First Approach – Wenn bereits Modell Klassen bestehen.

Zu Beginn der Implementation bestand keine Datenbank. Der Visual Designer, ein Entity Data Model Tool in Visual Studio, ist bekannt und bereits eingesetzt worden, jedoch ist man in

der Verwaltung von komplexen Datenbankmodellen nicht ausreichend erfahren. Demzufolge wird der Code First Approach ausgewählt.

Mithilfe des Code First Approachs werden lediglich die Model Klassen mit deren Eigenschaften definiert. Nach der Definition einer neuen Model Klasse wird der Datenbankkontext angepasst, welcher die Änderungen an die physikalische Datenbank übermittelt. Die Abbildung 32 hilft das Konzept des Code First Approachs mit dem Datenbankkontext zu verstehen.

Abbildung 32: Verwendung des Datenbankkontextes im Code First Approach



Quelle: (DbContext, kein Datum)

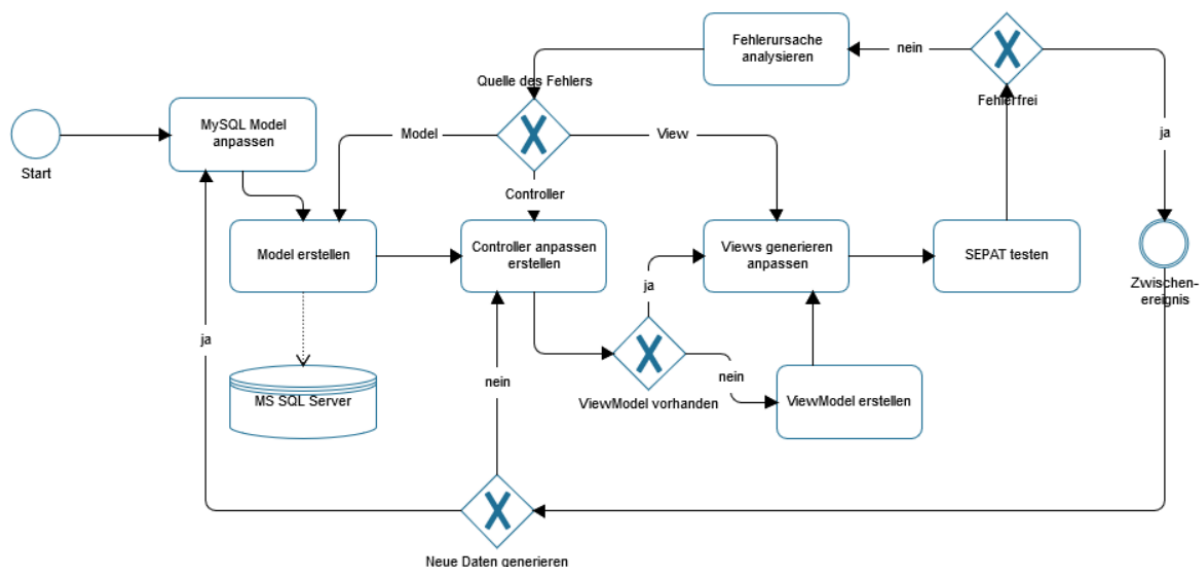
Wie aus der Abbildung 32 ersichtlich ist, werden beispielsweise über den Datenbankkontext die Entitäten oder Tabellenbeziehungen definiert oder Datenbankabfragen durchgeführt. Dies ermöglicht eine flexible und schnelle Implementierung hinsichtlich der Datenbankdefinition. Die Erstellung des Datenbankkontextes garantiert, dass die Schicht des Datenzugriffs und dessen Datentransferobjekte durch Entity Framework erstellt und verwaltet werden.

7. Die Implementationen und Lösungsansätze

Schrittweise wird die Implementation der SEPAT Software dokumentiert. In diesem Kapitel sind Pseudocodes erstellt worden. Auf den originalen Sourcecode wird anhand der Klassen- und Methodenbezeichnungen verlinkt oder an den Anhang angefügt.

Wie bei der Entwicklung vorgegangen wurde, wird auf der Abbildung 33 in Form eines **Business Process Model and Notation (BPMN)**-Models dargestellt.

Abbildung 33: Vorgehensweise der Softwareimplementierung



Nachdem die Technologien festgelegt worden sind, wurde mit dem Implementationsprozess gestartet. In einem ersten Schritt war die visuelle Anpassung des Datenbankmodells von Bedeutung. Dieser Prozess wurde mit Hilfe des Programmes MySQL Workbench¹⁶ durchgeführt. Diese Modellierung hatte keine Einflüsse auf das SEPAT System. Die Modellierung diente ausschliesslich der visuellen Kontrolle und Übersicht der aktuellen Tabellen und Datenentitäten.

Nach dem Modellierungsprozess lag der Fokus in der Erzeugung der neuen Tabelle in der MS SQL Datenbank. Für die Umsetzung ist die spezifische Model Klasse erstellt worden. Wenn die Anpassung der physischen Datenbank erfolgreich abschloss, wurde eine neue Controller Klasse erstellt oder eine bestehende angepasst. Die ViewModels wurden anhand der Mockups erstellt. Die Mockups wiedergaben die angeforderten Daten, die eine View anzeigen soll. Um ein ViewModel in eine View einzubinden, musste diese zuerst erzeugt werden. Auf der View fand die Gestaltung des GUI, mit den Technologien wie JavaScript, HTML5 und CSS3, statt. In einem weiteren Schritt wurde die Software ausgeführt und getestet.

¹⁶ Quelle: <https://www.mysql.com/de/products/workbench/>, 11.07.2017

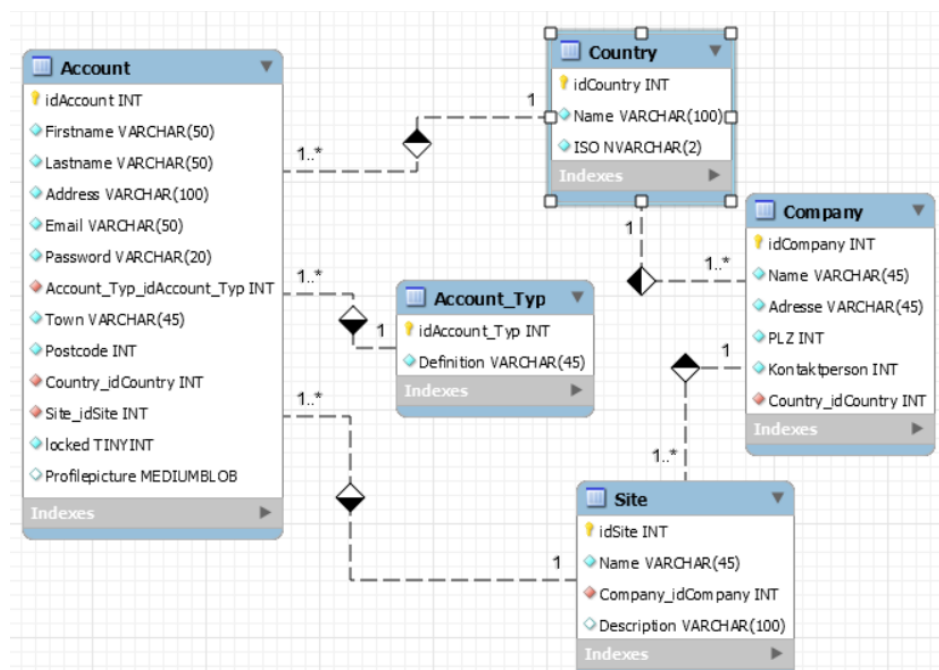
Bei einem fehlerfreien Ablauf konnte zu der nächsten Funktion übergegangen werden. Falls die Software abbrach oder fehlerhafte Daten in der View beziehungsweise auf der Datenbank anzeigte, musste analysiert werden, woher die Fehlerquelle stammt, um diese zu beheben.

7.1. Relationales Datenbank Modell

In diesem Kapitel wird das relationale Datenbankmodell im Detail erläutert. Das Datenbankmodell wurde mit der Software MySQL Workbench nachgebildet und diente während der Implementation als visuelle Kontrolle der Datenentitäten und Tabellenbeziehungen. Zur Übersicht ist im Anhang IX das gesamte Datenbankmodell aufgeführt.

7.1.1. Account-Management

Abbildung 34: Auszug des Datenbankmodells des Account-Managements



Die Abbildung 34 zeigt die Tabellen betreffend des Account-Managements. Ein Account wird klar durch einen vordefinierten Identifikator und einer angegebenen E-Mail-Adresse unterschieden. Da ein Account eine Registrierung oder ein Login durchführen kann, enthält die Tabelle einen Password Hash. Optional kann jeder Account ein Profilbild abspeichern. Bei der Registration werden die restlichen Stammdaten durch den neuen Anwender eingetragen.

Im Verlauf der Erstellung eines neuen Accounts wird zwischen vier Account Typen unterschieden. Diese Typen betreffen die erstellten Personas und bilden in der Datenbank die diversen Rollen ab. Folgende Rollen mit deren Identifikatoren sind vorhanden:

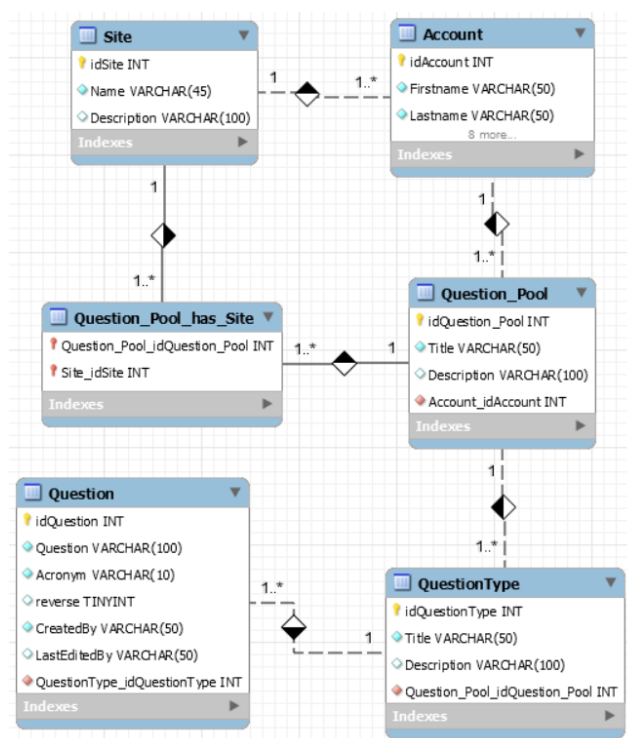
1. Server Administrator
2. Site Administrator
3. Master
4. User

Ein Account wird jeweils einer Site untergeordnet. Die Siteentitäten bestehen aus einem spezifischen Namen, einer optionalen Beschreibung und einem Fremdschlüssel einer Unternehmung. Denn ein Unternehmen besteht aus einer oder mehreren Sites. Bei den Unternehmungen ist die Sicherstellung von Stammdaten wie die Adresse, Postleitzahl und die Telefonnummer der Kontaktperson von Bedeutung. Die Accounts sowie die Unternehmen müssen einem Land angehören, um sich zu registrieren.

7.1.2. Analysefragen-Management

Wie die Analysefragen und die Analysefragenblöcke in der Datenbank verwaltet werden, ist in der Abbildung 35 illustriert.

Abbildung 35: Auszug des Datenbankmodells des Analysefragen-Managements



Die Tabelle der Analysefrage (Question) beinhaltet die Attribute eines generierten Identifikators, einer Frage, einem Kurztitel (Acronym) und der Umkehrungsmöglichkeit reverse. Die Reverse-Funktion besteht aus einem numerischen Wert. Standardmässig ist die Funktion deaktiviert mit dem Wert 0. Anhand einer 1 wird diese aktiviert.

Eine Analysefrage ist stets einem Fragetypen (QuestionType) untergeordnet. Die Fragetypen können von den Administratorenrollen und dem Master definiert werden. Bei der Erstellung eines neuen Fragetyps, ist der Anwender darauf angewiesen, einen Titel zu definieren. Der Identifikator wird inkrementiert und der Fremdschlüssel des Fragenpools (Question_Pool) wird von der aktuellen Site, in welcher der Anwender registriert ist, übernommen.

Ein Fragenpool umfasst die neben den Attributen des Titels und der optionalen Beschreibung ebenfalls einen Identifikator. Bei der Erstellung eines neuen Fragenpools wird dieser der Site des Anwenders zugewiesen.

Ein Fragenpool kann vom zugehörigen Anwender auf diverse Sites innerhalb des Unternehmens geteilt werden. Aus diesem Grund erhalten Anwender aus diversen Sites Zugriff auf die Fragen im Fragenpool.

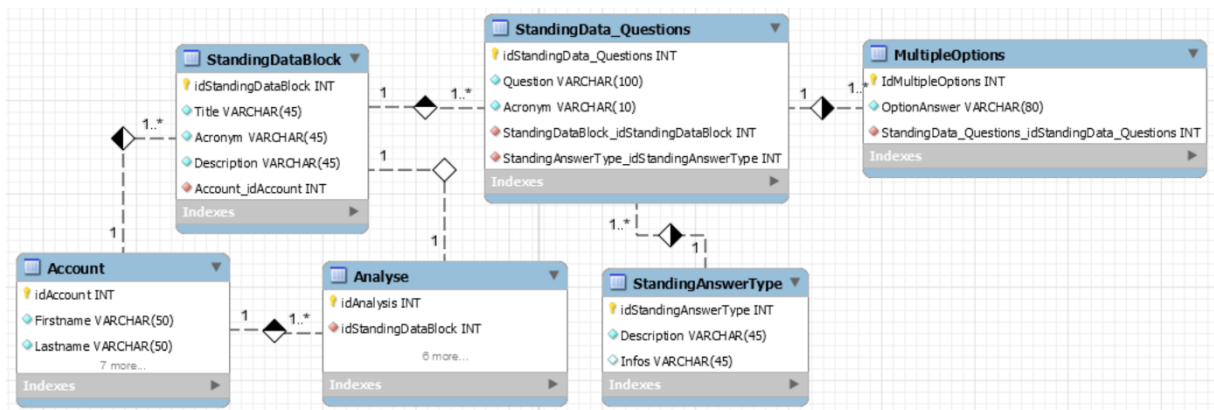
7.1.3. Stammdatenfragen-Management

Das Stammdaten-Management in der Datenbank ist in der Abbildung 36 verdeutlicht. Die Stammdatenfragen (StandingData) besitzen ähnliche Datenentitäten wie die Analysefragen, Somit wird bei der Erstellung eine Frage und einen Kurztitel erfasst. Zusätzlich beinhaltet eine Stammdatenfrage einen Fremdschlüssel zu einem Stammdatenblock und Stammdatenantworttyp. Die Stammdatenantworttypen sind durch den Serveradministrator vordefiniert. Zurzeit existieren vier Typen. Die Texteingabe, Single Choice, Multiple Choice und eine Dropdown Option zählen zu den vier Stammdatenantworttypen.

Abhängig vom Fremdschlüssel der Tabelle Stammdatenantworttyp ist die Anzahl der zugehörigen Antwortoptionen (MultipleOptions) einer Stammdatenfrage. Beispielsweise im Prozess der Erstellung einer neuen Stammdatenfrage besteht die Auswahl des Antworttyps. Wenn die Wahl des Typs mehrere Antwortoptionen verlangt, wie zum Beispiel die Option Multiple Choice, werden diese in die Tabelle der Antwortoptionen gesichert. Deshalb kann eine Stammdatenfrage mehrere Antwortoptionen besitzen. Es ist ausserdem möglich, dass eine Stammdatenfrage keine Antwortoption erfordert. Dies betrifft lediglich den Antworttyp der Texteingabe.

Der Stammdatenblock wird nicht in der Auswertung angezeigt, jedoch für den Export im CSV Format ist der Kurztitel für die Spaltenbeschreibung von Bedeutung. Der Titel und der Kurztitel sind bei der Erstellung als Pflichtfelder markiert, hingegen die Beschreibung optional ausgefüllt werden kann. Ein Stammdatenblock ist eindeutig einem Account und einer Analyse (Selbstevaluation) zugeteilt. Infolgedessen kann diese nicht siteübergreifend eingesetzt werden.

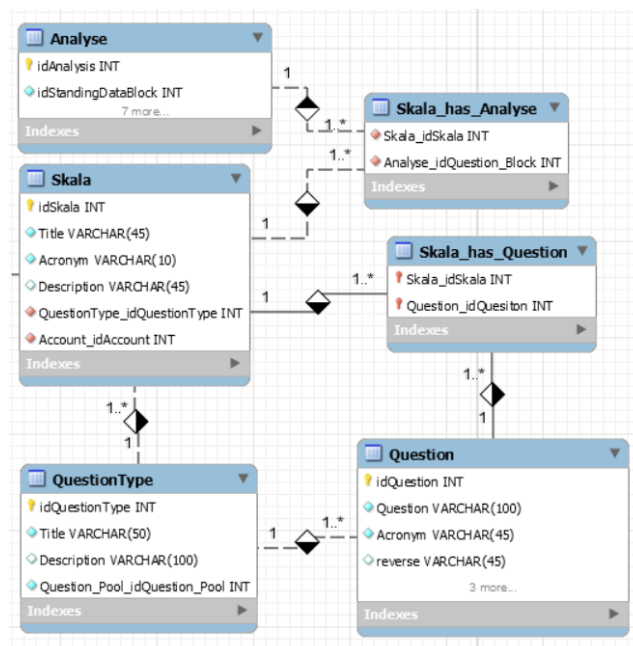
Abbildung 36: Auszug des Datenbankmodells des Stammdatenfragen-Managements



7.1.4. Skalen-Management

Laut der Abbildung 37 besteht eine N:M Beziehung zwischen den Tabellen der Analyse und der Skala. Dies erlaubt die Wiederverwendung der Skalen in mehreren Analysen (siehe Kapitel 4.3.4. Fragenbogen und Skalen-Management oder Anhang IV Skalenlogik – Skizze).

Abbildung 37: Auszug des Datenbankmodells des Skalen-Managements

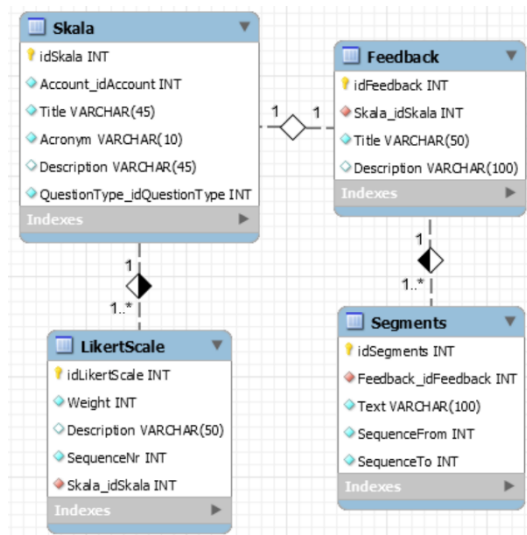


Für eine Skala generiert das System inkrementell einen Identifikator und speichert den Primärschlüssel des Anwenders ab. Der Titel sowie der Kurztitel sind einzutragen. Eine Skala beansprucht je einen Fragentypen. Folglich ist es möglich, thematisch zusammengehörige Fragen in mehreren Skalen abzulegen.

Die Skalen sind deshalb von Bedeutung, da diese die Fragen des Fragenpools enthalten. Durch die Skalen wird eine Verbindung mit den Analysen geschaffen. Diese sind massgebend für die Generierung einer Selbstevaluation.

Zum Skala-Management bezieht sich ebenfalls die Verwaltung der Feedbacks und deren Segmenten. Wie diese modelliert worden sind, ist in der Abbildung 38 verdeutlicht.

Abbildung 38: Auszug des Datenbankmodells des Feedbacks-Managements



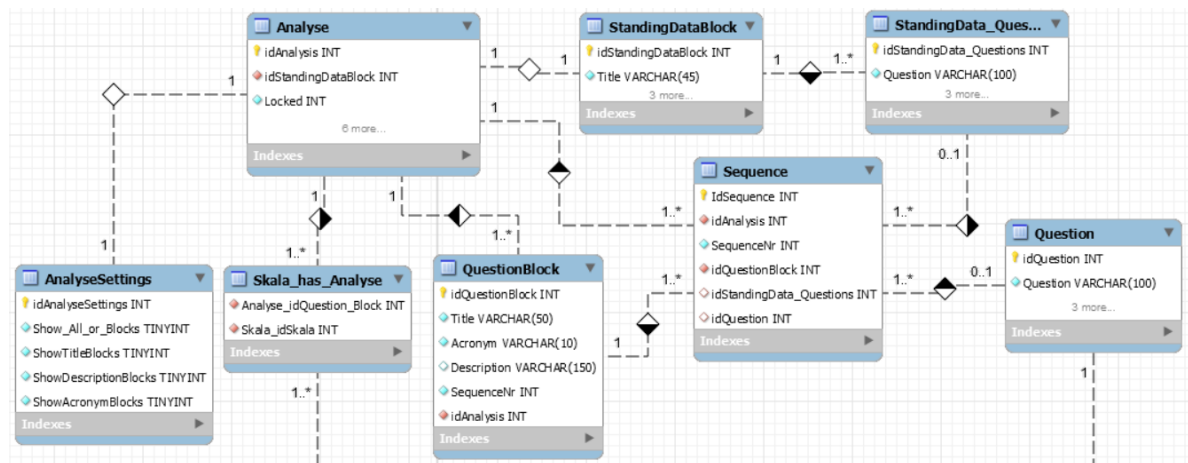
Ein Feedback ist eindeutig und deshalb nur einer Skala zugewiesen. Aufgrund der vorliegenden 1:1 Beziehung besitzt das Feedback denselben Identifikator wie die Skala. Das Feedback wurde absichtlich nicht in die Skala miteinbezogen, weil bei der Erstellung der Skala und des Feedbacks nicht gleichzeitig erfolgt.

Ein Feedback enthält einen Titel und eine optionale Beschreibung. Dem Anwender muss am Ende einer Selbstevaluation eine spezifische Auswertung erscheinen. Diese werden mithilfe der Tabelle Segmente erstellt. Ein Segment ist jeweils einem Feedback untergeordnet und enthält die Eingabe des gewählten Prozentbereichs der Skala.

7.1.5. Analysis-Management

Die Abbildung 39 visualisiert den Lösungsansatz zum Skala- und Stammdatenfragen-Management einer Selbstevaluation auf Seiten der Datenbank.

Abbildung 39: Auszug des Datenbankmodells des Analysis-Managements



Um eine Analyse durchzuführen, müssen die Analysefragen sowie die Stammdatenfragen zugeteilt werden. Dies wurde bereits mithilfe der Skalen und des Stammdatenblockes realisiert. In einem nächsten Schritt gilt es die Fragen zu strukturieren beziehungsweise die Reihenfolge festzulegen. Im SEPAT System ist der Administrator fähig, die Strukturierung nach verschiedenen Vorgehensweisen durchzuführen (siehe Kapitel 7.7. Sortierungsverfahren einer Selbstevaluation). Die Strukturierung erfolgt mit der Einbindung der Frageblöcke (QuestionBlock) und der Reihenfolge (Sequence) Tabellen.

Die Frageblöcke sind unabhängig von den Skalen und werden nur bei der Strukturierung der Selbstevaluation eingesetzt. Wie aus der Tabelle Selbstevaluationseinstellungen (AnalyseSettings) entnommen werden kann, enthält diese lediglich Benutzereinstellungen zur Darstellung. Diese Einstellungen betreffen die Attribute (Titel, Kurztitel und Beschreibung) der Fragebögen. Eine Analyse besteht aus mindestens einem oder mehreren Fragebögen. Die Fragebögen strukturieren die Fragen blockweise. Die Reihenfolge der Fragen im Frageblock werden mit der Tabelle Sequenz (Sequence) festgelegt.

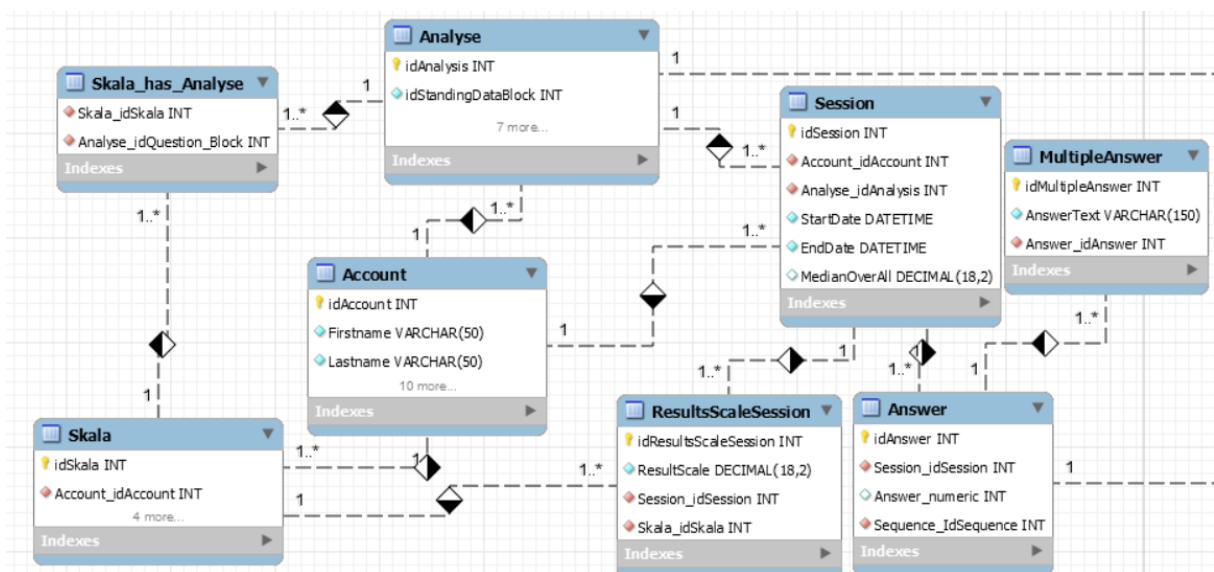
Jede neue Sequenz ist eindeutig, wird einem Frageblock zugeteilt und gehört zu einer Analyse. Über die Analysen Tabelle erhält der Anwender über alle Stammdaten- und Analysefragen Zugriff. Aufgrund der unterschiedlichen Fragetypen sind in der Tabelle Reihenfolge zwei Spalten mit jeweils optionalen Fremdschlüsseln vorhanden. Bei der Zuordnung einer Frage wird geprüft, ob es sich um eine Analyse- oder Stammdatenfrage handelt. Nach der Prüfung kann der entsprechende Identifikator als Fremdschlüssel in der

Sequence Tabelle abgespeichert werden. Da die Reihenfolge verändert werden kann, wird diese Problemstellung nicht mit inkrementierenden Identifikatoren gelöst. Die Entität namens SequenceNr definiert die festgelegte Reihenfolge auf dem GUI.

7.1.6. Antworten-Management

Das Antworten-Management verwaltet die resultierenden Benutzereingaben einer Selbstevaluation. Für die Umsetzung sind die Tabellen Sitzung (Session), Resultat per Skala (ResultsScaleSession), Antwort (Answer) und mehrere Antworten (MultipleAnswer) erstellt worden. Die Tabellen sind in der Abbildung 40 dargestellt.

Abbildung 40: Auszug des Datenbankmodells des Antworten-Managements



Eine neue Sitzung wird beim Ausführen einer Selbstevaluation zur Datenbank hinzugefügt. Jede Sitzung wird eindeutig einem Account zugewiesen. Die ausgewählte Selbstevaluation sowie der Zeitpunkt werden ebenfalls in die Tabelle abgespeichert. Die Spalte «MedianOverAll» setzt sich aus den Ausprägungen der Skalen, die in der Tabelle Resultat per Skala abgespeichert sind, zusammen. Diese Sitzungen erlauben, dass ein Anwender eine Selbstevaluation mehrmals ausführen kann und die Resultate voneinander trennt.

Die Antworten der Analyse- und Stammdatenfragen sind unter Verwendung der Tabelle Antwort abgewickelt. Eine Antwort der Analysefragen liefert einen numerischen Wert der Likert Skala. Infolgedessen wird dieser Wert in die Spalte «Answer_numeric» abgespeichert. Bei der Antwortspeicherung der Stammdatenfragen wird eine andere Vorgehensweise durchgeführt. Abhängig vom Stammdatenantworttyp müssen mehrere Antworten abgespeichert werden. Diese Antworten werden in die Tabelle «MultipleAnswer» abgelegt. Die Tabelle «MultipleAnswer» referenziert auf die Antwort Tabelle mittels einem Fremdschlüssel.

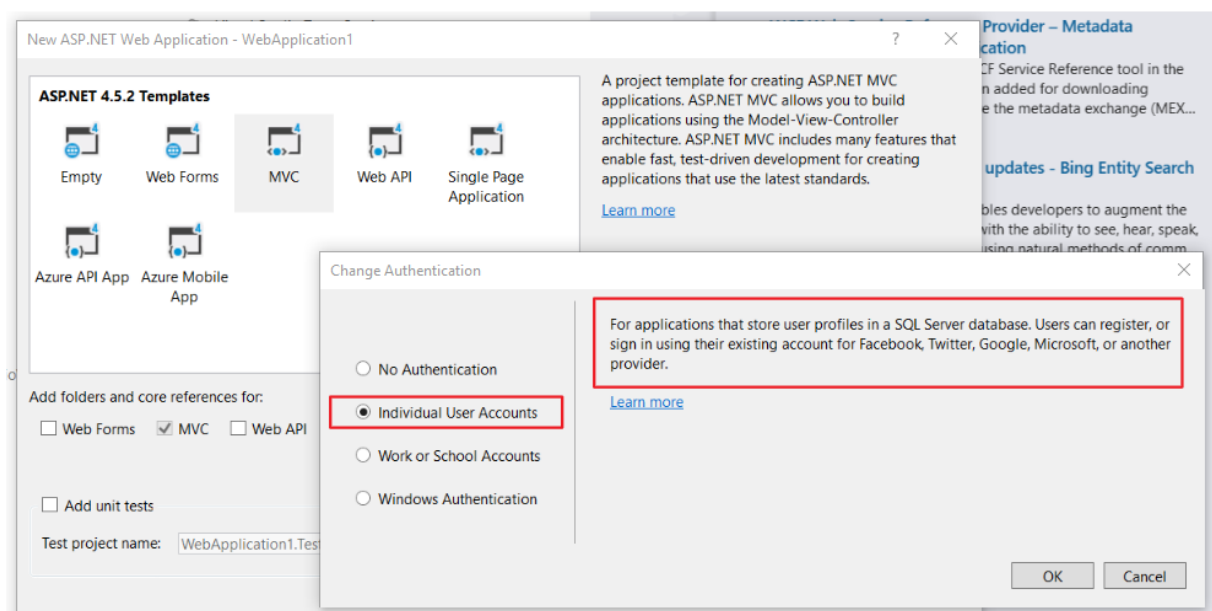
Die eingegebene Antwort des Anwenders wird in Textform gesichert. Damit diese Datensätze für die Auswertung genutzt werden können, ist eine Identifikator eingesetzt worden.

