Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften und Informatik
Institut für Datenbanken und Informationssysteme

Entwicklung eines Usability-Konzepts für ein prozessorientiertes Informationssystem zum Austausch von Nachhaltigkeitsdaten

Bachelorarbeit an der Universität Ulm

Vorgelegt von:
Laurens Lang
laurens.lang@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
M. Sc. Gregor Grambow

2013

Fassung 24. Oktober 2013

© 2013 Laurens Lang

This work is licensed under the Creative Commons. Attribution-NonCommercial-ShareAlike 3.0 License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/3.0/de/ or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.

Satz: PDF-LaTeX 2_ε

Kurzfassung

Das Interesse an nachhaltigen Produkten und Prozessen steigt. Gesetzlicher und öffentlicher Druck zwingt Unternehmen zur Sammlung und Bewertung von Nachhaltigkeitsdaten. Heutzutage sind die Entwicklung und Produktion von vielen Produkten Ergebnis von intransparenten und komplexen Lieferketten mit weltweit verteilten Lieferanten. Dementsprechend erweist sich die Sammlung von Daten über die Lieferkette als ein sehr aufwendiges und komplexes Verfahren. Weiterhin erschwert der ständige Wandel der Wertschöpfungsketten das Prozedere. Speziell für Nachhaltigkeitsdaten mangelt es an einer Standardisierung des Datenaustauschs und somit fehlen häufig Angaben von Lieferanten oder diese sind aufgrund von fehlender Vollständigkeit oder Genauigkeit unbrauchbar. Bisherige Informationssysteme werden den Anforderungen dieser speziellen Datenverarbeitung nicht gerecht. Das EU-Projekt SustainHub will genau diese Lücke schließen mithilfe einer Plattform, die die Datensammlung über die Lieferkette unterstützt. Zudem soll diese automatisierte Aufgaben, wie zum Beispiel Datenverifikation, Validierung oder Autorisierung, bereitstellen. Dazu soll neben weiteren technischen Lösungen ein Webinterface entwickelt werden, um auch kleinere und mittlere Firmen zu erreichen. Um die Qualität dieses Webinterfaces aus Benutzersicht zu gewährleisten, werden die Methoden des Usability Engneerings vorgestellt. Die Ergebnisse, die sich aus der Anforderungsanalyse ergaben, werden hier dokumentiert. Auf dieser Grundlage wurden Mockups entworfen und daraufhin iterativ evaluiert und optimiert. Als Resultat ist ein Styleguide entstanden, der die Gestaltung des User Interfaces regelt. Zum Schluss wird als Ausblick die Vorgehensweise der Implementierung dieses Usability-Konzepts vorgestellt und mittels eines Proof of Concepts die Machbarkeit bewiesen.

1 Danksagung

An dieser Stelle möchte ich allen Personen danken, die mit ihrer Unterstützung zum Gelingen dieser Arbeit beigetragen haben. Ich möchte meinen Betreuern M. Sc. Gregor Grambow und Dipl. Medieninf. Vivian Steller danken. Sie haben mir nicht nur mit großer Hilfsbereitschaft und kritischen Hinweisen bei der Erstellung dieser Arbeit geholfen, sondern haben sich die Zeit genommen, mir einen tieferen Einblick in die Arbeit an Projekten zu verschaffen.

Außerdem möchte ich Herrn Prof. Dr. Manfred Reichert danken, der es mir ermöglicht hat, diese Arbeit am Institut für Datenbanken und Informationssysteme zu schreiben.

Weiterhin danke ich meinen Eltern, die mir durch ihre uneingeschränkte Unterstützung das Studium und dadurch auch diese Arbeit ermöglicht haben.

Inhaltsverzeichnis

1 Danksagung v

1 Einleitung **1**

1.1 Motivation 2

1.2 Aufgabenstellung 2

1.3 Aufbau dieser Arbeit 3

2 Einführung in SustainHub **5**

2.1 Überblick 5

2.2 Architektur 7

2.3 Datenmodell 9

2.4 Akteure 10

3 Technologien **13**

3.1 Basistechnologien 14

3.1.1 Rich Internet Application 14

3.1.2 Asynchronous JavaScript and XML 14

3.1.3 Google Web Toolkit 15

3.1.4 Java Servlets 15

3.1.5 SCSS 15

3.2 Vaadin 18

3.2.1 Themes 18

4 Usability Engineering **19**

4.1 Definition Usability 20

4.2 Modelle der Mensch-Maschine-Interaktion 21
4.2.1 IFIP-Modell für Benutzerschnittstellen 21
4.3 Kriterien für Usability Entwurf 22
4.3.1 Wahrnehmungstheorien 22
4.3.2 EN ISO 9241 Teil 110: Grundsätze der Dialoggestaltung 23
4.3.3 Usability Guidelines 24
4.4 Referenzmodell nach Daimler AG 26
4.4.1 Anforderungsanalyse 27
4.4.2 User-Interface-Entwurf 29

5 Anforderungsanalyse 31
5.1 Geschäftsziele und Einsatzziele 32
5.2 Analyse Ist-Stand 32
5.3 Benutzerprofilanalyse 32
5.3.1 Normale Nutzer 33
5.3.2 Administratoren 34
5.3.3 Fazit 35
5.4 Aufgabenanalyse des Participant Interfaces 35
5.4.1 Global erreichbare Funktionen 35
5.4.2 Administration und System 36
5.4.3 Datenaustausch 38
5.4.4 Kontakte 40
5.4.5 Dashboard 42
5.5 Aufgabenanalyse des Administration Interfaces 43
5.5.1 Global erreichbare Funktionen 43
5.6 Hardware- und Softwarerandbedingungen 44
5.6.1 Hardwarerandbedingung 44
5.6.2 Softwarerandbedingungen 45
5.7 Generelle Entwurfsprinzipien 46

6 Usabilityentwurf **49**
6.1 Konzeptueller Usabilityentwurf 49
6.1.1 Detailseite 50
6.1.2 Wizards 51
6.1.3 Tabellen 52
6.1.4 Formularkomponenten 52
6.1.5 Tokenfield 53
6.2 Participant Interface 55
6.2.1 Aufgabenhierarchie 55
6.2.2 Aufbau und Struktur der Benutzeroberfläche 56
6.2.3 Administration und System 58
6.2.4 Datenaustausch 59
6.2.5 Kontakte 68
6.2.6 Data 70
6.2.7 Dashboard 71
6.3 Administration Interface 71
6.3.1 Costumer Data 72
6.3.2 IHS verwalten 73

7 Styleguide **75**
7.1 Generelle Gestaltrichtlinien 75
7.1.1 Typografie 76
7.1.2 Farbkonzept 77
7.1.3 Favicon 77
7.2 Wiederkehrende Oberflächenelemente 78
7.2.1 User-Interface-Komponenten 78
7.2.2 Gruppierung von Formularfeldern 81
7.2.3 Pagination 81
7.3 Aufbau des User-Interfaces 82
7.3.1 Toolbar 82
7.3.2 Navigation und Subnavigation 83
7.3.3 Dialogarten 84

8 Alternative Konzepte **89**
8.1 Seitenaufbau 90
8.2 Wizards 92
8.3 Organisationsmodell 92

9 Implementierung eines Usability-Konzepts **95**
9.1 Vorgehensmodell zur Implementierung von UI-Konzepten 96
9.1.1 Masterpage 97
9.1.2 Analyse der Seiten 97
9.1.3 Implementierung 98
9.2 Proof of Concept 98
9.2.1 Masterpages 99
9.2.2 Analyse der Seiten 99
9.2.3 Implementierung 103
9.2.4 Dokumentation 106
9.3 Ergebnis 107

1

Einleitung

In der heutigen globalisierten Industrie basiert die Entwicklung und Produktion von vielen Produkten auf intransparenten und komplexen Lieferketten mit zahllosen Gliedern, die weltweit verteilt sind. Um die gemeinsame Produktion zu koordinieren, gibt es sogenannte Stücklisten, auf englisch Bill of Material (BOM), bei denen es sich um strukturierte Zusammenstellungen von Bauteilen handelt. Das Ergebnis ist, dass die BOMs häufig sehr groß und unübersichtlich sind, weil die Produkte sehr komplex sind. Europaweite Regelungen, wie beispielsweise die REACH-Verordnung (Registration, Evaluation, Authorisation and Restriction of Chemicals) und immer mehr Konsumenten fordern nachhaltige Produkte und Prozesse. Somit werden Nachhaltigkeitsdaten immer wichtiger. [GMSR13].

1.1 Motivation

Die Sammlung von Nachhaltigkeitsdaten gestaltet sich sehr aufwendig, da die Daten entlang der Lieferkette zwischen den Unternehmen ausgetauscht werden müssen. Weitere Problemfelder ergeben sich, da aufgrund der fehlenden Standardisierung die Daten sehr heterogen oder unvollständig angegeben werden. Zudem unterziehen sich die Lieferketten einem kontinuierlichen Wandel [GMS+13]. Einige Informationssysteme, wie z.B. das International Material Data System, lösen Teilprobleme. Allerdings existieren bisher keine Informationssysteme, die diese Problemstellung in ihrer Komplexität und Heterogenität ausreichend lösen können [GMSR13]. SustainHub will genau diese Probleme lösen, indem ein prozessorientiertes Informationssystem entwickelt wird, welches alle beteiligten Unternehmen mit einbezieht. Um dies zu ermöglichen, soll neben externer Systeme, welche mit der SustainHub-Plattform kommunizieren, ein Webinterface entwickelt werden. Dieses Webinterface stellt besondere Anforderungen an das User Interface. Usability beeinflusst die Nutzerakzeptanz und hat sich somit zu einem entscheidenden Qualitätskriterium eines Softwareprodukts herausgestellt. Um diese Qualitätsanforderung praktikabel und vollständig zu erfüllen, werden die Methoden des Usability Engineerings benutzt. Diese stellen mithilfe verschiedener Konzepte und Prozesse die Gebrauchstauglichkeit des Systems sicher.

1.2 Aufgabenstellung

Anhand von SustainHub soll ein Usability-Konzept für ein prozessorientiertes Informationssystems entwickelt werden. Es sollte die speziellen Anforderungen des Austauschs von Nachhaltigkeitsdaten unterstützen. Um das System den Anforderungen der Nutzer gerecht zu werden, sollen die Methoden des Usability Engineerings angewandt werden.

1.3 Aufbau dieser Arbeit

Nachdem in diesem Kapitel eine Motivation in diese Arbeit einleitet, wird im zweiten Kapitel das Projekt SustainHub vorgestellt. Darin werden besonders die Punkte vorgestellt, die wichtig zum Verständnis dieser Arbeit sind. Daraufhin werden im dritten Kapitel die eingesetzten Technologien beschrieben und erklärt, insbesondere das Framework Vaadin und dessen Basistechnologien. Das vierte Kapitel beinhaltet die Grundlagen und Methoden des Usability Engineerings, die im Zuge der Arbeit angewandt wurden. Das darin beschriebene Referenzmodell dient als Grundlage für die folgenden Kapitel: Im fünften Kapitel werden die Ergebnisse der Anforderungsanalyse dokumentiert. Neben einer Aufgabenanalyse werden auch die Benutzer analysiert und technische Randbedingungen erarbeitet. Die erarbeiteten Aufgaben, Bedingungen und Probleme des Usability-Konzepts dienen als Basis für die Entwicklung der Usabilityentwürfe, deren Ergebnisse im sechsten Kapitel vorgestellt werden. Die meist abstrakten Entwürfe wurden iterativ optimiert und erweitert. Das Resultat wird in Form eines Styleguides ausführlich im siebten Kapitel beschrieben. Nicht alle Entwürfe wurden für den Styleguide übernommen. Im achten Kapitel werden diese alternativen Konzepte vorgestellt und somit ein Einblick in den Entwicklungsprozess gegeben. Im neunten Kapitel wird eine Vorgehensweise zur Implementierung eines Usability-Konzepts beschrieben. Mithilfe eines Proof of Concepts wird diese daraufhin auf ihre Durchführbarkeit geprüft.

2

Einführung in SustainHub

Die Basis für die Entwicklung eines Usability-Konzepts eines Informationssystems für Lieferketten war der *Sustainability Data Exchange Hub*, kurz *SustainHub*. Um den Inhalt der Arbeit besser zu verstehen, werden die wesentlichen Bestandteile und Konzepte im Folgenden erläutert.

2.1 Überblick

Nachhaltigkeit hat sich zunehmend als ein zentrales Prinzip in Politik und Gesellschaft etabliert. Gesetzgebung und Verbraucher fordern von Unternehmen, besonders in der Elektronik- und Automobilindustrie, umweltfreundliche und nachhaltige Produkte. Spezielle Indikatoren werden, auch gesetzlich, genutzt, um die Qualität von Nachhaltigkeit der Prozesse und Produkte erfassen und bewerten zu können. Wenn Produkte und Prozesse

Ergebnisse von Wertschöpfungsketten sind, stehen die Unternehmen vor dem Problem, dass Daten von allen beteiligten Organisationen zusammengetragen werden müssen. Es existieren zwar einige Softwaresysteme, die die Sammlung, Verwaltung und Verarbeitung von Daten über die gesamte Lieferkette unterstützen. Allerdings erweist sich der dynamische Prozess der Datenerhebung und -verwaltung von Nachhaltigkeitsdaten in Lieferketten als komplizierte und zeitintensive Prozedur, da herkömmliche Systeme nicht die Anforderungen dieser spezifischen Datenverarbeitung erfüllen: Daten müssen von allen Lieferanten in der Lieferkette erfasst und berechnet werden. Dies betrifft sowohl direkte als auch indireke Lieferanten. Indirekte Lieferanten sind zum Beispiel Lieferanten der Lieferanten, also Kettenglieder, die in der Lieferkette in keiner direkten Verbindungen stehen. Indikatoren betreffen soziale, ökologische und strategische Aspekte. Aufgrund dieser Vielseitigkeit und der hohen Anzahl an Teilnehmern besteht kein einheitliches Format der Daten. Durch diese Heterogenität können Lieferkettenglieder häufig nur unbrauchbare Daten, die die Qualitätsanforderungen nicht erfüllen, anbieten. Mit *SustainHub* wird eine Plattform entwickelt, die genau diese Probleme lösen soll und eine Reihe von automatisierten Aufgaben, wie zum Beispiel Datenverifikation, Validierung oder Autorisierung, bereitstellt. Datenaustausch kann mit Hilfe sogenannter In-House-Solutions (IHS) erfolgen. Eine IHS ist ein externes System, welches mit SustainHub verbunden ist. Allerdings rentiert sich die Einrichtung eines solchen Systems bei bei kleineren und mittelständischen Unternehmen nicht. Für diese soll ein Webinterface entwickelt werden. Hierfür können Kunden sich und ihre Organisationen registrieren und untereinander Nachhaltigkeitsdaten austauschen. Dies kann durch einen Datenaustausch stattfinden. Ein Kunde sendet an das Zielunternehmen einen sogenannten Request. In diesem spezifiziert er, welchen Indikator und welche Anforderungen er an die Daten benötigt. Bei diesen Anforderungen handelt es sich um qualitative, z.B. ob es sich um gemessene oder geschätzte Daten handelt, zeitliche und weitere Informationen. Der Request wird an den zuständigen Teilnehmer weitergeleitet und kann mit einem Response beantwortet werden. Daten können auch ohne einen Request durch ein sogenanntes Push-Szenario ausgetauscht werden. Hierfür werden Daten zuerst bereitgestellt und daraufhin von den entsprechenden Unternehmen entgegengenommen.