Nach der Durchführung der Selbstevaluation wird die Ausprägung per Skala berechnet und eine neue Zeile in der Tabelle «ResultsScaleSession» angelegt. Dabei wird zusätzlich der Identifikator der Skala und der Sitzung abgespeichert.

7.2. Google Authentifikation (OAuth 2)

Die Google Authentifikation ist mit vordefinierten Zusatzpaketen von Microsoft umgesetzt worden. Die Einbindung dieser Zusatzpakete erfolgte teilweise bei der Projekterstellung von SEPAT. Die Abbildung 41 zeigt die ASP.NET MVC Projekterstellung auf.

Abbildung 41: Erstellung eines ASP.NET MVC Projektes mit ASP.NET Identity



Der SEPAT Prototyp wurde mit der ASP.NET MVC Technologie entwickelt, weshalb bei der Projekterstellung das MVC Template ausgewählt worden ist. Die Auswahl der Option «Individual User Accounts» generiert vordefinierte Controller-, Model-, ViewModel-, und View Klassen. Somit existiert bereits nach der Projekterstellung eine Webapplikation, die über ein Login- und ein Registrierungsverfahren verfügt. Die Option «Individual User Accounts» bindet in das Projekt automatisch Zusatzpakete wie EF, Core und OWIN ein (Galloway, Rastogi, & Dykstra, 2013).

Alleine die Einbindung der Zusatzpakete gewährleistet nicht die Authentifikation mittels Drittanbieterkonten. Die vordefinierten Klassen umfassen nur die Programmlogik einer Authentifikation. Deshalb muss die Verbindung zu den jeweiligen Anbietern ermöglicht werden. Aus diesem Grund wird die Middleware OWIN (**O**pen **W**eb Interface for **.NET**)¹⁷ im SEPAT Projekt eingebunden.

OWIN dient als standardisierte Schnittstelle für Komponenten zwischen .NET Webservern und Webanwendungen. Diese Komponenten werden ebenfalls als Middleware bezeichnet und können Anfragen empfangen und bearbeiten. Die Bearbeitung der Anfragen findet vor der Delegation zum Anwendungscode statt. Die Middleware Komponente leitet die Anfrage mittels Pipelines auf ein Framework weiter, welches die Webapplikation ausführt. Die Antworten werden durch dieselbe Reihenfolge der Pipelines, wie sie verschickt wurden, an den Absender (Client) zurückgesendet (Steyer & Softic, 2017, S. 377-378).

Das Core Zusatzpaket wurde in der SEPAT Software nicht angewendet. Laut der Microsoft Dokumentation wird Core als Kernschnittstelle für die Implementierung von verschiedenen Datenbanken wie Azure Table Storage, NoSQL Datenbanken eingesetzt (Galloway, Rastogi, & Dykstra, 2013). Im SEPAT Prototypen wird lediglich ein Microsoft SQL Server eingesetzt, weshalb auf das Core Paket verzichtet werden kann.

Wichtig für die Verbindung zu den Servern der Drittanbieter ist die Generierung der Klasse «Startup.Auth.cs» im ASP.NET Identity Paket. In dieser Klasse werden die Clientschlüssel eingebunden. Mithilfe eines Clientschlüssels wird die ausführende Webapplikation auf dem Webserver des Drittanbieters verifiziert. Bei einer Clientschlüsselgenerierung für die Google Authentifikation muss ein Gmail Konto eingesetzt werden. In der Google Cloud Platform¹⁸ wird die Schnittstelle «Google+ API» aktiviert, um die Verbindung zwischen der SEPAT Software und der Google Cloud Platform zu ermöglichen. Nachdem der Clientschlüssel generiert und die Verbindung zur Google Cloud Platform aktiviert ist, kann die Implementation starten (Anderson, 2015).

¹⁷ Quelle: <http://owin.org/>, 16.07.2017

¹⁸ Quelle: <https://cloud.google.com/>, 16.07.2017

7.2.1. Anwendung von ASP.NET Identity

Die Datenbanktabelle der Accounts verlangt Informationen über die Stammdaten eines neuen Anwenders. Daher müssen die Daten, welche an das individuelle Gmail Konto angebunden sind, übernommen werden. Um Zugriff über die Daten zu erhalten, wird die generierte Klasse «Startup.Auth.cs» wie in der Abbildung 42 abgeändert.

Abbildung 42: Google Authentifikation und Zugriff in Startup.Auth.cs

```
var googleOptions = new GoogleOAuth2AuthenticationOptions()
{
    // keys to access
    ClientId = "31...om",
    ClientSecret = "Ql...B",
    // get data of gmail account
    Provider = new GoogleOAuth2AuthenticationProvider()
    {
        OnAuthenticated = (context) =>
        {
            foreach (var claim in context.User)
            {
                var claimType = string.Format("http://schemas.xmlsoap.org/ws/2005/05/identity/claims/{0}", claim.Key);
                string claimValue = claim.Value.ToString();
                context.Identity.AddClaim(new Claim(claimType, claimValue, "http://www.w3.org/2001/XMLSchema#string"));
            }

            return Task.FromResult(0);
        }
    }
};
app.UseGoogleAuthentication(googleOptions);
```

Die Google User Daten werden vom Authentifikationsprovider bezogen und in den Identity Kontext eingefügt. Das Format des Kontexts wird von der Windows Identity Foundation vorgegeben. In diesem Format sind die öffentlichen Felder (Public Fields) der ClaimTypes enthalten (ClaimTypes Fields, kein Datum). Um die Daten eines Google Users zu empfangen, wird die Schnittstelle des Authentication Manager der OWIN Komponente aufgerufen und dessen Identity Kontext gelesen. Mithilfe der ClaimTypes wird auf die Daten des Kontextes strukturiert zugegriffen. Wie die technische Umsetzung des Zugriffes implementiert wird, ist in der Abbildung 43 verdeutlicht.

Abbildung 43: Informationszugriff eines Google-Accounts in ExternalLoginCallback

```
// GET: /Account/ExternalLoginCallback
[AllowAnonymous]
// 0 references | 0 requests | 0 exceptions
public async Task<ActionResult> ExternalLoginCallback(string returnUrl)
{
    // takes an OAuth 2 login place
    var loginInfo = await AuthenticationManager.GetExternalLoginInfoAsync();
    if (loginInfo == null)
    {
        return RedirectToAction("Login");
    }
    // collect data from google
    if (loginInfo.Login.LoginProvider == "Google")
    {
        var externalIdentity = AuthenticationManager.GetExternalIdentityAsync(DefaultAuthenticationTypes.ExternalCookie);

        // save infos from gmail
        var emailClaim = externalIdentity.Result.Claims.FirstOrDefault(c => c.Type == ClaimTypes.Email);
        var lastNameClaim = externalIdentity.Result.Claims.FirstOrDefault(c => c.Type == ClaimTypes.Surname);
        var givenNameClaim = externalIdentity.Result.Claims.FirstOrDefault(c => c.Type == ClaimTypes.GivenName);
        var addressClaim = externalIdentity.Result.Claims.FirstOrDefault(c => c.Type == ClaimTypes.StreetAddress);
        var stateClaim = externalIdentity.Result.Claims.FirstOrDefault(c => c.Type == ClaimTypes.StateOrProvince);
        var postalClaim = externalIdentity.Result.Claims.FirstOrDefault(c => c.Type == ClaimTypes.PostalCode);
    }
}
```


Abbildung 45: Authentifikation eines externen Kontos in ExternalLoginCallback

```
// Sign in the user with this external login provider if the user already has a login
var result = await SignInManager.ExternalSignInAsync(loginInfo, isPersistent: false);
switch (result)
{
    // if the address exists
    case SignInStatus.Success:
        string aId = _db.Users.Where(x => x.Email == modelConfirm.Email).Select(x => x.Id).FirstOrDefault();
        int idSiteSession = _db.Users.Where(x => x.Email == modelConfirm.Email).Select(x => x.IdSite).FirstOrDefault();
        Session["UserID"] = aId;
        Session["Email"] = modelConfirm.Email;
        Session["AccountType"] = _db.Users.Where(x => x.Email == modelConfirm.Email).Select(x => x.IdAccountType).FirstOrDefault();
        Session["SiteID"] = idSiteSession;

        modelConfirm = null;
        return RedirectToAction("LoggedIn", "Dashboard");
    case SignInStatus.LockedOut:
        return View("Lockout");
    case SignInStatus.RequiresVerification:
        return RedirectToAction("SendCode", new { ReturnUrl = returnUrl, RememberMe = false });
    case SignInStatus.Failure:
    default:
        // If the user does not have an account, then prompt the user to create an account
        ViewBag.ReturnUrl = returnUrl;
        ViewBag.LoginProvider = loginInfo.Login.LoginProvider;
        ViewBag.RegEmail = modelConfirm.Email;
        return RedirectToAction("SelectSite", "Account");
}
```

Der wesentliche Unterschied zur Registration ist die manuelle Eingabe der Stammdaten. Bei der Authentifikation wird die Vollständigkeit der Pflichtfelder überprüft. Es ist möglich, dass ein Konto des Drittanbieters nicht die nötigen Informationen umfasst. Aus diesem Grund müssen die fehlenden Informationen manuell vom Anwender eingegeben werden. Diese Interaktion findet bei der Auswahl des Unternehmens und Site statt.

Die Methode External Login Confirmation speichert die Benutzereingaben ab und loggt den neuen Anwender ein. In der Abbildung 46 ist die Objekterstellung und die Speicherung des neuen Anwenders verdeutlicht worden. Die Erstellung erfolgt über das EF Model ApplicationUser. Hingegen die Speicherung findet über die Schnittstelle des ApplicationUserManagers von OWIN statt. Wie in der Abbildung 45 werden bei einer erfolgreichen Speicherung die Sessions erstellt und das Login durchgeführt.

Abbildung 46: Objekterstellung und Speicherung eines neuen Gmail Anwender:

```
// need infos about the site and country
var user = new ApplicationUser { UserName = modelConfirm.Email, Email = modelConfirm.Email, IdAccountType = IdType,
    Firstname = modelConfirm.Firstname, Lastname = modelConfirm.Lastname, Address = modelConfirm.Address, Town = modelConfirm.Town,
    Postcode = modelConfirm.Postcode, IdCountry = modelConfirm.IdCountry, IdSite = modelTempSelectSite.IdSite };

// save into db
var result = await UserManager.CreateAsync(user);
```

Aufgrund der Einbindung von ASP.NET Identity und der Einsetzung der Authentifikation, können die klassischen EF Befehle nicht für die Speicherung der Accounts eingesetzt werden. Die Kommunikation mit der Schnittstelle OWIN garantiert die vordefinierten Methoden.

7.2.2. Datenbankkontext

Der Datenbankkontext ist ein wesentlicher Bestandteil des EFs. In diesem Kontext werden die Datenobjekte eingefügt, verändert oder gelöscht. Mit einem spezifischen Aufruf wird der Kontext auf die physische Datenbank übernommen und persistiert.

Bei der Einbindung des Projektes ASP.NET Identity erstellt Visual Studio einen neuen Datenbankkontext. Dieser zielt auf die Accountverwaltung ab, was zu erheblichen Problemen für die Erweiterung des Kontexts führt. Der Kontext leitet sich von einer Schnittstelle ab, die lediglich die Accounts umfassen.

Es bestehen zwei Vorgehensweisen für die optimale Umsetzung des Kontextes. Einerseits kann ein weiterer Kontext erstellt werden. Dies hat zur Folge, dass mit zwei Kontexten gearbeitet werden muss. Dementsprechend ist der Kommunikationsaufwand der beiden Kontexten enorm hoch. Andererseits kann der vordefinierte Kontext auf einen neuen vererbt werden. Keine der erwähnten Vorgehensweisen wird auf der Microsoft Dokumentation eindeutig vorgeschlagen. Für die SEPAT Software ist die zweite Option umgesetzt worden, weil die Implementation mit einem Kontext weniger komplex und zeit effizienter umgesetzt werden kann. Somit wird ein neuer Kontext erstellt, welcher den vordefinierten Identity Kontext implementiert. Die Abbildung 47 zeigt, wie die Umsetzung des Datenbankkontextes im SEPAT Sourcecode umgesetzt wird.

Abbildung 47: Datenbankkontext des SEPAT Prototyps

```
// identityDbContext of the Individual User Accounts combined with custom one
25 references
public class SEPATContext : IdentityDbContext<ApplicationUser>
{
    10 references | 0 exceptions
    public SEPATContext()
    {
        : base("name=SEPATConnection", throwIfV1Schema: false)
    }

    // configurations for the db/ef
    //Configuration.LazyLoadingEnabled = true;
    //Database.SetInitializer(new MigrateDatabaseToLatestVersion<SEPATContext,
    //Migrations.Configuration>("name=SEPATConnection"));
}

// initialize the table for the dbcontext --> db
12 references | 0 exceptions
public DbSet<Site> Site { get; set; }
2 references | 0 exceptions
public DbSet<Company> Company { get; set; }
```

Die wichtigsten Code Abschnitte sind in der Abbildung 47 markiert. Bei der Klassendefinition wird der Identity Kontext implementiert. Im Konstruktor wird ein Verweis auf das «Web.Config» File des Projektes eingefügt. Dieser Verweis bindet die physische MS SQL Datenbank in das Projekt ein. Ohne diesen Verweis, könnte EF die Änderungen der Datenobjekte nicht in die Datenbank übernehmen. Dem Datenbankkontext müssen die

definierten Model Klassen explizit mitgeteilt werden. Anhand der DbSet Eigenschaft, werden diese in den Kontext eingebunden.

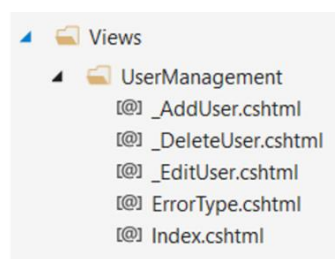
Aufgrund der Ableitung des Identity Kontextes unterscheiden sich die CRUD Funktionen der Accounts von den anderen Komponenten. Beispielsweise muss bei einem Insert über das Interface die vordefinierte Methode aufgerufen werden. Diese Methoden sind als asynchron definiert, weshalb die CRUD Funktionen des Accounts ebenfalls als solche definiert sind (siehe Kapitel 7.3.1- Account CRUD). Bei einer Komponente wie der Analysefrage wird lediglich das Objekt erstellt, die Daten in das Objekt eingefügt und mittels EF über den Datenbankkontext zur physischen Datenbank übertragen.

7.3. Account-Management

In diesem Kapitel wird die Umsetzung der CRUD Funktionen bezüglich der Accountverwaltung aufgezeigt. Die Vorgehensweise der CRUD-Implementationen wurden für die Komponenten der Verwaltung von Fragenpools, Skalen, Stammdatenblöcke, Analysefragen und der Stammdatenfragen übernommen.

Für jede neue Komponente, die in das Projekt und die Datenbank hinzugefügt wird, ist eine neue Controller Klasse notwendig. Dies fördert die Sourcecode Verwaltung und die logischen Zusammenhänge des Projektes. Somit sind die CRUD Funktionen jeweils einem Controller unterstellt und können übergreifend eingesetzt werden. Ein weiterer Faktor für die Aufteilung der Funktionen ist das MVC Prinzip. Bei der Erstellung eines Controllers generiert Visual Studio für die Views, die im neuen Controller verwaltet werden, einen Unterordner im Verzeichnis der View. Dieser Ordner wird nach dem Controller benannt. Dies hilft, die Umsetzung des MVC Prinzip optimal umzusetzen, denn in diesem Verzeichnis werden die zugehörigen Views erstellt. Als Beispiel dient die Abbildung 48, welche die Verzeichnisstruktur der Views anhand des User Managements aufzeigt.

Abbildung 48: Die Verzeichnisstruktur der Views vom User Management







In der Abbildung 49 ist die Anzeige der Accounts sowie der Buttons, welche für die Erstellung, Veränderung und Löschung eines Accounts sind, ersichtlich.

Abbildung 49: Account-Management aus SEPAT

Data Table of all users

New

Excel CSV PDF Column visibility Show 10 entries Search:

Definition	Email	Firstname	Lastname	ISO	Company	Site	Locked	Actions
Klient	ralph@visp.ch	ralph	zurbriggen	DE	Lonza	Supply Chain Management	0	 
Klient	kevin@visp.com	Kevin	Eggmann	CH	Lonza	Supply Chain Management	0	 

Die Formatierung der Tabelle wird mit JavaScript Klassen vorgenommen, die von der Website der DataTables¹⁹ bereitgestellt werden (siehe Kapitel 7.10. Export Gesamtausprägungen für die Definition der Export Buttons). Die Einbindung der Daten und der Konfigurationen der Datentabelle ist mit JQuery realisiert worden, da die JavaScript Klassen auf JQuery basieren. Die Konfigurationen einer DataTable sind in einem separaten JavaScript-File festgelegt. In der Abbildung 50 sind die Einstellungen der DataTable für die Accounts ersichtlich.

Abbildung 50: Konfigurationen der DataTable für die Anzeige der Accounts

```

14 $( '#tableUsers' ).DataTable({
15     // enable the paging if there are multiple pages
16     paging: true,
17     // change the quantity of rows per table
18     lengthChange: true,
19     // enable the searching function
20     searching: true,
21     // enable the ordering function per column
22     ordering: true,
23     info: true,
24     autoWidth: false,

```

Die Daten werden mittels einem ViewModel auf die Razor View übertragen. Das ViewModel enthält das Objekt eines Accounts. Dieses Objekt wird aus der Datenbank abgerufen und in die View eingebunden. Ein Beispiel zur Anzeige eines ViewModels mittels C#-Code ist in der Abbildung 51 anhand der Zeilengenerierung der Datentabelle aus der Abbildung 49 zu sehen.

¹⁹ Quelle: <https://datatables.net/>, 17.07.2017

Abbildung 51: Razor View des Account-Managements

```
<tbody>
@foreach (var item in Model)
{
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.AccountType.Definition)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Email)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Firstname)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Lastname)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Country.ISO)
        </td>
        <th>
            @Html.DisplayFor(model => item.Site.Company.Name)
        </th>
        <td>
            @Html.DisplayFor(modelItem => item.Site.Name)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Locked)
        </td>
        <td>
            <a href="#" class="btn btn-success" onclick="EditUser('@item.Id')"><i class="glyphicon glyphicon-pencil"></i></a>
            <button class="btn btn-success" onclick="DeleteUser('@item.Id')"><i class="glyphicon glyphicon-trash"></i></button>
        </td>
    </tr>
}
```

Die Account Daten können mittels der Abfragesprache Linq aus der Datenbank gelesen werden. Die Linq Abfrage umfasst lediglich ein «Select» auf die Tabelle der Accounts, die alle in derselben Site registriert sind. Die Datenbankresultate werden unmittelbar auf die View übertragen. Auf der View können diese in einer foreach Schleife durchiteriert und mittels den HTML Helper Tags angezeigt werden. Unter der Verwendung der ViewModels wird die dynamische Datenanzeige auf der View ermöglicht.

7.3.1. Account CRUD

Es sind Bootstrap Modals eingesetzt worden, um die Dynamik der Oberfläche zu optimieren. Ein Modal ist eine Dialogbox, die auf einer View eingebunden wird und beim Aufruf eine eigene Oberfläche darstellt. Die Bootstrap Modals sind Komponenten und somit ein Teil des Bootstrap Frameworks (Modal, kein Datum). In der Abbildung 52 ist die Erstellung eines neuen Anwenders durch einen Administrator ersichtlich.

Abbildung 52: Bootstrap Modal für die Erstellung eines neuen Accounts

Das Modal wird auf der Razor View, wo die Datentabelle definiert ist, eingebunden. Bei einem Klick auf den «New»-Button wird ein OnClick Event ausgeführt, welches eine Funktion in einem eingebundenen Skript File aufruft. Die AddUser Funktion initialisiert die URL für die Anfrage der spezifischen Controller-Methode. Ausserdem wird die Antwort, in diesem Fall eine PartialView, der Controller Methode mittels JQuery in das Bootstrap Modal eingefügt.

Die Generierung des Modals und der Aufruf zum Controller sind in der Abbildung 53 dargestellt.

Abbildung 53: AddUser Bootstrap Modal Aufruf aus externem Script File

```

<div class="modal" id="myModalAdd">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
        <h4 class="modal-title">Add New User</h4>
      </div>
      <div class="modal-body" id="myModalBodyAdd">
        <!-- Content of the modal body -->
      </div>
    </div>
  </div>
</div>
<!-- /.modal-content -->
</div>
<!-- /.modal-dialog -->
</div>
<!-- /.modal -->

```

PartialView _AddUser.cshtml

In externem Script File:

```

var AddUser = function () {
  var url = "/UserManagement/AddUser";

  $("#myModalBodyAdd").load(url, function () {
    $("#myModalAdd").modal("show")
  })
}

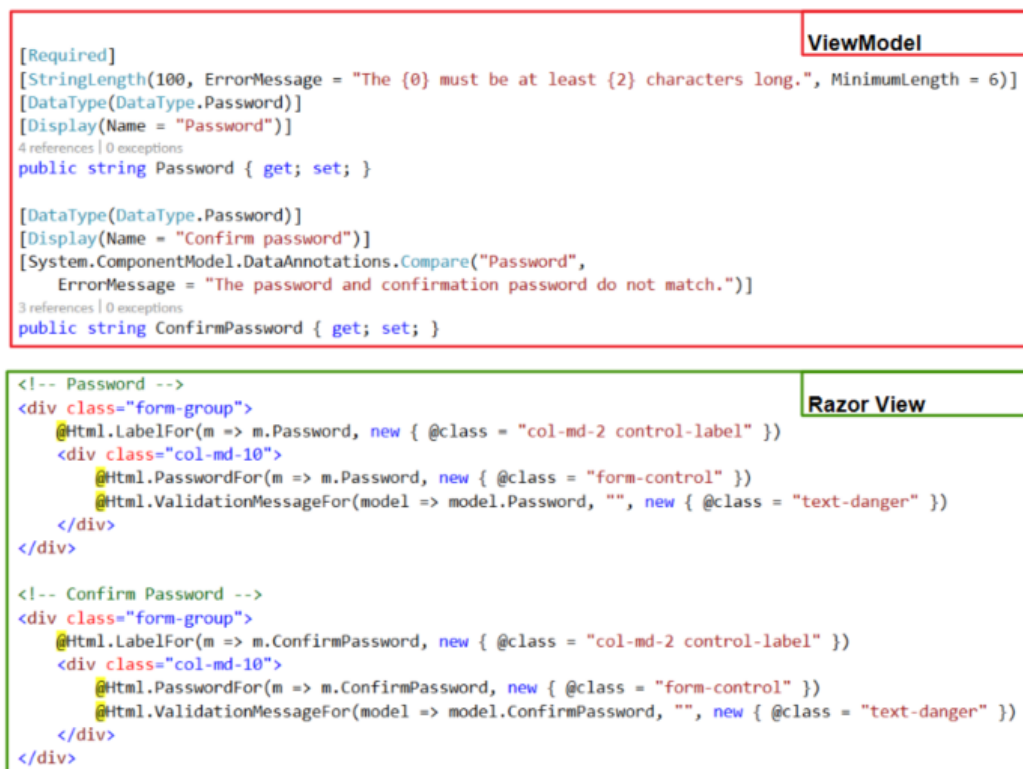
```

Um Fehlermeldungen bei unvollständigen Pflichtfeldern auszugeben, wird die «jquery.validate.unobtrusive.js» Klasse eingesetzt. Dies ermöglicht eine vereinfachte und schnelle Umsetzung der Fehlermeldungen. Die Klasse muss lediglich an die Form ausgerichtet werden. Damit das JavaScript File das spezifische Form Tag erreichen kann, muss der folgende Tag in einem Script Tag der Razor View erstellt werden.

```
<script>
    $.validator.unobtrusive.parse("#myForm");
</script>
```

Um die Fehlermeldungen für den Anwender auf der View einzubinden, werden die HTML Helper Tags eingesetzt. Da ein ViewModel auf der View angezeigt wird, kann die Fehlermeldung spezifisch auf das Objekt im ViewModel definiert werden. Die Einbindung der Meldungen erfolgen mittels dem HTML Helper Tag «@Html.ValidationMessageFor». Durch dieses Tag wird beim Unterlaufen eines Fehlers die Nachricht des ViewModels aufgerufen und angezeigt. Die Abbildung 54 verdeutlicht die Implementation der Fehlermeldungen.

Abbildung 54: Sourcecode Teile eines ViewModels AddUser und einer Razor View



Nachdem ein Anwender alle Pflichtfelder ausgefüllt hat, können die Eingaben gespeichert werden. Aus diesem Grund wird eine Form erstellt, die das ViewModel an den Controller zurücksendet. Zusätzlich umfasst die PartialView «AddUser» in Abbildung 55 zwei dynamische AJAX Funktionen. Bei der Eingabe einer Email Adresse, wird bereits während der Eingabe überprüft, ob die eingegebene Email bereits vorhanden ist. Die zweite Funktion ist die Autocompletion. Gibt der Anwender im Eingabefeld der Länder beispielsweise «GER» ein, erhält dieser von der Datenbank alle Ländernamen, die mit diesen Buchstaben beginnen. Die Ausführung der AJAX Abfragen werden im Form Tag beigefügt. Ein Beispiel eines erweiterten Form-Tags und die wesentliche Sicherheitsmassnahme, die vorgenommen wird, ist in der Abbildung 55 ersichtlich.

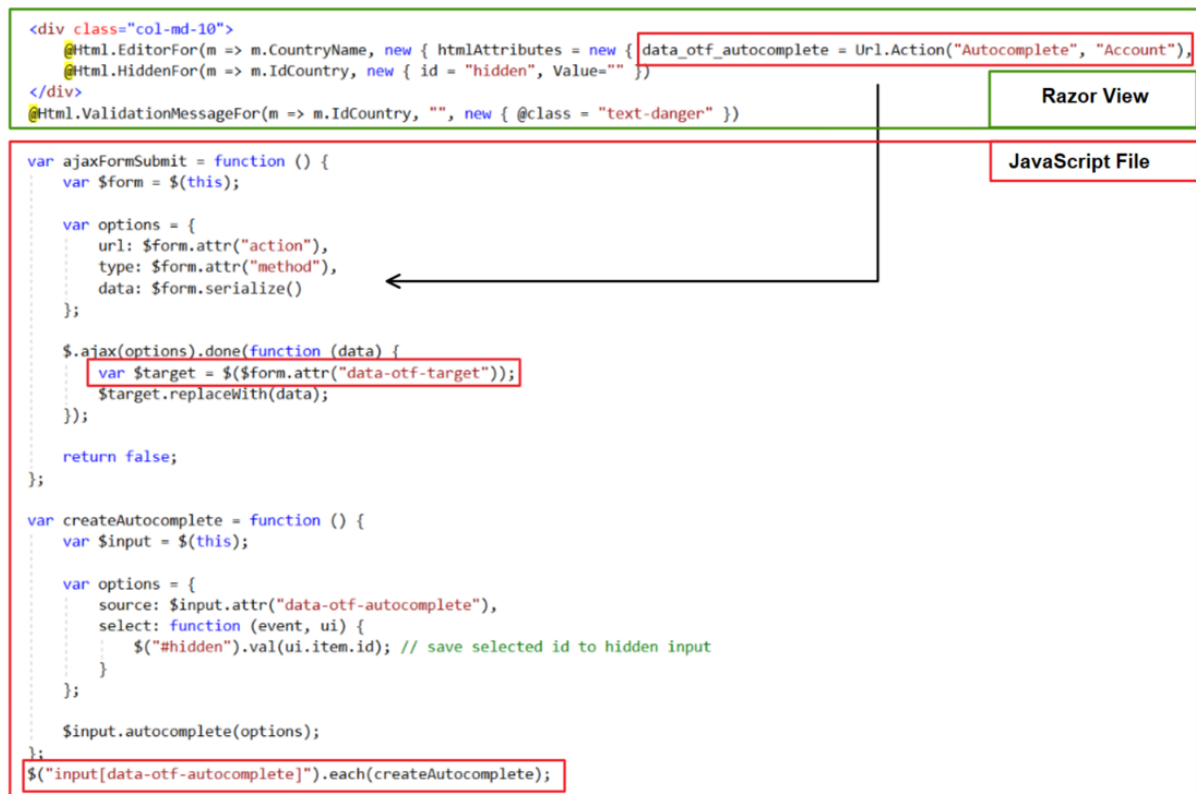
Abbildung 55: Razor PartialView Code-Ausschnitt der Funktionalität Add User

```
<form id="myForm" role="form" action="@Url.Action("AddUser", "UserManagement")" method="post" data-otf-ajax="true" data-otf-target="#countrylist">
  <div class="form-horizontal">
    @Html.ValidationSummary(true, "", new { @class = "text-danger" })
    @Html.AntiForgeryToken()

    <div class="form-group">
      @Html.LabelFor(model => model.Firstname, htmlAttributes: new { @class = "control-label col-md-2" })
      <div class="col-md-10">
        @Html.EditorFor(model => model.Firstname, new { htmlAttributes = new { @class = "form-control" } })
        @Html.ValidationMessageFor(model => model.Firstname, "", new { @class = "text-danger" })
      </div>
    </div>
  </div>
```

Die erweiterte Form veranschaulicht ein Post der eingegebenen Daten auf den Controller. Die Controller-Methode muss wiederum als Post-Methode gekennzeichnet werden. Die Tags «data-otf-ajax» und «data-otf-target» werden für die Validierung der Eingabefelder der Länder und der Email Adresse eingesetzt. Diese Tags ermöglichen die dynamische Ausführung eines Ajax Request auf den Controller, welcher die Eingabe verarbeitet und mögliche Resultate mittels JSON Formates an die View zurücksendet. Die Abbildung 56 visualisiert das Beispiel der Autocompletion.