2.2 Architektur

SustainHub ist ein großes und komplexes System. Die Aufteilung in Schichten bzw. Layer reduziert die Komplexität der Abhängigkeiten innerhalb des Systems. Dadurch wird eine geringe Kopplung bei gleichzeitig stärkerer Kohäsion der einzelnen Schichten erzielt. Somit wird durch diese Modularisierung der Funktionalität die Entwicklung und Wartung erleichtert. Diese Schichten sind weiter aufgeteilt in Komponenten und Subkomponenten. Abbildung 2.1, die aus internen Dokumenten stammt, zeigt diese. Im folgenden werden die Aufgaben dieser konzeptionellen Schichten skizziert:

Client Layer ermöglicht die Darstellung und Entgegennahme der Daten durch einen Browser.

Presentation Layer ermöglicht den Zugriff auf SustainHub vom Client. Hier befindet sich die Schnittstelle, um Daten abzurufen sowohl für Benutzer über den Browser als auch für Web-Service-Clients.

Application Layer beinhaltet die Datenspeicherung, die Orchestrierung, die SustainHub Logik und deren Schnittstellen.

Integration Layer fasst das Service Provider Interface und den externen Datenaustausch zusammen.

Persistence Layer kapselt den Zugriff auf persistente Daten und die dazu verwendeten Techniken.

Infrastructure Layer stellt die Umgebung des SustainHub Systems dar.

Für das Usability-Konzept ist die Präsentationsschicht/Presentation Layer von besonderer Bedeutung, da diese das User Interface (UI) beinhaltet. Diese Schicht ist unterteilt in das Participant Interface, das das Front-End und somit die Webschnittstelle für alle SustainHub-Kunden darstellt und ein Administration Interface, welches das Back-End des Systems darstellt und zur einfachen Verwaltung des Systems durch Administratoren dient.

Das Participant-Interface setzt sich aus den folgenden Komponenten zusammen. Bei der Erstellung dieses Prototyp stehen die wesentliche Ziele im Fokus. Die ausgegrauten

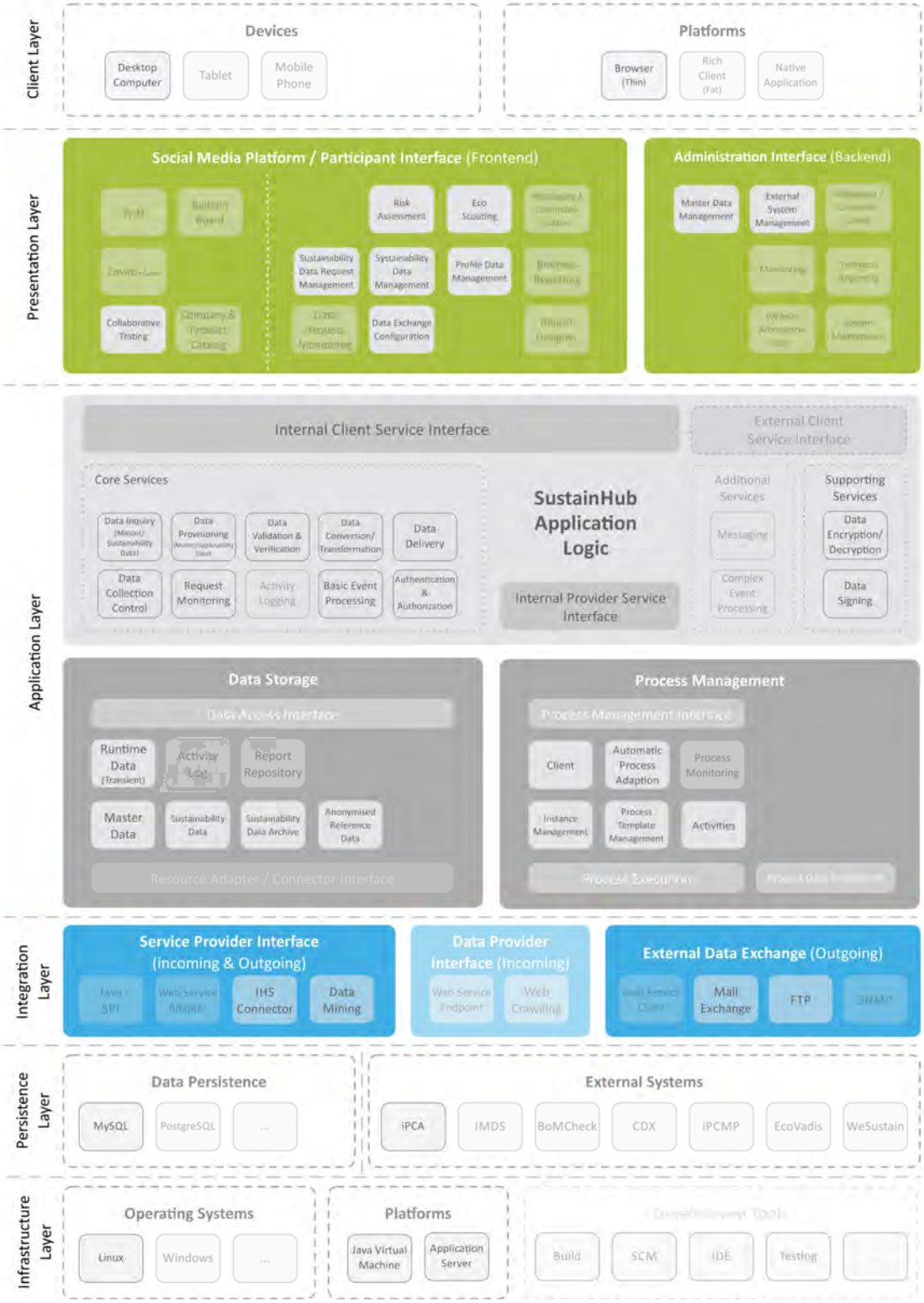


Abbildung 2.1: Architektur von SustainHub: Komponenten und Subkomponenten (Ausgegraute Komponenten sind "out-of-scope")

Komponenten werden deshalb nicht realisiert. Die Komponenten *Risk Assessment*, *Collaborative Testing* und *Eco Scouting* von anderen Projektteilnehmern realisiert:

Sustainability Data Request Management Einleitung, Anzeige und Steuerung von Requests und Responses

Sustainability Data Management Verwaltung von Daten, die durch einen Data-Push oder einen Request übermittelt worden sind.

Profile Data Management Verwaltung des Benutzerprofils und der Unternehmensstruktur.

Data Exchange Configuration Verwaltung von Präferenzen und Vertrauen gegenüber anderen Organisationen.

Die Präsentationsschicht setzt sich neben dem Participant Interface auch aus dem zuvor erwähnten Administration Interface zusammen. In diesem Back-End lassen sich die Masterdaten erstellen, editieren oder löschen. Zudem können externe Systeme darüber konfiguriert und verwaltet werden.

2.3 Datenmodell

Über das Webinterface können Daten sowohl angezeigt, gelöscht als auch verändert werden. Im weiteren Verlauf dieser Arbeit werden einzelne Entitäten genauer vorgestellt. Deshalb wird in diesem Abschnitt das konzeptionelle Datenmodell von SustainHub grob skizziert. In Abbildung 2.2 werden alle wichtigen Entitätskategorien gezeigt. Das Datenmodell ist in vier Bereiche unterteilt. Diese Informationen sind aus einem internen Dokument des Projekts entnommen.

Master Data sind Daten, die für alle SH-Costumer verfügbar sind.

Costumer Data sind spezifische Daten, die für einzelne Teilnehmer bereitgestellt sind.

Runtime Data sind flüchtige Daten, die im Rahmen eines Datenaustauschs zwischen Teilnehmern gespeichert werden müssen.

Sustainability Data sind konkrete Nachhaltigkeitsdaten, die von SustainHub-Teilnehmern z.B. durch einen Response bereitgestellt wurden.

Das Datenmodell umfasst Daten über Organisationen, deren Struktur und ihren Mitarbeitern. Es können Produkte, Komponenten, Materialien und Substanzen gespeichert werden. Mithilfe der BoM-Entität können Relationen zwischen Komponenten dargestellt werden, wie es üblicherweise bei Firmen mithilfe der BOMs geschieht. Es existieren Nachhaltigkeitsindikatoren mit zugehörigen Anforderungen. Für diese können Requests, Responses und Push-Daten erstellt werden.

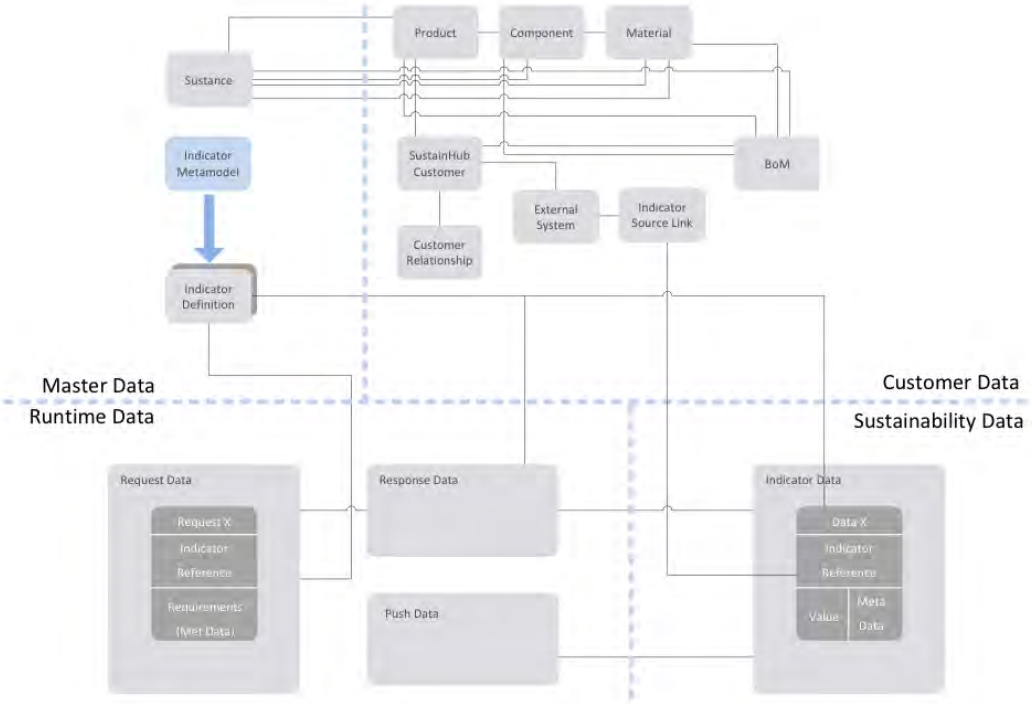


Abbildung 2.2: Datenmodell von SustainHub)

2.4 Akteure

Verschiedene Akteure interagieren mit dem System. Dies betrifft natürliche Akteure. Ein natürlicher Akteur kann je nach Anwendungsfall ein Nicht-Mitglied, Mitglied, Requester,

Responder, Responder (nicht registriert), Service Provider, Assessor, Administrator oder System Administrator sein. Auch technische Akteure, wie das System, IHS oder externe Systeme, kommunizieren mit dem System. Alle Akteure werden in folgender Tabelle 2.4 zusammengefasst und vorgestellt. Diese stammt aus internen Dokumenten.

Name	Description	Synonyms	Comments
Requester	Ein SustainHub-Kunde, der Daten von anderen verbundenen oder nicht registrierten Akteuren anfragt	OEM, Sender, Member, Participant, Lieferant, Kunde, Supplier	Jeder Requester ist ein registriertes Mitglied
Responder	Ein registriertes Mitglied, das im Zuge eines Requests via Web-Form Informationen anbietet.	Supplier, Responder, Member, Participant, Kunde, Customer	Jeder Responder ist ein registriertes Mitglied
Responder (nicht registriert)	Ein nicht registriertes Mitglied, das im Zuge eines Requests offline Informationen anbietet (über ein anderes Mitglied)	Supplier, Responder	Ein solcher Responder ist nicht registriert.
Non-Member	Jemand, der nicht auf SustainHub registriert ist	Disconnected Answerer, Disconnected Customer	
Service Provider	Jemand, der im Zuge eines Push/Pull-Szenarios Daten anbietet	Non-Participant	Services o. Daten können von SustainHub selbst zur Verfügung gestellt werden.
Assessor	Jemand, der mit SH verbunden ist und Daten von anderen Mitgliedern überprüft	Data Provider, External Party, Member(/Participant), Content provider	Jeder Assessor muss ein SustainHub-Mitglied sein.
Administrator	Verantwortliche Person eines SH Kunden, die für die Einrichtung und Administration der IHS zuständig ist.	External Party	
System Administrator	Personal von SustainHub, das für administrative Aufgaben zuständig ist.		
System	Technischer Akteur, der das SH System repräsentiert	SustainHub Administrator	Wird benutzt, um das SustainHub System z.B. in einem Modell zu repräsentieren.
IHS	Technischer Akteur, der ein externes, verbundenes System repräsentiert.	SustainHub	Eine IHS muss von SH registriert und geprüft sein.
Externes System	Technischer Akteur, der ein externes System repräsentiert, welches nicht dauerhaft mit SustainHub verbunden ist. Verbindung zu SH ist ad-hoc aufgebaut.	Registrierter Service Provider	Ein registriertes System kann Funktionen (aktiv) triggern.

3

Technologien

In diesem Kapitel werden die verwendeten Technologien vorgestellt. Das Hauptaugenmerk liegt dabei auf dem Framework Vaadin. Vaadin wurde ausgewählt, weil sich damit eine moderne Webapplikation umsetzen lässt. Zudem lässt es sich in die bisherige Systemarchitektur gut einbinden und alle Projektteilnehmer haben bereits Erfahrungen damit. Vaadin baut auf den Konzepten der Rich Internet Application, Asynchronous JavaScript and XML, Google Web Toolkit Java-Servlets und Syntactically Awesome Stylesheet auf. Vaadin setzt sich aus einem serverseitigem Framework und einer clientseitigen Engine zusammen.

3.1 Basistechnologien

Anhand der Erklärung der Basistechnologien, wird detailiert beschrieben, wie Vaadin aufgebaut ist und funktioniert.

3.1.1 Rich Internet Application

Der Begriff *Rich Internet Application* ist nicht fest definiert, sondern dient nur dazu, sich von simpleren Webapplikationen zu unterscheiden. Die Usability von und die Interaktionsmöglichkeiten mit UIs sind bei herkömmlichen Webapplikationen sehr beschränkt, da diese meist nur aus HTML bestehen. Diese bieten nur ein eingeschränktes Repertoire an Funktionen. Rich Internet Applications bieten anspruchsvollere grafische Komponenten und effizientere Interaktionsfähigkeiten, die ein Look-and-Feel ermöglicht, das Desktop Applikationen in nichts nachsteht. Die Datenverarbeitung findet sowohl auf dem Server als auch auf dem Client statt. Der Datenaustausch wird durch asynchrone Datenübertragung optimiert [BK09].

3.1.2 Asynchronous JavaScript and XML

Dies wird im Falle von Vaadin mit *Asynchronous Javascript and XML* (*AJAX*) realisiert. Man versteht darunter eine Webtechnologie, die asynchrone Datenübertragung zwischen Server und Browser ermöglicht, indem diese in mehrere Requests aufgeteilt wird und jede Anfrage nur die Informationen beinhaltet, die auch wirklich benötigt werden. Somit können kontinuierlich Elemente des UIs neu geladen und berechnet werden. *Cascading Style Sheets* (*CSS*) definieren das visuelle Erscheinungsbild einer Website, während HTML/XHTML die Struktur und Formatierung einer Webpage festlegt. Über das *Document Object Model* (DOM) kann ein JavasSript-Programm diese manipulieren. DOM beschreibt die Darstellung von HTML- oder XML-Dokumenten im Hauptspeicher, welches es ermöglicht den Inhalt, die Struktur und das Layout eines Dokuments zu verändern [LWW13]. Mithilfe der JavaScript Object Notation (JSON) werden Daten serialisiert und zwischen Anwendungen ausgetauscht werden. Für den Nutzer bedeutet der

Einsatz von AJAX, dass Inhalte und Komponenten dynamisch verändert werden können ohne die Seite neuzuladen [Gro10].