Abbildung 56: Implementierung der Autocompletion



Die Definition von data-otf-autocomplete im HTML Helper Tag EditorFor ist für die Navigation des Datenstromes und der Auslösung der AJAX Anfrage zuständig. Durch die angefügten data-otf Tags in der Form, wird dynamisch eine Anfrage zum Server ausgelöst. Diese Anfrage wird über das JavaScript File definiert und ausgeführt. Die URL.Action aus dem EditorFor Tag füllt die AJAX Optionen im JavaScript File. Ohne diese Informationen besteht kein Routing auf den Server. Die JQuery Definition \$.ajax(options).done() wartet auf die AJAX Antwort und fügt die Daten ins Target ein. Anhand der data-otf Attributen kann aus dem Form Tag den Auslöser festgestellt werden. Anschliessend werden die Daten eingefügt. In diesem Fall wird lediglich der Identifikator des Landes in ein HiddenFor (Hiddenfield) gesetzt. Dieses HiddenFor ist direkt mit dem ViewModel verbunden, weshalb bei einem Submit dieser Identifikator an den Controller weitergeleitet wird.

Die Einsetzung des Tags @HTML.AntiForgeryToken() unterstützt die Sicherheit der Webapplikation. Primär werden Angriffe wie Cross-Site Request Forgery erschwert. Um Fehler zu vermeiden, muss die Controller Methode ebenfalls mit der Annotation [ValidateAntiForgeryToken] des Antifälschungstokens markiert werden (Microsoft Corporation, kein Datum).

In der Abbildung 57 ist eine Post-Methode für die Eröffnung eines neuen Anwenders ersichtlich. Die zwei Annotationen [HttpPost] und [ValidateAntiForgeryToken] sind ausdrücklich anzugeben. Ohne die Post Annotation kann die View nicht auf die Methode verwiesen werden. Die Post Annotation teilt dem Server mit, dass Daten aus der View auf die Controller Methode einzufügen sind. Ähnlich wie bei der Post Annotation ist es beim Antifälschungstoken. Aufgrund der Positionierung des HTML Helper Tags @Html.AntiForgeryToken() in der Form, muss der Controller Methode mitgeteilt werden, dass der Inhalt durch den Antifälschungstoken gesichert wurde.

Abbildung 57: HttpPost Methode AddUser

```
[HttpPost]
[ValidateAntiForgeryToken]
// references | 4 requests | 0 exceptions
public async Task<ActionResult> AddUser(UserAddViewModel modelTrans)
{
    // check if the user has the permission
    if (modelTrans.IdAccountType == 2 || modelTrans.IdAccountType == 1
        && DashboardController.sessionAccountType != 4)
    {
        ViewData["MessageError"] = "You are not able to create a new Site Admin!";
        return View("ErrorType");
    }
    var user = new ApplicationUser...;

    // create user with password and save the hash
    IdentityResult result = await UserManager.CreateAsync(user, modelTrans.Password);
    if (result.Succeeded)
    {
        ViewData["MessageUpdate"] = "Successfully inserted";
    }
    return RedirectToAction("Index", "UserManagement");
}
```

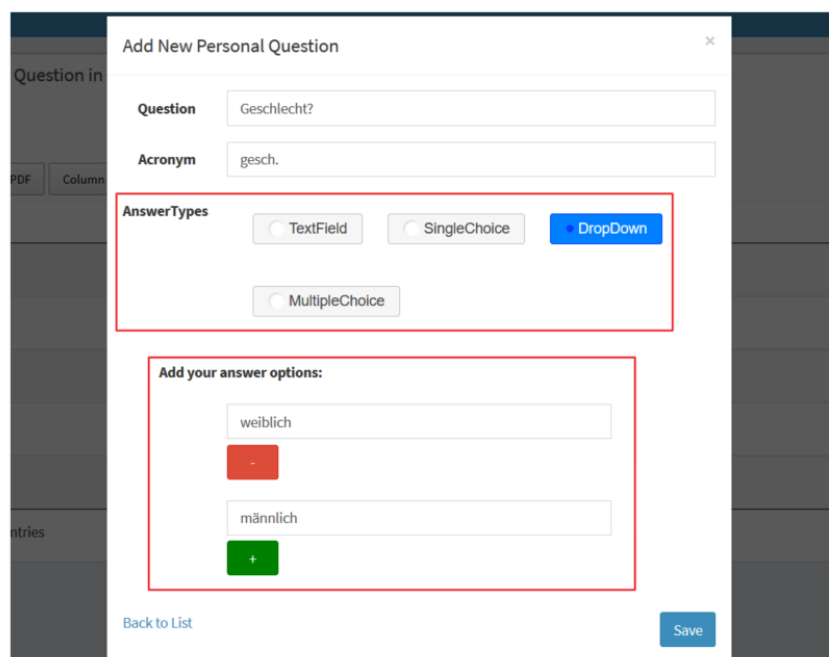
Für die Update und Delete Funktionen des User Managements werden dieselben Vorgehensweisen angewendet. Lediglich unterscheidet sich der Datenstrom bei diesen Funktionen. Beim Delete werden die ViewModels auf den Identifikator und den Namen des Accounts beschränkt. Bei einer Entfernung eines Users erscheint dem Anwender ein Bootstrap Modal, welches prüft, ob er den Datensatz eindeutig aus der Datenbank löschen möchte. Auf diesem Modal ist nur der Name oder Titel des ausgewählten Datensatzes ersichtlich. Der Identifikator wird an den Controller weitergeleitet, um den korrekten Datensatz aus der Datenbank zu löschen.

Bei der Update-Funktion wird das gewählte Objekt aus der Datenbank gelesen. Spezifisch wird für den Anwender das ViewModel erstellt. Je nach View oder Berechtigung des Anwenders differenzieren sich die Eigenschaften eines ViewModels. Somit sind nicht alle Daten des gewählten Objektes editierbar.

7.4. Stammdatenfragen Antwortoptionen

Die Abbildung 58 visualisiert die Erstellung einer neuen Stammdatenfrage. Während der Erstellung kann der Anwender den Stammdatenantworttypen festlegen. Nach der Auswahl muss das UI dementsprechend angepasst werden. In der Abbildung 58 ist eine Dropdownliste ausgewählt worden, weshalb der Anwender dynamisch die Antwortoptionen bestimmen kann. Bei der Umsetzung von dynamischen Benutzereingaben wird die JavaScript-Library JQuery benötigt. Der JQuery Code führt nach der Auswahl eines Antworttyps eine UI Änderung durch. Diese ist in der Abbildung 58 im Bereich «Add your answer options» gekennzeichnet.

Abbildung 58: Definition der Antwortoptionen einer Stammdatenfrage



Die View Logik wurde in einem Pseudocode zusammengeführt um die Verständlichkeit zu verbessern. Der Pseudocode ist in der Abbildung 59 ersichtlich. Der Originalcode ist im beigelegten .NET Projekt in der Klasse «_AddStandingDataQuestion.cshtml» aufgeführt.

Abbildung 59: Pseudocode der dynamischen Erstellung von Antwortoptionen

```
// wo die Elemente gezeichnet werden
definiere das TargetPane

// setze das aktuelle Target als invisible
TargetPane.visibility = "hidden"

if(antwortTyp == 1){
    // setze das neue Target als visible
    TargetPane.visibility = "visible"

    // zeichne das neue Target
    TargetPane.empty()
    TargetPane.append("HTML Tags hier");
}

if(antwortTyp == 2,3 oder 4){
    // setze das neue Target als visible
    TargetPane.visibility = "visible"

    // zeichne das neue Target
    click event addElement {
        TargetPane.empty()
        TargetPane.append("HTML Tags hier");

        // wenn ein weiteres element generiert wird
        newElement = getVorherigesElement.next
        // an das letzte Element anfügen
        getVorherigesElement.after(newElement)

        // löschen eines elements
        click event removeElement = {
            getAusgewähltesElement(selectedId)
            removeAusgewähltesElement.remove()
        }
    }
}
```

Ein TargetPane ist im Sourcecode der View als ein div eingesetzt worden. Dieses div enthält einen Identifikator, damit aus dem Script Bereich Änderungen vorgenommen werden können. Beim Ausführen des Bootstrap Modals ist das TargetPane als leer und unsichtbar deklariert. Nach der Auswahl eines Antworttyps über die Radiobuttons, fügt der Script Bereich die nötigen HTML Tags in das TargetPane hinzu. JQuery ermöglicht die dynamische Generierung der Tags, weshalb kein Pageload notwendig ist. Um die entsprechenden HTML Tags in das TargetPane einzufügen, wird vorerst geprüft, welcher Antworttyp selektiert worden ist. Wie aus dem Pseudocode ersichtlich ist, wird für die Antworttypen Multiple Choice, Single Choice und der Dropdownliste identischer HTML Code generiert. Es können beliebig viele Antwortoptionen eingefügt wie auch entfernt werden. Der Antworttyp eines Textfeldes verlangt keine weiteren Antwortoptionen. Bei der Durchführung einer Selbstevaluation wird ein leeres Textfeld auf die View gesetzt.

Nach der Erstellung der Antwortoptionen durch den Administrator, müssen die Daten gespeichert werden. Das ViewModel übermittelt lediglich die statisch eingegeben Daten. Somit umfasst das ViewModel ausschliesslich die Frage und die Abkürzung. Die dynamischen Elemente konnten nicht in das ViewModel eingebunden werden. Als Alternative für die Datenübermittlung an den Controller wird die FormCollection eingesetzt. Damit die FormCollection die Daten identifizieren kann, werden die Elemente innerhalb des Form-Tags

erstellt. Demzufolge können die generierten Daten aus der `FormCollection` der View aufgerufen werden. Diese muss lediglich als Parameter an die Methode des Controllers übergeben werden.

Der Zugriff auf die individuellen Antwortoptionen wird durch die Identifikatoren der View Tags garantiert. Die Abbildung 60 zeigt die Controller Methode, welche die Antwortoptionen aus der `FormCollection` liest und in eine Liste der Antwortoptionen einfügt. Anschliessend werden die gewonnenen Daten in die Datenbank abgespeichert.

Abbildung 60: Antwortoptionen aus der `FormCollection` lesen

```
private void saveMultipleOptionsDB(StandingDataQuestionAddViewModel model, FormCollection formData)
{
    int nrOptions = model.NrOptions;
    int idQuestionSaved = getRelatedQuestionId(model);
    // prepare to save options to db
    List<MultipleOptionsType> listOptionsToDb = new List<MultipleOptionsType>();

    for (int i = 1; i <= nrOptions; i++)
    {
        // read data from formcollection
        if (formData["field" + i] != null)
        {
            MultipleOptionsType m = new MultipleOptionsType();
            m.Option = formData["field" + i];
            m.IdStandingDataQuestion = idQuestionSaved;
            listOptionsToDb.Add(m);
        }
    }

    try
    {
        foreach(var item in listOptionsToDb)
        {
            _db.MultipleOptionsType.Add(item);
        }
        _db.SaveChanges();
    }
    catch(Exception e)
    {
        ViewData["MessageUpdate"] = "Something went wrong. Try one more time!";
    }
}
```

7.5. Skalen-Management

In der Abbildung 61 wird das Skalen-Management aufgezeigt. Die markierten Blöcke unterteilen die View in zwei Bereiche. Einerseits dient die Unterteilung zur Kennzeichnung der beiden `PartialViews` und andererseits zum Aufzeigen, dass die Daten der `PartialView` unterschiedlich bezogen werden. Die `PartialView 2` ist von der `PartialView 1` abhängig. Die `PartialView 1` präsentiert die verfügbaren Fragenpools und deren Fragentypen. Hingegen zeigt die `PartialView 2` die Analysefragen an, die zum spezifischen Fragentypen angehören. Die Analysefragen, die bereits einer Skala zugehören sind in der `PartialView 2` blau markiert. In der `PartialView2` findet ebenfalls die Selektion der Analysefragen statt. Mit einem Klick auf die gewünschte Zeile werden diese hinzugefügt beziehungsweise entfernt.

Abbildung 61: Übersicht der Fragenzuordnung im Skalen-Management

1

Add Questions to the Scale

QuestionPool

Supply Chain Management Pool

▼

QuestionType

Selbsteinschätzung im Berufsalltag

▼

2

Save Questions

Questions

[Check Row Count](#)

Show 10 entries

Search:

QuestionType	Question	Acronym	Reversed?
Selbsteinschätzung im Berufsalltag	Wenn sich Widerstände auftun, finde ich Mittel und Wege, mich durchzusetzen.	WID	0
Selbsteinschätzung im Berufsalltag	Die Lösung schwieriger Probleme gelingt mir immer, wenn ich mich darum bemühe	LÖS	0
Selbsteinschätzung im Berufsalltag	Es bereitet mir keine Schwierigkeiten, meine Absichten und Ziele zu verwirklichen.	ZIL	0
Selbsteinschätzung im Berufsalltag	In unerwarteten Situationen weiss ich immer, wie ich mich verhalten soll.	USI	0
Selbsteinschätzung im Berufsalltag	Auch bei überraschenden Ereignissen glaube ich, dass ich gut mit ihnen zurechtkommen kann.	ÜRE	0

Die Daten werden sequentiell auf die View übertragen. In einem ersten Schritt werden die Fragenpools sowie die Fragentypen im Controller an das ViewModel angefügt. Der Anwender kann in der PartialView 1 über die Dropdownlisten der Fragenpools und Fragentypen die Analysefragen auf die PartialView 2 laden. Vom ausgewählten Fragenpool wird der Identifikator und vom Fragentypen der Name in jeweils ein Hiddenfield gesetzt. Sobald der Fragentyp geändert wird, versendet die PartialView 1 einen AJAX Request an den Controller. Die Controller Methode führt eine Datenbankabfrage durch, um die passenden Daten in das ViewModel der PartialView einzusetzen. Dem AJAX Aufruf wird die generierte PartialView 2 zurückgesendet. In einem vordefinierten div Tag wird die retournierte PartialView 2 eingebunden. Demzufolge wird kein Pageload durchgeführt, welcher die Usability beeinträchtigt. Der Pseudocode in der Abbildung 62 zeigt die Verarbeitung der Benutzerinteraktionen mit den Dropdownlisten der Fragenbogen und Fragentypen an (siehe Projekt Klasse «AddQuestionsToScale.cshtml»).

Abbildung 62: JQuery Code der PartialViews vom Skalen-Management

```

document.ready(){
    aktiviereDropDownliste für Fragenpool
    DropDownlistModifikation(){
        clear renderFragenTargetPane()

        setHiddenFieldGewählterFragenpool
        clearHiddenFieldGewählterFragenTyp

        AjaxRequest für die Fragentypen im gewählten Fragenpool
        AjaxRequest.success{
            clearFragentypenDropDownListe

            FragentypenDrowdownListe.append("HTML Tags hier");

            disableFragenpoolDropDownListe
        }

        AjaxRequest.error{
            "Fehlermeldung"
        }
    }

    click event SelectDropDownListeFragenpool {
        renderFragenTargetPane
    }

    // wenn Fragen bereits zugeordnet worden sind
    if (HiddenFieldGewählterFragenTyp != 0){
        renderFragenTargetPane
    }

    renderFragenTargetPane(){
        siehe Abbildung 49
    }
}

```

Die Implementation der Funktion `renderFragenTargetPane` ist in der Abbildung 63 dargestellt. Diese Funktion fügt in das `div` die `PartialView 2` ein. Falls der Controller keine Analysefragen zum ausgewählten Fragentypen und Fragenpool findet, wird ein Null Wert an den AJAX Aufruf zurückgesendet. Aus diesem Grund wird eine Prüfung durchgeführt, bevor der HTML Code der `PartialView 2` auf die View eingesetzt wird.

Abbildung 63: Methode `renderPaneSection` für die Analysefragen

```

function renderPaneSection() {
    // create rendersection in paneQuestions - where questions will be loaded
    $("#paneQuestions").append("<div id='renderQuestionsOfTypePool'></div>")
    // div where to load the partial view
    var $QuestionOfPool = $("#renderQuestionsOfTypePool");

    if ($("#QuestionTypesOfPool").val() != "--select--") {
        // set selected id into hidden field
        $("#IdSelectedType").val($("#QuestionTypesOfPool").val())
        // transmit id pool and type name
        var url = "/AnalysisScales/QuestionsOfTypePool?QuestionTypesOfPool=" + $("#IdSelectedType").val() +
            "&QuestionPool=" + $("#IdSelectedPool").val();

        $QuestionOfPool.load(url, function (response, status, xhr) {
            // if html is empty there are no questions
            if ($QuestionOfPool.html() == "") {
                sweetAlert("Oops...", "There are no Questions!", "info");
            } else {
                $QuestionOfPool.html();
            }
        });
    } else {
        $QuestionOfPool.remove();
    }
}

```

7.5.1. Fragen zur Skala zuordnen

Um die selektierten Fragen abzuspeichern, wird im ViewModel der PartialView 2 eine zusätzliche Eigenschaft erstellt. Die Eigenschaft enthält einen numerischen Wert namens `SelectedQuestion`. Beim Laden der PartialView 2 wird in der Datenbank geprüft, ob die Analysefragen bereits zur ausgewählten Skala angehören. Falls dies zutrifft, wird der dementsprechende Identifikator eingesetzt. Beim Einfügen der Daten in die PartialView 2 validiert ein Script der DataTable, ob die Eigenschaft `SelectedQuestion` einen Wert enthält oder nicht. Hinsichtlich dieser Prüfung werden die Zeilen markiert, die bereits in der Datenbank bestehend sind.

Die Analysefragen werden mittels Linq aus der Datenbank gelesen. Linq eignet sich optimal für komplexe Datenbankabfragen durchzuführen und das spezifische ViewModel mit den geeigneten Daten zu füllen. In der Abbildung 64 ist die Select-Abfrage der Analysefragen ersichtlich. Es werden lediglich die Analysefragen der selektierten Skala in die Liste des ViewModels `QuestionsInScaleViewModel` hinzugefügt.

Abbildung 64: Linq Datenbankabfrage der Analysefragen in einer Skala

```
// get selected pool id
int idPool = Int32.Parse(QuestionPool.ToString());
int idType = Int32.Parse(QuestionTypesOfPool.ToString());

// get questions of types in current pool
List<QuestionsInScaleViewModel> list =
    (from q in _db.Question
     from t in _db.QuestionType
     where t.IdQuestionType == q.IdQuestionType
           && q.IdQuestionType == idType
           && t.IdQuestionPool == idPool
     select new QuestionsInScaleViewModel
     {
         IdQuestion = q.IdQuestion,
         Demand = q.Demand,
         Acronym = q.Acronym,
         Reversed = q.Reversed,
         IdQuestionType = q.IdQuestionType,
         QuestionTypeName = q.QuestionType.Name,
         // is it already connected to the scale?
         SelectedQuestion = (from s in _db.Scale
                             from q2 in s.Questions
                             where s.IdScale == selectedScale
                                   && q2.IdQuestion == q.IdQuestion
                             select q2.IdQuestion).FirstOrDefault()
     }).ToList();
```

Die Speicherung der selektierten Analysefragen wird anhand eines Pseudocodes in der Abbildung 65 erklärt (siehe Sourcecode in Anhang XII). In einem ersten Schritt wird das aktuelle Skala Objekt aus der Datenbank bezogen und deren bestehenden Analysefragen in eine Liste abgespeichert. Zusätzlich wird eine neue Liste initialisiert, um den entgültigen Speichervorgang durchzuführen.

Damit die Objekte des ViewModels durchiteriert werden können, wird eine foreach Schleife erstellt. Jedes Objekt wird geprüft, ob es auf der PartialView 2 selektiert worden ist oder nicht. Bei einer Selektion wird ein neues Analysefragenobjekt erstellt und in die Liste hinzugefügt. Nach der Iterierung der foreach Schleife wird die Datenbank angepasst und den Anwender zurück auf die View navigiert.

Abbildung 65: Speichern der neuen und Anpassung der bestehenden Analysefragen

```
SaveSelectedQuestionsInScale(Liste des ViewModels)
{
    Skala = GetAktuelleSkala von der Datenbank

    bestehende Fragen einer Skala = Skala.Fragen

    Neue Liste aus Fragen initialisieren

    foreach(Fragen im ViewModel)
    {
        if(Wenn SelectedQuestion != 0)
        {
            Initialisiere ein neues Fragenobjekt
            Fülle das Fragenobjekt mit den ViewModel Frage
            Setzte das Objekt in die FragenListe
        }
    }

    Die Fragenliste speichern

    zurück zur View des Skalen-Management navigieren
}
```

7.6. Segmentierung

Die Segmentierung wird für die spezifische Auswertung einer Selbstevaluation durchgeführt. Ziel bei der Umsetzung ist es, die Usability gegenüber Selevor zu steigern. Aus diesem Grund wird eine dynamische Segmentierung erstellt. In der Abbildung 66 ist ein Beispiel einer Segmentierung ersichtlich.

Abbildung 66: Übersicht der Segmentierung einer Skala

Title	Text	From	To	Actions
Sehr tiefe Selbstwirksamkeitserwartung	Ihre Einschätzung betreffend Ihre Selbst...	0	25	
Mittlere Selbstwirksamkeitserwartung	Ihre Einschätzung betreffend Ihre Selbst...	51	75	
Hohe Selbstwirksamkeitserwartung	Ihre Einschätzung betreffend die Selbst...	76	100	

Damit die Verwaltung, beziehungsweise die CRUD Funktionen, auf einer View erledigt werden können, ist JQuery, JQuery-UI und AJAX eingesetzt worden.

Die Datentabelle der bestehenden Segmente wird mit den Daten des ViewModels gefüllt. Die Modifikationen durch den Anwender in der Datentabelle werden ebenfalls unter der Verwendung des ViewModels an den Controller überwiesen. Nach der Speicherung der Modifikationen oder Entfernung eines Segmentes, wird die PartialView erneut generiert. Die dynamische Einfügung der angepassten PartialView in die grafische Oberfläche wird mit AJAX durchgeführt. Die HTML Helper Tags gewährleisten die Erstellung von AJAX Formen in der Razor View. Es muss kein zusätzlicher JavaScript Code implementiert werden, der den AJAX Aufruf definiert und die Antwort einfügt. Diese Aufgaben werden vom AJAX Form Tag übernommen. Wie die AJAX Form implementiert wird, ist in der Abbildung 67 dargestellt.

Abbildung 67: AJAX Form in Razor View

```
@using (Ajax.BeginForm("AddSegment", "Wizard",
    new AjaxOptions {
        InsertionMode = InsertionMode.Replace,
        UpdateTargetId = "navmenu" },
    new { id = "myFormSegments"
    })
{
    @Html.AntiForgeryToken()
    @Html.Hidden("IdScale", ViewData["IdScaleSelected"])
    @Html.Hidden("NewSelected", "false")
}
```

Damit die AJAX Form in der Razor View Veränderungen durchführen kann, unterstützt eine JavaScript Klasse namens «jquery.unobtrusive-ajax.js». Die Einbindung dieser Klasse bezweckt die JQuery Verarbeitung des AJAX Form Tags. Anstelle des HTML Helper Tags `Ajax.BeginForm` könnten beispielsweise JQuery AJAX Methoden eingesetzt werden. Ein Beispiel für diese Vorgehensweise ist im Kapitel 7.3.1. – Account CRUD erklärt worden. Dies ist lediglich eine weitere Vorgehensweise, wie eine grafische Oberfläche dynamisch verändert werden kann. Der Vorteil des HTML Helper Tags liegt darin, dass weniger Sourcecode implementiert werden muss. Durch die `AjaxOptions` ist der Unterschied ersichtlich. In den `AjaxOptions` wird das `div` per zugewiesenem Identifikator aktualisiert. Die Angabe des `div`, welches verändert werden soll, wird in der Eigenschaft `UpdateTargetId` festgelegt. Die Aktualisierung ermöglicht der `InsertMode` mit der Eigenschaft `InsertMode.Replace`. Somit wird der Inhalt des `navmenu` Tags durch den HTML Code der neu generierten PartialView ersetzt.

Die Erstellung eines Multisliders (Regler in Abbildung 66) ist mit der JavaScript Library JQuery-UI implementiert worden. In der Abbildung 66 sind beispielsweise drei Sliders auf einem Multislider generiert worden. Folglich werden mehrere Sliders auf einem Bereich (Multislider) eingesetzt. Hierbei liegt die Problemstellung der Implementierung bei der JQuery-UI Dokumentation. Die JQuery-UI Dokumentation bietet ausschliesslich Lösungen eines Sliders an.

Vor der Generierung des Multisliders müssen die einzelnen Slider eruiert werden. Ein Slider symbolisiert jeweils ein bestehendes oder ein neues Segment. Die Implementierung der Sliders ist in der Abbildung 68 als Pseudocode illustriert. Der vollständige Code der Abbildung 68 und Abbildung 69 ist im beigelegten Projekt in der Klasse «_Segments.cshtml» erstellt worden.

Abbildung 68: Pseudocode der Einbindung von den Segmenten in den Multislider

```

Array allerExistierendenSegmente
Array Segmente für den Multislider
Int Total der Segmente = 0

funktion Ajax GET existierende Segmente () {
  // Antwort Formate JSON
  AjaxResponse.success {
    for(iteriere durch die JSON Objekte) {
      Segment Objekt erstellen
      Segment Objekt in Array allerExistierendenSegmente ablegen
    }
    Int DifferenzVorNach Differenz zwischen aktuellen Start und vorherigen End Segment initialisieren
    Int Y für die Iterierung in While Schleife
    Int I für die Iterierung allerExistierendenSegmente
    Boolean SegmentierungErledigt wenn der Array allerExistierendenSegmente durchiteriert wurde

    while(!SegmentierungErledigt) {
      if(Daten vorhanden?)
      {
        bestehendes RangeObjekt für in den Multislider initialisieren und einfügen
        y++;
        Total der Segmente berechnen
        if(i == 0) {
          if(erstes Element von allerExistierendenSegmente.Start != 0) {
            lehres RangeObjekt für in den Multislider am Anfang initialisieren und einfügen
            y++;

            if(Wenn nur das aktuelle Element von allerExistierendenSegmente existiert) {
              lehres RangeObjekt für in den Multislider am Ende initialisieren und einfügen
              y++;
            }
          } else if(nächstes Element von allerExistierendenSegmente nicht existiert) {
            lehres RangeObjekt für in den Multislider am Ende initialisieren und einfügen
            y++;
          }
        } else {
          berechne DifferenzVorNach
          if(DifferenzVorNach > 1) {
            lehres RangeObjekt für in den Multislider initialisieren und einfügen
            y++;
          }
          if(Falls das Element das Letzte, das Total != 100, allerExistierendenSegmente.End != 100)
            lehres RangeObjekt für in den Multislider am Ende initialisieren und einfügen
            y++;
        }
        i++
        if(!nächstes Element vorhanden)
          SegmentierungErledigt = true
      } else {
        lehres RangeObjekt für in den Multislider initialisieren und einfügen
        y++;
      }
    }
  }
}

```

Der grün markierte Bereich fügt die existierenden Segmente als Sliders in den Multislider ein. Hingegen der rote Bereich des Pseudocodes bezweckt die Festlegung der Sliders, die nicht in der Datenbank existieren. Verglichen mit der Abbildung 66, sind die gelben Sliders im roten Bereich und die roten Slider im grünen Bereich definiert.

Die Funktionsweise des Codes befolgt jeweils dieselbe Struktur. Diese Struktur bezieht sich immer auf das vorherige Segment. Falls kein Segment unmittelbar am aktuellen Segment

angebunden ist, wird ein Slider zwischen diesen beiden Segmenten eingefügt. Das erste sowie das letzte Segment muss explizit überprüft werden. Wenn beispielsweise das erste Segment nicht mit dem Wert 0 beginnt und es gleichzeitig das Einzige Segment ist, müssen vor und nach dem Segment jeweils ein Slider für neue mögliche Segmente gesetzt werden. Ein ähnliches Verfahren gilt für das letzte Segment im Array der bestehenden Segmente.

In der Abbildung 69 ist der Pseudocode der Handhabung des Multisliders ersichtlich. In diesem Codeabschnitt werden die Benutzerinteraktionen abgefangen und in den Multislider übernommen.

Abbildung 69: Pseudocode der Generierung des Multisliders

```
wenn AJAXResponse aus Abbildung 54 ist fertig (){  
    Start Wert des ausgewählten Slider initialisieren = function(UI)  
    End Wert des ausgewählten Slider initialisieren = function(UI)  
    Index Wert des ausgewählten Slider initialisieren = function(UI)  
  
    // (grün(false) od. gelb(true)  
    Status Wert des ausgewählten Slider initialisieren = function(UI)  
  
    Slider initialisieren = {  
        min: 0  
        max: 100  
        Segmente des Multisliders einfügen  
        Slider Event = function(Event aus dem UI){  
            Start = Start Wert Funktion aufrufen  
            End = End Wert Funktion aufrufen  
            Index = Index Wert Funktion aufrufen  
            Status = Status Wert Funktion aufrufen  
  
            if(Status = true){  
                setze Start auf neuen Input des Sliders  
                setze Ende auf neuen Input des Sliders  
            }else{  
                setze Start auf existierenden Input des Sliders  
                setze Ende auf existierenden Input des Sliders  
            }  
        }  
    }  
}
```

Die Handhabung des Multisliders wird mit vordefinierten JQuery-UI Funktionen implementiert. Nach dem AJAX Aufruf und der Ausrichtung der bestehenden Segmente auf dem Multislider werden Funktionen für die Start-, End-, Status- und Index Werte angelegt. Diese Funktionen werden durch die Veränderung eines Sliders vom JQuery-UI aufgerufen. Aus diesem Grund müssen die vier Funktionen das UI als Parameter erhalten um die ausgewählten Werte festzulegen. Bei diesen vier Werten handelt es sich jeweils um die vorhandenen Daten des selektierten Segmentes (Slider). Diese werden anschliessend direkt mittels JQuery an den spezifischen Slider weitergeleitet damit dieser neu gezeichnet werden kann.

7.7. Sortierungsverfahren einer Selbstevaluation

Die grafischen Elemente der Sortierungsverfahren sind von der Website JQuery Script.net²⁰ importiert worden. Der Import dieser JavaScript Library umfasst das Drag & Drop Prinzip und reduziert den Designaufwand des Sortierungsverfahrens. Das Sortierungsverfahren beinhaltet zwei Spalten mit den jeweiligen Fragen. Ein Beispiel ist in der Abbildung 70 dargestellt.