3.1.3 Google Web Toolkit

Das clientseitige Framework von Vaadin basiert auf dem Google Web Toolkit (GWT). Damit lassen sich User-Interfaces von Webapplikationen mit Java implementieren. Das Toolkit beinhaltet einen Compiler, der den Java Code der Clientkomponenten in optimierten HTML- und Javascript-Code übersetzt. Einer der Hauptaspekte, die dabei berücksichtigt werden, ist die Browserkompatibilität. Der AJAX-Code wird generiert, indem auf einer höheren Abstraktionsebene mittels Java implementiert wird. Die Folge ist, dass die strikte Trennung der Schichten vereinfacht wird und der Softwareengineeringsprozess beschleunigt wird. Die Interprozesskommunikation zwischen Server und Client erfolgt über Remote Procedure Calls, die es ermöglicht entfernte Funktionen- und Methoden auf dem Server aufzurufen [Gro10].

3.1.4 Java Servlets

Java Servlets sind Java-Klassen auf dem Webserver, die HTTP-Requests des Clients behandeln. Die vom Client gesendeten Daten werden gelesen. Die Antworten werden dynamisch generiert und z.B. in statisches HTML eingebunden. Dieses Dokument wird über eine HTTP-Response an den Client gesendet.

3.1.5 SCSS

Cascading Style Sheets ist eine deklarative Sprache für Stilvorlagen von strukturierten Dokumenten, mithilfe das konkrete visuelle Erscheinungsbild, wie Farben, Größen und Abstände, definiert wird. Somit kann abgesondert davon mittels HTML nur die inhaltliche Untergliederung eines Dokumentes beschrieben werden.

Syntactically Awesome Stylesheet, SASS, ist eine Erweiterung von Cascading Style Sheets 3. Es existieren zwei verschiedene Syntaxen. Die ”indented syntax” und der in

Vaadin verwendete ”Sassy CSS”, SCSS, die auf der CSS-Syntax aufbaut [CWE13]. Im Folgenden werden die Erweiterungen von SCSS mithilfe von Beispielen erklärt. Jedes Beispiel enthält zum einen den klassischen CSS Code und wie er mithilfe von SCSS vereinfacht werden kann.

Es können **Variablen** definiert werden. Hexadezimal-Farbwerte und spezielle Größen, die mehrmals auftauchen können somit an einer Stelle definiert werden. Im Beispiel aus Abbildung 3.1 wird ein Grün deklariert und in mehreren Klassen aufgerufen.

```
1   /* CSS */
2
3   .button{
4         background-color: #8dae33;
5         width: 40px;
6   }
7
8   .successText{
9         color: #8dae33;
10        font-weight: bold;
11  }
12
13
```

```
/* SCSS */

$sustainHubGreen = #8dae33;

.button{
      background-color: $sustainHubGreen;
      width: 40px;
}

.successText{
      color: $sustainHubGreen;
      font-weight: bold;
}
```

Abbildung 3.1: Variablen in SCSS

Zudem kann durch **Verschachtelung von Selectors und Properties** Wiederholungen der Klassennamen vermieden werden. Abbildung 3.2 zeigt dies.

```
1   /* CSS */
2
3   .panel{
4         width: 500px;
5   }
6
7   .panel h1{
8         text-align: center;
9   }
10
11
12
13
```

```
/*SCSS*/

.panel{
      width: 500px;

      h1{
          text-align: center;
      }

}
```

Abbildung 3.2: Verschachtelung von Selectors und Properties in SCSS

Mithilfe von **Mixins** lassen sich ganze CSS Fragmente wiederverwenden und für diese Parameter setzen. Abbildung 3.3 die unterschiedliche Umsetzung zweier ähnlichen Tabellen. Hierfür wird ein Mixin für die Größe und das Layout der Tabelle erstellt und

zwei Mixins für die jeweilige Farbkombination. Mit **@include** können diese aufgerufen werden.

```
1   /* CSS */
2
3   #bigwhitelist{
4         background-color: white;
5         color: black;
6         width: 300px;
7         font-size: 12px;
8         float: left;
9   }
10
11  #smallblacklist{
12        background-color: black;
13        color: white;
14        width: 100px;
15        font-size: 12px;
16        float: left;
17  }
18
19
20
21
22
23
24
25
26
27
```

```
/*SCSS*/

@mixin list($size){
      font-size: 12px;
      float: left;
      width: $size;
}

@mixin black{
      background-color: black;
      color: white;
}

@mixin white{
      background-color: white;
      color: black;
}

#bigwhitelist{
      @include list(200px);
      @include white;
}

#smallblacklist{
      @include list(100px);
      @include black;
}
```

Abbildung 3.3: Mixins in SCSS

SCSS unterstützt außerdem **Vererbung**. Mithilfe von **@extend** können komplette Definitionen von Selektoren übernommen werden ohne den Code zu duplizieren. In Abbildung 3.4 wird die komplette Definition des Selektors .warning in .error übernommen:

```
1   /* CSS */
2
3   .warning{
4         background-color: red;
5         color: black;
6   }
7
8   .error{
9         background-color: red;
10        color: black;
11        font-weight: bold;
12  }
13
```

```
/* SCSS */

.warning{
      background-color: red;
      color: black;
}

.error{
      @extend warning;
      font-weight: bold;
}
```

Abbildung 3.4: Vererbung in SCSS

3.2 Vaadin

Vaadin ist ein Webapplication Framework für die Erstellung von Rich Internet Applications. Diese werden mithilfe des Framworks in Java entwickelt. Entwickler können somit RIAs erstellen ohne sich mit Webtechnologien wie HTTP, HTML, Javascript und CSS zu befassen. Dies wird realisiert, indem Vaadin aus zwei Teilen besteht: dem serverseitigem Framework und der clientseitigen Engine. Die Anwendungslogik befindet sich in Java Servlets auf einem Webserver. Diese erhalten Requests des Clients, Events, die vom Eventlistener verarbeitet werden. Vaadin stellt eine Reihe von User-Interface-Komponenten und Layouts zur Verfügung. Dieses Sortiment kann mit Addons von Dritten erweitert werden. Zudem bietet Vaadin die Möglichkeit Komponenten eigens zu definieren.

3.2.1 Themes

Die Anwendungslogik ist in Vaadin von dem visuellen Erscheinungsbild getrennt. Dieses wird durch Themes mithilfe SCSS, HTML-Layouts und Grafiken definiert. Jede UI-Komponente besitzt mindestens eine CSS-Klasse. In der Datei styles.scss kann somit via SCSS oder CSS die Darstellung der Komponenten bestimmt werden. UI-Komponenten können auch individuell verschieden gestaltet werden. Dazu wird diesen ein Style-Klassenname zugewiesen. Ist dieser in der SCSS-Datei festgelegt, werden dadurch die allgemeinen Styles überschrieben.

4

Usability Engineering

In diesem Kapitel werden die Prinzipien und Methoden des Usability Engineerings vorgestellt. Dazu wird erst in Kapitel 4.1 definiert, was Usability ist. Danach werden drei wesentliche Fragen beantwortet:

1. Was lässt sich gestalten?
2. Nach welchen Kriterien wird ein Usability-Konzept entworfen und beurteilt?
3. Wie geht man dabei systematisch vor?

In Kapitel 4.2 wird geklärt, was sich gestalten lässt. Im Kapitel 4.3 werden die Kriterien vorgestellt, nach denen ein Usability-Konzept entworfen und beurteilt wird. Kapitel 4.4 stellt ein Referenzmodell vor, mit dem ein solches Konzept systematisch erstellt werden kann.

4.1 Definition Usability

Usability ist ein Qualitätsmerkmal eines Systems. Die ”Internationale Organisation für Normung”, kurz ISO, definiert Usability in ”EN ISO 9241 Teil 11: Anforderungen an die Gebrauchstauglichkeit” wie folgt: Die Qualität der Gebrauchstauglichkeit ist der Grad, in dem ein System von bestimmten Nutzern in einem bestimmten Nutzungskontext bestimmte Ziele *effektiv*, *effizient* und *zufriedenstellend* erreichen. Diese drei Leitkriterien werden weiter präzisiert:

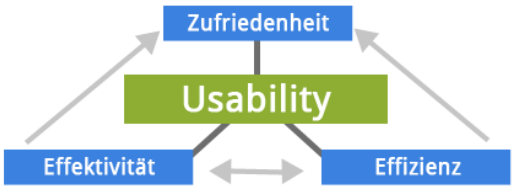


Abbildung 4.1: Usabilitykriterien und deren Relation

Effektivität beschreibt die Genauigkeit und Vollständigkeit, mit denen ein Nutzer ein bestimmtes Ziel erreicht.

Effizienz ist erreicht, wenn der Aufwand im Verhältnis zur Effektivität möglichst gering ist, um eine Aufgabe zu erledigen.

Zufriedenheit stellt die subjektive Komponente dar. Mit ihr ist die Freiheit von Beeinträchtigungen durch das System und die positive Einstellung des Nutzers gegenüber der Nutzung des Systems gemeint.

Abbildung 4.1 zeigt die Relation zwischen diesen Kriterien. Je effektiver und effizienter das System ist, desto mehr wächst die Zufriedenheit. Die Kriterien Effektivität und Effizienz stehen in einem Spannungsverhältnis: Je kleiner der Aufwand, desto ungenauer können Ziele erreicht werden. Je präziser das Ziel erreicht werden soll, desto mehr Aufwand muss vom Nutzer investiert werden. Hier gilt es das bestmögliche Verhältnis zu finden. Dieses beeinflusst die Zufriedenheit des Nutzers. Die Gewichtung dieser Kriterien kann je nach Nutzungskontext des Systems variieren.

4.2 Modelle der Mensch-Maschine-Interaktion

Bevor Benutzeroberflächen gestaltet werden können, muss geklärt werden, was gestaltet werden kann. Hierzu werden die zwei Modelle hergezogen, das Schalenmodell und das *International Federation for Information Processing Modell* (IFIP-Modell). Beide Modelle heben die Gesamtheit der Gestaltung hervor. Das Schalenmodell stellt allerdings die Reihenfolge der Gestaltungsprozessschritte in den Vordergrund, weshalb nur das IFIP-Modell vorgestellt wird.

4.2.1 IFIP-Modell für Benutzerschnittstellen

Das IFIP-Modell verschafft einen Überblick, wie Mensch und Computer miteinander interagieren. Es unterteilt die Benutzerschnittstelle in die folgenden vier, voneinander unabhängige, Schnittstellen:

Ein-/Ausgabeschnittstelle bezieht sich auf die Eingabe durch den Benutzer und die Ausgabe, z.B. die Präsentation von Daten.

Dialogschnittstelle definiert die Abläufe der Arbeit z.B. Unterteilungen in mehrere Teilschritte.

Werkzeugschnittstelle umfasst Regeln der eigentlichen Aufgabe des Benutzers. Für den Benutzer werden Daten durch Arbeitsgegenstände aus der realen Welt symbolisiert und die Anwendungssoftware als Werkzeugansammlung, die die Arbeitsgegenstände manipulieren können.

Organisationsschnittstelle umfasst die Regeln, die die Aufgaben des Benutzers in Relation zu den Geschäftsprozessen, z.B. Arbeitsabläufe oder Arbeitsteilung, stellen. Damit sind sowohl technische als auch nichttechnische Aspekte gemeint.

Abbildung 4.2 zeigt, dass der Nutzer mit allen vier Schnittstellen interagiert. Die Organisation wird außerhalb der Benutzerschnittstelle angezeigt, da sie für die eigentliche Arbeit mit dem Rechner nicht wichtig ist. Um die Qualität der Benutzeroberfläche ganzheitlich zu verbessern, müssen allerdings alle Aspekte beachtet werden [Jan01]. Neben der Ganzheit der Gestaltung wird auch die Forderung aufgestellt, dass unabhängig eines

Abbildung 4.2: IFIP Modell

spezifischen UIs auf gleiche Eingaben bzw. Aktionen die gleichen Ausführungen folgen müssen. Solche anwendungsunabhängigen Benutzerschnittstellen führen dazu, dass die Erlernbarkeit neuer Systeme beschleunigt wird.

4.3 Kriterien für Usability Entwurf

In diesem Kapitel werden die Kriterien des Usability Engineerings vorgestellt, mit denen sich Usability-Konzepte beurteilen und entwerfen lassen.

4.3.1 Wahrnehmungstheorien

Die Mensch-Maschine-Interaktion basiert auf der gezielten Nutzung der menschlichen Sinneswahrnehmung. Durch die gezielte zeitliche und räumliche Anordnung von Elementen, lassen sich UIs verständlicher und intuitiver gestalten und die Aufmerksamkeit der Nutzer gezielt lenken. Die Gestaltpsychologie, die im folgenden Kapitel vorgestellt wird, ist dabei für User Interfaces besonders wichtig [WN12].

Gestaltpsychologie

Die Gestaltpsychologie beschäftigt sich mit der Untersuchung der grundsätzlichen kognitiven Mechanismen, die bei der Wahrnehmung von Phänomenen auftreten.

Die zentrale Idee der Gestaltwahrnehmung ist, dass ein Wahrnehmungseindruck nicht durch die Summe seiner Teile, sondern ein ganzheitlicher Wahrnehmungseindruck gebildet wird. Visuelle Reizwirkungen können verschieden interpretiert werden. Das Wahrnehmungssystem realisiert meistens nur eine einzige. Verständlichkeit und Intuitivität können durch den gezielten Einsatz dieser Gestaltgesetze erhöht werden. Die wichtigsten Gesetze werden anhand von Beispielen in Abbildung 4.3 erklärt [WN12].

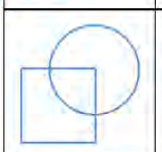
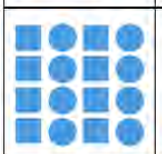
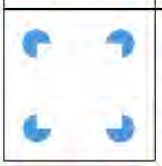
Beispiel	Name und Erklärung
	Gesetz der Nähe Elemente einer Menge, die sich in geringer Distanz zueinander befinden, werden, trotz möglicherweise verschiedener Form und/oder Farbe, als zusammengehörig erkannt. Die Abgrenzung verschiedener Gruppen beginnt dort, wo die Distanz zwischen Elemente größer ist. Vor allem findet dies Anwendung in der Menügestaltung, wo verschiedene Menüpunkte einer Kategorie in ein eigenes Menü zusammengefasst werden.
	**Gesetz der Prägnanz** Oft auch bekannt als „Gesetz der guten Form", besagt, dass komplexere Figuren in einfachere Formen, wie Kreise, Vierecke oder Dreiecke, reduziert werden.
	**Gesetz der Ähnlichkeit** Elemente einer Menge mit gleichem Erscheinungsbild werden vom Betrachter als zusammengehörig wahrgenommen. Bei Elementen mit mehreren Merkmalen ist die Wahrnehmung von der Anzahl der gleichen Merkmale abhängig.
	**Gesetz der Geschlossenheit** Das Gesetz der Geschlossenheit besagt, dass Flächen, die von nicht offenen Formen umrandet sind, eine Gesamtheit bildet, wodurch sich Elemente, die sich innerhalb der Abgrenzung befinden, als zusammengehörig empfunden werden

Abbildung 4.3: Gestaltgesetze [Dah05]

4.3.2 EN ISO 9241 Teil 110: Grundsätze der Dialoggestaltung

ISO 9241 ist ein mehrteiliger Standard der ISO, der verschiedene Aspekte der Mensch-Computer Interaktion normiert. In Teil 110 werden Grundsätze für die Bewertung und Ge-

staltung von Benutzerschnittstellen, die sogenannten sieben Gütekriterien der Software-Ergonomie, bestimmt [ISO06]:

Aufgabenangemessenheit Ein Dialog ist aufgabenangemessen, wenn der Nutzer dahingehend unterstützt wird, dass er seine Aufgaben effektiv und effizient durchführen kann. Dabei ist die Verfügbarkeit des Systems Grundvoraussetzung. Zudem sollten die angebotenen Funktionen nützlich sein, um die Aufgaben erledigen zu können. Komfort sollte dahingehend bestehen, dass alle Informationen und Funktionen so angeboten werden sollten, dass keine Handhabungsprobleme aufkommen und der Nutzer somit nicht von seiner Arbeit abgehalten wird. Übersichtlichkeit soll das Kurzzeitgedächtnis entlasten.