Abbildung 70: Darstellung einer manuelle Sortierung

Profilfragen	0. 1. Question Block
2. Ausbildung aktuell	1. Alter
4. Ihre Deutsch-Note (Schweizer System) im letzten Zeugnis/Bericht	4.0. Wenn sich Widerstände auftun, finde ich Mittel und Wege, mich durchzusetzen.
4. Selbsteinschätzung	4.3. In unerwarteten Situationen weiss ich immer, wie ich mich verhalten soll.
4.4. Auch bei überraschenden Ereignissen glaube ich, dass ich gut mit ihnen zurechtkommen kann.	1. 2. Question Block
4.5. Schwierigkeiten sehe ich gelassen entgegen, weil ich meinen Fähigkeiten immer vertrauen kann.	0. Geschlecht
4.6. Was auch immer passiert, ich werde schon klarkommen.	4.2. Es bereitet mir keine Schwierigkeiten, meine Absichten und Ziele zu verwirklichen.
4.7. Für jedes Problem kann ich eine Lösung finden.	4.1. Die Lösung schwieriger Probleme gelingt mir immer, wenn ich mich darum bemühe
4.8. Wenn eine neue Sache auf mich zukommt, weiss ich, wie ich damit umgehen kann.	3. Ihre Mathe-Note (Schweizer System) im letzten Zeugnis/Bericht
4.9. Wenn ein Problem auftaucht, kann ich es aus eigener Kraft meistern.	

In der Abbildung 70 ist ein manuelles Sortierungsverfahren dargestellt. Die farbliche Unterscheidung bezweckt die Erkennbarkeit der einzelnen Komponenten einer Selbstevaluation.

- Blau = Stammdatenfragen
- Grün = Analysefragen
- Grau = Fragenblöcke

Der SEPAT Prototype ermöglicht zusätzliche Sortierungsverfahren für eine Selbstevaluation. Die verschiedenen Sortierungsverfahren lauten wie folgt:

- Manuelles Drag & Drop durch den Anwender
- Automatisiert die Reihenfolge der Komponenten festlegen
- Automatisiert alle Skalen und Stammdatenblöcke als Fragenblöcke festlegen
- Automatisiert die Reihenfolge der Komponenten in einem Fragenblocke festlegen
- Automatisiert nur die Reihenfolge der Fragenblöcke festlegen

²⁰ Quelle: <http://www.jqueryscript.net/>, 18.07.2017

Die erwähnten Sortierungsverfahren basieren auf derselben PartialView. Je nach der Auswahl des Sortierungsverfahrens wird eine PartialView aufgerufen und dementsprechend modifiziert. Damit die Implementierung nachvollzogen werden kann, werden die Entwicklungen des manuellen und des automatisierten Sortierungsverfahrens analysiert.

Der Pseudocode in der Abbildung 71 ist in drei Arten von Blöcken aufgeteilt worden (siehe Sourcecode in Projektklasse «WizardController.cshtml» der Methode «SequenceDragDrop»). Der grüne Block beinhaltet die Listen, welcher die Daten für den linken oder rechten Bereich der PartialView vorbereitet. Diese Listen werden in der Methode SequenceDragDrop verarbeitet und am Ende der Durchführung in das ViewModel eingefügt um die Weiterleitung an die View zu gewährleisten. Der Unterschied zwischen den roten und den blauen Bereichen ist, dass bei den roten Bereichen eine automatisierte und bei den blauen eine manuelle Sortierung durchgeführt wird.

Abbildung 71: Verwaltung der Frageblöcke für die Anzeige

```

Methode SequenceDragDrop (ausgewählte Option){
  // für Zuweisung aus der DB
  Initialisiere und fülle Liste bestehender Analysefragen
  Initialisiere und fülle Liste bestehender Stammdatenfragen

  // für ViewModel - für rechte Spalte
  Initialisiere Liste für Frageblöcke

  if(bestehen Sequenzen){
    Initialisiere und fülle Liste der Skalen ohne Analysefragen in Sequenzen
    Initialisiere und fülle Liste der Stammdatenblöcke ohne Stammdatenfragen in Sequenzen

    ① if(Option == Shuffle von Blöcken, Fragen, Alles){
      Shuffling nach Option
    }else{
      ① Liste der Frageblöcke mit Blöcken und Sequenzen füllen
    }
  }else{
    Fülle Liste bestehender Skalen
    Fülle Liste bestehender Stammdatenblöcke
    Liste für Frageblöcke = null

    ② if(Option == Shuffle){
      Shuffling Skalen und Stammdatenblöcke
      Speicher als Frageblöcke - Abbildung 60
    }

    ③ if(Option == ScalesAsBlocks){
      Speichere Skalen/Stammdatenblöcke als Frageblöcke
      Speichere Fragen als Sequenzen
      Liste Skalen = null
      Liste Stammdatenblöcke = null
    }

    ② if(Option == DragDrop){
      Liste der Frageblöcke mit Blöcken füllen
    }
  }
  ViewModel füllen mit den erstellten Listen (Frageblöcke, Skalen, Stammdatenblöcke)
  PartialView zurücksenden
}

```

In der Implementation der `SequenceDragDrop` Methode werden beim Aufruf alle Analyse- und Stammdatenfragen in jeweils eine Liste abgelegt. Diese Listen werden in der restlichen Methode für die Datenzuweisungen der Listen eingesetzt, die an das `ViewModel` angefügt werden. Aufgrund des hohen Datenaustausches können diese Listen die Anzahl der Zugriffe auf den Datenbankkontext vermindern und die Performance optimieren.

7.7.1. Manuelles Sortierungsverfahren

Beim manuellen Sortierungsverfahren ist die Drag & Drop Funktion zu verstehen. Falls die gewählte Selbstevaluation Sequenzen und Frageblöcke umfasst, werden diese in die Liste der Fragenblöcke gelesen. Dadurch werden die Fragen der bestehenden Sequenzen nicht in die Listen der Skalen und Stammdatenblöcke miteingefügt. Folglich würde die View Fragenduplikate anzeigen.

Die zweite Option des manuellen Sortierungsverfahrens wird durchgeführt, wenn einer Selbstevaluation keine Sequenzen zugeteilt worden sind. In diesem Fall wird die Selbstevaluation auf bestehende Fragenblöcke geprüft und präsentiert.

Der Bereich Selbstevaluation befindet sich in der rechten Spalte der Abbildung 70. Die Daten des `ViewModels` sind lediglich Fragen, die zu keinem Fragenblock hinzugefügt sind. Die nicht zugeordneten Fragen sind in der linken Spalte positioniert. Die Abbildung 70 visualisiert das manuelle Sortierungsverfahren. Dafür sind zwei Fragenblöcke erstellt und jeweils Analyse- oder Stammdatenfragen zugeteilt worden.

In der linken Spalte werden die Daten mittels einem `ViewModel` auf die `PartialView` eingefügt. Hingegen in der rechten Spalte, wo die Reihenfolge der Fragenblöcke und deren Fragen definiert werden, fügt der Anwender individuell die Daten aus der linken Spalte ein. Bei bestehenden Sequenzen werden ebenfalls diese Fragenblöcke mittels einem `ViewModel` übertragen. Nach der Speicherung der aktuellen Fragen und Fragenblöcke in der rechten Spalte müssen diese bei einem Pageload der `PartialView` auf der rechten Spaltenseite erscheinen. Die zugeordneten Fragen dürfen nicht erneut auf der linken Spalte generiert werden (siehe Kapitel 7.7.3 - Speicherung des Sortierungsverfahren).

7.7.2. Automatisiertes Sortierungsverfahren

Das automatisierte Sortierungsverfahren umfasst zwei Sortierungsmöglichkeiten. Die erste Möglichkeit wird während der Implementation «`SetScalesAsBlocks`» benannt. Somit ist der Anwender in der Lage, alle existierenden Skalen und Stammdatenfragenblöcke einer Selbstevaluation als jeweils einen neuen Frageblock zu definieren. Diese Funktionalität überträgt per Ausführung die Skalen und Stammdatenblöcke auf die Seite der

Selbstevaluation. Die Skalen und Stammdatenblöcke werden in Fragenblöcke abgeändert. Die Implementation dieser Funktionalität ist in der Abbildung 72 als Pseudocode dargestellt. Im Anhang XIII ist der vollständige Sourcecode ersichtlich.

Abbildung 72: Implementierung der Methode SetScalesAsBlocks

```
Methode SetScalesAsBlocks (Liste der Skalen, Stammdatenblock Objekt){  
    foreach(Skala in Liste der Skalen){  
        Initialisieren eines Frageblock Objektes  
        Zuweisung der Skaleninformationen in Frageblock Objekt  
  
        Frageblock in DbContext speichern  
  
        foreach(Analysefrage in Skala){  
            Analysefrage als Sequence abspeichern  
        }  
    }  
  
    Initialisieren eines Frageblock Objektes  
    Zuweisung der Stammdatenblockinformationen in Frageblock Objekt  
  
    Frageblock in DbContext speichern  
  
    foreach(Stammdatenfrage in Stammdatenblock){  
        Stammdatenfrage als Sequence abspeichern  
    }  
  
    Sequence in DbContext speichern  
}
```

Im Pseudocode wird die zwischen der Transformation der Skalen und des Stammdatenblockes unterschieden. Eine Selbstevaluation beinhaltet mehrere Skalen jedoch nur einen Stammdatenblock, der verarbeitet werden muss. Anhand des roten Bereiches ist die Skalen Verarbeitung verdeutlicht worden. Im grünen Bereich wird lediglich ein neuer Frageblock für den Stammdatenblock erstellt.

Die zweite automatisierte Sortierungsmöglichkeit basiert auf dem Shuffle Prinzip. Das Shuffle Prinzip wird mithilfe des Fisher-Yates Shuffle Algorithmus durchgeführt. Ziel des gewählten Algorithmus ist die Durchmischung der Fragenblöcke oder Fragen einer Selbstevaluation. (Fisher-Yates shuffle, kein Datum).

7.7.3. Speicherung des Sortierungsverfahren

Für die Datenspeicherung wird kein ViewModel generiert. Grund dafür sind HTML Helper Tags, die an das ViewModel gebunden sind. Diese können nur erschwert mittels JavaScript generiert werden. Aus technischen Gründen wäre es realisierbar, jedoch müssten die HTML Helper Tags vordefiniert werden und mittels JavaScript eingeblendet beziehungsweise ausgeblendet werden. Beim aktuellen Lösungsansatz wird die Listenstruktur einer Spalte in ein JSON Format formatiert. Nachfolgend wird ein Auszug aus einem generierten JSON File angezeigt:

```
[{"idQuestionblock":190,"children":[{"idStandingdataquestion":132},{"idStandingdataquestion":128}}],  
{"idQuestionblock":191,"children":[{"idQuestion":19},{"idQuestion":20}]}
```

Dieser Auszug des JSON Files beinhaltet zwei JSON Parent-Objekte mit jeweils zwei Child-Objekten. Die Parent-Objekte stellen die Frageblöcke mit deren unterstellten Analyse- oder Stammdatenfragen dar.

Diese JSON formatierte Listenstruktur wird in einem Script verarbeitet und mittels einem AJAX Post an den Controller des Wizards gesendet. Die Controller Methode enthält eine Liste von Objekten als Parameter. Dementsprechend erkennt ASP.NET MVC die Struktur des einkommenden JSON Files und gewährleistet den Übertrag in die Liste der Objekte. Als Voraussetzung gilt, dass die Objekte der Liste und das JSON Resultats eine identische Struktur aufweisen. Falls die Liste keine Werte beinhaltet, wird eine JSON Antwort an die View zurückgesendet. Dabei handelt es sich um eine Fehlermeldung, dass die Daten nicht den Controller erreicht haben oder ein interner Fehler stattgefunden hat. Andernfalls wird der Speichervorgang gestartet. Wie der Speichervorgang durchgeführt wird, ist in der Abbildung 73 als Pseudocode oder im Anhang XIV als Originalcode ersichtlich.

Abbildung 73: Methode zur Speicherung der Sequenzen

```

Methode PutSequence(Liste von Models der JSON Objekte){
    if(modelJSON == null){
        sende eine JSON Fehlermeldung zurück
    }
    Initialisiere Liste NEWSeq der neuen Sequenzen

    Int SequenzenIteration für die Reihenfolge
    Int Identifikator eine Blockes
    Int BlockIteration für die Reihenfolge

    foreach(Fragenblock in modelJSON){
        BlockIteration++

        if(Wenn eine Skala als gesamtes eingefügt wurde){
            Initialisiere Fragenblock Objekt
            Initialisiere Skala Objekt und fülle mit Daten aus DB
            if(Skala Objekt != null){
                Fragenblock Objekt = Skala Objekt
            }
            Initialisiere Stammdatenfragenblock und fülle mit Daten aus DB
            if(Stammdatenfragenblock != null){
                Fragenblock Objekt = Stammdatenfragenblock Objekt
            }
            speichere den Fragenblock Objekt
            neu erstellter Identifikator dem Fragenblock Objekt zuweisen
        }else{
            Anpassung Reihenfolge Wert des Fragenblockes
        }
    }

    foreach(Frage in Fragenblock.Children){
        SequenzenIteration++
        if(bestehen Sequenzen){
            if(Frage bereits in DB Sequenzen){
                Neue Sequence erstellen
                Objekt in NEWSeq
            }else{
                Anpassung Reihenfolge Wert
                Anpassung Fragenblock Identifikator
                Objekt in NEWSeq
            }
        }else{
            Neue Sequence bei Ersterstellung
        }
    }

    Lösche Fragenblöcke ohne Sequenzen
    Speichern der Fragenblöcke
    Update und Speichern der Sequenzen in Liste
}

```

Bei der Generierung der neuen Fragenblöcke und dessen zugeteilten Analyse- und Stammdatenfragen wird eine Liste der neuen Fragen erstellt. In diese Liste werden alle neu ausgewählten sowie die bestehenden Fragen hinzugefügt. Die Liste wird anschliessend in die Datenbank gespeichert. Somit werden die Fragenblöcke durchgehend aktualisiert.

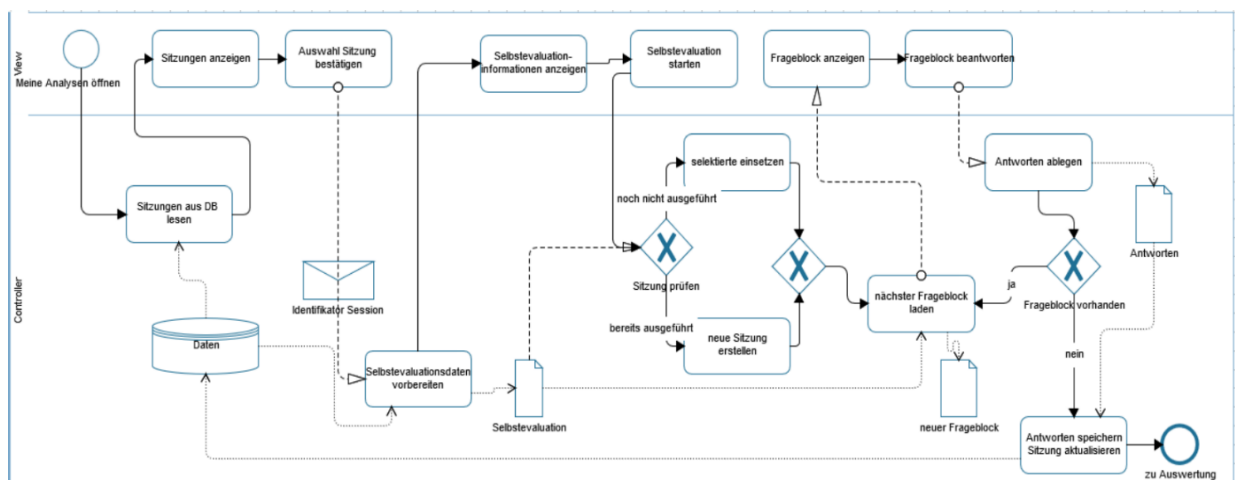
Die Implementierung iteriert durch die Fragenblöcke und deren zugehörigen Fragen. Falls eine Skala oder ein Stammdatenblock in den rechten Bereich der Selbstevaluation verschoben wurde, wird dieser als Frageblock abgespeichert. Nach der Speicherung des Frageblockes wird der Identifikator aus dem Datenbankkontext gelesen. Dieser Schritt ist von Bedeutung, da die neuen Fragen die Zuordnung des Fragenblockes beanspruchen. Die Iteration der Child-Objekte wird nach der Speicherung des Fragenblockes durchgeführt. Falls keine Sequenzen (siehe Kapitel 7.1.5 - Analysis-Management) in der aktuellen Selbstevaluation bestehen,

werden direkt die neuen Sequenzen erstellt. Andernfalls wird geprüft, ob eine Frage bereits in der Datenbank als Sequenz vorhanden ist.

7.8. Selbstevaluation durchführen

Die Abbildung 74 visualisiert den Workflow der Selbstevaluation. Die Prozessschritte der Auswertung sind in der Abbildung nicht einbezogen worden. Mithilfe dieser Abbildung wird die Vorgehensweise einer Selbstevaluationsdurchführung erläutert.

Abbildung 74: Workflow der Selbstevaluation



Die Umsetzung der Anzeige und Navigation beansprucht drei Controller Methoden. Diese werden nach der Festlegung einer Sitzung ausgeführt. Die Methoden werden im Controller Play, Analysis und End bezeichnet. Die Methode Play ist in Abbildung 74 nach dem Task «Selbstevaluation starten» aufzufinden. Der Pseudocode der Abbildung 70 zeigt deren Funktionalität auf (siehe Anhang XV für den Sourcecode).

Abbildung 75: Implementation der Methode Play

```

Methode Play() {
    if(Session nicht aktuell){
        Abbrechen der Selbstevaluation
    }

    if(Session bereits eingesetzt){
        Initialisiere neues Session Objekt
        Speichere Session in DbContext
    }else{
        Startzeit in Session abspeichern
    }
    Speichern der Session in Datenbank

    Initialisiere Liste mit Fragen für die View

    if(Selbstevaluation auf mehreren Seiten präsentieren){
        Liste mit Fragen für View einen Frageblock zuweisen
    }else{
        Liste mit Fragen für View alle Frageblöcke zuweisen
    }

    Anzahl der Sequenzen in ViewData
    Anzahl der aktuellen Sequenzen in ViewData
    Identifikator der Sitzung in ViewData

    Anzeige der View mit der Liste der Fragen
}

```

Nachdem der Anwender den ersten Fragenblock beantwortet hat und die Resultate mittels eines Klicks auf «Submit» an den Controller schickt, wird die Methode Analysis aufgerufen. Die Analysis Methode ist für fortlaufende Datenverarbeitung der Selbstevaluation zuständig. Hingegen wird die Play Methode lediglich beim Start der Selbstevaluation aufgerufen. Die Hauptaufgabe der Methode Analysis ist die Zwischenspeicherung der Antworten in Listen und die Generierung des nächsten Fragenblockes für die View. Die Daten werden mittels einem ViewModel zur Methode und zur View geleitet. Die Abbildung 76 visualisiert die Implementation (siehe Anhang XVI für den Sourcecode).

Abbildung 76: Implementation der Methode Analysis

```
Methode Analysis(Liste der Antwortobjekten){  
  if(ViewModel Daten korrekt){  
    if(Selbstevaluation auf einer Seite präsentieren){  
      Speichern der Liste mit Antwortobjekten  
      Weiter zur Auswertung  
    }else{  
      Speichern der Antwortobjekten in Liste  
  
      if(Wurden alle Fragen beantwortet){  
        Weiter zur Auswertung  
      }  
    }  
  }  
  ViewModel leeren und nächste Fragen zuweisen  
  Anzeige der View mit Liste der Fragen  
}
```

Die Auswahl der Anzeigeeoption «Block per Page» in den Einstellungen der Selbstevaluation wird pro Seite jeweils ein Fragenblock dargestellt. Falls diese Anzeigeeoption nicht selektiert wird, werden alle Fragen der Fragenblöcke auf einer Seite generiert.

Bei der Auswahl «Block per Page» kommuniziert die View mit der Methode Analysis bis alle Fragen beantwortet werden. Dieser Überprüfung wird durch eine zusätzlich Eigenschaft im ViewModel durchgeführt. Die Eigenschaft `alreadySet` enthält einen Wahrheitswert (Boolean), welcher bei True gesetzt wird. Sind alle Eigenschaften der Fragen mit True markiert, werden die Antworten definitiv in die Datenbank übernommen. Wie die Zwischenspeicherung und der entgültige Speichervorgang durchgeführt wird, ist im Pseudocode der Abbildung 77 abgebildet (siehe Anhang XVII für den Sourcecode).

Abbildung 77: Implementation der Methode SaveAnswers

```
Methode SaveAnswers(Liste der Antworten, bool notFinished)
    Initialisiere Antwort Objekt
    // für Multiple Choice
    Initialisiere Liste für mehrere Antwortoptionen
    Initialisiere Stammdatenantwort Objekt

    foreach(Antwort in Liste der Antworten){
        Sitzungs Identifikator in Antwort Objekt einfügen
        Sequenzen Identifikator in Antwort Objekt einfügen

        if(Antwort eine Analysefrage){
            Numerischen Antwortenwert in Antwort Objekt zuordnen
            Antwort Objekt in DbContext einfügen
            Numerischen Antwortenwert zu globaler Skalenresultat Variable addieren
        }else{
            Antwort Objekt in DbContext einfügen
            Speichern des Antwort Objektes
            Identifikator des Antwort Objekt dem Stammdatenantwort Objekt zuordnen

            if(Stammdatenantworttyp == Textfeld){
                Antworttext in Stammdatenantwort Objekt einfügen
                Stammdatenantwort Objekt in Liste mehreren Antwortoptionen einfügen
            }
            if(Stammdatenantworttyp == Dropdown){
                Antworttext in Stammdatenantwort Objekt einfügen
                Stammdatenantwort Objekt in Liste mehreren Antwortoptionen einfügen
            }
            if(Stammdatenantworttyp == Single Choice){
                Antworttext in Stammdatenantwort Objekt einfügen
                Stammdatenantwort Objekt in Liste mehreren Antwortoptionen einfügen
            }
            if(Stammdatenantworttyp == Multiple Choice){
                foreach(Antwortmöglichkeit in Multiple Choice){
                    Initialisiere Stammdatenantwort Objekt
                    Identifikator des Antwort Objekt dem Stammdatenantwort Objekt zuordnen

                    if(Antwortmöglichkeit ist selektiert worden){
                        Antworttext in Stammdatenantwort Objekt einfügen
                        Stammdatenantwort Objekt in Liste mehreren Antwortoptionen einfügen
                    }
                }
            }
        }
    }

    if(!notFinished){
        DbContext in Datenbank übertragen
        Speichern der Liste mehreren Antwortoptionen
    }
}
```

Die Antwortgebung der Anwender werden während der Generierung eines neuen Frageblockes in eine temporäre Liste gespeichert. Es ist möglich, dass ein Anwender die Selbstevaluation abbricht. Folglich würde eine Sitzung ohne Resultate in der Datenbank bestehen. Nach der Beantwortung des letzten Frageblockes wird ein Wahrheitswerte als Parameter an die SaveAnswers Methode übertragen. Dieser Wert löst die Speicherung der erungenen Antworten einer Sitzung aus.

Den Antworten der Analysfragen werden lediglich die ausgewählten numerischen Werte der Likert Skala zugeordnet. Hingegen für die Stammdatenfragen müssen zusätzliche Antwortobjekte erstellt werden. In diese wird der Antworttext abgespeichert. Diese werden jedoch an das individuelle erstellte Antwortobjekt angeknüpft.

Die End Methode wird nach der Speicherung aller Antworten aufgerufen. Als Paramter der Methode wird der Identifikator von der aktuelle Sitzungen übergeben. Anhand der Sitzung wird

die Vollständigkeit der Antworten untersucht. Falls Antworten zu der Sitzung existieren, wird die Feedback View vorbereitet und aufgerufen. Andernfalls müssen die einzelnen Skalenausprägungen berechnet werden. Nach der Addition der Skalenausprägungen wird die Endausprägung über die gesamte Selbstevaluation berechnet. Anschliessend müssen die berechneten Resultate in die Datenbank eingefügt werden.

Die End Methode wird aufgrund ihrer Struktur multifunktional eingesetzt. Deshalb erreicht ein Anwender die View der Resultate ebenfalls über die Endmethode. Dies garantiert der Identifikator der Sitzung als Parameter.

In der Abbildung 78 ist die Entwicklung der «End» Methode dargestellt (siehe Anhang XVIII für den Sourcecode).

Abbildung 78: Implementation der Methode End

```
Methode End(Identifikator der Sitzung){
    Endzeit in Sitzung speichern

    // Addition der Antworten der Likert Skalen einer Skala
    Liste der Skalen-Antworten erstellen
    // Ausprägungen per Skala berechnen
    Liste der Skalen-Resultaten erstellen

    if(Sitzung nicht vollendet){
        Endzeit in Sitzung einsetzen
        Totale Ausprägung der Sitzung berechnen und einfügen

        Anpassung der Sitzung in Datenbank
        Speichern der Resultaten
    }

    Initialisiere Feedback ViewModel
    Feedback ViewModel Identifikator der Sitzung zuordnen
    Feedback ViewModel Skalen und Resultate zuordnen
    Feedback ViewModel spezifisches Feedback zuordnen

    Anzeige der Auswertung und Einbindung des ViewModels
}
```

7.9. Auswertung anzeigen

Die Auswertungen werden anhand der JavaScript Library ChartJS in der View visualisiert. ChartJS bietet mehrere Ansichten an, um die Daten anzuzeigen. Für den SEPAT Prototypen sind die Ansichten eines Bar und Radar Chart umgesetzt worden.

ChartJS basiert auf der Auszeichnungssprache HTML5. Anhand des HTML5 Tags Canvas ist dies zu begründen. Canvas unterstützt die Zeichnung von Grafiken mit Skriptsprachen. Üblicherweise erfolgt dies mit JavaScript. Um die Grafik einzubinden, wird in der View ein «div» mit einem Identifikator «chart» und das Canvas Tag erstellt. Mithilfe von JavaScript wird das Canvas Tag optisch angepasst und die Daten eingefügt. Die Abbildung 79 zeigt, wie ein Canvas Tag aus einem JavaScript File angepasst wird.

Abbildung 79: ChartJS Definition

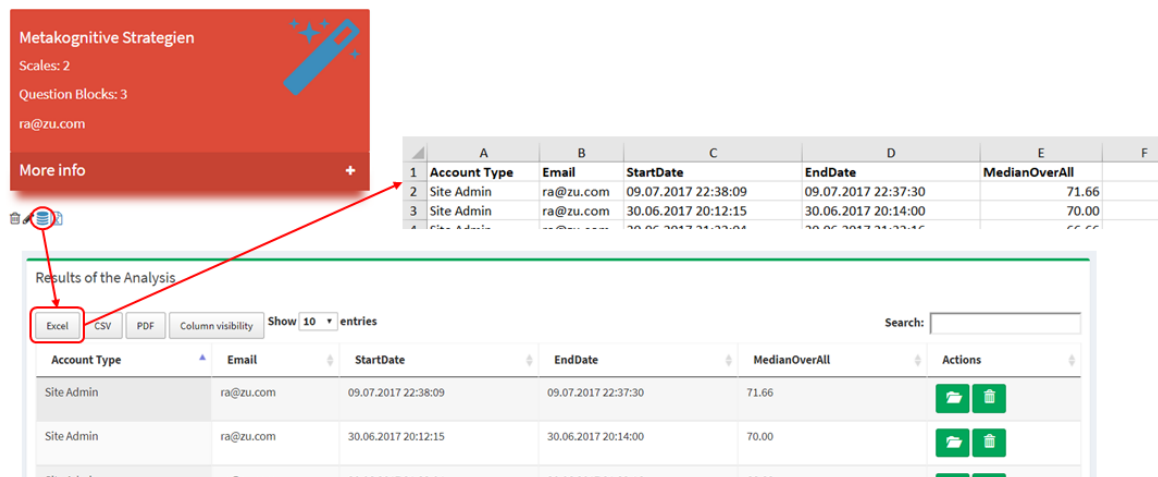
```
var barChartCanvas = $("#barChart_" + index).get(0).getContext("2d");
var barChart = new Chart(barChartCanvas, {
  type: 'bar',
  data: {
    // names of scales
    labels: labelQuestionNames,
    datasets: [
      {
        label: "Values per question",
        fillColor: "rgba(60,141,188,0.9)",
        strokeColor: "rgba(60,141,188,0.8)",
        pointColor: "#3b8bba",
        pointStrokeColor: "rgba(60,141,188,1)",
        pointHighlightFill: "#fff",
        pointHighlightStroke: "rgba(60,141,188,1)",
        // median of each scale from session
        data: dataValues
      }
    ]
  },
  options: {
    scales: {
      yAxes: [{
        // begin and end value
        ticks: {
          beginAtZero: true,
          max: maxY_Axes
        }
      }]
    }
  }
});
```

Der Variablen `barChartCanvas` wird mittels JQuery das Canvas Tag der View zugeordnet. An dieses Canvas Tag werden danach die Konfigurationen und Daten definiert. Anhand der Type Konfiguration wird die Ansichtsart vorgegeben. Beispielsweise bewirkt der Wert «radar» die Anzeige eines Netzdiagrammes. Für die X-Achsenbeschriftungen werden die Daten des ViewModels übernommen, da diese die Skalen- und Fragen Abkürzungen beinhalten. Im Dataset sind mehrere Einstellungen bezüglich der Farben, Breite und Informationen definiert. Zusätzlich werden die Resultate der Skalen oder Fragen hinzugefügt. Die Festlegung der statischen X- und Y-Achse findet in den Options statt.

7.10. Export Gesamtausprägungen

Laut Kundenauftrag soll es dem Administrator ermöglicht werden, Resultate zu exportieren. Die Resultate umfassen alle Antworten der Analyse- sowie Stammdatenfragen. Im Prototype SEPAT wurde der File Export der Resultate teilweise implementiert. SEPAT stellt den Export der Gesamtausprägungen bereit. Aus diesem Grund sind nicht alle gesammelten Daten einer Selbstevaluation im exportierten File vorhanden. Die Abbildung 80 visualisiert den Prozess in der Software.

Abbildung 80: Excel Export der Gesamtausprägungen einer Selbstevaluation



Um die Gesamtausprägungen zu exportieren, muss das Icon der Datenbank ausgewählt werden. Anschliessend navigiert das System auf die Übersicht aller durchgeführten Selbstevaluationen. Wie aus der Abbildung 80 erkennbar ist, kann SEPAT eine Excel, CSV oder PDF Datei erstellen und exportieren. Zusätzlich ermöglicht dieses GUI die Verwaltung der Selbstevaluationen durch die Löschfunktion sowie die Anzeige des Resultates.

Das Design und die Funktionalität der Tabelle in Abbildung 80 wird anhand der Files (JS- und CSS- Klassen) festgelegt. In den JavaScript Klasse sind ebenfalls die Funktionen des Suchfeldes, der Sortierung der Spalten, die Dropdown-Liste für die Definition der «Anzahl pro Seite» und das allgemeine Designverhalten. Die Abbildung 81 zeigt die Imports, die auf der Razor View eingefügt sind.