Selbstbeschreibungsfähigkeit Jeder einzelne Dialogschritt ist durch Rückmeldungen des Systems unmittelbar verständlich oder wird auf Anfrage erklärt.

Steuerbarkeit Der Nutzer muss in der Lage sein, den Dialogablauf zu steuern, also Ablauf, Richtung und Geschwindigkeit, bis das Ziel erreicht ist.

Erwartungskonformität Ein Dialog ist erwartungskonform, wenn er den Kenntnissen des Nutzers auf seinem Arbeitsgebiet, seiner Ausbildung oder Erfahrung entspricht. Der Dialog ist außerdem konsistent gestaltet und stimmt mit anerkannten Konventionen überein.

Fehlertoleranz Der Nutzer kann, trotz erkennbarer Fehlereingaben, mit keinem oder minimalem Korrekturaufwand das beabsichtigte Arbeitsergebnis erreichen.

Lernförderlichkeit Der Nutzer wird beim Nutzen der Anwendung assistiert und angeleitet.

Individualisierbarkeit Der Nutzer muss den Dialog an die Erfordernisse der Arbeitsaufgabe bzw. seine Vorlieben und Fähigkeiten anpassen können.

4.3.3 Usability Guidelines

Neben generellen psychologischen Erkenntnissen, die für die Erstellung von User-Interfaces wichtig sind, existieren Guidelines, mit deren Richtlinien die Qualität von

Benutzeroberflächen beurteilt und verbessert werden kann. Am bekanntesten sind die *zehn Design Heuristiken nach Molich und Nielsen*. Obwohl diese aus dem Jahre 1990 stammen, werden sie als zeitlose Richtlinien in der Usability-Welt angesehen. Etwas neuer hingegen sind die sogenannten *acht goldenen Regeln des Interaction-Designs*. Um Wiederholungen zu vermeiden, werden im folgenden die wichtigsten Punkte aus beiden Guidelines vorgestellt [WN12].

Feedback Der Nutzer sollte durch Rückmeldungen stets darüber informiert sein, was das System macht. Diese sollten zeitnah und verständlich auftauchen, damit sie richtig zugeordnet werden [Nie04].

Metaphern und Mappings Analogien aus der realen Welt helfen technischen Laien das System zu verstehen [Nie04].

Konsistenz Gleiche Sachverhalte sollten einheitlich dargestellt werden. Eine konsistente Gestaltung führt zu Vorhersagbarkeit und vermittelt dem Nutzer ein Gefühl der Kontrolle [Pru13].

Fehlervermeidung Besser als gute Fehlermeldung ist, wenn das Design dazu führt, dass Fehler verhindert werden [Pru13].

Benutzerkontrolle und Zufriedenheit Nutzer machen bei der Systembedienung Fehler. Unerwünschte Aktionen sollten rückgängig gemacht werden können. Dies ermöglicht zudem, dass der Nutzer das System explorativ erlernen kann [Nie04].

Mininale mentale Belastung Das Kurzzeitgedächtnis hat eine beschränkte Größe. Mögliche Objekte, Aktionen und Optionen sollten sichtbar sein, damit diese erkannt werden. Der Nutzer muss sich diese nicht merken und Kurzzeitgedächtniskapazitäten stehen der Erlernbarkeit des System oder Inhalten zur Verfügung [Nie04].

Nutzererfahrung Steigt die Erfahrung der Nutzer mit dem System bedarf es Shortcuts und Spezialistenoptionen. Diese beschleunigen die Interaktionsschritte [Nie04].

Simples Design Dialoge sollten auf die wichtigen Informationen reduziert sein. Um die Nutzerakzeptanz zu erhöhen, hilft ein ästhetisches und minimales Design [Pru13].

Hilfestellung in Fehlersituationen Treten Fehler auf, sollten diese präzise beschrieben werden. Fehlermeldungen sollten konstruktiv und lösungsorientiert formuliert sein [Pru13].

Hilfe Optimal sollte ein System ohne Hilfe oder Dokumentation bedienbar sein. Neben einer klassischer Hilfe, sollten Wizards, Einblendungen und digitale Assistenten dem Nutzer helfen [Nie04].

4.4 Referenzmodell nach Daimler AG

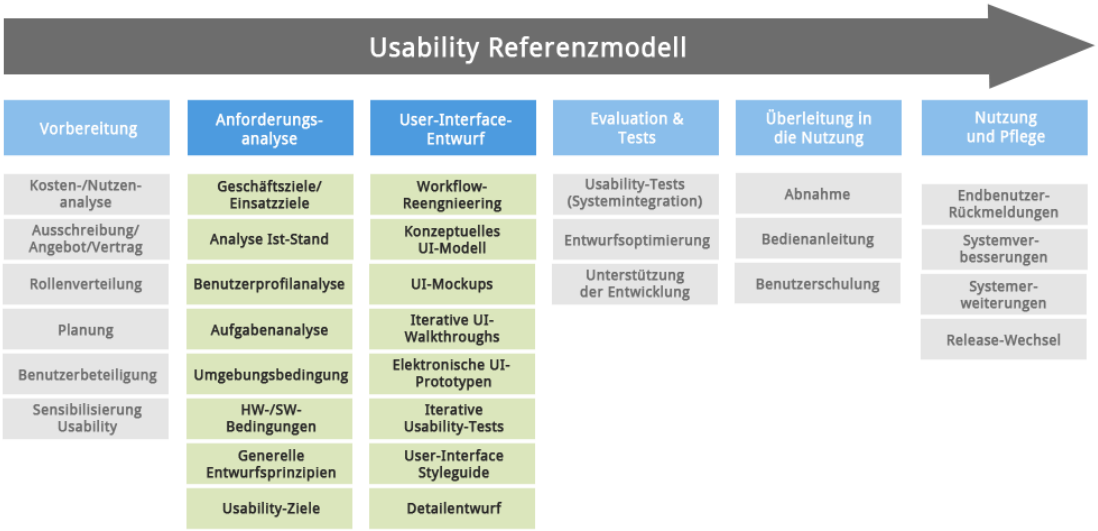


Abbildung 4.4: Referenzmodell nach Daimler AG

Das Referenzmodell nach Daimler AG ist ein Systementwicklungsprozess, das Usability Engineering in den Mittelpunkt stellt. Für die Erarbeitung des Usability-Konzepts habe ich mich an dem *Referenzmodell für benutzerzentrierte User-Interface-Entwicklung nach Daimler AG* orientiert, da sich Usability ganzheitlich, systematisch, empirisch und durchgängig in eine Systementwicklung miteinbeziehen lässt. Herkömmliche Vorgehensmodelle der Softwaretechnik beziehen nur selten User-Interface-Entwicklung mit ein. Dies führt zu schlechten Benutzerschnittstellen, welche entweder die Qualität des

Systems wesentlich einschränken oder mit viel Aufwand geändert werden müssen. Ziel dieses Modells ist es ein System zu erstellen, das den Bedürfnissen der späteren Nutzer entspricht. Wie Abbildung 4.4 zeigt, umfasst das Referenzmodell 6 Schritte, die wiederum in Ebenen unterteilt sind. Je nach Bedarf können Teile ausgelassen werden. Die Phase *Vorbereitung* umfasst Teile, die für SustainHub keine Bedeutung haben. Diese Schritte wurden entweder schon vor meiner Arbeit getätigt oder sind im wissenschaftlichen Kontext nicht relevant. Auf die Phase *Evaluation und Tests* wird ebenfalls verzichtet aufgrund der zeitlichen Begrenzung und der fehlenden Benutzerbeteiligung. *Überleitung in die Nutzung*, *Nutzung und Pflege* werden zwar teilweise durchgeführt, allerdings werden diese nicht in dieser Arbeit dokumentiert. Die Schritte, die im Zuge von SustainHub nicht stattfinden, sind ausgegraut in der Abbildung zu sehen. Alle Informationen bezüglich des Referenzmodells stammen aus dem Skript ”Usability Engineering” [Off12].

4.4.1 Anforderungsanalyse

In der Anforderungsanalyse werden Informationen erarbeitet, die für die Gestaltung von User Interfaces relevant sind. Dies können Eigenschaften, Merkmale, Randbedingungen und Vorlieben der späteren Nutzer, sowie ihre Aufgaben sein. Das Ziel der Anforderungsanalyse ist es, die wesentlichen Anforderungen an das System aus Usabilitysicht zu erhalten. Hierfür nehmen *Geschäfts- und Einsatzziele* einen großen Einfluss auf die Gestaltung. Aus ihnen lassen sich Randbedingungen und Ziele in Bezug auf die Aufgabenangemessenheit und Handhabbarkeit des User-Interfaces schlussfolgern. Unter der *Analyse Ist-Stand* versteht man die Untersuchung existierender Altsysteme oder Konkurrenzprodukte, um Informationen zu erhalten, wie man das User Interface verbessern kann. Da es im Fall SustainHub kein Konkurrenzprodukt und kein Altsystem existiert, kann dies nur für einzelne Teilprobleme getan werden. Anschließend erfolgen die Schritte *Benutzerprofilanalyse*, die *Aufgabenanalyse* und die *Umgebungsanalyse*. Diese drei Schritte werden oft nicht streng getrennt voneinander durchgeführt, weshalb man diese zusammengefasst auch *Kontextanalyse* nennt. In diesem Schritt werden die folgenden Kontextfragen geklärt:

- Wer sind die Nutzer der Software?

- Welche Aufgaben haben die Nutzer?
- Wann führt der Nutzer eine Aufgabe aus? (Auslöser)
- Warum führt der Nutzer diese Aufgabe aus? (Handlungsziel)
- Wie oft wird eine Aufgabe ausgeführt?
- Was ist beim Durchführen einer Aufgabe im Einzelnen zu tun?
- Welche zusätzlichen Mittel benötigt der Nutzer?
- Welche Wünsche hat der Nutzer im Bezug auf die Dialoggestaltung?
- Welche Ausnahmefälle/Sonderfälle gibt es?

Im nächsten Schritt folgt die Suche nach *generellen Entwurfsprinzipien*. Gemeint sind damit Gestaltungsprinzipien, die vor dem Projekt schon bestehen. Dies können Styleguides von Betriebssystemen und Frameworks oder auch Coporate Design Regeln, die eingehalten werden müssen, sein. Zuletzt erfolgt das Festlegen von *generellen Usability-Zielen*. Sich genau zu vergegewärtigen, welche Usability-Ziele angestrebt werden, führt zu einer Fokussierung der User Interface Gestaltung. Dadurch können Entwurfsentscheidungen einfacher diskutiert und getroffen werden und somit der Prozess beschleunigt werden. Usabilityziele können unterschiedlich festgelegt werden. Es gibt zwei Arten von Usabilityzielen:

Qualitative Ziele sind nicht direkt messbare, allgemeine Ziele. Eine solche Zielsetzung, z.B. durch die Gütekriterien, ist nützlich, um abstrakte Ideen zu diskutieren.

Quantitative Ziele sind messbar. Diese können weiter unterschieden werden in:

- **Ease-of-Use-Ziele** sind Geschwindigkeiten, Effizienz und Flexibilität, die in der Interaktion zwischen Experten und des Systems angestrebt werden.
- **Ease-of-Learning-Ziele** sind angestrebte Ziele von Anfängern mit dem System.

Aufgrund der fehlenden Benutzerbeteiligung können quantitative Ziele nicht mit Messungen abgeglichen werden und werden deshalb nicht definiert.

4.4.2 User-Interface-Entwurf

In der Phase des User-Interface-Entwurfs werden die Ergebnisse der Anforderungsanalyse in Skizzen, Low-fidelity Mockups und High-fidelity Mockups umgesetzt. In mehreren Iterationen werden diese daraufhin evaluiert, diskutiert und optimiert bis konkrete Entwürfe für Benutzeroberflächen entstehen. Die erarbeiteten Usabilityziele der Anforderungsanalyse müssen erfüllt sein. Der User-Interface-Entwurf beinhaltet die Prozessschritte Workflow Reengineering, Konzeptuelles UI-Modell, User-Interface-Mockups, Iterative UI-Walkthroughs, Elektronische UI-Prototypen, Iterative Usability-Tests, User-Interface-Styleguide sowie Detailentwurf. Da SustainHub ein neues System ist, ist ein *Workflow-Reengineering* nicht erforderlich. In der Phase des *konzeptuellen UI-Modells* wird die Struktur und der Aufbau der Oberfläche geklärt und entworfen. Daraus lassen sich Regeln für wiederkehrende Oberflächenelemente bestimmen. Dadurch wird beim Nutzer ein mentales Modell erzeugt. Dieses unterstützt den Nutzer die Dialogsysteme schneller zu erlernen und es effizienter und effektiver zu benutzen. Daraufhin werden alle Dialoge skizziert. In mehreren Iterationen werden diese Mockups evaluiert, präzisiert und optimiert in ständigem Vergleich mit den Anforderungen. In dieser Arbeit werden mithilfe der Wireframing-Software *Balsamiq Mockups* Low-fidelity Mokckups erstellt. Low-fidelity Mockups sind grobe Skizzen von UIs. Die schnelle und einfache Produktion dieser Mockups fördert die Fokussierung auf die wesentlichen Aspekte. Erlangen diese einen gewissen Reifegrad, werden elektronische Prototypen, sogenannte High-fidelity Mockups erstellt, die mit Hilfe von Photoshop CS5 so erzeugt werden, dass sie dem Endsystem sehr ähnlich sehen. Auch diese Prototypen unterziehen sich ständiger Iterationen, in denen Verbesserungen eingebaut werden. Im Anschluss daran werden die detailierten Design-Regeln des User-Interfaces in einem Styleguide manifestiert. Daraufhin werden alle Dialogbereiche und konkreten Dialogelemente im Detail definiert.

5

Anforderungsanalyse

In diesem Kapitel werden die Anforderungen an das System aus Usabilitysicht systematisch analysiert. Hierfür ist die Erarbeitung von *Geschäfts- und Einsatzielen* Grundvorraussetzung. Daraufhin werden in *Analyse Ist-Stand* Konkurrenzsysteme untersucht. Im Anschluss daran folgt eine *Benutzerprofilanalyse*. In der *Aufgabenanalyse* werden die Aufgaben gesammelt, kategorisiert und für diese Anforderungen gestellt. Ein Leitfaden bei der Erstellung sind dabei die *Kontextfragen*. Im Anschluss werden *Hardware- und Softwarerandbedingungen* und *Generelle Gestaltungsprinzipien* erarbeitet, um Einschränkungen rechtzeitig zu erkennen und bei späteren Entwürfen beachten zu können.

5.1 Geschäftsziele und Einsatzziele

Mithilfe von SustainHub werden Firmen innerhalb einer Lieferkette Nachhaltigkeitsdaten austauschen. Das Nachhaltigkeitsdatenmanagement soll damit verbessert werden. Eine der Hauptziele ist, dass es keine Einschränkungen im Hinblick auf Branchen und Unternehmen gibt. Jede Organisation, die Nachhaltigkeitsdaten austauschen will, kann SustainHub nutzen. Speziell für das Webinterface ist vorgesehen, dass Firmen ohne eine In-House-Solution darüber Nachhaltigkeitsdaten austauschen können. Die Benutzungsschnittstelle dient als ”digitaler Vermittler” von Nachhaltigkeitsdaten. Aus softwareergnomischer Sicht sollte somit berücksichtigt werden, dass das System für Nutzer aus allen Branchen verständlich und leicht bedienbar ist. Der Hauptfokus liegt beim Austausch von Nachhaltigkeitsdaten.

5.2 Analyse Ist-Stand

SustainHub ist ein neues Projekt, weshalb keine Altsysteme existieren. Allerdings existieren andere Informationssysteme, die ähnliche Dienste anbieten. Das *International Material Data System* (IMDS) ist ein System, mit dem sich Materialdaten für den Fahrzeugbau archivieren, verwalten und austauschen lassen. Das System BOMCheck ist eine Internet-Datenbank zur Stoffdeklaration. Diese unterstützen einen Datenaustausch von Nachhaltigkeitsdaten. Leider ist ein Zugriff ohne authentifizierten Account auf diese Systeme nicht erlaubt. Diese werden nur an Firmen vergeben. Eine Analyse des User-Interfaces ist somit nicht möglich.

5.3 Benutzerprofilanalyse

Einige Gestaltungsentscheidungen lassen sich nur mit Kenntnis über den Nutzer treffen, da Nutzer mit verschiedenen Hintergründen unterschiedlich mit Benutzeroberflächen interagieren. Deshalb lohnt sich eine Auseinandersetzung mit dem Nutzer. Im SustainHub-Projekt war keine Auseinandersetzung mit realen Benutzern möglich, dennoch lassen

sich Rückschlüsse auf den Nutzer ziehen. Anstatt verschiedene Benutzergruppen und für diese Gütekriterien zu identifizieren, folgt ein Versuch der Priorisierung der Gütekriterien für die Benutzerschnittstellen insgesamt, um so die Anforderungen aus ergonomischer Sicht an die Benutzeroberfläche zu definieren. Die Web-Interfaces von SustainHub werden von Mitarbeitern in kleineren, mittleren und großen Unternehmen benutzt, um Nachhaltigkeitsdaten auszutauschen. Es handelt sich um Personen, die in ihren Betrieben für die Nachhaltigkeit zuständig sind. In größeren Unternehmen handelt es sich um auf fachlich spezialisierte Nachhaltigkeitsexperten, in kleineren Unternehmen zumindest um Personen, die eine Nachhaltigkeitskompetenz aufweisen. Nachhaltigkeitsmanagement ist auf strategischer Ebene einer Organisation anzusiedeln, was auf eine priveligiertere Ausbildung und Position der Person schließen lässt. Es handelt sich um einen Expertenarbeitsplatz. Der Nutzer benutzt täglich bei der Arbeit einen Computer und ist somit mit grafischen Benutzeroberflächen vertraut. Es sind erfahrene Computernutzer, allerdings handelt es sich nicht um Computerexperten. Die Weboberfläche wird in zwei unterteilt, dem Participant Interface und dem Administration Interface. Folgende detailierte Benutzerprofilanalyse zeigt den Unterschied der Anforderungen aus softwareergonomischer Sicht an das User-Interface.

5.3.1 Normale Nutzer

Die meisten Nutzer lassen sich hier in die Benutzerkategorie **geübter und sporadischer Benutzer** einordnen. Daraus ergibt sich die Rangfolge und Gewichtung der Gütekriterien dieser Benutzerkategorie in der Tabelle [Off12]. Wie oft das System benutzt wird, lässt sich nicht pauschalisieren. Die meisten Nutzer benutzen das System nicht regelmäßig. Eine Schulung am System findet nicht statt. Mit einer technischen Sprache und technischen Details sollte der Nutzer somit nicht konfrontiert werden.

Nr.	Priortät	Gütekriterium
1.	Hoch	Kooperations- und Kommunikationsförderlichkeit
2.	Hoch	Datenschutz und Datensicherheit
3.	Hoch	Nützlichkeit
4.	Hoch	Komfort
5.	Hoch	Erwartungskonformität
6.	Hoch	Selbstbeschreibungsfähigkeit
7.	Hoch	Lernförderlichkeit
8.	Mittel	Fehlerrobustheit
9.	Mittel	Übersichtlichkeit
10.	Gering	Steuerbarkeit
11.	Gering	Individualierbarkeit
12.	Gering	Verfügbarkeit

5.3.2 Administratoren

Nr.	Priortät	Gütekriterium
1.	Hoch	Kooperations- und Kommunikationsförderlichkeit
2.	Hoch	Datenschutz und Datensicherheit
3.	Hoch	Verfügbarkeit
4.	Hoch	Nützlichkeit
5.	Hoch	Komfort
6.	Hoch	Steuerbarkeit
7.	Hoch	Erwartungskonformität
8.	Mittel	Individualierbarkeit
9.	Mittel	Selbstbeschreibungsfähigkeit
10.	Mittel	Lernförderlichkeit
11.	Gering	Übersichtlichkeit
12.	Gering	Fehlerrobustheit

Dieser Benutzertyp ähnelt stark dem vorher beschriebenen. Der wesentliche Unterschied ist, dass er mehr Zeit am System arbeitet. Die Kategorisierung *geübter und regelmäßiger Benutzer* trifft allerdings auch nicht zu, da mit *regelmäßig* gemeint ist, dass sich der

Nutzer mehrere Stunden am Tag mit dem System beschäftigt, wovon nicht auszugehen ist. Dies verlangt einen leichteren Einstieg in das System, weshalb die Erlernbarkeit höher eingestuft wird. Gleiches gilt für die Erwartungskonformität. Die Priorität der Individualisierung wird abgeschwächt, da der Nutzer nicht so viel Zeit an dem System verbringt, dass er personalisierte Oberflächen benötigt. Die Tabelle zeigt die Rangfolge und Gewichtung der softwareergonomischen Gütekriterien basierend auf denen des *geübten und regelmäßigen Benutzers* mit diesen Änderungen [Off12].

5.3.3 Fazit

Trotz der vielen Gemeinsamkeiten der zwei Benutzergruppen, variieren einige Nutzereigenschaften. Sowohl die Regelmäßigkeit der Systemnutzung als auch das Vorwissen der Nutzer unterscheidet sich. Somit entstehen auch unterschiedliche Anforderungen an das Participant Interface und das Administration Interface. Das System sollte sowohl einen einfachen Einstieg für Novizen ermöglichen und gleichzeitig Experten, vor allem im Administration Interface, effizientes Arbeiten ermöglichen. An wichtigen Funktionen des Systems ist es sinnvoll verschiedene Dialoge anzubieten: einen für Experten, einen für Novizen. Die definierten Prioritäten der Gütekriterien sollten somit als Orientierung dienen.

5.4 Aufgabenanalyse des Participant Interfaces

In diesem Kapitel werden systematisch alle Use-Cases des Participant Interfaces genannt und beschrieben.

5.4.1 Global erreichbare Funktionen

Einige Werkzeuge, Funktionen und Informationen sollten von jeder Stelle im System aus erreichbar sein. Der Nutzer sollte wissen, dass und wo er sich auf SustainHub befindet. Der Nutzer sollte bei der Suche nach Informationen unterstützt werden und die angebotenen Funktionen sollten vorgestellt werden. Hierfür eignet sich eine Navigation. Für

eine gezielte Suche sollte eine Suchfunktion angeboten werden. Zudem sollte angezeigt werden, ob und unter welchem Account der Nutzer angemeldet ist. Die Hauptaufgaben des SustainHubs Projekt sind Requests erstellen und Daten bereitzustellen. Um diese möglichst schnell zu erreichen, sollten Deeplinks, Verknüpfungen von Unterseiten, die in der Navigationshierarchie unten sind, zu diesen Aufgaben, erstellt werden. Dem Nutzer sollte es erspart bleiben jedes Mal nachzuschauen, ob es Änderungen bei den Dataexchanges gab. Eine Funktion, die ihn darauf aufmerksam macht, sollte miteinbezogen werden.

5.4.2 Administration und System

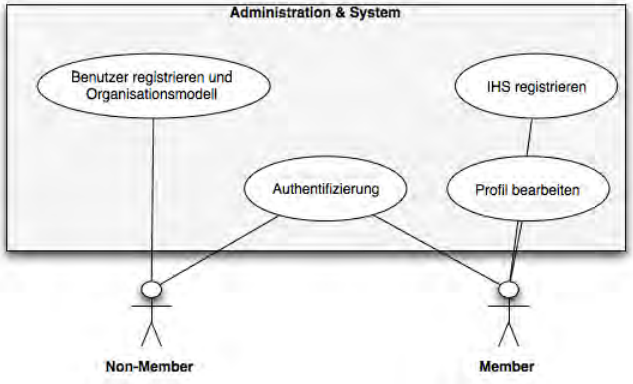


Abbildung 5.1: Use-Case-Diagramm: Administration und System

Um einen Überblick über den Umfang zu erhalten, werden im folgenden die Aufgaben, die die Administration und das System betreffen, zuerst genannt und daraufhin konkretisiert. In der Softwarearchitektur sind diese in der Komponente *Profile Management* anzusiedeln. Ziel ist es die Aufgaben vollständig zu erfassen und Prioritäten zu definieren. Die Use-Cases sind teilweise aus projektinternen Dokumenten entnommen und wurden durch weitere Aufgaben ergänzt.

Registrierung: Nutzer registrieren und Organisationsmodell

Bevor ein Nicht-Mitglied auf den vollen Funktionsumfang von SustainHub zugreifen kann, muss er sich und seine Organisation registrieren. Diese Bewerbung muss von einem System-Administrator überprüft werden, damit sichergestellt wird, dass Nutzer nur ihre eigene Organisation anmelden. Der Bewerber wird über das Ergebnis des Genehmigungsprozess via Mail informiert. Wenn die Bewerbung genehmigt wird, wird ein Nutzer angelegt, der nun als System-Administrator agiert. Dieser kann die Organisationsstruktur seines Unternehmens spezifizieren und somit Verantwortlichkeiten innerhalb von SustainHub zuweisen.

Aus Datenmodell-Sicht ist die ”Participant Data”-Entität betroffen (vgl. Abbildung 2.2). Das System muss ein Formular für diese Daten anbieten. Dabei sollte beachtet werden, dass es sich um den ersten Dialog mit dem System handelt und der Nutzer ohne Vorkenntnisse agiert.

Authentifizierung

Der Nutzer muss sich authentifizieren. Hierfür muss das System ein Anmeldeformular bereitstellen, wenn der Nutzer nicht eingeloggt ist. Ist er eingeloggt, soll ihm sein Name angezeigt werden und die Möglichkeit angeboten werden, die Login-Session zu beenden.

Profil bearbeiten

Neben der Manipulierung der Daten, die schon bei der Registrierung relevant sind, soll hier auch die Möglichkeit bestehen die Benutzeroberfläche zu individualisieren. Hier kann eingestellt werden, ob der Nutzer anstatt des Request-Wizards das Expertenformular nutzen will und wieviele Elemente ihm pro Seite von einer Tabelle oder eines Suchergebnisses angezeigt werden.

5.4.3 Datenaustausch

In diesem Kapitel werden die softwareergonomischen Anforderungen an das System im Bezug auf das Austauschen von Daten vorgestellt. In Abbildung 5.2 werden alle Use Cases aufgelistet, die das Austauschen von Daten betrifft. Mit Datenaustausch werden zwei unterschiedliche Abläufe zusammengefasst: Er kann mit einem *Request* eingeleitet werden und daraufhin via *Response* beantwortet werden oder über *Provide Data* in SustainHub abgespeichert werden und von berechtigten Mitgliedern eingesehen werden.

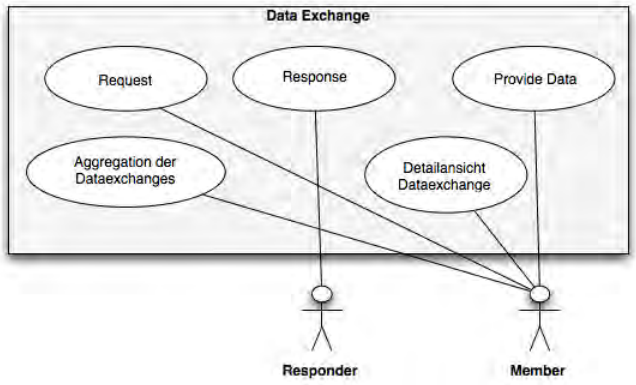


Abbildung 5.2: Use-Case-Diagramm: Data Exchange

Request

Ein Request ist eine Anfrage, die man als Mitglied an eine Organisationen stellen kann, um Nachhaltigkeitsdaten von diesen zu erhalten. Um möglichst präzise Antworten zu erhalten, werden neben der Empfangsorganisation und dem Indikator, Qualität der Daten, die zeitliche Einordnung und der Geltungsbereich definiert. Aus Datenmodell-Sicht muss die ”SustainHub Request”-Entität ausgefüllt werden (vgl. Abbildung 2.2). Ein Wizard sollte Novizen durch diesen Prozess leiten. Experten sollte es ermöglicht sein, effizienter zu arbeiten. Das User-Interface sollte Fehleingaben minimieren. Sehr häufig müssen gleiche Requests an verschiedene Organisationen verschickt werden. Außerdem muss

häufig direkt nach einem Request ein ähnlicher Request, der sich nur in einzelnen Punkten vom vorherigen unterscheidet, direkt danach gestellt werden.

Response

Ein Response ist eine Antwort auf einen Request. Hier müssen die angefragten Daten und die dazugehörigen Metadaten eingetragen werden. Aus Datenmodell-Sicht muss die "SustainHub Response"-Entität ausgefüllt werden (vgl. Abbildung 2.2). Um Falscheingaben zu minimieren, sollte der Nutzer jederzeit vergleichen können, ob seine Daten den angefragten Daten entsprechen.

Aggregation der Datenaustausche

Um auf Requests zu antworten, bedarf es einer Übersicht, in der die eingehenden Requests angezeigt werden. In dieser werden sowohl die Requests als auch die Responses, die den Nutzer bzw. seine Organisation betreffen, angezeigt. Ziel ist es, aus einer Reihe von Datenaustausche einzelne schnell vergleichen zu können und das gewünschte auszuwählen. Die Requests unterscheiden sich häufig nur in Details. Deshalb sollte die Übersicht nach verschiedenen Kriterien sortierbar sein und eine Suchfunktion soll einen schnellen Zugriff gewährleisten.

Ein Request kann in den folgenden Status sein, die ebenfalls in der Aggregation angezeigt werden sollten:

- Open
- Processing
- Partly finished
- Finished

Die Übersicht sollte zudem einen Verweis auf eine Detailansicht der einzelnen Requests und im Falle eines eingehenden Requests den Response-Wizard anbieten. Requests, bei denen Änderungen, z.B. den Status, durch andere Parteien getätigt wurden, sollten

hervorgehoben sein. Bestimmte Datenaustausche können von höherer Priorität sein und sollten unter den anderen schnell gefunden werden.