Abbildung 81: Einbindung der JS und CSS Klassen der DataTable

```
@section Styles {
    <!-- datatable css -->
    <link rel="stylesheet" type="text/css" href="https://cdn.datatables.net/1.10.15/css/jquery.dataTables.min.css" />
    <link rel="stylesheet" type="text/css" href="https://cdn.datatables.net/buttons/1.3.1/css/buttons.dataTables.min.css" />
}

@section Scripts {
    <!-- datatables scripts -->
    <script type="text/javascript" src="https://cdn.datatables.net/1.10.15/js/jquery.dataTables.min.js"></script>
    <script type="text/javascript" src="https://cdn.datatables.net/buttons/1.3.1/js/dataTables.buttons.min.js"></script>
    <script type="text/javascript" src="//cdnjs.cloudflare.com/ajax/libs/jszip/3.1.3/jszip.min.js"></script>
    <script type="text/javascript" src="https://cdn.rawgit.com/bpampuch/pdfmake/0.1.27/build/pdfmake.min.js"></script>
    <script type="text/javascript" src="https://cdn.rawgit.com/bpampuch/pdfmake/0.1.27/build/vfs_fonts.js"></script>
    <script type="text/javascript" src="https://cdn.datatables.net/buttons/1.3.1/js/buttons.html5.min.js"></script>
    <script type="text/javascript" src="https://cdn.datatables.net/buttons/1.3.1/js/buttons.colvis.min.js"></script>

    <script type="text/javascript" src="~/Scripts/results.handling.js"></script>
}
```

Die Events der Export Buttons sind individuell zu erstellen. Aus diesem Grund wird das JavaScript File «results.handling.js» eingesetzt. In diesem File werden die Export Buttons definiert, die auf der Oberfläche angezeigt werden sollen. Ebenfalls wird die Anzahl der

Spalten definiert, die exportiert werden sollen. Die Implementation dieser Export Buttons ist in der Abbildung 82 dargestellt.

Abbildung 82: Definition der Export Buttons

```
// set the export buttons
dom: 'B1frtip',
buttons: [
  $.extend(true, {}, buttonCommon, {
    // call function in js imported js class
    extend: 'excelHtml5',
    exportOptions: {
      // specify the columns to export related to datatable on UI
      columns: [0, 1, 2, 3, 4]
    }
  }),
  $.extend(true, {}, buttonCommon, {
    extend: 'csvHtml5',
    exportOptions: {
      columns: [0, 1, 2, 3, 4]
    }
  }),
  $.extend(true, {}, buttonCommon, {
    extend: 'pdfHtml5',
    exportOptions: {
      columns: [0, 1, 2, 3, 4]
    }
  })
],
'colvis'
```

7.11. Responsive Design

Um das Responsive Design für den SEPAT Prototypen umzusetzen, werden die Entwicklungen auf dem Template AdminLTE²¹ eingefügt. Dieses Template besteht aus mehreren vordefinierten Webseiten. Beispielsweise konnte das Login, Registration und das Dashboard übernommen werden. Damit die Struktur des Template über den gesamten Prototypen übernommen werden konnte, war die Dokumentation des Template von Bedeutung. Die Dokumentation erläutert die Einsetzung die Stylesheets und zusätzlichen JavaScript Klassen.

Die eigenen Entwicklungen werden in das Template mittels Bootstrap eingebunden. Aus diesem Grund besteht nur ein zusätzliches Stylesheet namens «custom.css». In diesem Stylesheet werden die Steuerelemente, wie z. B. Textfelder, Radiobuttons, etc., benutzerfreundlich positioniert. Diese Steuerelemente werden durch die Spaltenverwaltung und inbegriffenen Klassen des Bootstraps Framework ausgerichtet.

²¹ Quelle: <https://adminlte.io/themes/AdminLTE/index2.html>, 23.07.2017

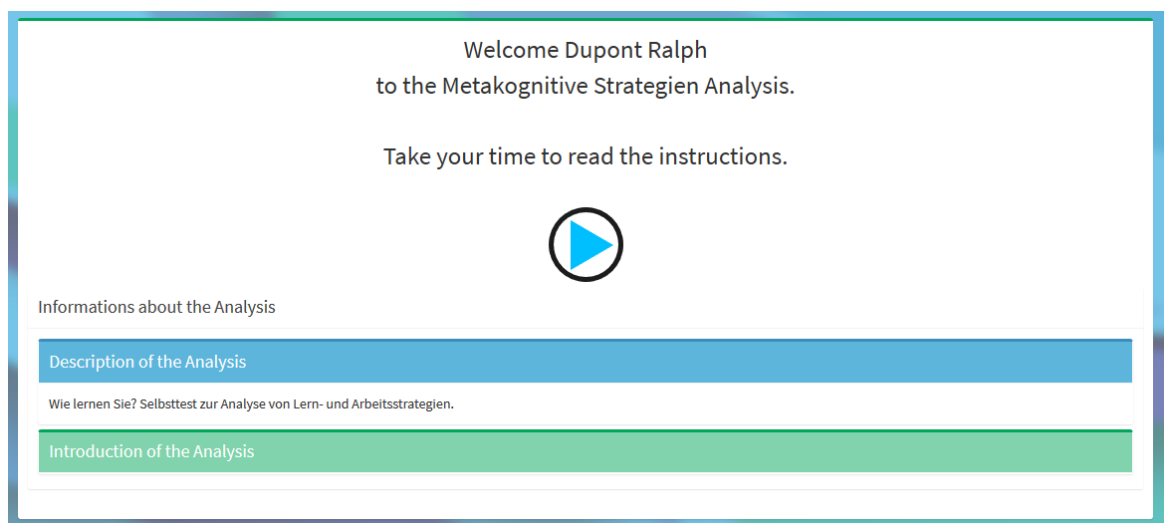
8. SEPAT versus Selevor

In diesem Kapitel wird der SEPAT Prototyp dem Selevor Plug-In gegenübergestellt. In einem ersten Schritt werden drei grafische Oberflächen des SEPAT Prototypen dargestellt. Anschliessend werden die relevantesten Erweiterungen des Kundenauftrages vorgestellt und deren Vorteile erwähnt.

8.1. Durchführung einer Selbstevaluation

In diesem Kapitel wird die Durchführung einer Selbstevaluation des SEPAT Prototypen aufgezeigt. Nach der Auswahl einer Selbstevaluation erscheint die grafische Oberfläche aus der Abbildung 83. In dieser Oberfläche erhält der Anwender Informationen über die Selbstevaluation. Bei einem Klick auf «Description of the Analysis» erscheint die Beschreibung unterhalb des Titels. Das gleiche Verfahren wird bei «Introduction of the Analysis» angewendet.

Abbildung 83: Startoberfläche einer Selbstevaluation in SEPAT



Beim Ausführen der Selbstevaluation wird die Oberfläche aus der Abbildung 84 angezeigt. Diese Oberfläche enthält die Fragen der ausgeführten Selbstevaluation. Wie aus der Abbildung entnommen werden kann, sind nicht ausschliesslich Analysefragen mit deren Likert Skala ersichtlich. Die Umsetzung des Skalen-Managements ist deswegen aus der Abbildung 84 zu entnehmen. Es können anhand des Sortierungsverfahrens Analyse- mit Stammdatenfragen in Fragenblöcken zusammengefügt werden.

Abbildung 84: Durchführung einer Selbstevaluation in SEPAT

Metakognitive Strategien

0%

1. Question Block

Self-generated Question Block for Analysis Metakognitive Strategien

Ich prüfe nach jeder Etappe, ob mein Plan bisher aufgegangen ist.

stimmt nicht
stimmt kaum
stimmt eher
stimmt genau

✓

Wenn eine neue Sache auf mich zukommt, weiss ich, wie ich damit umgehen kann.

stimmt nicht
stimmt kaum
stimmt eher
stimmt genau

✓

Geschlecht

m

Ausbildung aktuell

Berufslehre/Berufsschule

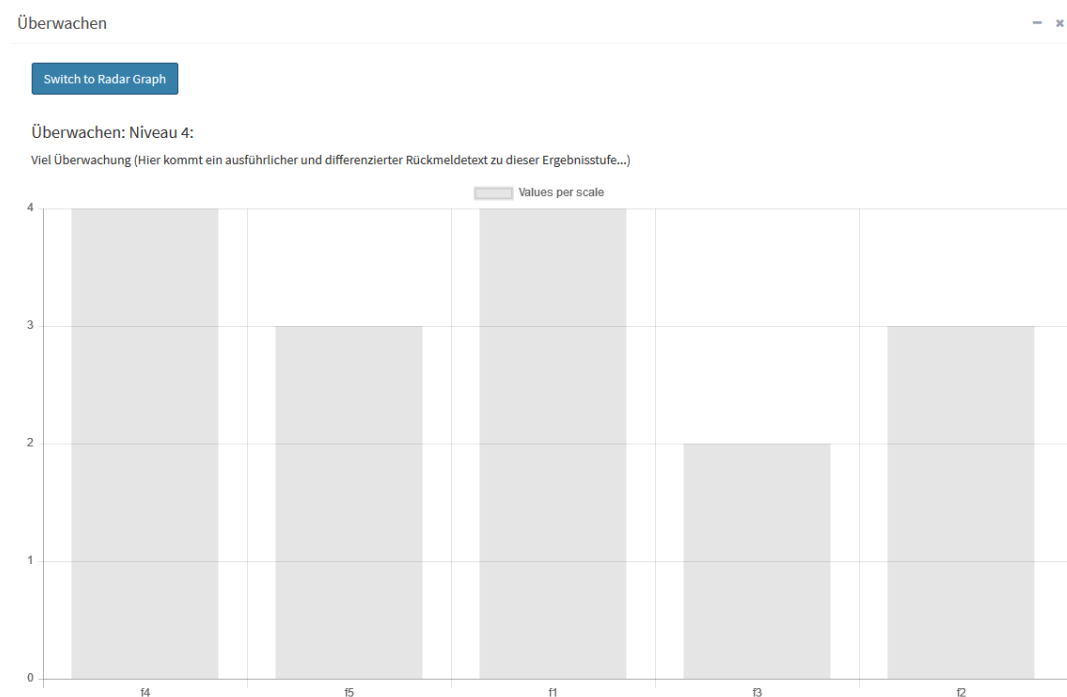
Fachhochschule

Höhere Fachschule

8.2. Anzeige der Auswertung

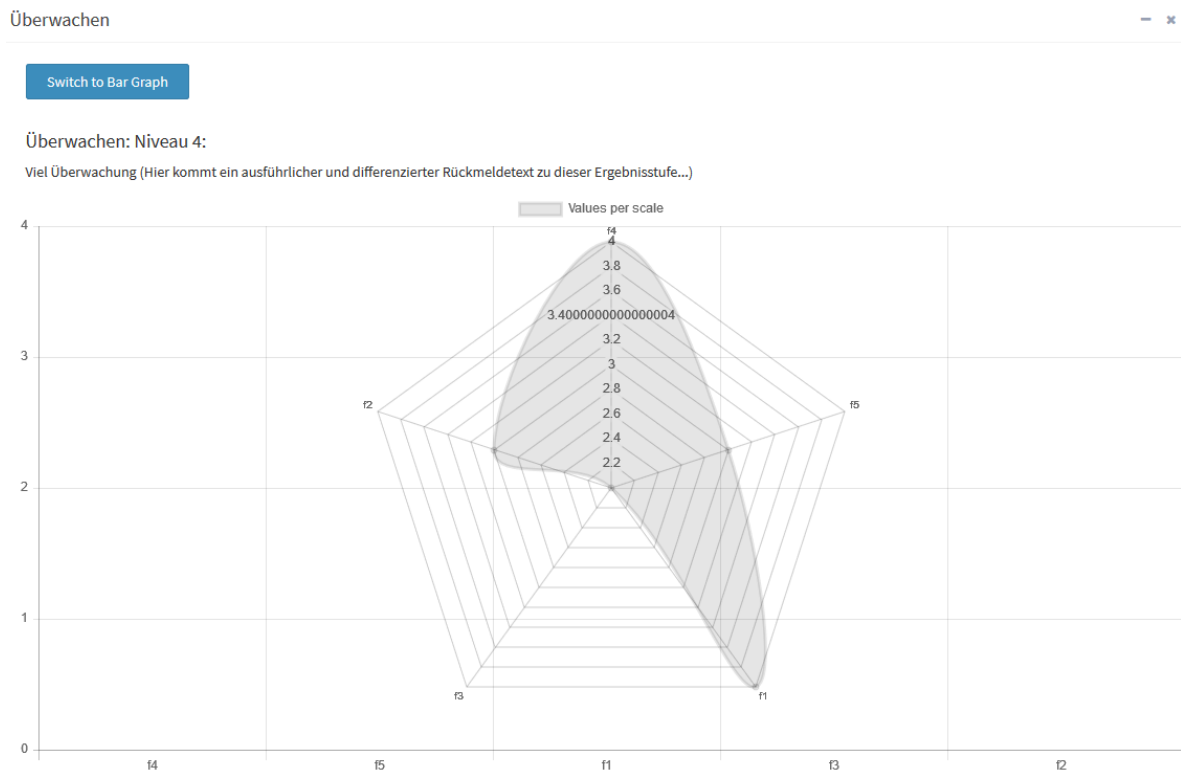
In der Abbildung 85 wird die Auswertung des Fragenblocks «Überwachen» dargestellt. Die Anzeige der Resultate und Antworten erfolgt mittels einem Bar Chart.

Abbildung 85: Bar Chart Auswertung in SEPAT



Der Button «Switch to Radar Graph» wechselt die Ansicht der bestehenden Auswertung von einem Bar Chart in ein Netzdiagramm. In der Abbildung 86 ist ein Netzdiagramm ersichtlich.

Abbildung 86: Netzdiagramm Auswertung in SEPAT

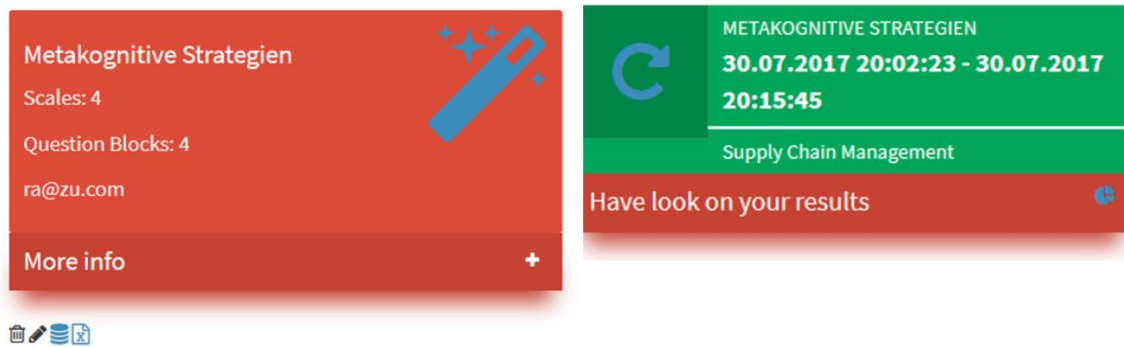


8.3. User Experience

Das allgemeine Design des Prototyps ist auf die Personas ausgerichtet. Laut Kundenauftrag soll die Oberfläche ansprechend für die Administratoren sowie Anwender einer Selbstevaluation sein. Die ausgewählte Template Vorlage «AdminLTE» unterstützt diese Grundlage. Durch dieses Template wird die UX der Personas weitgehend gedeckt. Das Template ist fortgeschritten und besitzt zugleich die aktuellsten Technologien. Somit werden die Bedürfnisse und Anliegen an die UX der verschiedenen Ebenen von der SEPAT Software erfüllt. Vor allem zielt die Benutzerfreundlichkeit auf die Ebenen des Servers und der Sites ab.

Der User-Ebene werden grafische Elemente bei der Verwaltung deren zugeteilten Analysen zur Verfügung gestellt. Diese sind gezielt ausgerichtet worden, dass ein User die Übersicht auf dem Endgerät nicht verliert. Ebenfalls die Einsetzung der JavaScript Library ChartJS bietet interessante Auswertungen für den User (siehe Abbildung 85 & Abbildung 86). In der Abbildung 87 sind standardisierte Blöcke dargestellt. Die grafische Erscheinung eines Blockes enthält je nach Komponente andere Daten.

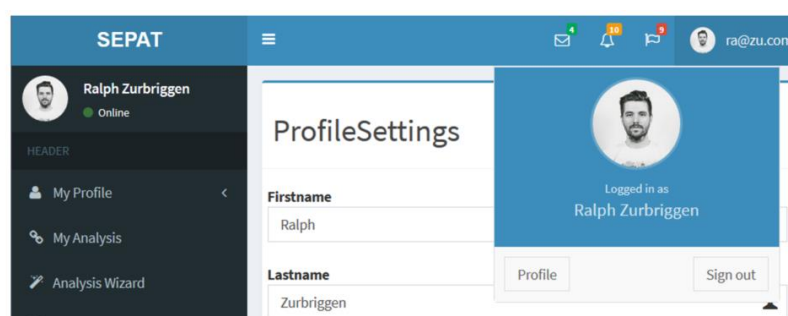
Abbildung 87: Anzeigeblöcke für die Komponenten



Das Dashboard wird optimal für die Administrations-/Siteebene angewendet. Die Master und Siteadministratoren verfügen über eine angenehme Einsetzung der verschiedenen Funktionen. Diese Funktionen werden durch Icons, logische Bereitstellung auf der View sowie wiederkehrende Prozessstrukturen der verschiedenen Komponenten gewährleistet. Beispielsweise ist die Verwaltung der Stammdatenfragen ähnlich aufgebaut wie diejenige der Analysefragen oder des Accountmanagements.

Nach einem erfolgreichen Login in die SEPAT Software stellt das System dem Anwender individuell seine Arbeitsumgebung zur Verfügung. Die Verwaltung der eigenen Arbeitsumgebung wirkt anwenderfreundlicher und teilt den Aspekt einer Social Media Plattform. In der Abbildung 88 ist ein Beispiel einer persönlicheren Arbeitsumgebung ersichtlich.

Abbildung 88: Profil eines Accounts



Im Selevor Plug-In ist der Anwender in der Lage, sein persönliches Profil zu bearbeiten. Die Oberfläche des Profils ist grafisch nicht gleich ausgearbeitet wie bei der SEPAT Anwendung. Im Selevor Plug-In scheint die Anwendung der Profile sekundär zu sein. Hingegen beim SEPAT Prototypen sind Social Media Komponenten für die Gewinnung von Usern relevant.

Bei einer direkten Gegenüberstellung des SEPAT Prototypen mit dem Selevor Plug-In sind lediglich Parallelen in den Grundfunktionalitäten aufzuweisen. Im Design und der Usability weichen diese voneinander ab.

8.4. Usability

Die Bedienerfreundlichkeit der SEPAT Software übertrifft diejenige des Selevor Plug-Ins. Die Nachfolgenden Neuentwicklungen der SEPAT Software verdeutlichen die massgebenden Unterschiede zum Selevor Plug-In.

8.4.1. Wiederkehrende Prozessstrukturen

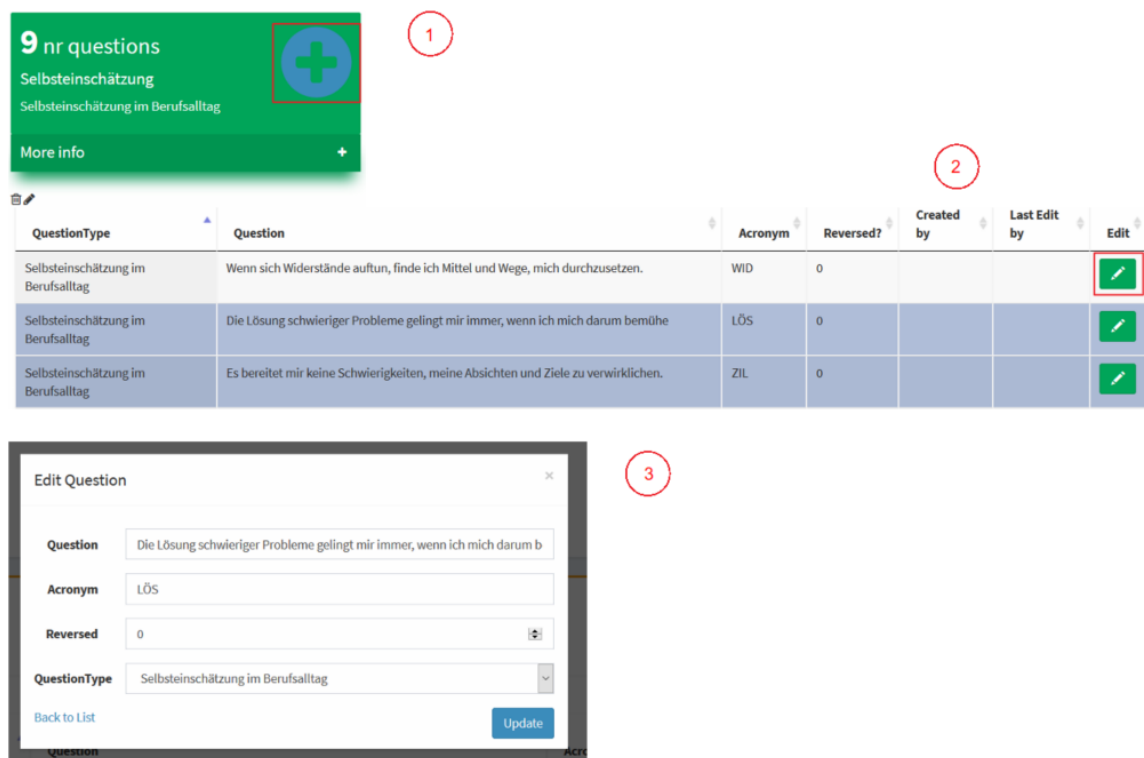
Bei dem Selevor Plug-In werden die möglichen Aktionen, die durch den Anwender vorgenommen werden können, mehrheitlich in Dropdown-Listen präsentiert. Nach der Auswahl einer Aktion navigiert das System auf eine neue Oberfläche. Ein Beispiel ist in der Abbildung 89 ersichtlich. In dieser Abbildung wird eine Frage im Fragenblock bearbeitet

Abbildung 89: Bearbeitung einer Analysefrage im Selevor Plug-In

The screenshot displays the SEPAT software interface. The top section, titled 'FRAGEBLÖCKE (1 - 5 von 5)', contains a table with columns: Titel, Abkürzung, Beschreibung, Anzahl Fragen, Anzahl Feedbacks, Feedback-Block vollständig, and Aktionen. The table lists four question blocks: 'Profildaten', 'Selbsteinschätzung', 'Planen', and 'Überwachen'. The 'Aktionen' column for the 'Planen' block is highlighted with a red box, showing options like 'Fragen bearbeiten', 'Feedback bearbeiten', 'Blockinformationen bearbeiten', and 'Block löschen'. Below the table, the 'FRAGE BEARBEITEN' form is shown, with a red circle highlighting the 'Frage' input field. The form also includes fields for 'Kurztitel' and a checkbox for 'Skala umkehren'.

Diese Vorgehensweise ist im SEPAT System optimiert worden. Die Optimierungen betreffen die UX wie auch die Usability. Die nachfolgende Abbildung 90 veranschaulicht den Unterschied zur Verarbeitung einer Analysefrage zum Selevor Plug-In. Diese Vorgehensweise wird durch die gesamte Struktur des Systems beibehalten. Von der Anzeige bis zur Erstellung oder Verwaltung ist die Umsetzung bei den diversen anderen Komponenten ähnlich aufgebaut worden. Dies dient zur besseren Erinnerung der Prozessabwicklungen und steigert somit die positive Einstellung der Anwender gegenüber der Software.

Abbildung 90: Bearbeitung einer Analysefrage im SEPAT Prototypen



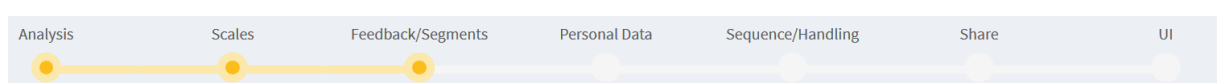
8.4.2. Wizard

Beim Kundenauftrag wird die Umsetzung eines Autorenwerkzeuges für die Erstellung von Selbstevaluationen enorm gewichtet. Aus diesem Grund ist ein Wizard implementiert worden, der den Administratoren bei der Erstellung der Komponenten einer Selbstevaluation assistiert.

Die Fertigstellung einer Selbstevaluation im Selevor Plug-In ist im Kapitel 3.3 – Selevor – das bestehende Plug-In erläutert. Das Vorgehen bei der Erstellung umfasst keine statisch definierte Reihenfolge, weshalb Selevor für unerfahrene Anwender verwirrend wirken kann.

Hingegen bei der SEPAT Software können die Komponenten wie Analyse- oder Stammdatenfragen im Vorfeld oder während der Ausführung des Wizards erstellt werden. Der Wizard bietet zusätzlich einen Progress Balken, der dem Anwender aufzeigt, welche Komponenten noch bearbeitet werden müssen. In der Abbildung 91 ist der Progress Balken ersichtlich.

Abbildung 91: Progress Balken vom Wizard Manager



Neben der visuellen Darstellung der laufenden Bearbeitung eines Komponenten verdeutlicht der Progress Balken ebenfalls die wichtigsten Einstellungen. Beispielsweise befinden sich im Selevor Plug-In die Einstellungen in einem separaten Bearbeitungsort. Die Einstellungen des SEPAT Prototypen sind direkt im Wizard aufzufinden. In der Abbildung 91 sind die Einstellungen anhand des Prozesses «UI» erkennbar.

8.5. Skalen-Management

Wie das Skalen-Management funktioniert ist im Wizard ersichtlich. Im Prozess «Sequence/Handling» ist die Thematik der Abbildung 23 in der Software abgewickelt worden. Eine Visualisierung des Skalen-Managements ist in der Abbildung 70 bereits aufgezeigt worden. Auf der linken Seite sind die ausgewählten Skalen und der Stammdatenblock der Selbstevaluation ersichtlich. Die Unterelemente der Skalen und des Stammdatenblockes definieren die Analyse- oder Stammdatenfragen. Hingegen auf der rechten Seite werden die selektierten Analyse- oder Stammdatenfragen in der spezifischen Reihenfolge dargestellt.

Der grundlegendste Mehrwert gegenüber dem Selevor Plug-In liegt im Skalen-Management. In einer klassischen Erstellung einer Selbstevaluation im Selevor Plug-In wird immer ein Frageblock, beziehungsweise einen Stammdatenblock, erstellt. Anschliessend werden einzeln die Fragen innerhalb des Blockes generiert.

Der SEPAT Prototyp erweitert dieses System. Einerseits wird dieser Prozessablauf bei der Durchführung des Wizards gewährleistet. Andererseits kann ein Administrator Skalen und Stammdatenblöcke unabhängig von Selbstevaluationen erstellen. Diese können zu mehreren Selbstevaluationen geteilt werden. Ausserdem sind die Analysefragen innerhalb einer Skala wiederum mehrfach einsetzbar in diversen Skalen.

8.6. Antwortausrichtungen der Stammdatenfragen

Die erweiterte Antwortausrichtung der Stammdatenfragen bietet einen Mehrwert in der Durchführung der Selbstevaluationen. Hinsichtlich der zwei Antwortausrichtungen im Selevor Plug-In kann im SEPAT Prototypen zwischen dem Textfeld, der Dropdown-Liste, der Single und der Multiple Choice Auswahl unterschieden werden. Für zukünftige Weiterentwicklungen bietet das generierte Datenbankmodell weitere Antworttypen an. Ebenfalls kann die Selbstevaluation mit grafischen Elementen der Antwortausrichtung interessanter und vielseitiger gestaltet werden.

8.7. Einsetzbarkeit des Prototyps

Die Umsetzung der Erweiterung aus Kapitel 4.3.1. Grobarchitektur von SEPAT ermöglicht eine zukünftige Integration der Webapplikation in verschiedenen Unternehmen. Unternehmen sind nicht von einem bestehendem LMS abhängig wie beim Selevor Plug-In. Beim SEPAT Prototypen reicht ein Interview mit dem Softwarebetreiber aus, welcher auf der Siteebene die Unternehmung/Abteilungen erstellt und zusätzlich die notwendigen Accounts für die Verwaltung der Sites und Unternehmen generiert. Nach der Erstellung dieser Sites und Accounts können die Kunden die Webapplikation in deren Unternehmen integrieren.

Schlussfolgerung

In dieser Bachelorarbeit wurde zunächst ein Überblick über den aktuellen Stand des Selevor Plug-Ins geschaffen. Mit Hilfe der Analyse des Selevor Plug-Ins wurden die grundlegenden Funktionen aufgezeigt, die in den neuen SEPAT Prototypen übernommen werden sollten. Aus dieser Analyse wurden die Grundlagen des Kundenauftrages geschaffen. In einem weiteren Schritt wurden die Erweiterungen für den SEPAT Prototypen formuliert.

Nach der Festlegung der Erweiterungsfunktionalitäten wurden diese sowie die grundlegenden Funktionalitäten in die Spezifikation eingebunden. Wichtig bei der Durchführung der Spezifikation waren die Personas. Die Personas visualisierten die potenziellen Zielgruppen, welche die Webapplikation benutzen würden. Hauptgrund der Generierung von den vier Personas war die Benutzung während der GUI-Spezifikation. Anhand der erstellten Personas konnte der Kundenauftrag leichter verstanden und optisch dargestellt werden. Zusätzlich konnte mithilfe der Personas die Erweiterung der Grobarchitektur besser kommuniziert und verstanden werden.

Anschliessend fand die Priorisierung der einzelnen Aufgabenfelder statt. Die Erstellung der Prioritätenliste optimierte durchaus die Implementation. Die Liste unterstützte das Zeitmanagement bei der Einteilung der verschiedenen Komponenten auf die Implementierungsphasen.

In einem nächsten Schritt wurde die Softwarespezifikation im Detail ausgearbeitet. Als Vorgehensweise wurde die GUI-Spezifikation gewählt. Die GUI-Spezifikation konnte optimal in die Zusammenarbeit mit dem Auftraggeber eingesetzt werden. Dank den erstellten Mockups erhielt der Auftraggeber bereits nach zwei Wochen einen ersten Einblick, wie die Software funktionieren wird. Die Ideen sowie die Vorschläge von Seiten des Auftragsgebers betreffend den Funktionalitäten und dem Design der Software wurden unmittelbar während der Sitzung eingebaut. Diese Vorgehensweise ermöglichte die Bedürfnisse des Auftraggebers besser zu verstehen.