Detailansicht eines Datenaustauschs

Ein Datenaustausch muss dem Nutzer detailiert angezeigt werden. Je nach Status liegen verschiedene Daten vor. Es muss deutlich werden, welche Daten von wem stammen und in welchem Status sich der Datenaustausch befindet. Hier sollten auch auf Werkzeuge wie das Verändern, Löschen oder Erstellen eines ähnlichen Requests verwiesen werden. Von hier aus sollte der Nutzer schnell auf verwandte Requests zugreifen können. Sinnvolle Assoziationen wären andere Datenaustausche mit den gleichen Parteien und den gleichen Indikatoren.

Provide Data

Möchte ein Nutzer Nachhaltigkeitsdaten über die SustainHub-Plattform bereitstellen, muss ein Formular bereitgestellt werden, mithilfe dessen die Daten und Metadaten eingetragen werden können. Aus Datenmodell-Sicht muss die ”SustainHub Push Data”-Entität ausgefüllt werden (vgl. Abbildung 2.2).

5.4.4 Kontakte

In diesem Kapitel werden die Anforderungen an das System im Hinblick auf die Kontakte untersucht. Unter Kontakten werden andere SustainHub Mitglieder verstanden. Neben der Anzeige aller SustainHub Mitglieder sollte eine detailierte Ansicht eines Mitglieds und einer Organisation bestehen. Zudem sollte man sich mit Mitgliedern verbinden können. Alle Use-Cases werden in Abbildung 5.3 angezeigt.

Aggregation der Kontakte

Kontakte müssen in einer listenähnlichen Struktur angezeigt werden. Kontakte, mit denen der Nutzer bereits verbunden ist, sollten darin angezeigt werden können, aber

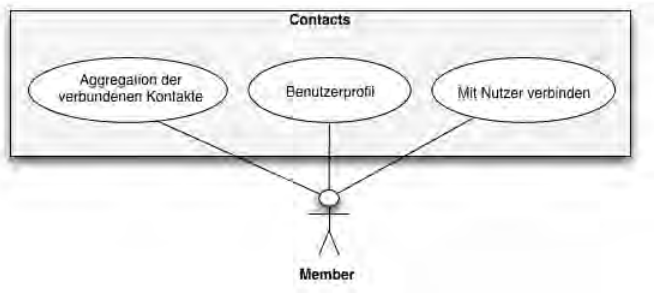


Abbildung 5.3: Use-Case-Diagramm: Kontakte

auch alle SustainHub Mitglieder. Ziel ist es einen gewünschten Kontakt aus diese Liste auszuwählen. Mitglieder lassen sich am Besten über ihre Namen, den Namen ihrer Organisation und ihrer Funktion in der Lieferkette im Bezug auf den eigenen Nutzer, identifizieren. Um dies zu gewährleisten, sollte die Liste nach diesen Kriterien sortierbar sein. Eine Suchfunktion soll den Nutzer bei einer gezielten Suche unterstützenn. Details von Mitgliedern sollten anzeigt werden ohne die Seite neu zu laden, um einen schnellen Vergleichen von ähnlichen Mitgliedern zu garantieren. Hier sollten die Informationen aus dem Benutzerprofil dienen.

Benutzerprofil

Eine Detailansicht soll Details über einen Nutzer ausgeben. Durch Kategorisierung und Gruppierung soll es trotz der großen Menge einzelne Informationen schnell gefunden werden. Neben der Anzeige der Daten des SustainHub-Mitglieds, sollten auch Informationen zu der Verbindung zwischen den beiden Nutzern und zu der Organisation des Mitglieds angezeigt werden. Um Datenaustausche über Kontakte zu finden, müssen hier diese angezeigt werden, die mit diesem Mitglied erfolgt sind. Ein Verweis auf die Organisationsansicht sollte ebenfalls angeboten werden. Der Nutzer kann über einen Button auf das *Mit Nutzer verbinden*-Formular zugreifen. Zudem sollte angezeigt werden, ob die Nutzer verbunden sind und, sollte dies der Fall sein, eine entsprechende Anzeige für ”Level of Trust” und ”Rating”. Es handelt sich um Bewertungen von Organisationen z.B. ihre Vertrauenswürdigkeit, die andere definieren können. Aus Datenmodellsicht

sind somit die Entität SustainHub Participant und SustainHub Customer Relationship betroffen (Vgl. Abbildung 2.2).

Aggregation der Organisationen

Die Aggregation der Organisationen sollte ähnlich wie die der Kontakte angezeigt werden. Diese lassen sich neben des Namens der Organisation auch über die Branche, das Land und den Ort identifizieren. Auch hier bietet sich eine kurze Übersicht zum schnellen Vergleich von Organisationen an. So können zum Beispiel verschiedene Tochterfirmen mit ähnlichen Namen unterschieden werden.

Organisationsansicht

In der Organisationsansicht sollten die Informationen, die zur Firma existieren, angezeigt werden. Dazu die Nutzer, die für diese Organisationen tätig sind. Um Datenaustausche über die Organisationen zu finden, müssen hier die entsprechenden Austausche angezeigt werden, die mit dieser Organisation erfolgt sind. Zudem bietet es sich an, die Produkte aufzulisten, an der diese Organisation beteiligt ist.

Mit Nutzer verbinden

Aus Datenmodellsicht ist die Entität ”Participant Relationship” betroffen (Vgl. Abbildung 2.2). Diese Daten sollten ausgefüllt werden können.

5.4.5 Dashboard

Obwohl die Anzahl der Aufgaben hoch ist, lassen sich einige Aufgaben identifizieren, die sehr oft benutzt werden. Um einen schnellen Einstieg für erfahrene Nutzer zu bieten und somit aufwändiges Navigieren durch die Hierarchie zu verhindern, dient als Startseite für eingeloggte Nutzer ein Dashboard. Besonders der Datenaustausch, die Verbindung mit

anderen Nutzern und die Regulations sind wichtig. Somit werden folgende Informationen auf dem Dashboard plaziert:

1. Die 5 aktuellsten Dataexchanges
2. Die neusten 5 verbundenen Kontakte
3. Die 5 aktuellsten Regulationen

5.5 Aufgabenanalyse des Administration Interfaces

Im Gegensatz zum Participant Interface liegt beim Administration Interface der Fokus nicht auf der einfachen Erlernbarkeit, sondern auf einer schnellen und komfortablen Bedienung. Abbildung 5.4 zeigt die Use-Cases im Bezug auf das Administration Interface an.

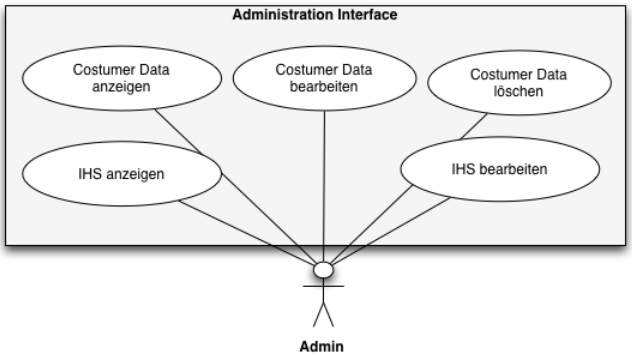


Abbildung 5.4: Use-Case-Diagramm für das Administration Interface

5.5.1 Global erreichbare Funktionen

Wie in Kapitel 5.3 erarbeitet, haben die Nutzer des Administration Interfaces unterschiedliche Anforderungen an das System als Nutzer des Participant Interfaces. Daraus folgend müssten auch die global erreichbaren Funktionen andere sein. Da sich diese allerdings nur minimal von denen der Partizipanten unterscheiden, ist der Aufwand einen neuen

Seitenaufbau zu realisieren unverhältnismäßig groß zum Nutzen und es wird somit der Aufbau des *Participant Interface* genutzt.

Costumer Data

Alle Costumer Daten müssen erstellt, angezeigt, bearbeitet und gelöscht werden können. Folgende Entitäten sind betroffen:

- Product
- Component
- Material
- BoM

IHS Konfiguration

Im Administration Interface müssen die IHS-Daten aus der Entität *External System* angezeigt werden und manipuliert werden.

5.6 Hardware- und Softwarerandbedingungen

In diesem Kapitel folgt die Analyse der Hardware- und Softwarerandbedingungen.

5.6.1 Hardwarerandbedingung

Um ergonomische Anforderungen festzustellen, bedarf es auch einer Analyse der Hardware des Nutzers. Mit welcher Hardware der Nutzer auf das Webinterface von SustainHub zugreift, lässt sich nicht bestimmen. Allerdings ist das Ziel möglichst niedrige Systemanforderungen an die Hard- und Software zu stellen, damit möglichst vielen Nutzern die Nutzung ermöglicht wird. Da SustainHub eine Arbeitssoftware ist, werden sich die meisten Nutzer in einem Büro oder einem ähnlichen Ort befinden und mit einem

Computer, der über einen Monitor, eine Tastatur und eine Maus oder Mousepad verfügt, arbeiten.

Eine wichtige Rolle spielt die **Bildschirmauflösung**. Aus der w3school Bildschirmauflösungsstatistik [Ref13b], die in Abbildung 5.5 zu sehen ist, lässt sich ableiten, dass rund 99 % der Internetnutzer die Auflösung 1024x768 oder höher benutzt, welche sich somit als Systemanforderung eignet.

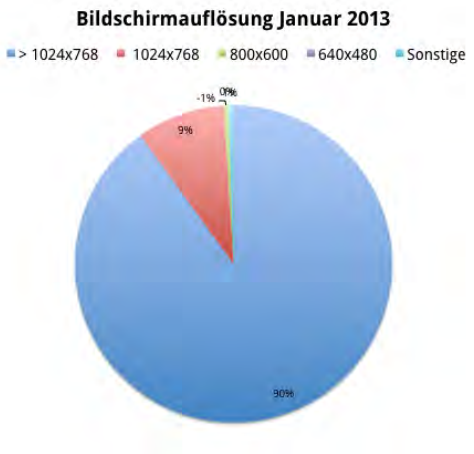


Abbildung 5.5: Verteilung von Bildschirmauflösungen der Monitore im Januar 2013 laut w3school

5.6.2 Softwarerandbedingungen

Da SustainHub als Webapplikation realisiert wird, sind keine Anforderungen an das Betriebssystem vorhanden. Laut einer Statistik der w3school [Ref13a] benutzen über 80 % der Internetnutzer einen Windowsrechner. Gestaltprinzpien aus Windows sind somit unter den meisten Nutzern bekannt und können somit übernommen werden. Außerdem kann davon ausgegangen werden, dass nur ein vernachlässigbarer Anteil der Rechner die Mindestvorraussetzungen unterschreiten wird. Eine Softwarerandbedingung ist ein mit Vaadin kompatibler Browser, in dem Javascript aktiviert ist. Abbildung 5.6 zeigt die Verteilung der Browser im August 2013 aller Internetnutzer. Vaadin ist mit allen darin genannten Browsern kompatibel. Zu den restlichen 1,4 % der Browser weltweit lassen

sich keine Aussage treffen bezüglich der Kompatibilität mit Vaadin. Da unter sonstigen Browsern mobile Browser ebenfalls enthalten sind und diese für SustainHub keine Rolle spielen und andere Browser auch Vaadin unterstützen können, wird erwartet, dass über 99 % aller Internetnutzer mit ihrem Browser SustainHub nutzen können [Vaa12].

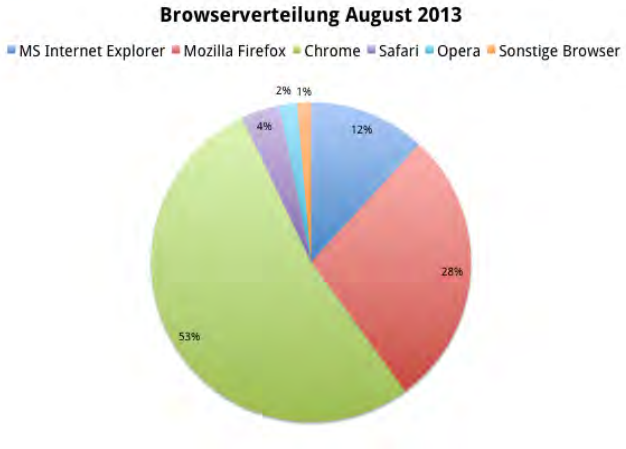


Abbildung 5.6: Browserverteilung im August 2013 laut w3school

Zur Anzahl der Nutzer, die Javascript deaktiviert haben, gibt es keine seriöse Statistik. Allerdings wird dieser Teil unter 5% geschätzt. Für die Entwicklung der Webapplikation bedeutet dies optimale Vorraussetzungen. Es müssen kaum Randbedingungen eingehalten werden.

Auch im Hinblick auf Vaadin lassen sich kaum Einschränkungen finden. Trotz der vielen Vorgaben lassen sich eigene Komponenten erstellen und bereits existierende Komponenten modifizieren. Allerdings muss beachtet werden, dass Vaadin keine Scrollevents unterstützt und sich diese schwer realisieren lassen.

5.7 Generelle Entwurfsprinzipien

In diesem Kapitel werden die bereits existierenden Gestaltungsrichtlinien erarbeitet, die als harte Randbedingungen für die Gestaltung der Benutzeroberfläche gelten. Um ein einheitliches und positives Bild des Projektes nach außen zu tragen und durch einen

hohen Wiedererkennungswert den Bekanntheitsgrad zu steigern, sollte das gesamte Erscheinungsbild von SusainHub konsistent sein. SustainHub als Projekt besitzt bereits ein solches Coporate Design. Die wesentlichen Aspekte dessen sind die Wort-/Bildmarke und das Farbkonzept. In Abbildung 5.7 ist das Logo inklusive aller Hauptfarben des Projektes vertreten.

Abbildung 5.7: Farbkonzept anhand SustainHub Logo

6

Usabilityentwurf

In diesem Kapitel sind die Entwürfe, die im Zuge der Entwicklung des Usability-Konzepts entstanden sind, dokumentiert. Nach dem konzeptuellen Usabilityentwurf in Kapitel 6.1 folgen die entsprechenden Entwürfe zu den einzelnen Aufgaben. Diese folgen der Reihenfolge aus der Anforderungsanalyse, wobei einige Aufgaben, die kaum Unterschiede zu anderen Aufgaben aufweisen, ausgelassen werden, um Wiederholungen zu meiden.

6.1 Konzeptueller Usabilityentwurf

Im System gibt es Teilprobleme, die an mehreren Stellen auftauchen. Damit ein mentales Modell entsteht, müssen diese konsistent gelöst werden. Hierfür müssen entsprechende Dialoge identifiziert und Regeln erstellt werden. Diese haben sich nach den Entwür-

fen herauskristallisiert und werden hier vorweg beschrieben, um Wiederholungen zu vermeiden.

6.1.1 Detailseite

Auf einer Detailseite (vgl. Abbildung 6.1) werden Daten zu einzelnen Objekten angezeigt, indem links angezeigt wird, um was für eine Information es sich handelt und rechts der Inhalt. Mehrere Informationen können gruppiert werden durch einen Panel. Zudem können ähnliche oder verwandte Objekte angezeigt werden. Hierfür wird eine Tabelle angezeigt. Unter den Panels können Buttons plaziert sein. Detailseiten tauchen auch am Ende eines Wizards zur Datenverifikation auf.