Beim Abschluss der Softwarespezifikation und Priorisierung wurden die Technologien und die Softwarearchitektur festgelegt. Es standen mehrere Technologien zur Auswahl. Die Wahl fiel für die Microsoft Technologie ASP.NET MVC. Obwohl mit ASP.NET WebForms programmiert wurde, waren die Vorteile der ASP.NET MVC Technologie überzeugender.

Aufgrund der zeitlich engen Implementierungsphase konnten nicht alle Funktionalitäten in den Prototypen übernommen werden. Der Prototyp deckt die Funktionalitäten von der Prioritätenliste bis «Auswertungen generieren» ab. Somit wurde das Self Assessment

umgesetzt. Die Erstellung der Wizard Lösung, die als Autorenwerkzeug eingesetzt wird, ist ein wichtiger Bestandteil des SEPAT Prototypen. Ein weiterer Faktor ist die Einsetzbarkeit des Systems. Somit kann der Prototyp in Bildungsanstalten wie auch in Personalentwicklungsabteilungen in verschiedenen Unternehmen eingeführt werden.

Eine Selbstevaluation setzt sich aus mehreren Komponenten zusammen. Dies hatte enorme Auswirkungen auf die Implementation. Das Ziel bestand primär in der Umsetzung der Grundfunktionalitäten aus der Prioritätenliste. Sekundär wurden die generierten Ausnahmen behoben.

In nächster Zeit werden mit dem Auftraggeber, Herrn Tribelhorn, weiterführende Gespräche stattfinden. Die Software wird aus diesem Grund in den kommenden Monaten weiterentwickelt. In der nächsten Implementierungsphase werden überwiegend Einschränkungen und Ausnahmen des bestehenden Systems behandelt. Zudem werden die GUIs der Selbstevaluation ausgearbeitet.

Literaturverzeichnis

- Anderson, R. (03. April 2015). *Code! MVC 5 App with Facebook, Twitter, LinkedIn and Google OAuth2 Sign-on (C#)*. Abgerufen am 19. Juli 2017 von <https://docs.microsoft.com/en-us/aspnet/mvc/overview/security/create-an-aspnet-mvc-5-app-with-facebook-and-google-oauth2-and-openid-sign-on>
- Baumgartner, P., Häfele, H., & Maier-Häfele, K. (2002). In *E-Learning Praxishandbuch - Auswahl von Lernplattformen: Marktübersicht - Funktionen - Fachbegriffe* (S. 24). Innsbruck-Wien: Studien-Verlag.
- Borg, I. (07. Juli 2017). Von <https://portal.hogrefe.com/dorsch/likert-skala/> abgerufen
- Büsching, T., & Goderbauer-Marchner, G. (2014). In *E-Publishing-Management* (S. 99,100). Berlin: Gabler Verlag. Abgerufen am 06. Juli 2017 von <https://books.google.ch/books?id=OUUIBAAAQBAJ>
- ClaimTypes Fields*. (kein Datum). Abgerufen am 19. Juli 2017 von https://msdn.microsoft.com/en-us/library/microsoft.identitymodel.claims.claimtypes_fields.aspx
- Cooper, A., Reimann, R., Cronin, D., & Noessel, C. (2014). In W. J. Sons (Hrsg.), *About Face: The Essentials of Interaction Design* (4. Ausg., S. 64). Hoboken, New Jersey, USA: Wiley John + Sons. Abgerufen am 09. Juli 2017 von <https://books.google.ch/books?id=w9Q5BAAAQBAJ>
- Database First development with Entity Framework*. (kein Datum). Abgerufen am 11. Juli 2017 von <http://www.entityframeworktutorial.net/choosing-development-approach-with-entity-framework.aspx>
- DBContext*. (kein Datum). Abgerufen am 19. Juli 2017 von <http://www.entityframeworktutorial.net/EntityFramework4.3/dbcontext-vs-objectcontext.aspx>
- El Moussaoui, H., & Zeppenfeld, K. (2008). In *AJAX: Geschichte, Technologie, Zukunft* (S. 25-33). Heidelberg: Springer-Verlag. Abgerufen am 12. Juli 2017 von <https://books.google.ch/books?id=8P4dBAAAQBAJ>
- Fisher-Yates shuffle*. (kein Datum). Abgerufen am 21. Juli 2017 von <http://www.programming-algorithms.net/article/43676/Fisher-Yates-shuffle>
- Flanagan, D. (2007). In *JavaScript: das umfassende Referenzwerk* (S. 1-10). Heidelberg: dpunkt.verlag. Abgerufen am 12. Juli 2017 von <https://books.google.ch/books?id=KyAHa6gmUgUsC>

- Galloway, J., Rastogi, P., & Dykstra, T. (17. Oktober 2013). *Introduction to ASP.NET Identity*. Abgerufen am 19. Juli 2017 von <https://docs.microsoft.com/en-us/aspnet/identity/overview/getting-started/introduction-to-aspnet-identity>
- Gelbmann, M. (13. August 2012). Abgerufen am 10. Juli 2017 von https://w3techs.com/blog/entry/jquery_now_runs_on_every_second_website
- Gelbmann, M. (10. April 2017). *Nginx reaches 33.3% web server market share while Apache falls below 50%*. Abgerufen am 11. Juli 2017 von https://w3techs.com/blog/entry/nginx_reaches_33_3_percent_web_server_market_share_while_apache_falls_below_50_percent
- Getting Started with Kanban for Software Development*. (kein Datum). Abgerufen am 27. Juli 2017 von <https://dzone.com/refcardz/getting-started-kanban>
- Häusel, H.-G. (2016). In *Brain View: Warum Kunden kaufen* (4. Ausg., S. 249). Freiburg im Breisgau: Haufe-Lexware. Abgerufen am 07. Juli 2017 von <https://books.google.ch/books?id=xAACDAAAQBAJ>
- ILIAS open source e-Learning e.V. (kein Datum). *About ILIAS*. Abgerufen am 03. Juli 2017 von https://www.ilias.de/docu/goto.php?target=cat_580&client_id=docu
- ILIAS open source e-Learning e.V. (kein Datum). *List of Members*. Abgerufen am 03. Juli 2017 von https://www.ilias.de/docu/goto.php?target=cat_3650&client_id=docu
- Jendryschik, M. (28. Dezember 2010). *Mit Personas Projekte menschlich und motivierend gestalten*. Abgerufen am 10. Juli 2017 von <http://jendryschik.de/weblog/2010/12/28/mit-personas-projekte-menschlich-und-motivierend-gestalten>
- jQuery*. (08. April 2012). Abgerufen am 10. Juli 2017 von <http://www.itwissen.info/jquery.html>
- Kirchbergerknorr, D. (kein Datum). *CSS-Grundlagen*. Abgerufen am 10. Juli 2017 von [Css-lernen.net: http://www.css-lernen.net/css-grundlagen.html](http://www.css-lernen.net/css-grundlagen.html)
- Kuhn, C. (2013). Eine Anleitung für User Experience, Design und Webentwicklung. In *UX Design für Tablets*. Frankfurt: entwickler.Press. Abgerufen am 10. Juli 2017 von <https://books.google.ch/books?id=dsZ-DAAAQBAJ>
- Lackers, R., & Siepermann, M. (kein Datum). *Definition HTML*. (S. G. Verlag, Herausgeber) Abgerufen am 10. Juli 2017 von <http://wirtschaftslexikon.gabler.de/Definition/html.html>
- Markle, B. (10. April 2013). *What is bootstrap?* Abgerufen am 27. Juli 2017 von <http://www.inmotionhosting.com/support/edu/joomla-3/using-bootstrap/what-is-bootstrap>

- Microsoft Corporation. (kein Datum). *ASP.NET MVC Overview*. Abgerufen am 14. Juli 2017 von [https://msdn.microsoft.com/en-us/library/dd381412\(v=vs.108\).aspx](https://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx)
- Microsoft Corporation. (kein Datum). *HtmlHelper.AntiForgeryToken-Methode*. Abgerufen am 20. Juli 2017 von [https://msdn.microsoft.com/de-de/library/dd470175\(v=vs.118\).aspx](https://msdn.microsoft.com/de-de/library/dd470175(v=vs.118).aspx)
- Modal. (kein Datum). Abgerufen am 16. Juli 2017 von <https://v4-alpha.getbootstrap.com/components/modal/>
- Mrknowing. (08. November 2013). *Wie funktioniert die 3-Schichten-Architektur?* Abgerufen am 19. Juli 2017 von <http://www.mrknowing.com/2013/11/08/wie-funktioniert-die-3-schichten-architektur/>
- Mullen, T., & Anderson, R. (14. Januar 2017). *Razor Syntax Reference*. Abgerufen am 19. Juli 2017 von <https://docs.microsoft.com/en-us/aspnet/core/mvc/views/razor>
- Nielsen, J. (04. Januar 2012). *Usability 101: Introduction to Usability*. Abgerufen am 08. Juli 2017 von <https://www.nngroup.com/articles/usability-101-introduction-to-usability/>
- Roberts, T. (2006). In *Self, Peer and Group Assessment in E-Learning* (S. 3-9). Hershey: Idea Group Inc (IGI). Abgerufen am 12. Juli 2017 von <https://books.google.ch/books?id=oV-9AQAAQBAJ>
- Rouse, M., & Christiansen, S. (Juni 2014). *Datenbank-Management-System (DBMS)*. Abgerufen am 10. Juli 2017 von <http://www.searchenterprisesoftware.de/definition/Datenbank-Managementsystem-DBMS>
- Rouse, M., & Ligan, J. B. (Juli 2015). *Web server*. Abgerufen am 10. Juli 2017 von <http://whatis.techtarget.com/definition/Web-server>
- Schultens, L. (31. Mai 2010). *GUI- Graphical User Interface*. Abgerufen am 10. Juli 2017 von <http://www.vorlesungen.info/node/1138>
- Spona, H. (2007). In *Jetzt lerne ich Dreamweaver CS3: Websites entwickeln mit HTML, CSS, JavaScript, PHP und ASP* (S. 53-54). Hallbergmoos: Pearson Deutschland GmbH. Abgerufen am 11. Juli 2017 von <https://books.google.ch/books?id=m6NnQfgMcsQC>
- Steyer, M., & Softic, V. (2017). *ASP.NET MVC im Zusammenspiel mit Web APIs und JavaScript-Frameworks*. In *Moderne Webanwendungen mit ASP.NET MVC und JavaScript* (S. 15-17, 377-378). Hamburg: BoD - Books on Demand. Abgerufen am 16. Juli 2017 von <https://books.google.ch/books?id=fs0mDwAAQBAJ>
- Stringfellow, A. (03. Mai 2017). *What are CRUD Operations? Examples, Tutorials & More*. Abgerufen am 09. Juli 2017 von <https://stackify.com/what-are-crud-operations/>

Thin-Client. (09. November 2013). (ITWissen.info, Produzent) Abgerufen am 16. Juli 2017 von <http://www.itwissen.info/Thin-Client-thin-client-TC.html>

TIOBE Software BV. (10. Juli 2017). *TIOBE Index*. Abgerufen am 11. Juli 2017 von Tiobe.com: <https://www.tiobe.com/tiobe-index/>

Tribelhorn, T. (04. Mai 2017). Erstes Meeting: Kundenauftrag spezifizieren. (G. Zurbriggen, & R. Schumann, Interviewer)

Welk, K. (2009). In *Techniken für mobile Datenbanken in Informationssystemen* (S. 35-37). Hamburg: Diplomica Verlag GmbH. Abgerufen am 10. Juli 2017 von <https://books.google.ch/books?id=YcdoAQAAQBAJ>

What is Entity Framework? (kein Datum). Abgerufen am 19. Juli 2017 von <http://www.entityframeworktutorial.net/what-is-entityframework.aspx>

Wübbenhorst, K. (kein Datum). *Likert-Skalierung*. (S. G. Verlag, Herausgeber) Abgerufen am 10. Juli 2017 von Gabler Wirtschaftslexikon: <http://wirtschaftslexikon.gabler.de/Archiv/12660/likert-skalierung-v6.html>

Anhang I: SEPAT Manual

Die folgenden Informationen wurden von Herrn Thomas Tribelhorn bereitgestellt, damit der Auftrag am Anfang der Bachelorarbeit besser verstanden wurde.

SEPAT

Was ist SEPAT?

SEPAT ist ein Online-Service, der Sie bei der Situationsanalyse unterstützt. Mit SEPAT können Sie auf flexible Art Online-Fragebogen und –Checklisten erstellen, die anschliessend durch Ihre Klientel oder durch Sie selbst ausgefüllt werden, je nach Situation.

Wie kann man SEPAT einsetzen?

Selbstanalyse als Online-Service

Stellen Sie einen Online-Selbsttest auf Ihrer Webseite zur Verfügung, der durch Ihre Zielgruppe ausgefüllt wird. Dazu erfassen Sie vorgängig einen Online-Fragebogen mit Hilfe entsprechender Analysefragen. Sie können der Klientel direktes Online-Feedback zur Verfügung stellen, nachdem der Onlinetest ausgefüllt wurde, inklusive einer grafischen Darstellung der Ergebnisse. Diese können durch die Klientel auch als PDF exportiert werden.

Beispiel: ...

Situationsanalyse in einer Beratung

SEPAT dient Ihnen auch zur strukturierten Bestandsaufnahme in Beratungssituationen. Erfassen Sie dazu vorgängig eine Checkliste in SEPAT mit zielführenden Analysefragen. Im Beratungsinterview werden Sie durch die Checkliste geführt und erfassen die Antworten Ihrer Klientel direkt online. Am Schluss können Sie die grafische Auswertung direkt mit Ihrer Klientel besprechen und weitere Massnahmen bestimmen. Der PDF-Report lässt sich ebenfalls ausdrucken. Checklisten erscheinen nicht online auf einer Webseiten, sondern in Ihrem persönlichen Cockpit.

Beispiel: ...

Wer kann SEPAT nutzen?

Registrierte Benutzerinnen und Benutzer können in SEPAT Fragebogen und Checklisten generieren und zur Verfügung stellen. Je nach Berechtigungsstufe können Sie bestehende Messinstrumente (Fragebonge, Checklisten) einsetzen oder auch selber generieren. SEPAT bietet drei Berechtigungsstufen.

User

Als User wird in SEPAT die Klientel bezeichnet, die z.B. Onlinefragebogen ausfüllt oder direkte Beratung erfährt.

Master

Als Master werden Personen bezeichnet, die Zugriff auf bestehende Fragebogen oder Checklisten haben, diese nutzen und ihrer Klientel online zur Verfügung stellen können oder sie in Beratungssituationen selber ausfüllen, als Hilfsmittel für ein strukturiertes Analyseinterview.

Site Administrator

Site Administratoren verwalten grössere Bereiche, in denen Gruppen von Messinstrumenten (Fragebogen, Checklisten) zur Verfügung gestellt werden. Eine Site kann z.B. einer Arbeitsgruppe, einer Abteilung oder einem Unternehmen zur Verfügung gestellt werden. Diese verwalten in ihrer Site die eigenen Messinstrumente.

Elemente und Terminologie

Fragebogen

Ein Online-Fragebogen, der durch Nutzerinnen und Nutzer, d.h. durch eine bestimmte Klientel ausgefüllt wird. Nach Abschluss wird eine unmittelbare Rückmeldung angezeigt, die auch als PDF-Report ausgedruckt werden kann.

Checkliste

Ein Online-Fragebogen, der nicht auf einer Webseite, sondern nur im persönlichen Benutzerkonto von Nutzern mit entsprechenden Rechten (Master, Site Administrator) erscheint. Checklisten werden zur Durchführung strukturierter Interviews mit einer Klientel benutzt. Beratungspersonen (mit Master-Berechtigungen) erfassen die Antworten auf die Fragen einer ausgewählten Checkliste selber. Die Gesamtauswertung dient als übersichtliche Situationsanalyse zur Bestimmung weiterer Massnahmen mit der Klientel.

Messinstrument

Überbegriff für Fragebogen und Checklisten.

System Administrator

Verwaltet den Service und den Server. Erstellt Sites (grössere Bereiche) für Fakultäten, Institute, Unternehmen etc., die ihrerseits eigene Unterbereiche (Homepages) eröffnen und verwalten können.

Site Administrator

Verwaltet den operativen Teil für die Nutzerinnen und Nutzern. Erstellt Unterbereiche (Homepages) für Gruppierungen zur Verwaltung der eigenen Messinstrumente.

Master

Verwendet bestehende Messinstrumente (Fragebogen, Checklisten), passt diese an und stellt sie der Klientel (User) zur Verfügung oder nutzt sie selber zur Analyse. Mit erweiterten Rechten können Master selber Messinstrumente generieren und für andere freigeben.

User

Klientel, die entweder Onlinefragebogen ausfüllt, um direktes Onlinefeedback zu erhalten oder Personen, die von Mastern unter Nutzung von Checklisten beraten werden.

Benutzerverwaltung

Durch System oder Site Administratoren sowie durch Master können Usern persönliche Benutzerkonten auf SEPAT eröffnet werden. User können sich auch selber auf SEPAT registrieren.

Cockpit

Landingpage für das persönliche Benutzerkonto. Zusätzlich werden alle Messinstrumente angezeigt, auf die Benutzerinnen und Benutzer Zugriffsrechte haben.

Site

Bereich für eine Fakultät, ein Institut, ein Unternehmen etc. Eine Site ist eine eigene SEPAT-Instanz für die entsprechenden Nutzerinnen und Nutzer mit vollen Verwaltungsrechten durch Site Administratoren.

Homepage

Unterbereiche einer Site. Site Administratoren können beispielsweise für firmeninterne Abteilungen, Fakultäten für Institute oder Projektgruppen separate Homepages erstellen, auf denen die zugehörigen Nutzerinnen und Nutzer ihre eigenen Messinstrumente verwalten können.

Fragenpool

Im Fragenpool erfassen Nutzer mit entsprechenden Rechten ihre Fragen, die anschliessend zu Skalen gebündelt werden. Seriöse Messinstrumente setzen sich aus wissenschaftlich fundierten Erhebungsinstrumenten zusammen wie psychometrische Skalen (z.B. Selbstwirksamkeitserwartung, Handlung-Lageorientierung, Freiburger Persönlichkeitsinventar usw.). Forschungsteams können eigene Fragenpools zu Forschungszwecken generieren.

Skala

Mehrere thematisch zusammengehörige Fragen werden zu Skalen gebündelt. Die Antworten auf alle Fragen einer Skala werden miteinander verrechnet. Im Feedback wird pro Skala eine Ausprägung angezeigt, die auf dem Mittelwert aller Antworten auf die Fragen einer Skala beruht.

Stammfragen

Mit Stammfragen werden persönliche Angaben der User erfasst. Neben biografischen Merkmalen können damit beispielsweise unabhängige Variablen erfasst werden, die später mit den Ergebnissen der Analysefragen verglichen bzw. verrechnet werden.

Analysefragen

Analysefragen dienen zur Erfassung der User-Antworten, welche die abhängigen Variablen betreffen, beispielsweise in Forschungsprojekten. Wenn ein Fragebogen als Online-Service zur Verfügung gestellt wird, so schätzen die User sich anhand der Analysefragen selber ein. Die entsprechenden Antworten bilden die Basis für das direkte Online-Feedback.

Block

Als Block wird die gebündelte Darstellung einer Skala oder eine Fragengruppe innerhalb des Messinstrumentes bezeichnet.

Analysefragenblock

Fragenblock mit Analysefragen.

Stammfragenblock

Block mit Stammfragen

Antwortformat

Für die Fragen in SEPAT stehen verschiedene Antwortformate zur Verfügung. Matrix, Single-Choice, Multiple-Choice, Drop-Down und Textfeld:

- Matrixfragen: Mehrere Fragen, die nur eine Antwortauswahl ermöglichen, können als kompakte Tabelle dargestellt werden. Die Links-Rechts-Ausprägung und die Abstufung der Antworten ist bei allen Fragen dieselbe.
- Single-Choice Fragen: Frage, die die Wahl nur einer Antwort aus den Antwortmöglichkeiten zulässt.
- Multiple-Choice Fragen: Mehrere Antworten der Auswahl können angewählt werden.
- Drop-Down Fragen. Vergleichbar mit Single-Choice, die Antwort wird jedoch aus einem Ausklapp-Menü gewählt.
- Textfeld-Fragen: Fragen mit offenem Antwortfeld, das die Eingabe alphanumerischer Werte zulässt.

Antwortrichtung

In der Regel werden Fragen positiv formuliert, so dass User mit einer Antwort ganz rechts die volle Zustimmung zur Frage geben. Um User zu identifizieren, die "gedankenlos" das Messinstrument

"durchklicken", können Fragen gestellt werden, die bei einer positiven Beantwortung eigentlich auf der anderen Seite des Pols angewählt werden müssen. Inhaltlich wird die Frage also quasi "umgekehrt" gestellt. Für die Antwortauswahl muss die Links-Rechts-Polarität also mit den Voreinstellungen gedreht werden für die Verrechnung. Die Darstellung im Messinstrument kann aber konsistent mit den restlichen Fragen gehalten werden.

Feedback

Als Feedback wird die direkte Rückmeldung an den/die User/in bezeichnet. Ein grafisches Feedback wird in SEPAT automatisch erstellt, verbales Feedback muss von Administratoren oder Masterern vorgängig erfasst werden.

Segment

100% der Summe einer Skala (Maximalwert aller Antworten der zugeordnete Fragen) können mehrere Segmente unterteilt werden (maximal zehn). Für jedes Segment kann vorgängig ein separater Feedbacktext erfasst werden. Beispiel: Das Feedback für eine Skala wird in vier gleich grosse Segmente unterteilt (0% – 25% / 26% - 50% / 51% - 75% / 76% - 100). Wenn eine Person auf einer Skala nun einen Gesamtpunktwert von unter 25% erreicht, wird ein anderer Feedbacktext angezeigt, als bei Punktwerten in anderen Sektoren.

Einzelanalyse

Die Segmentierung ist die wesentliche Möglichkeit, um den Usern ein direktes Online-Feedback zu geben, das aber dennoch differenziert ausgestaltet werden kann. Jeder User erhält eine Einzelanalyse aufgrund der eigenen Antworten. In Beratungssituationen können Master die Einzelanalyse als Grundlage für die weitere Beratung nutzen.

Gruppen

Mehrere User können zu Gruppen zusammengefasst werden. Die einzelnen Userinträge werden damit zu einem Mittelwert verrechnet.

Vergleich

Die grafischen Darstellungen von Userinträgen (einzeln oder Gruppen) können übereinandergelegt werden. So kann beispielsweise die Selbsteinschätzung einer Einzelperson mit dem Mittelwert der Fremdeinschätzung einer Gruppe von Usern direkt verglichen werden (z.B. im Unterricht), oder es können die Mittelwerte der Einschätzungen unterschiedlicher Gruppen visuell miteinander verglichen werden. Gruppen werden erstellt, oder Nutzer mit höheren Rechten) einen Gruppencode generieren in SEPA. Alle Userinträge via Gruppencode werden zu einem Mittelwert berechnet und entsprechend visualisiert.

Report

Das Online-Feedback, sowohl grafisch als textlich, kann von den Nutzerinnen und Nutzern in eine PDF-Datei exportiert und ausgedruckt werden.

Anhang II: Entwicklungshinweise

Die folgenden Informationen wurden von Herrn Thomas Tribelhorn bereitgestellt, damit der Auftrag am Anfang der Bachelorarbeit besser verstanden wurde.

Entwicklungshinweise

Freie Kommentare, Tipps und Hinweise zur Entwicklung

Nutzung

Durch die Gesamtarchitektur kann SEPAT als Service für andere Institutionen angeboten werden.

Beispiele

Beispiele für Services, die von der Bedienbarkeit und dem Erscheinungsbild als Vorbilder dienen könnten

[Socrative.com](https://www.socrative.com/)

[H5P.org](https://www.h5p.org/)

Anhang III: Rechtenmatrix

Die folgenden Informationen wurden von Herrn Thomas Tribelhorn bereitgestellt, damit der Auftrag am Anfang der Bachelorarbeit besser verstanden wurde.

Benutzerverwaltung: Hier werden alle Benutzer/innen zugewiesen, editiert oder gelöscht durch Admin mit entsprechenden Rechten.	USER	MASTER	SITE ADMIN	SYS ADMIN
Benutzer erstellen und löschen			1	1
Rechte zuweisen			1	1
Benutzer sperren			1	1
Benutzerdaten importieren (csv)			1	1
Benutzerdaten editieren (Benutzername und Passwort) bei direkt registrierten			1	1
Benutzerlisten exportieren (csv)				
[+ alle untergeordneten Rechte]			1	1

Cockpit: Zentraler Zugriffsort zur Verwaltung des eigenen Profils. Hier erscheinen alle zugewiesenen Elemente mit entsprechenden Berechtigungen	USER	MASTER	SITE ADMIN	SYS ADMIN
Eigene Benutzerdaten ändern	1	1	1	1
Eigenes Passwort ändern	1	1	1	1
Eigenes Profilbild hochladen, ersetzen	1	1	1	1
Zugriff auf eigene Fragenpools, Fragen, Fragebogen, Checklisten, Reports (gemäss zugewiesener Rechte)	1	1	1	1
[+ alle untergeordneten Rechte]	1	1	1	1

Site: Umgebung für Fakultäten, Departemente, Unternehmen etc. Hub für den Zugriff auf bereichsspezifische Homepages mit Fragebogen	USER	MASTER	SITE ADMIN	SYS ADMIN
Server verwalten				1
Site eröffnen (Umgebung für Abteilung, Departement etc.)			1	1
Site löschen			1	1
Site editieren (Titel und Beschreibung)			1	1
Site-Rechte zuweisen und entfernen			1	1
[+ alle untergeordneten Rechte]			1	1

Homepage: Zugriffsort für alle Elemente eines Bereiches (Fragebogen und Checklisten, Fragenpools und Fragen).Sichtbar mit entsprechenden Rechten.	USER	MASTER	SITE ADMIN	SYS ADMIN
Homepage Templates erstellen, kopieren, editieren, löschen			1	1
Homepage eröffnen/kopieren und löschen			1	1
Homepage editieren (CMS)			1	1
Fragenpool erstellen und zuweisen			1	1
Fragenpool kopieren und zuweisen			1	1
Fragenpool löschen			1	1
Homepage-Rechte zuweisen und entfernen			1	1
[+ alle untergeordneten Rechte]			1	1

Fragenpool: Sammlung aller Fragen für einen Fragebogen/eine Checkliste eines Bereiches (Arbeitsgruppe, Projekt, Institut etc.)	USER	MASTER	SITE ADMIN	SYS ADMIN
Fragenpool eröffnen und löschen				
Fragen direkt im erstellten Pool generieren und löschen				
Fragenpool (Kopie) für andere Site freigeben				
Rechte zum Editieren für Fragenpool zuweisen				
Fragenpool editieren: Titel, Beschreibung, (Besitzer/in ist mit Login-ID angegeben)				
[+ alle untergeordneten Rechte]			1	1

Fragen: Analysefragen und Stammfragen mit unterschiedlichen Antworttypen (Single-Choice oder Drop-Down, Multiple-Choice, Matrix, Textfeld)	USER	MASTER	SITE ADMIN	SYS ADMIN
Analysefrage erstellen (Titel, Kürzel, Beschreibung, Antworttyp, Antwortrichtung)				
Antworttyp einer Analysefrage bestimmen				
Antwortauswahl erfassen pro Analysefrage				
Stammfragen erstellen: Titel, Kürzel, Beschreibung				
Antworttyp einer Stammfrage bestimmen				
Antwortauswahl erfassen pro Stammfrage				
Fragen editieren ('erstellt'/'zuletzt geändert von' ist mit Login-ID angegeben)				
[+ alle untergeordneten Rechte]				

Stammdatenblock: Block mit Fragen zur Erhebung User-spezifischer Merkmale.	USER	MASTER	SITE ADMIN	SYS ADMIN
Stammfragenblock erstellen: Titel, Kürzel, Beschreibung		1	1	1
Stammfragen einem Stammfragenblock zuweisen und daraus entfernen		1	1	1
Stammfragenblock einem Fragebogen/einer Checkliste zuweisen		1	1	1
Reihenfolge der Stammfragen im Stammfragenblock manuell sortieren mit drag-n-drop		1	1	1
[+ alle untergeordneten Rechte]			1	1

Skala: Gruppierung mehrerer inhaltlich zusammenhängender Fragen, deren Antwortsummen total den Skalenwert ergeben (als Basis für das Feedback)	USER	MASTER	SITE ADMIN	SYS ADMIN
Skala eröffnen und löschen		1	1	1
Skala editieren (Titel, Kürzel, Beschreibung)		1	1	1
Analysefragen einer Skala zuordnen und entfernen (gemäss Berechtigungen)		1	1	1
Skala einem Fragebogen zuordnen (gemäss Berechtigungen)		1	1	1
Rating-Stufen einer Skala bestimmen (vier-, fünf-, sechs-, zehnstufig etc.)		1	1	1
[+ alle untergeordneten Rechte]			1	1

Feedbacks: Rückmeldungen zu den berechneten Userantworten pro Skala. Pro Skala ist eine Skalierung (Skalenaufteilung) möglich	USER	MASTER	SITE ADMIN	SYS ADMIN
Feedback für eine Skala eröffnen und löschen		1	1	1
Feedback editieren (Titel, Beschreibung)		1	1	1
Anzahl Feedbackabschnitte (Segmente) bestimmen		1	1	1
Ausprägung für jeden Feedbackabschnitt (Segment) mit Regler bestimmen (Prozentbereich)		1	1	1
Feedbacktexte für die Feedbackabschnitte (Segmente) erfassen, editieren, löschen		1	1	1
[+ alle untergeordneten Rechte]			1	1

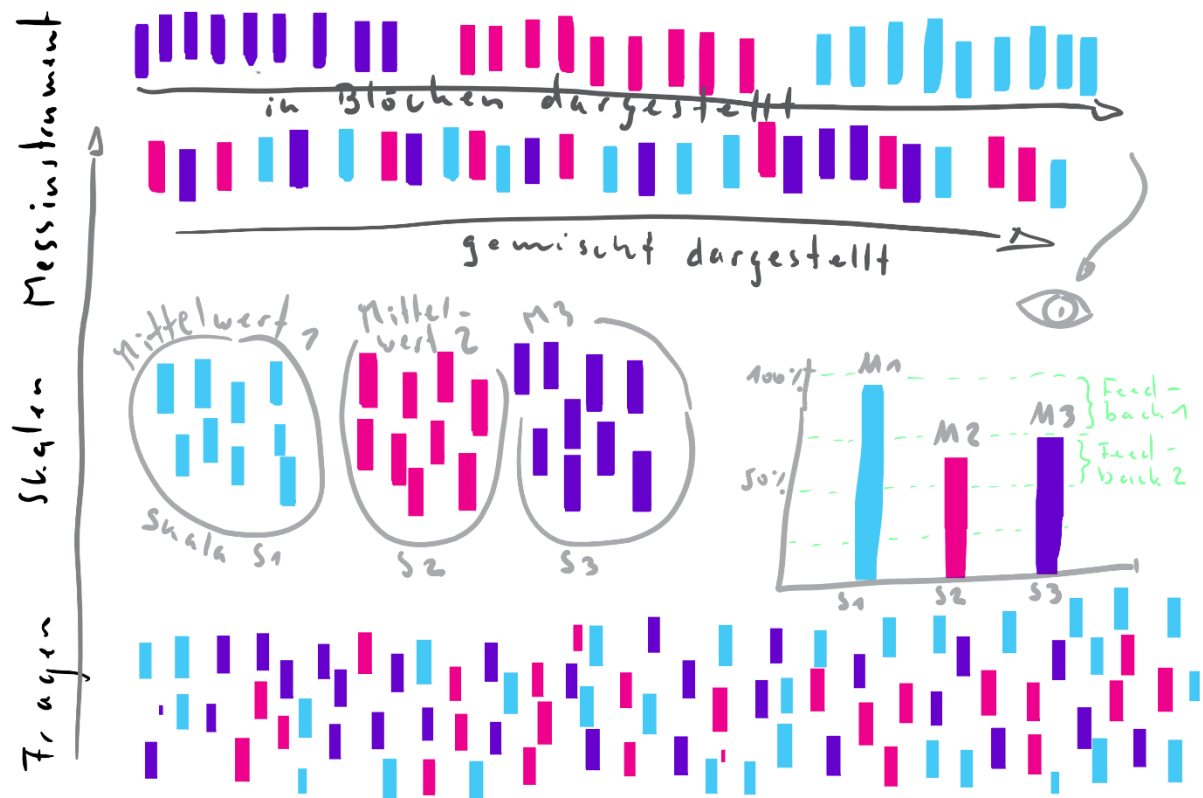
Fragebogen: Gruppierung von Skalen/Fragenblöcken. Voreinstellung <u>pro Messinstrument</u> . Checklisten erscheinen nur im Master-Cockpit (nicht Frontend)	USER	MASTER	SITE ADMIN	SYS ADMIN
Fragebogen/Checkliste generieren und löschen		1	1	1
Fragebogen/Checkliste online / offline stellen		1	1	1
Kopie eines Fragebogens erstellen (gemäss Berechtigungen)		1	1	1
Fragebogen editieren: Titel, Beschreibung, Einleitungstext, Abschlusstext		1	1	1
Fragen im Fragebogen direkt generieren		1	1	1
Fragen aus Fragenpool zuweisen (gemäss Berechtigungen)		1	1	1
Skalen einem Fragebogen als Frageblock zuweisen		1	1	1
Bestehenden Fragebogen nutzen/freigeben für User		1	1	1
Darstellung im Fragebogen / in der Checkliste				
Skalen als Frageblöcke darstellen (default = aktiviert; deaktiviert=alle Fragen in einer Liste ohne Zwischentitel und Unterteilung)		1	1	1
Zufällige Reihenfolge der Fragen im Fragebogen aktivieren, nur bei Darstellung ohne Blöcke möglich (default=deaktiviert)		1	1	1
Reihenfolge der Fragen im Fragenblock manuell sortieren mit drag-n-drop		1	1	1
Zufällige Reihenfolge der Fragen im Fragenblock aktivieren pro User (default = deaktiviert)		1	1	1
Zufällige Reihenfolge der Frageblöcke im Fragebogen aktivieren pro User (default = deaktiviert)		1	1	1
Reihenfolge der Frageblöcke im Fragebogen manuell sortieren mit drag-n-drop		1	1	1
Darstellung der Frageblöcke im Fragebogen bestimmen (pro Seite ein Block / alle auf einer Seite = default)		1	1	1
Titel der Frageblöcke für User anzeigen (aktivieren /deaktivieren)		1	1	1
Beschreibung der Frageblöcke für User anzeigen (aktivieren /deaktivieren)		1	1	1
Kürzel der Frageblöcke für User anzeigen (aktivieren /deaktivieren)		1	1	1
Darstellung in der Auswertung / im PDF-Report				
Feedback-Übersichtstafel anzeigen (aktivieren/deaktivieren)		1	1	1
Titel der Frageblöcke im Feedback anzeigen (aktivieren/deaktivieren)		1	1	1
Beschreibung der Frageblöcke im Feedback anzeigen (aktivieren/deaktivieren)		1	1	1
Einzelfeedback-Texte anzeigen (aktivieren/deaktivieren)		1	1	1
Einzelfeedback-Grafiken anzeigen (aktivieren/deaktivieren)		1	1	1
Fuss- und Kopfzeile des Reports editieren		1	1	1
Logo für Report hochladen (Pixelbeschränkung, Prüfen der Auflösung, Grösse wird automatisch angepasst)		1	1	1

Gruppen: Bündelung ausgewählter User, deren Input zu einem Gesamtwert verrechnet und dargestellt wird (nur grafisches Feedback).	USER	MASTER	SITE ADMIN	SYS ADMIN
Gruppencode generieren: Alle User-Einträge via Gruppencode werden zu einem Durchschnittswert pro Antwort verrechnet		1	1	1
Manuelle Gruppe erstellen: Registrierte User manuell einer Gruppe zuweisen und entfernen		1	1	1
Manuelle Gruppe editieren: Bezeichnung, Gruppenbeschreibung		1	1	1
Gruppen einem Fragebogen zuweisen und daraus entfernen		1	1	1
[+ alle untergeordneten Rechte]		1	1	1

Ergebnisse: Darstellung und Verarbeitung der berechneten User-Resultate.	USER	MASTER	SITE ADMIN	SYS ADMIN
Übersichtstabelle mit den Ergebnissen pro Fragebogen/Checkliste ansehen: Zeitstempel, Dauer, ...		1	1	1
Ergebnisse einzelner User ansehen (per Klick auf Usernamen/ID in der Übersicht)		1	1	1
Total der Ergebnisse als Exzeldatei exportieren		1	1	1
Gruppendarstellung der Ergebnisse ein-/ausblenden (alle oder einzelne)		1	1	1
Eigene Ergebnisse ansehen	1	1	1	1
Statistische Varianz zu den Ergebnissen (Mittelwerte) ein-/ausblenden (Boxplot bei Balken, Schattenbereich bei Spinne und Links-Rechts-Profil)	1	1	1	1
Eigene Ergebnisse als Master zusammen mit Gruppenergebnissen anzeigen (anzeigen/ausblenden=default)		1	1	1
Grafischen Darstellung der eigenen Ergebnisse wechseln zwischen Balkendiagramm, Spinnendiagramm, Links-Rechts-Profil	1	1	1	1
Eigene Ergebnisübersicht als PDF exportieren	1	1	1	1
[+ alle untergeordneten Rechte]			1	1

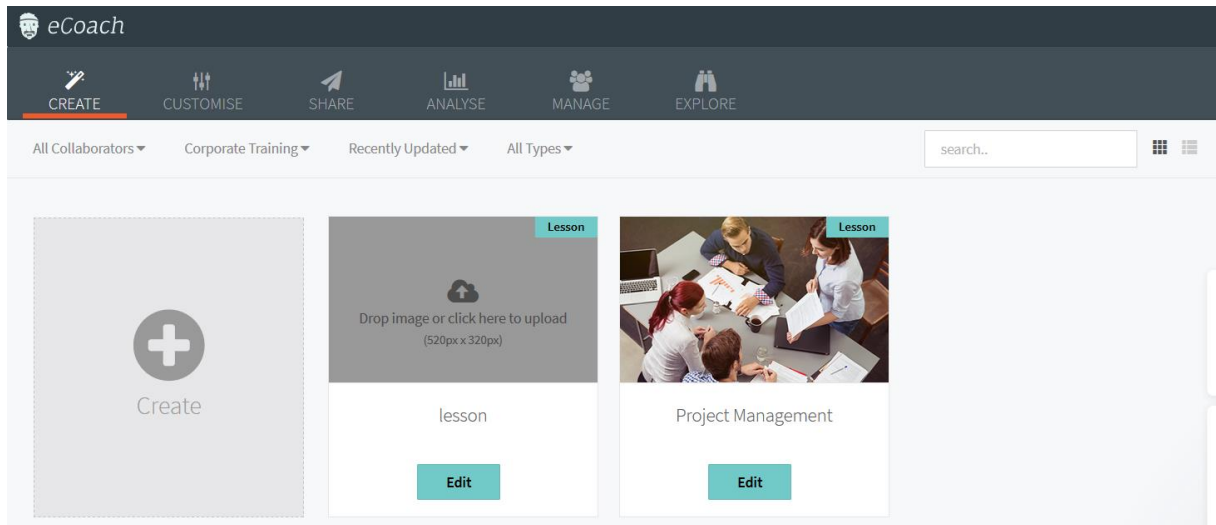
Anhang IV: Skalenlogik – Skizze

Die folgenden Informationen wurden von Herrn Thomas Tribelhorn bereitgestellt, damit der Auftrag am Anfang der Bachelorarbeit besser verstanden wurde.

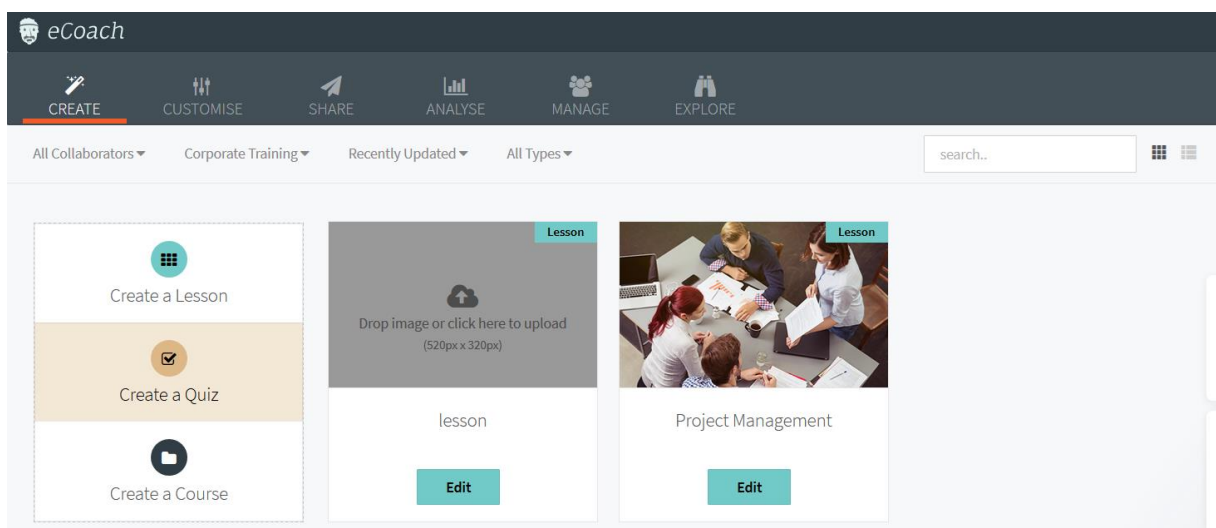


Anhang V: Erstellung eines Quiz/Selbstevaluation Projektes in eCoach

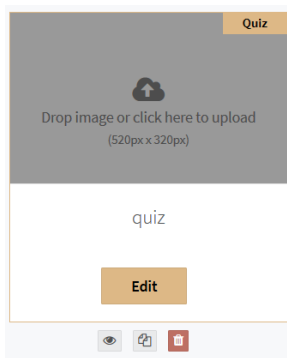
1. Startseite beim Login:



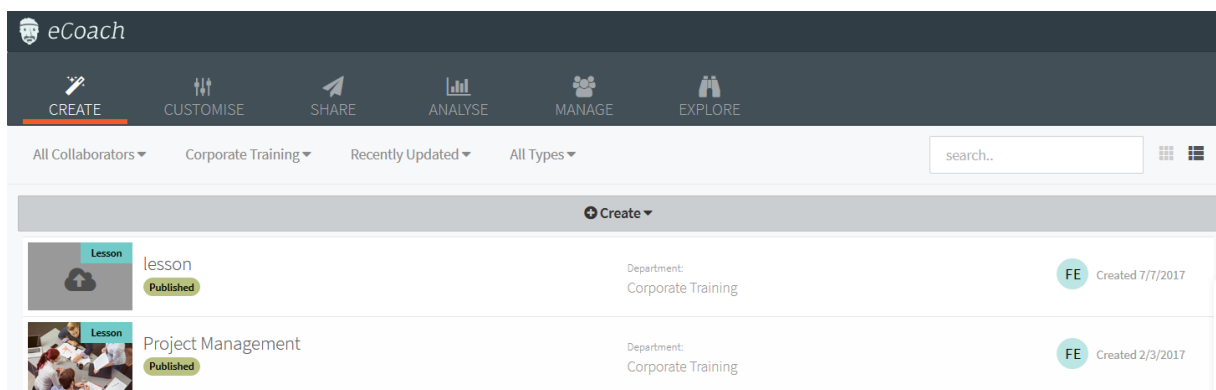
2. Neues Quiz erstellen



3. Quiz mit Schnellzugriffen



4. Änderung der Ansichtsoption



Anhang VI: Erstellung eines Selbstevaluation Projektes in Selevo

1. Startseite beim Login


The screenshot shows the ILIAS University of Bern homepage. At the top, there is a black header with the text "ILIAS Universität Bern". Below this is a navigation bar with the logo "u^b" and several menu items: "Persönlicher Schreibtisch", "Magazin", "Support", and "Portale". The main content area has a light blue header with the TUW logo and a navigation bar with "Inhalt", "Info", "Einstellungen", "Mitglieder", "Lernfortschritt", "Metadaten", "Export", "Rechte", and "Voransicht als Mitglied aktivieren". Below this is a sub-navigation bar with "Zeigen", "Verwalten", "Sortierung", and "Seite gestalten". A red button "Neues Objekt hinzufügen" is located on the right. The main content area is titled "INHALT" and lists three items: "BWL Keep Up To Date Kurs", "Evening Lecture Live Voting", and "Metakognitive Strategien". Each item has a small icon and a dropdown arrow on the right.

2. Neues Objekt hinzufügen

The screenshot shows the ILIAS University of Bern homepage with the "Neues Objekt hinzufügen" menu open. The menu is divided into four columns: "Inhalt", "Organisation", "Kommunikation & Kollaboration", and "Leistungen". The "Inhalt" column lists: Datei, DigiSem, Weblink, Glossar, Mediacast, SWITCHcast Channel, Medienpool, Lernmodul ILIAS, Lernmodul HTML, Lernmodul SCORM, Webfeed, and Datensammlung. The "Organisation" column lists: Ordner, Sitzung, Objektblock, Forum, Blog, Wiki, Etherpad, and Gruppe. The "Kommunikation & Kollaboration" column is empty. The "Leistungen" column lists: Übung, Test, Fragenpool für Tests, Portfoliovorlage, Feedback & Evaluation, Abstimmung, LiveVoting, Umfrage, Fragenpool für Umfragen, and Selbstevaluation. The "Selbstevaluation" option is highlighted in blue.

3. Schnellzugriffe einer Selbstevaluation


ILIAS Universität Bern





Persönlicher Schreibtisch ▾Magazin ▾Support ▾Portale ▾

Neues Objekt hinzufügen ▾

INHALT


BWL Keep Up To Date Kurs

Evening Lecture Live Voting



Metakognitive Strategien

Testumgebung mit Mockup-Test **Wie lernen Sie?** Selbsttest zur Analyse von Lern- und Arbeitsstrategien.

nutzungsablauf

Status: Offline

Bearbeiten

Info

Löschen

Verknüpfen

Verschieben

Auf Schreibtisch legen

Kommentare

Notizen

115

Anhang VII: Exposé Zeitplanung

Die folgende Zeitplanung wurde während der ersten Woche der Bachelorarbeit erstellt. Diese Liste diente zur Grobeinschätzung der zur Verfügung stehenden Zeit.

KW	von	bis	Aufgaben	Report: Deadlines
19	08.05.2017	14.05.2017	- Exposé beenden - Tasks Besprechung + Priorisierung - Datenbank modellieren - Entwicklungsumgebung bereitstellen	
20	15.05.2017	21.05.2017	- Wahl des Bootstrap Design + Einbindung in Source Code - Entity Framework aufbereiten + Datenbank erstellen - Accountmanagement (Login, Registration, Passwort vergessen, Zugangscode)	
21	22.05.2017	28.05.2017	- UI Cockpit eines Kunden, Masters, Site Administrator, System Administrator - Stammdaten + Fragenbogen Management	
22	29.05.2017	04.06.2017	- Antworten Management - UI Antwortformate	
23	05.06.2017	11.06.2017	- Skala/Segments Aufbereitung (für Auswertung) - Segments UI	
24	12.06.2017	18.06.2017	- UI Auswertungen	
25	19.06.2017	25.06.2017	- UI Auswertungen inkl. Gruppen - PDF Export (ohne unternehmensspezifisches Template) - Thesis	Kapitel 1 (ohne Einleitung Text)
26	26.06.2017	02.07.2017	- Testing & Bug fixes - Thesis	Kapitel 2
27	03.07.2017	09.07.2017	- Thesis	Kapitel 3 & 4
28	10.07.2017	16.07.2017	- Thesis	Kapitel 5 & 6 & Anhang
29	17.07.2017	23.07.2017	- Nacharbeit der Thesis - Implementieren / Software Testing	Einleitung & Schluss Abbildungsverzeichnis Formatierung
30	24.07.2017	30.07.2017	- Nacharbeit der Thesis - Plagiat Check durchführen - Implementieren / Software Testing	
31	31.07.2017	06.08.2017	- Aufbereitung des Codes für Abgabe - Abgabe Thesis (02.08)	

Anhang VIII: Mockups

Nachfolgend werden die erstellen Mockups aufgezeigt.

Baumstrukturverwaltung

My Profile

My Analysis

My Savings

Usermanagement

Anzahl Clients 20
Anzahl Masters 5

Benutzerliste
[sort v]
[show 20 v]

Email	Firstname	Lastname	Addressed	Role	Site	Group	...
~	~	~	~	~	~	~	edit + delete
~	~	~	~	~	~	~	edit + delete
~	~	~	~	~	~	~	edit + delete
~	~	~	~	~	~	~	edit + delete
~	~	~	~	~	~	~	edit + delete
~	~	~	~	~	~	~	edit + delete
~	~	~	~	~	~	~	edit + delete
~	~	~	~	~	~	~	edit + delete
~	~	~	~	~	~	~	edit + delete
~	~	~	~	~	~	~	edit + delete

No person has been added yet.

+ New User

Baumstrukturverwaltung INSERT

EPAIT

New Account

Select Account type:

Client ▼

Email: _____

Password: _____

Confirm Password: _____

Firstname: _____

Lastname: _____

Address: _____

Town: _____

Country: _____

>> Navigation to Insert

The image shows two hand-drawn sketches of web pages, likely for a user management system.

Top Page: Benutzeranmeldung (User Login)

- Header:** "Benutzeranmeldung" and "WS08/9" (with a small icon).
- Navigation:** "Navigation zu anderen U2-Seiten" (with a small icon).
- Main Content:**
 - "Neues Account" (with a small icon).
 - "Select a company" (with a small icon).
 - "Um die" (with a small icon).
 - "Select a site/department" (with a small icon).
 - Buttons: "Wirtschaft", "Industrie", "JUS", "Administration".
- Footer:** "Benutzeranmeldung Edit" (with a small icon).

Bottom Page: Benutzeranmeldung Edit (User Edit)

- Header:** "Benutzeranmeldung Edit" (with a small icon).
- Main Content:**
 - "Edit - Home Master" (with a small icon).
 - Form fields: "Email", "Password", "First name", "Last name", "Address", "Town", "Post code", "Country", "Company", "Site".
 - Buttons: "Cancel", "Save".
- Footer:** "Benutzeranmeldung Edit" (with a small icon).

Benutzerverwaltung Delete

Pop-Up

Delete - Haupt-Muster

?

Passwort

Cancel Delete

3

Benutzerverwaltung Export

Pop-Up

Export - Benutzerlist Site

Datenname

URL Path

Cancel Export

4

The image contains two hand-drawn sketches of mobile application screens, likely for a user profile or account management system.

Top Screen: Profile Settings

- Header:** "Profile Settings" with a back arrow icon on the left and a settings gear icon on the right.
- Left Sidebar:** A vertical list of menu items: "My Profile" (highlighted with a yellow background), "Profile Settings", "Account Settings", and "My Sites".
- Main Content Area:**
 - Fields for "First Name" and "Last Name" (each with a text input field).
 - Fields for "Address" and "Town" (each with a text input field).
 - Fields for "Postcode" and "Country" (each with a text input field).
 - A "Save" button at the bottom right.

Bottom Screen: Account Settings

- Header:** "Account Settings" with a back arrow icon on the left and a settings gear icon on the right.
- Left Sidebar:** A vertical list of menu items: "My Profile", "Profile Settings", "Account Settings" (highlighted with a yellow background), and "My Sites".
- Main Content Area:**
 - An "Upload your profile picture" section with an "Upload..." button and a circular placeholder for a profile picture.
 - Fields for "Email" (with a text input field).
 - Fields for "Password" and "Confirm Password" (each with a text input field).
 - A "Save" button at the bottom right.

Site Mgmt → Site Admin

SEPAT

My Site (New) (Current active Version: 1.0)

DELETE

My Profile

Name

My Account

Company

My Surveys

My Site

Description

Save

Site Mgmt / Fragepool erstellen

SEPAT

Question Pool Management

My Question Pool:

→ no question Pool

+

→ Nav. to 11

External Question Pools:

Pool of 17

Questions: 157

Creator: msh@bch

Pool of 17 Rep.

Questions: 43

Creator: skw@bch

Nav. to 13

Pool erstellen Montag

Create Question Pool

Title

Description

Save

Pop-Up

Pool of 17

Questions: 157

Creator: msh@bch

Edit

Delete

Pop-Up

Delete - Pool of 17

Assignment

Cancel

Save

Create Question Type

Title

Description

Question Types:

HR ID

BWL IX

XLIX

FBG IX

Re Bu IX

Cancel

Save

Question Management Dienstag

SEPAT

Pool of 17

New Question

All Questions

17

Question Type

Site Mgmt

Question Pool

Manage Question

Pop-Up

Edit Question

Assignment

Question

Type

Cancel

Save

Stam-freige Mgmt Mittwoch

SEPAT

My Standing Data Blocks

Standing DATA UPS

Survey → Question Block 1

Questions: 5

Edit

Delete

SD for BWL Survey

→ Question Block 2

Questions: 6

Edit

Delete

SD BWL 1 Semester

→ Question Block 3

Questions: 3

Edit

Delete

Nav. to 20

Personen

Edit Standing Data Block

Title

Assignment

Question Date

Description

Question Block

Block 1

Cancel

Save

Create Standing Data Block

Title

Assignment

Question Date

Description

Cancel

Save

Delete Standing Block

Assignment

Cancel

Save

Donnerstag

Stammdaten folgen Eistella, Bookmaker, Lurda

SEPAT =

Standing data of Standing Data UBS (UBS)

New Question

Existing Standing Data Questions (Sort) (Change)

ID Anonym Question Answer type

Question Value

Standing Data

Standing Data Value

change sequence of questions

Edit Standing Data abc

Question

Answer type

Single choice Multiple choice

Textfield Dropdown

Cancel Save

Delete Question abc

Question

Password

Cancel Save

Freitag

Fragebogen erstellen

SEPAT =

Overview Question Blocks NEW Nov 27

Title abc

Creator: gerd@upch

Date: 11.11.17

Standing Data

Fragebogen

Question

Edit Delete

Pop-Up

general window

Edit Question Block ABC

Title

Description

Intro Text

Conclusion Text

Edit Question?

Standing Data Analysis

Cancel Save

Choose a Scale:

Scale 1 Motivation

Scale 2 Personal best

Scale 3 Team build

Add Question? NEW

Existing Question:

ID Anonym Question Type

Edit Question

Edit Question

Edit of Scale

Fragebogen erstellen

SEPAT =

Question Block

Create New Question Block (Analysis)

Title

Description

Question Block

Overview

BB Wizard

Intro text

Conclusion text

NEXT

Create New Scale:

Title

Area

Question 1

Create Scale

Title

Area

Description

Play or Stop!

Cancel Save

Scale spezifizieren

SEPAT =

Select a Question Pool

Pool 1 X Pool 2 X Pool 3 X

Select a type of Question

Block X Intro X Final X

ID Area Question Type

No. Selected: 1

to 28

Back Create Feedback Save

Create Feedback

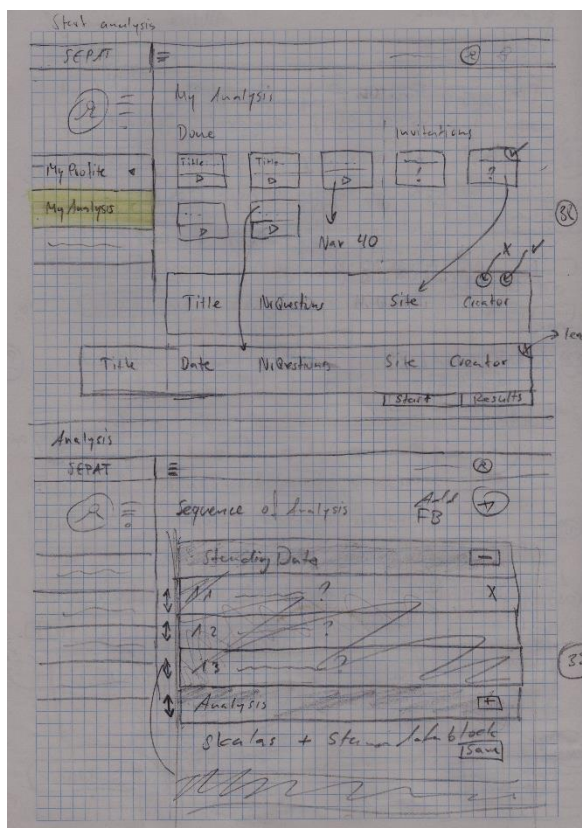
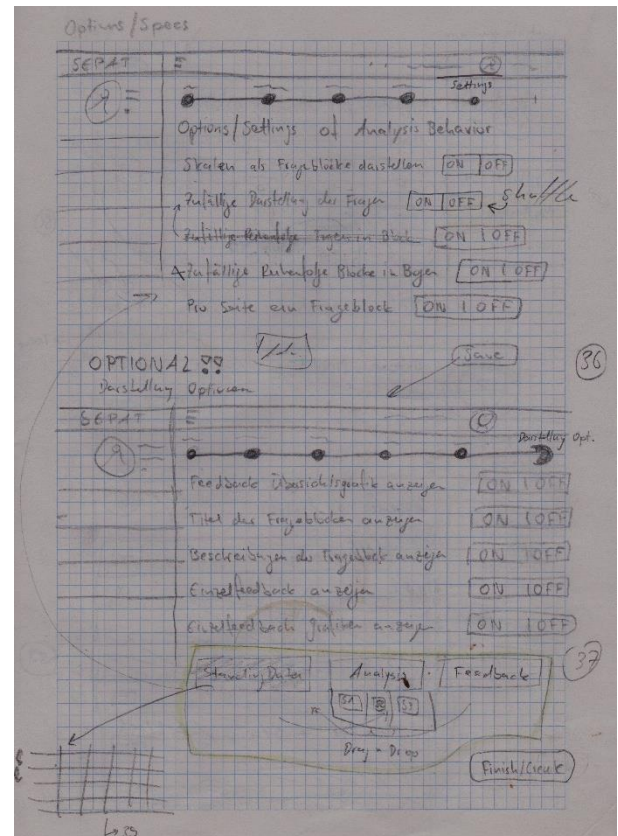
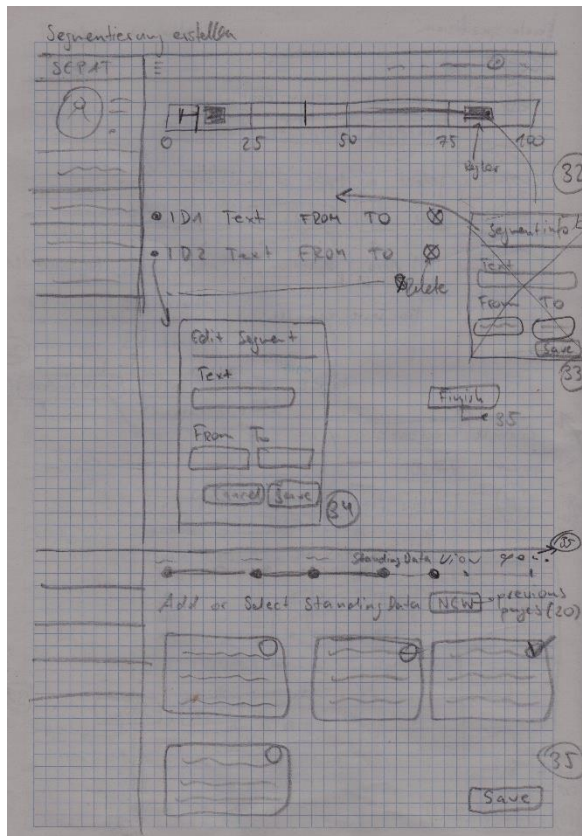
SEPAT =