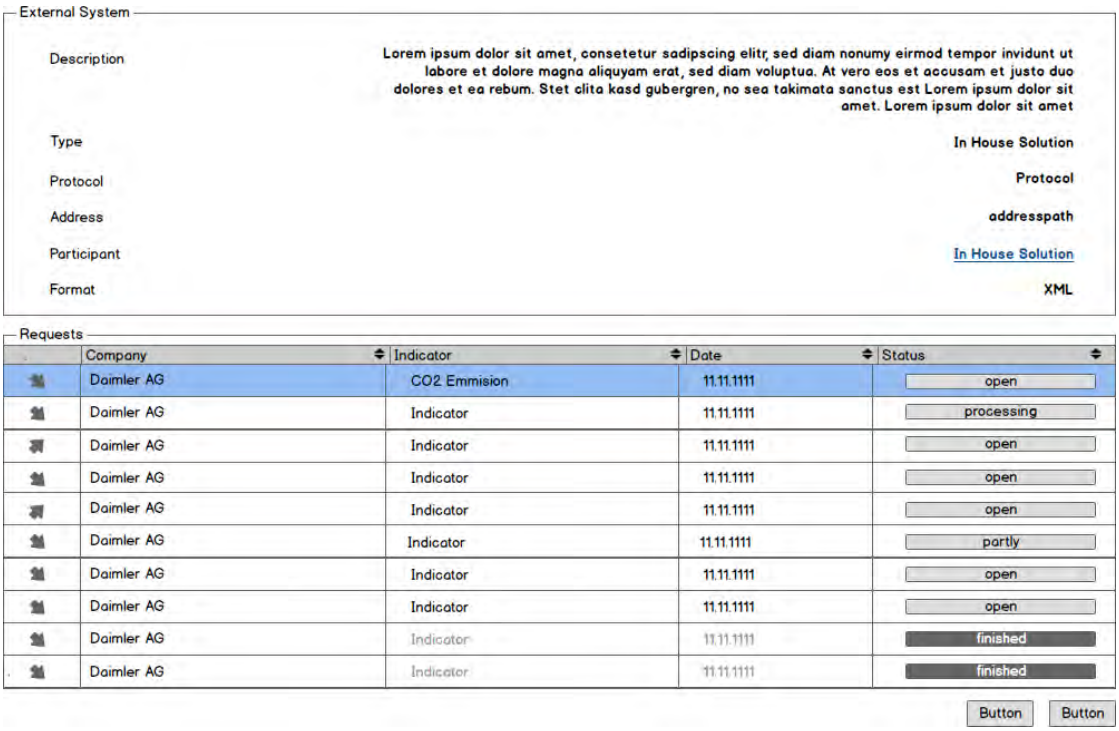


Abbildung 6.1: Low-fidelity-Mockup einer Detailseite

6.1.2 Wizards

Die Aufgaben *Registrierung*, *Request*, *Response* und *Provide Data* sind umfangreiche und komplexe Eingabeprozesse. Es bedarf einer Möglichkeit übersichtlich und strukturiert Informationen vom Nutzer abzufragen. Hierfür eignen sich *Wizards*. Große Formulare werden hierbei in mehrere Teilschritte aufgeteilt. Wenn eine Aufgabe in Einzelschritte aufgeteilt wird, wird eine Anzeige benötigt, die dem Nutzer den Fortschritt innerhalb dieses Tasks anzeigt. Außerdem sollte eine Speicherfunktion angeboten werden. Wird der Prozess abgebrochen, können teilweise ausgefüllte Formulare somit zu einem späteren Zeitpunkt komplettiert werden. Der Nutzer muss zudem durch durch den Wizard navigieren können und Eingaben aus vorherigen Teilschritten ändern können.

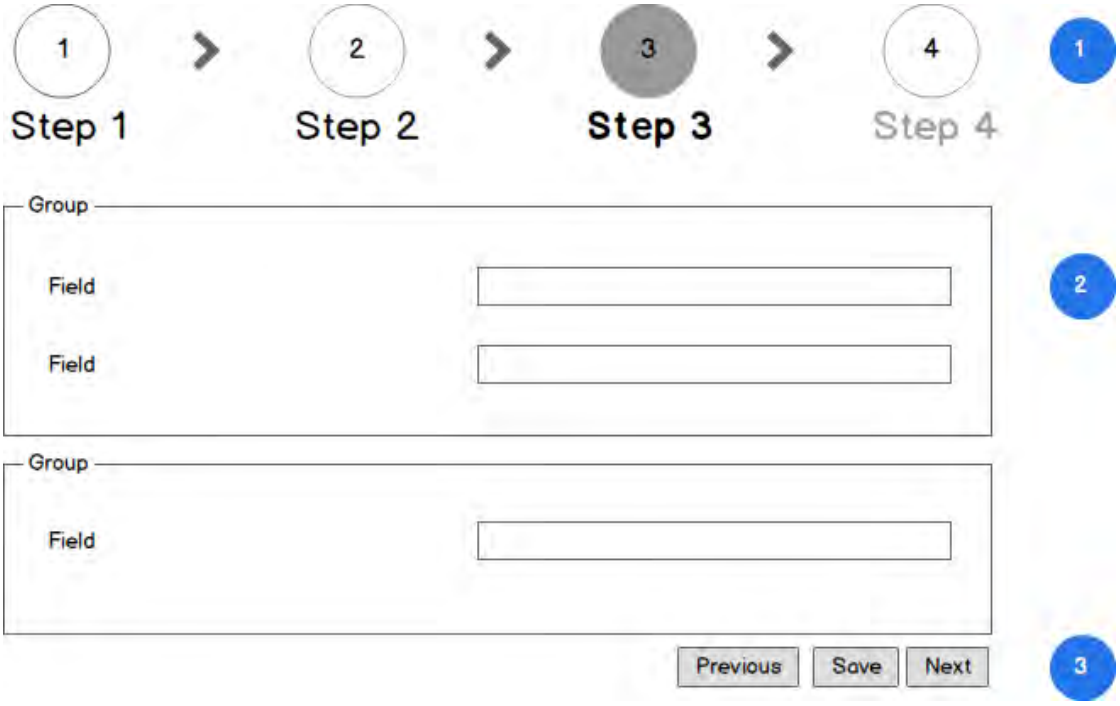


Abbildung 6.2: Low-fidelity-Mockup eines Wizards

In Abbildung 6.2 ist ein hypothetischer Wizard skizziert:

1. Der Fortschrittsindikator zeigt die Unterschritte an. Außerdem wird angezeigt, in welchem Unterschritt sich der Nutzer befindet. Da die Formulare in den Unter-

schritten thematisch zusammenhängend sein sollen, wird diesen Unterschritten ein Name zugewiesen.

2. In diesem Bereich befinden sich die gruppierten Formularfelder. Es können mehrere Gruppen auf einer Seite abgefragt werden.

3. Hier sind die Bedienelemente platziert. Mithilfe des ”Previous”- und des ”Next”-Buttons kann die Richtung innerhalb des Wizards gesteuert werden. Mit dem ”Save”-Button können bisherige Daten gespeichert werden ohne das Formular abzusenden.

Im Anschluss des Wizards werden dem Nutzer zur Verifizierung auf einer *Detailseite* die eingegeben Daten angezeigt, die er bestätigen kann.

6.1.3 Tabellen

Mit Tabellen können eine große Anzahl von Objekten angezeigt werden. Mithilfe von einigen wenigen Eigenschaften lassen sie sich identifizieren. Wird der Spaltenkopf angeklickt, können die Elemente nach dieser Eigenschaften sortiert werden.

Aggregationsseiten

Auf Aggregationsseiten werden Tabellen, um einige Funktionen erweitert. Mithilfe von Buttons über der Tabelle lassen sich die Tabelleninhalte nach bestimmten Eigenschaften selektieren. Wird ein Objekt ausgewählt, erscheint rechts eine Übersicht des einzelnen Objekts. Der Nutzer kann somit mehrere Objekte schnell vergleichen (vgl. Abbildung 6.11). Auf der rechten Seite werden die wichtigsten Informationen ausgegeben. Daneben ist auch ein Button, der zur Detailansicht führt, plaziert.

6.1.4 Formularkomponenten

Abbildung 6.3 zeigt, welche Fomularkomponenten für die jeweiligen Datentypen benutzt werden.

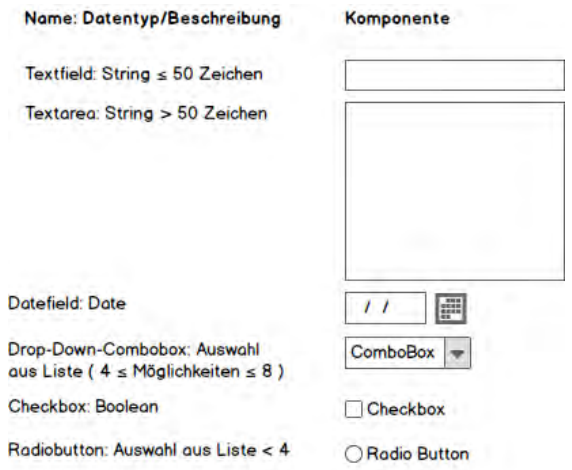


Abbildung 6.3: Low-fidelity Mockup für Formularkomponenten

6.1.5 Tokenfield

An einigen Stellen müssen eine oder mehrere Punkte aus einer Liste ausgewählt werden z.B. bei der Auswahl der Lieferanten bei einem Request oder dem Organisationsmodel. Hierfür werden häufig Auswahllisten oder Listen, vor deren Elementen Checkboxen angebracht sind, benutzt.

Im Falle von SustainHub müssen häufig mehrere Elemente aus einer sehr großen Liste ausgewählt werden. Mit Auswahllisten oder Checkboxlisten muss sich der Nutzer aufwendig durch lange Listen scrollen. Außerdem geht sehr schnell der Überblick verloren, welche Elemente bereits ausgewählt worden sind.

Eine komfortablere Alternative bieten sogenannte *Tokenfields*. Mithilfe des Low-fidelity Mockup Walkthrough in 6.4 werden die verschiedenen Funktionen dieser Tokenfields beschrieben.

1. Anfangs wird ein Textfeld angezeigt. Ein Plusicon rechts im Feld dient als Affordance und impliziert somit ein solches Tokenfield. Eine Affordance beschreibt eine einfache Auffindbarkeit von möglichen Aktionen, die eine UI-Komponente anbietet.

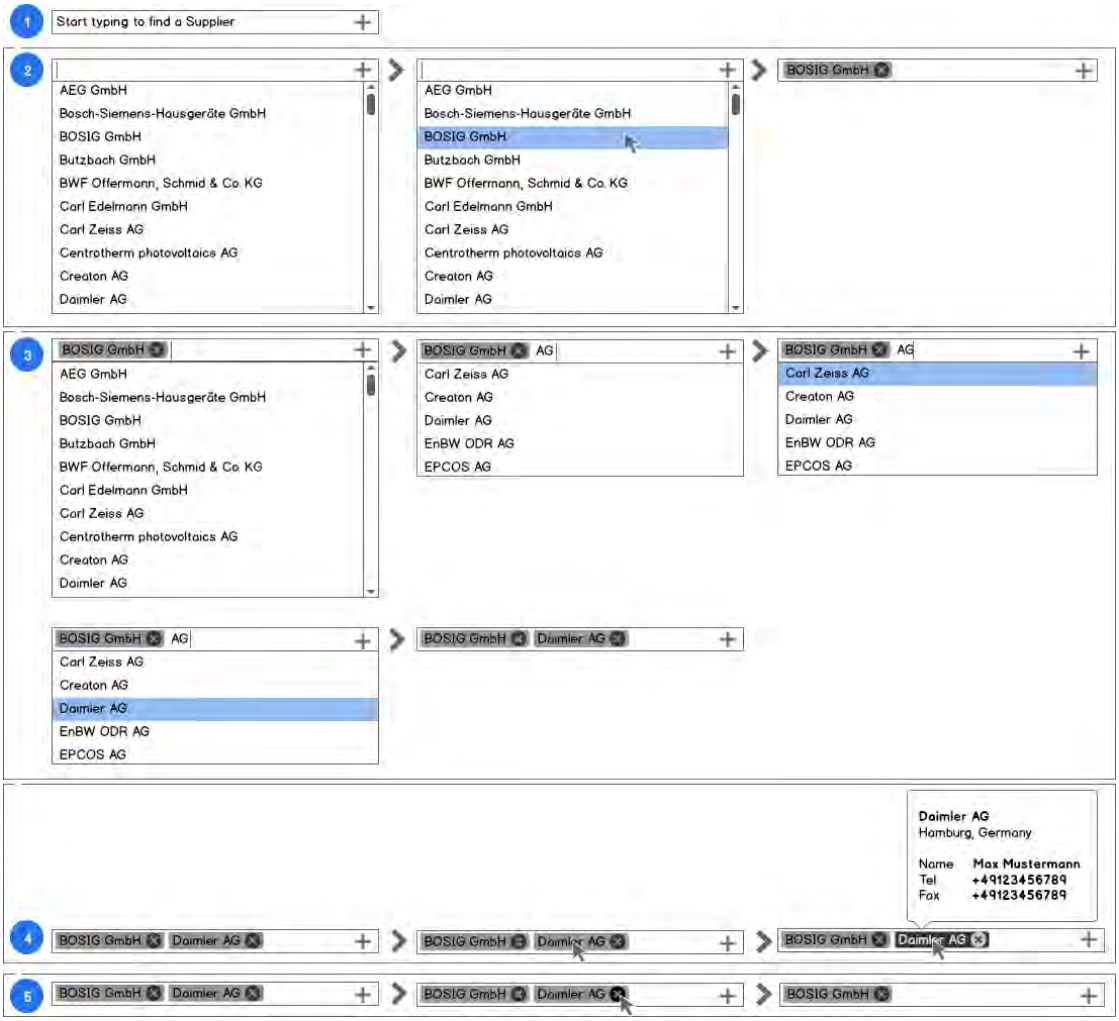


Abbildung 6.4: Low-fidelity Mockup Walkthrough für Tokenfield

2. Fokusiert der Nutzer das Feld mit einem Klick, wird die Liste der auswählbaren Elemente angezeigt. Nun kann der Nutzer sich durch die Liste scrollen und ein Element auswählen. Die Auswahl wird angezeigt, indem das Element als grauer Tag in dem Suchfeld angezeigt wird.
3. Zudem besteht die Möglichkeit via Texteingabe die Liste einzuschränken. In diesem Fall gibt der Nutzer ”AG” ein. Es erscheinen alle Elemente, die ”AG” enthalten. Mithilfe der Pfeiltasten kann das gewünschte Element fokusiert werden und mit Enter ausgewählt werden. Dem Nutzer wird somit eine vollständige Eingabe erspart.
4. Ist der Nutzer sich bei seiner Eingabe nicht sicher, kann er sich Informationen zu diesem Element anzeigen lassen, indem er auf den Tag klickt. Es erscheint ein Tooltip mit relevanten Daten.
5. Fehleingaben können leicht korrigiert werden. Alle Tokens besitzen ein Kreuzicon, mit dem Token aus der Auswahl gelöscht werden können.

6.2 Participant Interface

In diesem Kapitel werden die Entwürfe vorgestellt, die speziell das Participant Interface betreffen.

6.2.1 Aufgabenhierarchie

In diesem Kapitel wird untersucht, wie sich die Aufgaben gruppieren und ordnen lassen. Die folgende Statistik stellt somit die Grundlage für die Navigation dar.