New Feedback for Scale 1 x 2

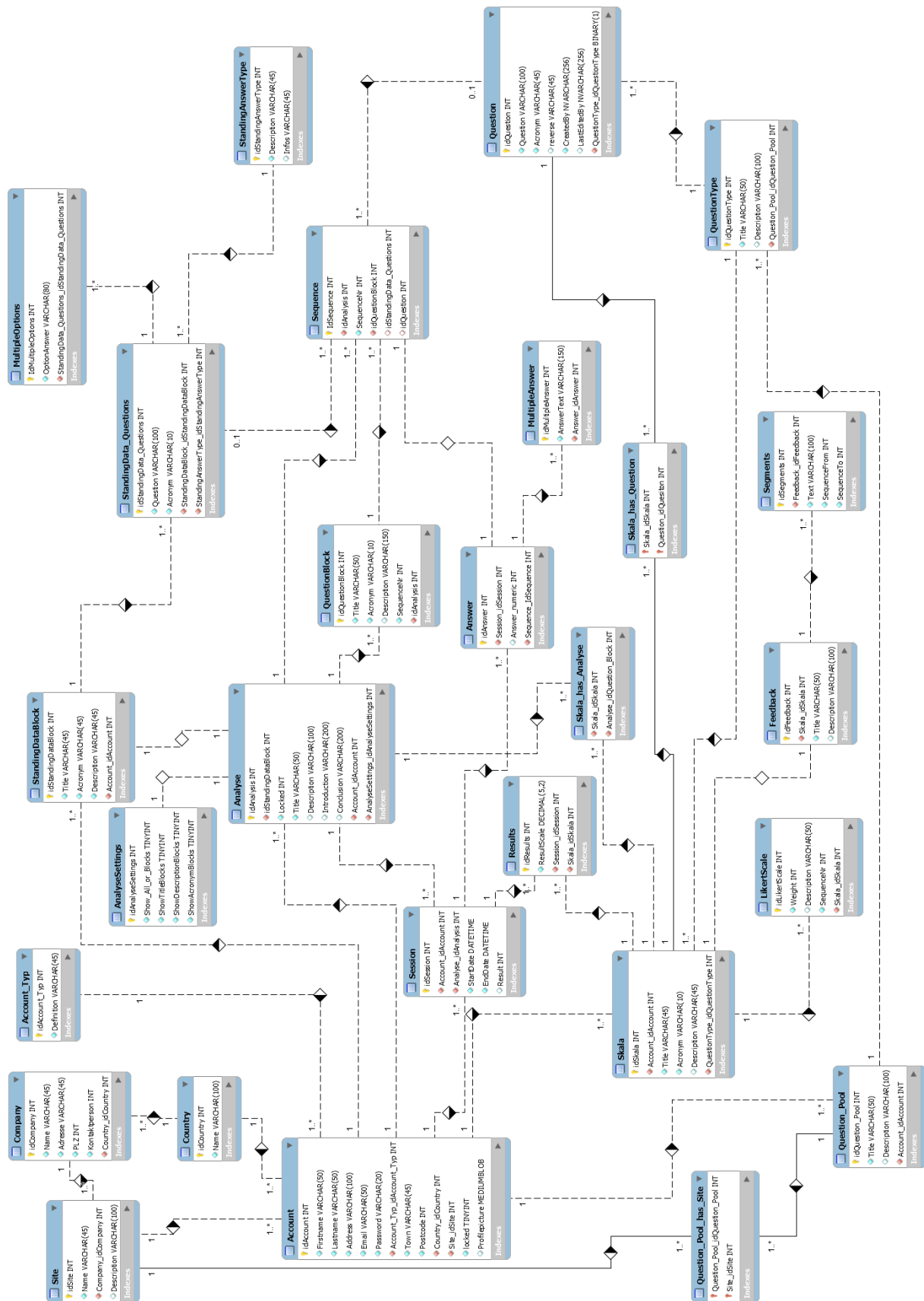
Title

Description

Cancel Save



Anhang IX: Datenbankmodell SEPAT



Anhang X: Implementationsaufwand

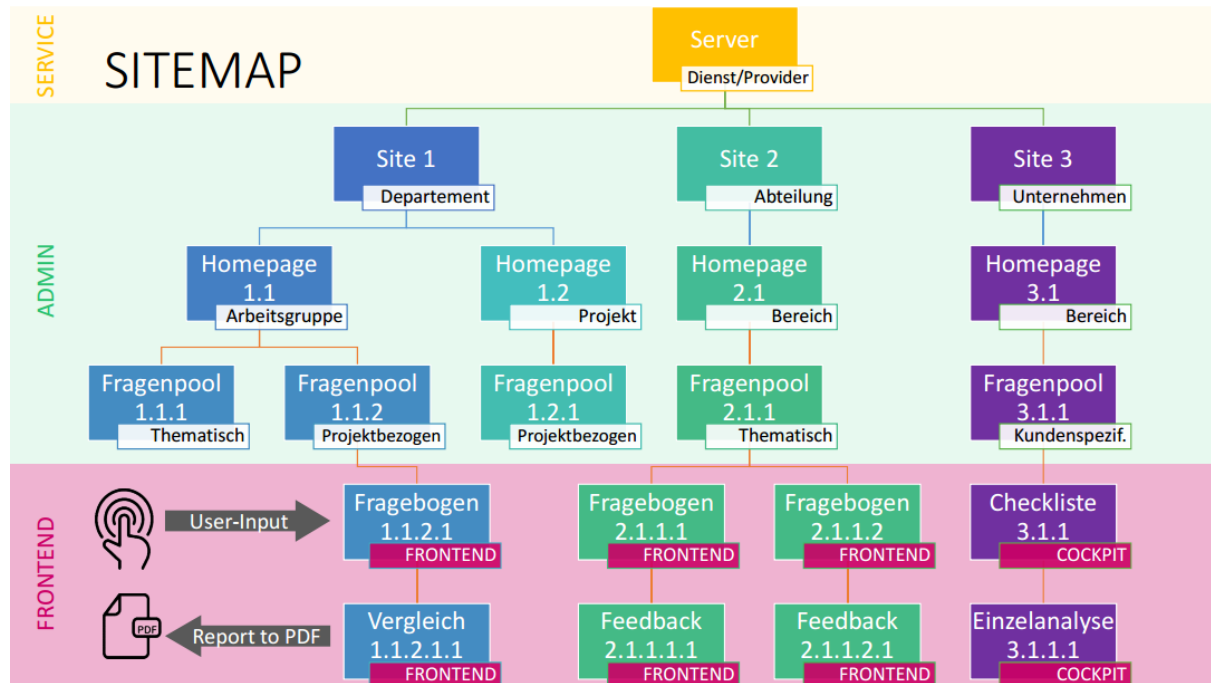
Die Liste des Implementationsaufwandes wurde erstellt, um Fehler oder Probleme zu notieren und zukünftig nach Möglichkeit zu beheben. Die Informationen in der Spalte «Notes» zeigen mögliche Probleme oder Fehlermeldungen des SEPAT Prototypen auf. Anhand dieser Liste wird für den Auftraggeber nach der Bachelorarbeit ein Dokument erstellt, welches die Fehler im Detail beschreibt. Dies ist bei einer Weiterführung des Projektes massgebend.

coding&research worktime SEPAT				total(h)	261.05
w	date	task	time spent	notes	
Code Week 1	19.05.2017	registration & login	11		
	20.05.2017	template integration	2		
	21.05.2017	google auth	2		
	22.05.2017	db modelling	1.5		
	22.05.2017	profile mgmt	4		
	22.05.2017	user mgmt	2		
	23.05.2017	user mgmt	2.5		
	24.05.2017	user mgmt	1.5		
	24.05.2017	design user mgmt	1.5		
	25.05.2017	all users mgmt	2		
	25.05.2017	table of users	2		
	25.05.2017	crud	2.5		
	26.05.2017	crud modals	4		
	27.05.2017	create update finish	5.5		
	27.05.2017	delete finish	1.2		
	27.05.2017	user list export	2.75		
	28.05.2017	export + specify list export done	1.2		
	28.05.2017	menu mgmt	0.5		
	28.05.2017	site mgmt(crud)	3.2	delete not finished, as admin handling and for users!!	
	28.05.2017	prof acc desing boxes	0.5		
Code Week 2	29.05.2017	prepareing for questionpool	5.25	share question pool in edit!	
	29.05.2017	question type in db	0.2		
	29.05.2017	question typ create delete	2		
	30.05.2017	delete question typ	0.75	navigation of modals and design optimation!!	
	30.05.2017	design & share pools to sites	4	pass value from view to controller	
	30.05.2017	question crud	2	if no pool nor typ no questions!!	
	31.05.2017	show question	3	new question typ as master!	
	31.05.2017	standing data blocks crud	2	assign to an analyse!!	
	01.06.2017	db modelling of standing data	0.5		
	01.06.2017	crud of standing data	5		
	02.06.2017	options of answer types	5		
	04.06.2017	planning next period	0.5		
				working with formcollection -> to optimize work with knockoutjs	
	04.06.2017	options of types save to db	2.5	edit standing data question is not done	
Code Week 3	05.06.2017	options	1		
	06.06.2017	save likerts	2		
	06.06.2017	questionnaire	3		
				select question from shared pools for the scale!	
	07.06.2017	scale crud	4	check names with autocompletion!!	
	08.06.2017	scale crud fini	2.25		
	08.06.2017	scale choose questions	3		
	09.06.2017	scale choose questions	7	redirect back to existing questions in scale!	
	10.06.2017	save scale questions	4		
		rename scalequestion classes(before wizard)			
	10.06.2017	and searching for timelines	1.5	TEST ALL FOR MASTERS!	
	10.06.2017	timeline partial view	1	timeline navigation!	
	10.06.2017	starting with wizard - create analysis	0.5		

Code Week 4	12.06.2017	analysis	0.2	
	12.06.2017	timeline navigation	1.2	
	13.06.2017	timeline navigation finish	0.2	
				scales handling for wizard and analysis scale controller -> not copy it, if no question you cant save scale!
	13.06.2017	scale	4	Scales can have multiple feedbacks(for specific analysis)
	13.06.2017	feedback	3	create without wizard! Will not be in the prototyp
	13.06.2017	segments	1.5	
	14.06.2017	segments slider	9.75	optimize it dynamically! Modify values from inputs! Add better constraints
	15.06.2017	segment save	2	handle a feedback status (e.g. feedback must have segments ->100%)
	15.06.2017	choose personal data and bug fixes	2	
				add to creation standing datablock the selection of an analysis! Question handling inside wizard of standing datablocks! Mark standing block which already have an analysis except current one
	15.06.2017	standing data selection - database modification - navigation	4.5	
	15.05.2017	design blocks (sweetalert)	1.5	
	16.06.2017	sequence implement db models + create view	7	
	17.06.2017	sequence save	10	optimize constraints of drag n drop! GENERICs to save questionblock
Code Week 5	19.06.2017	sequence show updated stuff	5.5	optimize design
	19.06.2017	finish sequence	5	
	19.06.2017	other sequence options	5	combine it with the drag and drop option
				check if the same question is multiple time in the new analysis and give warnings! Show with the analysis block(overview) the process in percent!
	20.06.2017	finish other sequence options	5	
	20.06.2017	nav + error mgmt	6	QUESTION DELETION
	21.06.2017	share analysis to users	5.2	ROLE HANDLING !!
	21.06.2017			Edit question blocks!!
		show analysis to execute for the users	0.2	CHECK EVERYTHING BEFORE UNLOCK ANALYSIS
	21.06.2017	likert crud on scale	5	better constraints!!
	22.06.2017	settings analysis + likert finish	6	ajax forgery!!
	22.06.2017	design and setup of execution	2	WORKING WITH RETURN URL!. Asyncs?
	23.06.2017	execution load data and put on view	4	
Code Week 6	23. - 25.06.2017	bug fixes plus one page finish	3	
	26.06.2017	optimizing data management	4	
	27.06.2017	save answer into db	4	
	28.06.2017	new session if exists, navigation with block, show block infos	3	
	29.06.2017	import and configue chartjs	4	
	29.06.2017	design of chartjs	2	
	30.06.2017	pdf export with textsharp	5	
	30.06.2017	pdf export with RazorPDF	2	
	01.07.2017	pdf export with RazorPDF	2	
	01.07.2017	pdf export with RotativaPDF	4	
	02.07.2017	handle issues on segmentation of feedbacks	6	
	03.07.2017	designing of home, analysis execution	6	
	03.07.2017	catch and handle unsolved issues	5	

Anhang XI: Grobkonzept von SEPAT

Die folgenden Informationen wurden von Herrn Thomas Tribelhorn bereitgestellt, damit der Auftrag am Anfang der Bachelorarbeit besser verstanden wurde.



Anhang XII: Sourcecode Methode «SaveSelectedQuestionsInScale»

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult SaveSelectedQuestionsInScale(List<QuestionsInScaleViewModel> model)
{
    if (ModelState.IsValid)
    {
        // get current scale to save the question
        Scale a = _db.Scale.Where(s => s.IdScale == selectedScale).Select(s => s).FirstOrDefault();
        // get existing questions in scale
        var existingScaleQuestions = a.Questions;

        // update them
        List<Question> updatedScaleQuestions = new List<Question>();
        foreach (var item in model)
        {
            // check if selected
            if (item.SelectedQuestion != 0)
            {
                Question newQuestion = _db.Question
                    .Where(s => s.IdQuestion == item.SelectedQuestion)
                    .Select(s => s).FirstOrDefault();
                updatedScaleQuestions.Add(newQuestion);
            }
        }
        a.Questions = updatedScaleQuestions;

        try
        {
            _db.SaveChanges();
        }
        catch (Exception e)
        {
            ViewData["MessageUpdate"] = "Something went wrong!";
            return View("Index");
        }

        return RedirectToAction("AddQuestionsToScale", "AnalysisScales", new { idScale = selectedScale });
    }

    ViewData["MessageUpdate"] = "Something went wrong!";
    return View("AddQuestionToScale", model);
}
```

Anhang XIII: Sourcecode Methode «SetScalesAsBlocks»

```

public void SetScalesAsBlocks(List<ScaleViewModel> listScales, StandingDataBlockViewModel listStandingDataBlocks)
{
    // prepare indeces for the sequences
    int seqIteration = 0;

    // save scales as question blocks
    foreach (var itemScales in listScales)
    {
        // get scale you want to put as a question block
        Scale scale = _db.Scale.Where(s => s.IdScale == itemScales.IdScale).Select(s => s).FirstOrDefault();

        QuestionBlock block = new QuestionBlock();
        block.Title = scale.Title;
        block.Acronym = scale.Acronym;
        block.Description = scale.Description + ". (generated from scale)";
        block.IdAnalysis = newAnalysis.IdAnalysis;

        // persist blocks
        try
        {
            _db.QuestionBlock.Add(block);
            _db.SaveChanges();
        }
        catch (Exception e)
        {
            ViewData["MessageUpdate"] = "Something went wrong. Try one more time.";
        }

        // get ids to save the sequences
        foreach (var itemQuestions in itemScales.Questions)
        {
            // add sequences
            seqIteration++;
            children childsQuestion = new children();
            childsQuestion.IdQuestion = itemQuestions.IdQuestion;

            AddNewSequence(seqIteration, block.IdQuestionBlock, childsQuestion);
        }
    }

    // save standingdatablock as question block
    StandingDataBlock blockStandingData = _db.StandingDataBlock
        .Select(s => s).FirstOrDefault()
        .Where(s => s.IdStandingDataBlock == listStandingDataBlocks.IdStandingDataBlock);

    QuestionBlock blockFromPersonalData = new QuestionBlock();
    blockFromPersonalData.Title = blockStandingData.Title;
    blockFromPersonalData.Acronym = blockStandingData.Acronym;
    blockFromPersonalData.Description = blockStandingData.Description + ". (generated from personaldata block)";
    blockFromPersonalData.IdAnalysis = newAnalysis.IdAnalysis;

    // persist blocks
    try
    {
        _db.QuestionBlock.Add(blockFromPersonalData);
        _db.SaveChanges();
    }
    catch (Exception e)
    {
        ViewData["MessageUpdate"] = "Something went wrong. Try one more time.";
    }

    // save standingdataquestion into new question block
    foreach (var itemQuestions in blockStandingData.Questions)
    {
        // add sequences
        seqIteration++;
        children childsQuestion = new children();
        childsQuestion.IdStandingdataquestion = itemQuestions.IdStandingDataQuestion;

        AddNewSequence(seqIteration, blockFromPersonalData.IdQuestionBlock, childsQuestion);
    }

    // save sequences of standingdata block
    try
    {
        _db.SaveChanges();
    }
    catch (Exception e)
    {
        ViewData["MessageUpdate"] = "Something went wrong. Try one more time.";
    }
}

```

Anhang XIV: Sourcecode Methode «PutSequence»

```
[HttpPost]
public JsonResult PutSequence(List<ResonseSequenceViewModel> model)
{
    if (model == null)
        return Json(new { error = true, message = "An Error Has occoured" });

    // list to update the context
    List<Sequence> listSequence = new List<Sequence>();
    List<QuestionBlock> listBlocks = new List<QuestionBlock>();

    // list to delete existing sequences - save updated added sequences -> delete sequences which are not included
    var listUpdateOrDelete = new List<Sequence>();

    // add or update sequence
    bool existsSequence = _db.Sequence.Any(s => s.Analysis.IdAnalysis == newAnalysis.IdAnalysis);
    if (existsSequence)
    {
        listSequence = (from a in _db.Analysis
                        from s in a.Sequences
                        where a.IdAnalysis == newAnalysis.IdAnalysis
                        select s).ToList();

        listBlocks = (from a in _db.Analysis
                     from b in a.QuestionsBlocks
                     where a.IdAnalysis == newAnalysis.IdAnalysis
                     select b).ToList();
    }

    // prepare indeces for the sequences and blocks
    int seqIteration = 0;
    int idBlock = 0;
    int seqBlockIteration = 0;

    // iterate through blocks
    foreach (var blockItem in model)
    {
        seqBlockIteration++;

        // if entire scale was selected - create a block
        if (blockItem.idQuestionblock == 0)
        {
            QuestionBlock block = new QuestionBlock();

            // add new questionblock for the scale
            Scale scale = _db.Scale.Where(s => s.IdScale == blockItem.idScale).Select(s => s).FirstOrDefault();
            if (scale != null)
            {
                block.Title = scale.Title;
                block.Acronym = scale.Acronym;
                block.Description = scale.Description;
                block.IdAnalysis = newAnalysis.IdAnalysis;
                block.SequenceNr = seqBlockIteration;
            }

            // add new questionblock for standingdata question block
            StandingDataBlock stBlock = _db.StandingDataBlock
                .Where(b => b.IdStandingDataBlock == blockItem.idStandingdata)
                .Select(s => s).FirstOrDefault();

            if (stBlock != null)
            {
                block.Title = stBlock.Title;
                block.Acronym = stBlock.Acronym;
                block.Description = stBlock.Description;
                block.IdAnalysis = newAnalysis.IdAnalysis;
                block.SequenceNr = seqBlockIteration;
            }

            try
            {
                _db.QuestionBlock.Add(block);
                _db.SaveChanges();
            }
            catch (Exception e)
            {
                ViewData["MessageUpdate"] = "Something went wrong. Try one more time.";
                return Json(new { error = true, message = "An Error Has occoured by adding a new Question Block" });
            }
        }
    }
}
```

```
        // put the new blockid to scale
        blockItem.idQuestionblock = block.IdQuestionBlock;
    }
    else
    {
        // update sequence nr of the block
        var updateBlockSeq = listBlocks
            .Where(b => b.IdQuestionBlock == blockItem.idQuestionblock)
            .Select(b => b).FirstOrDefault();
        updateBlockSeq.SequenceNr = seqBlockIteration;
    }
    idBlock = blockItem.idQuestionblock;

    // iterate through the questions
    foreach (var questionItem in blockItem.children)
    {
        seqIteration++;

        if (existsSequence)
        {
            // check if questions in sequence
            bool existsQuestion = listSequence.Any(s => s.IdQuestion == questionItem.idQuestion ||
                s.IdStandingDataQuestion == questionItem.idStandingdataquestion &&
                s.IdQuestionBlock == blockItem.idQuestionblock);

            // add new sequence
            if (!existsQuestion)
            {
                Sequence newSequ = AddNewSequence(seqIteration, idBlock, questionItem);
                // add updated to delete list
                listUpdateOrDelete.Add(newSequ);
            }
            // update sequence iteration
            else
            {
                var updateSeq = listSequence.Where(s => s.IdQuestion == questionItem.idQuestion ||
                    s.IdStandingDataQuestion == questionItem.idStandingdataquestion &&
                    s.IdQuestionBlock == blockItem.idQuestionblock).FirstOrDefault();
                updateSeq.SequenceNr = seqIteration;
                updateSeq.IdQuestionBlock = blockItem.idQuestionblock;

                // add updated to delete list
                listUpdateOrDelete.Add(updateSeq);
            }
        }
        // no sequence exists right now - just add them
        else
        {
            AddNewSequence(seqIteration, idBlock, questionItem);
        }
    }
}

DeleteQuestionBlocks();

try
{
    // push sequences into db
    _db.SaveChanges();
}
catch (Exception e)
{
    return Json(new { error = true, message = "Check your Sequences" });
}

// update the sequences
if(existsSequence)
    DeleteSequence(listUpdateOrDelete);

return Json(new { success = true, message = "The Sequence is updated!" });
}
```


Anhang XV: Sourcecode Methode «Play»

```
public ActionResult Play()
{
    List<PlayViewModel> specificListQuestionAnalysis = new List<PlayViewModel>();
    Session newSeesion = new Session();
    string message = null;

    // get session object
    if (currentSession == null)
    {
        message = "Something went wrong";
        return GoBackIndex(message);
    }

    // check if session was already used -> update or create a new one
    if (currentSession.EndDate != null)
    {
        // create a new session
        newSeesion = new Session();
        newSeesion.Id = currentSession.Id;
        newSeesion.IdAnalysis = currentSession.IdAnalysis;
        newSeesion.StartDate = DateTime.Now;
        newSeesion.MedianOverAll = null;

        _db.Session.Add(newSeesion);
        currentSession = newSeesion;
    }
    else
    {
        currentSession.StartDate = DateTime.Now;
    }
    // save new or update existing session
    try
    {
        _db.SaveChanges();
    }
    catch (Exception e)
    {
        return GoBackIndex("Something went wrong by the session creation");
    }
    // load questions
    listQuestionForView = GetQuestionsForView();
    // list for the view
    specificListQuestionAnalysis = new List<PlayViewModel>();

    // prepare list if ids of question blocks
    // check analysis settings
    bool onePage = currentAnalyseSettings.Show All or Blocks;
    if (!onePage)
    {
        // settings for questionblocks on view
        specificListQuestionAnalysis = GetBlockPerPage();
    }
    else
    {
        // put all questions on one page
        specificListQuestionAnalysis = listQuestionForView.OrderBy(q => q.SequenceNr).ToList();
    }
    // for the progressbar
    ViewData["nrSequences"] = listQuestionForView.Count;
    ViewData["nrCurrentSequences"] = specificListQuestionAnalysis.Count;
    ViewData["idSession"] = currentSession.IdSession;
    ViewData["nameAnalysis"] = currentAnalysis.Title;
    return View(specificListQuestionAnalysis);
}
```

Anhang XVI: Sourcecode Methode «Analysis»

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Analysis(List<PlayViewModel> model)
{
    // check if all question on one page or block per page
    bool onePage = Convert.ToBoolean(currentAnalyseSettings.Show_All_or_Blocks);
    List<PlayViewModel> specificListQuestion = new List<PlayViewModel>();

    // save the answers already into db
    if (model != null)
    {
        if (onePage)
        {
            SaveAnswers(model, false);
            return JavaScript("window.location = '" +
                Url.Action("End", "Execution", new { idSession = currentSession.IdSession }) + "'");
        }
        else
        {
            // are there any other blocks? check if every question is set
            bool areAllSet = listQuestionForView.Any(a => a.alreadySet == false);
            SaveAnswers(model, areAllSet);

            // analysis is finished -> go to the results
            if (!areAllSet)
            {
                return JavaScript("window.location = '" +
                    Url.Action("End", "Execution", new { idSession = currentSession.IdSession }) + "'");
            }
        }
    }

    // clear the modelstate
    ModelState.Clear();

    specificListQuestion = new List<PlayViewModel>();

    // get all question for the one page option
    if (onePage)
    {
        // put all on the list
        specificListQuestion = listQuestionForView;
    }
    else
    {
        // list of block with sequences/questions
        specificListQuestion = GetBlockPerPage();
    }
    return PartialView("_Analysis", specificListQuestion);
}
```

Anhang XVII: Sourcecode Methode «SaveAnswers»

```
public void SaveAnswers(List<PlayViewModel> model, bool notFinished)
{
    // list with answers object
    Answer tempAnswer = null;
    MultipleAnswer tempAnswerMultiple = null;

    foreach (var item in model)
    {
        // temp object to put into list
        tempAnswer = new Answer();
        tempAnswer.IdSession = currentSession.IdSession;
        tempAnswer.IdSequence = item.idSequence;

        if (item.idQuestion != null)
        {
            // selected likert value
            tempAnswer.Answer_numeric = item.idLikertSelected;

            // add to list to save at the end
            finalList.Add(tempAnswer);
            // add result to total of likert
            resultLikert = resultLikert + item.idLikertSelected;
        }
        else
        {
            finalList.Add(tempAnswer);

            tempAnswerMultiple = new MultipleAnswer();
            // id of answer
            tempAnswerMultiple.Answer = tempAnswer;

            // textfield
            if (item.idStandingDataAnswerType == 7)
            {
                tempAnswerMultiple.AnswerText = item.inputTextfield;
                listMultipleAnswers.Add(tempAnswerMultiple);
            }
            // dropdown
            if (item.idStandingDataAnswerType == 9)
            {
                // selected dropdown value
                tempAnswerMultiple.AnswerText = item.MultipleOptions[0].Option;
                listMultipleAnswers.Add(tempAnswerMultiple);
            }
            // single choice
            if (item.idStandingDataAnswerType == 8)
            {
                // selected single choice value
                tempAnswerMultiple.AnswerText = item.MultipleOptions[0].Option;
                listMultipleAnswers.Add(tempAnswerMultiple);
            }
            // multiple choice
            if (item.idStandingDataAnswerType == 10)
            {
                foreach (var res in item.TempSelections)
                {
                    tempAnswerMultiple = new MultipleAnswer();
                    // id of answer
                    tempAnswerMultiple.IdAnswer = tempAnswer.IdAnswer;

                    // if selected = true
                    if (res.isSelected)
                    {
                        tempAnswerMultiple.AnswerText = res.SelectedOption;
                        listMultipleAnswers.Add(tempAnswerMultiple);
                    }
                }
            }
        }
    }
}
```

```
// can the data be persisted?
if (!notFinished)
{
    try
    {
        if(finalList.Count > 0)
        {
            foreach(var questions in finalList)
            {
                _db.Answer.Add(questions);
            }
        }
        if (listMultipleAnswers.Count > 0)
        {
            foreach (var itemStanding in listMultipleAnswers)
            {
                _db.MultipleAnswer.Add(itemStanding);
            }
        }
        _db.SaveChanges();
    }
    catch (Exception e)
    {
        GoBackToPlay("Something went wrong!");
    }
}
```

Anhang XVIII: Sourcecode Methode «End»

```
public ActionResult End(string idSession)
{
    FeedbackViewModel model = new FeedbackViewModel();
    List<TempAnswerScale> scaleAnswersResults = new List<TempAnswerScale>();
    List<ResultsScaleSession> listResultsScalesOfSession = new List<ResultsScaleSession>();
    decimal resultOverall = 0;
    Session session = new Session();

    int IdSessionSelected = Int32.Parse(idSession.ToString());
    currentSession = _db.Session
        .Where(s => s.IdSession == IdSessionSelected)
        .Select(s => s).FirstOrDefault();

    // save end time of session and median from likert scale
    Session updateSession = _db.Session
        .Where(s => s.IdSession == currentSession.IdSession)
        .Select(s => s).FirstOrDefault();

    // get all scale results and sum of the likertscale
    scaleAnswersResults = GetResultsOfScalesFromSequencesSUM(currentSession.IdSession);
    // calculate result per scale
    listResultsScalesOfSession = CalculateResultsOfScalesSequences(scaleAnswersResults);

    // attention if the user updates just the page
    if (updateSession.EndDate == null)
    {
        updateSession.EndDate = DateTime.Now;
        // calculate the result overall for the session
        resultOverall = CalculateResultOverall(listResultsScalesOfSession, scaleAnswersResults);
        // save results per scale and the final result overall scales
        session = _db.Session
            .Where(s => s.IdSession == currentSession.IdSession)
            .Select(s => s).FirstOrDefault();
        session.MedianOverAll = resultOverall;

        try
        {
            foreach (var itemResult in listResultsScalesOfSession)
            {
                _db.ResultsScaleSession.Add(itemResult);
            }
            _db.SaveChanges();
        }
        catch (Exception e)
        {
            return GoBackToPlay("Something went wrong by saving your result. Please try one more time.");
        }
    }

    // show feedback on view
    model = new FeedbackViewModel();

    // set the id of session on the view
    model.IdSession = IdSessionSelected;

    // show scales with their results from the session - overall and per scale
    TotalScalesFeedbackViewModel modelScale = null;
    model.ListTotalScales = new List<TotalScalesFeedbackViewModel>();

    // get data of scale -> question, result of user, feedback, conclusion text
    List<ScaleInfos> listModel = new List<ScaleInfos>();
    model.ListFeedbackPerScale = new List<ScaleInfos>();

    // prepare each feedback
    ScaleInfos infos = new ScaleInfos();
```

```
// gather the data for the total feedback and the individual feedback for the scales
foreach (var item in listResultsScalesOfSession)
{
    // scale where to take the data
    Scale scale = currentScales
        .Where(s => s.IdScale == item.IdScale)
        .Select(s => s).FirstOrDefault();

    // fill the total scale model
    modelScale = new TotalScalesFeedbackViewModel();
    modelScale.NameScale = scale.Title;
    modelScale.ScaleValue = item.ResultOfScale;

    // allocate the feedback for each scale feedback
    infos = new ScaleInfos();
    infos.NameScale = scale.Title;

    // to get the correct segment
    decimal resultOfScale = db.ResultsScaleSession.Where(r => r.IdScale == scale.IdScale &&
        r.IdSession == currentSession.IdSession)
        .Select(r => r.ResultOfScale).FirstOrDefault();

    // get feedback (segment)
    string [] feedbackinfos = GetSpecifcFeedbackPerScale(scale.IdScale, resultOfScale);
    infos.FeedbackTitle = feedbackinfos[0];
    infos.Feedbacktext = feedbackinfos[1];

    infos.MaxLikertValue = scale.LikertScalesOptions.Count;
    infos.ListScaleQuestionValues = GetQuestionAndResultsForFeedback(scale);

    // add to list of feedback view model
    model.ListTotalScales.Add(modelScale);
    model.ListFeedbackPerScale.Add(infos);
}
return View(model);
}
```

Selbstständigkeitserklärung des Verfassers

"Ich bestätige hiermit, dass ich die vorliegende Bachelorarbeit alleine und nur mit den angegebenen Hilfsmitteln realisiert habe und ausschliesslich die erwähnten Quellen benutzt habe. Ohne Einverständnis des Studiengangsleiters und des für die Bachelorarbeit verantwortlichen Dozierenden sowie des Forschungspartners, mit dem ich zusammengearbeitet habe, werde ich diesen Bericht an niemanden verteilen, ausser an die Personen, die mir die wichtigsten Informationen für die Verfassung dieses Berichts geliefert haben und die ich nachstehend aufzähle: Prof. Dr. René Schumann, Thomas Tribelhorn".

Siders, 2. August 2017

Gerd Zurbriggen