1. Dashboard
2. Exchange
    a) **All Requests**: Aggregation der Datenaustausche
        i. **Response**: Response
    b) **New Request**: New Request

   c) **Provide Data**: Provide Data

3. Contacts
   a) **Contacts**: Aggregation der Kontakte
      i. **Contact**: Benutzerprofil
   b) **Organisations**: Aggregation der Organisationen
      i. **Organisation**: Organisationsprofil
4. Data
   a) **Costumer Data**
5. Benutzermenü
   a) **My profile**: Benutzerprofil
   b) **My Organization**: Organisationsmodell
   c) **Settings**: Settings
   d) **Logout**: Logout

6.2.2 Aufbau und Struktur der Benutzeroberfläche

Alle Dialoge in SustainHub sind einheitlich aufgebaut. Grundsätzlich besteht die Benutzeroberfläche aus drei Teilen, wie Abbildung 6.5 zeigt:

1. Header
2. Inhalt
3. Footer

Header Hier sind alle Elemente, die laut 5.4.1 überall verfügbar sein sollen, untergebracht. Auch dieses lässt sich weiter unterteilen:

Toolbar Die lateinische Schrift ist eine waagerechte rechtsläufige Schrift. Die Schrift wird von links nach rechts und von oben nach unten gelesen, was

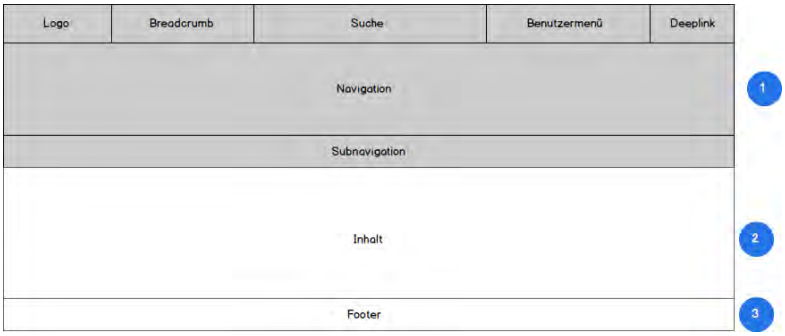


Abbildung 6.5: Grober Benutzeroberflächenaufbau

dazu führt, dass der erste Blick des Nutzers automatisch oben links beginnt. Deshalb ist es sinnvoll hier die Toolbar zu positionieren. Der Nutzer wird als erstes das **Logo** wahrnehmen. Neben der Information, auf welcher Website er sich informiert, ist direkt daneben ein **Breadcrumb**, was ihm vermittelt, wo er sich innerhalb der Webapplikation befindet. Die **Suche** und das **Benutzermenü** sind mittig und rechts plaziert. Nutzer sind diese Anordnung gewohnt. Ein Blick auf die 10 bekanntesten Websites zeigt dies: Acht der zehn meistbesuchtesten Websites haben diese Funktionen dort plaziert. [Ale13] Ein Deeplink führt zu den Hauptfunktionen des Systems.

Wie in Kapitel 6.2.1 festgestellt, ist die Navigation ein Baum mit der Tiefe $t \leq 2$, weswegen diese in zwei Teile geteilt wird:

Navigation Hier werden die Elemente aus der Hierarchiestufe 1 mit einem Icon angezeigt

Subnavigation

Inhalt Hier ist der individuelle Inhalt einer Seite plaziert.

Footer Hier ist Platz beispielsweise für einen Impressumslink, welcher rechtlich leicht erkennbar, unmittelbar erreichbar und ständig verfügbar sein muss.

6.2.3 Administration und System

In diesem Kapitel werden die Entwürfe des Participant Interfaces im Bezug auf die Administration und das System vorgestellt. Die dazugehörigen Anforderungen wurden in Kapitel 5.4.2 vorgestellt.

Registrierung

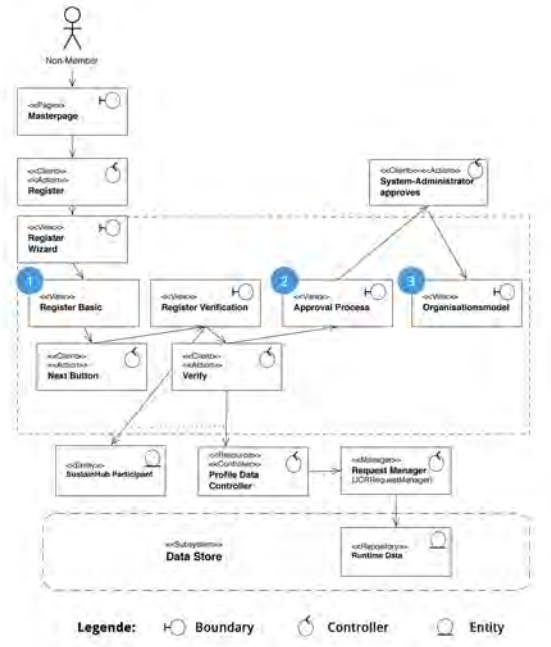


Abbildung 6.6: Robustnessdiagramm: Registrierung

Da eine hohe Anzahl an Daten eingegeben werden, um alle Entitäten der Kategorie ”Participant Data” abzudecken, ist eine Aufteilung in kleinere Schritte mit Teilzielen gemäß 6.1.2 nötig. In Abbildung 6.6 ist diese Aufteilung aufgezeigt. Die blauen Kreise referenzieren auf die betroffenen Bereiche des Datenmodells in Abbildung 6.7. Die ausgegrauten Entitäten sind für interne Daten und somit für das Formular nicht von Bedeutung.

Diese werden folgendermaßen benannt:

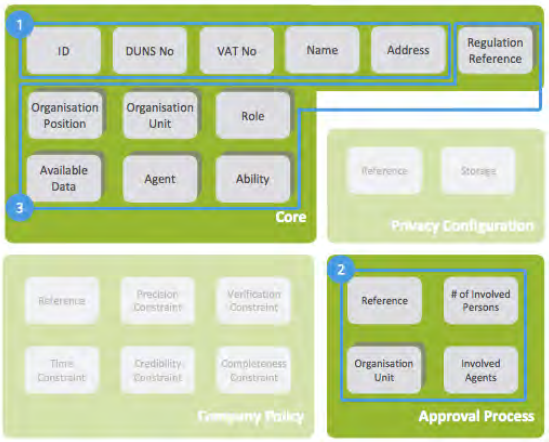


Abbildung 6.7: Entitätskategorie ”Participant Data” aus dem Datenmodell

1. Basic
2. Approval Process
3. Organisationsmodel

6.2.4 Datenaustausch

In diesem Kapitel folgen die UI-Entwürfe des Participant Interfaces im Bezug auf den Austausch von Daten. Diese sind auf Grundlage der Anforderungen aus Kapitel 5.4.3 entstanden.

Aggregation der Datenaustausche

Der Low-Fidelity-Mockup aus Abbildung 6.8 ist das Ergebnis mehrerer Iterationen. Anfangs bestand dieser aus einer einfachen Tabelle. Die ”Quick-Overview” und die einzelnen Punkte des Kontextmenüs wurden später hinzugefügt. Die Austausche werden in einer Tabelle angezeigt. Diese lassen sich am Besten mit Hilfe des Namens der Firma der anderen Partei, des Indikators oder des Erstellungsdatums identifizieren. Zusätzlich wird dem Nutzer angezeigt, ob es sich um einen eingehenden oder ausgehen-

den Request handelt. Die Tabellen lassen sich sortieren, indem die Tabellenkopffelder ausgewählt werden.

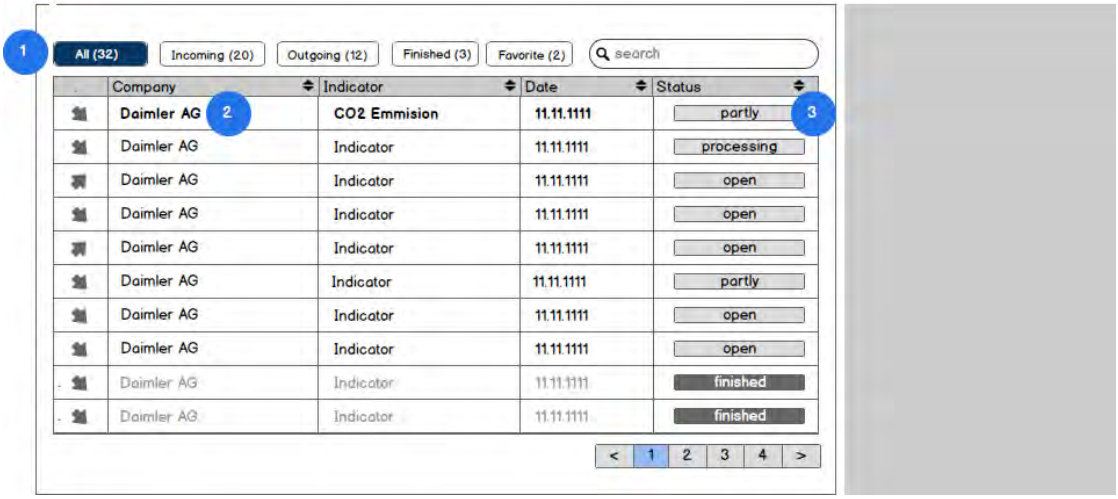


Abbildung 6.8: Low-fidelity Mockup Walkthrough für Aggregation der Dataexchanges

Die in Abbildung 6.8 nummerierten Callouts referenzieren auf folgende Punkte:

1. Die Elemente können nicht nur sortiert, sondern auch gefiltert werden. Dazu dient das Kontextmenü. Neben der Suche, können die Datenaustausche nach folgenden Kritierien gefiltert werden:

   All bedeutet, dass alle Requests ohne Filterung angezeigt werden. Dies ist standardisiert ausgewählt. Es wird vorne angezeigt, weil damit andere Filterungen behoben werden können.

   Incoming und Outgoing Hier kann ausgewählt werden, ob nur Elemente angezeigt werden,die eingehend oder ausgehend sind.

   Finished Datenaustausche mit dem Status ”Finished” sind von besonderer Bedeutung, da es sich hierbei, um die Historie handelt.

   Favorite Hiermit lassen sich Austausche von besonderer Bedeutung filtern, die der Nutzer als Favorit gesetzt hat.

   Um die Intuitivität zu steigern, sollen Icons für diese Selektionsoperationen genutzt werden um diese zu repräsentieren. Abbildung 6.9 zeigt die Entwürfe der Icons

für eingehende, ausgehende und favorisierte Datenaustausche. Für ”All Requests” wurde keine passende Metapher gefunden, die eindeutig die Funktion beschreibt. Hier soll ein Button mit ”All”- Beschriftung genutzt werden.

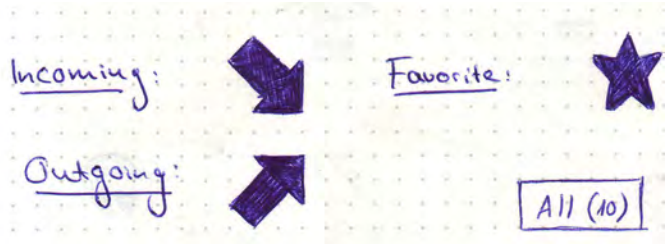


Abbildung 6.9: Iconentwurf für Selektionsoperationen

2. Elemente, an denen seit dem letzten Login neue Änderungen vorgenommen worden sind, werden fett angezeigt. Elemente, die leicht ausgegraut angezeigt werden, sind ausgehende Requests, die noch nicht fertig ausgefüllt sind.

3. Für die vier Status bietet es sich an diese mithilfe Icons anzuzeigen. Abbildung 6.10 zeigt die Entwürfe. Die Grundidee ist, dass ein Datenaustausch als drei horizontale Balken dargestellt werden. Damit ist im minimalistischen Stil ein Datensatz dargestellt, der einen Austausch repräsentiert. Dieses Icon erinnert entfernt auch an das auszufüllende Formular, was später vorgestellt wird. Die Füllung zeigt an, wie weit diese ausgefüllt sind. Open hat somit keine, Partly finished in Teilen und Finished und Processing folgerichtig die gesamte Füllung. Diese unterscheiden sich durch ein weiteres Symbol rechts oben: Ein Haken soll Finished und ein runder Pfeil, der dem Nutzer die Verarbeitung verdeutlicht, implizieren.

Quick-Overview Wird ein Austausch ausgewählt, erscheint rechts eine Übersicht dessen. Der Nutzer kann somit mehrere Austausche schnell vergleichen (vgl. Abbildung 6.11). Hier werden die wichtigsten Informationen ausgegeben. Zudem besteht die Möglichkeit sich mit Tabs weitere Informationen zu diesem Request anzeigen zu lassen. Neben Informationen zu dem Unternehmen der anderen Partei, können alle Daten, die vom Requester angegeben worden sind, hier ausgegeben werden.

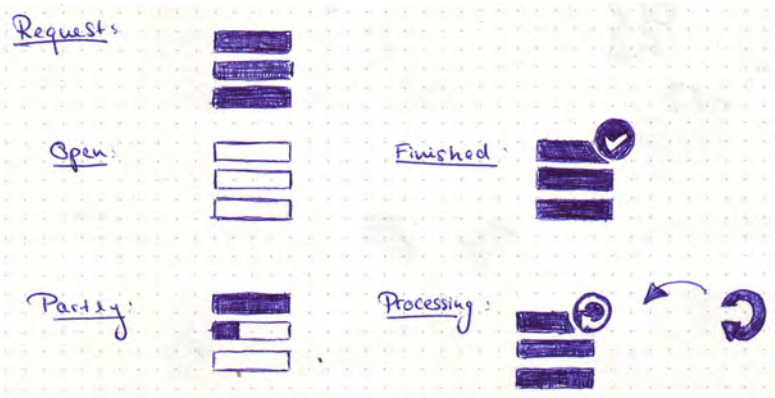


Abbildung 6.10: Iconentwurf für Status eines Request

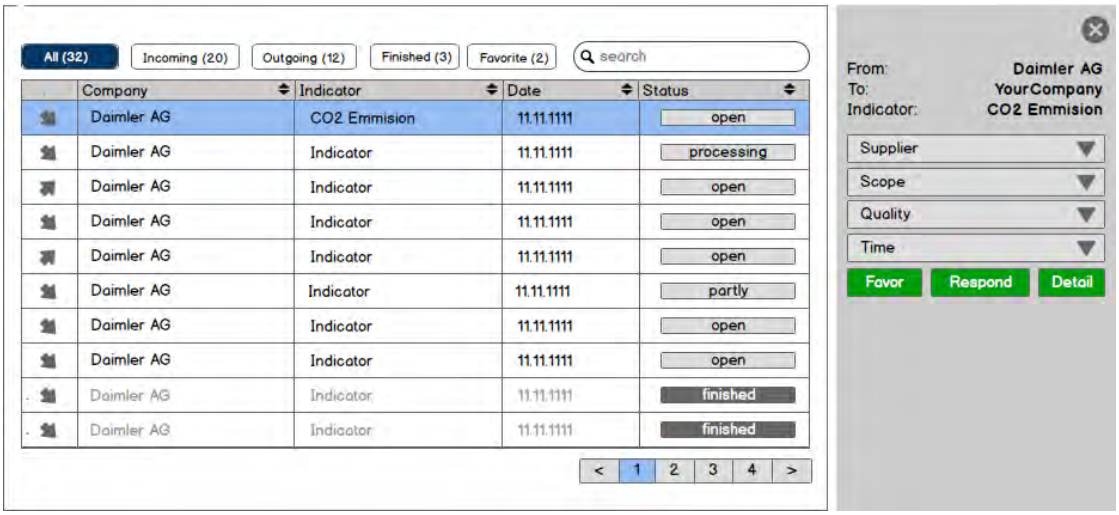


Abbildung 6.11: Low-fidelity Mockup für Quick-Overview eines offenen Requests

Wenn ein Datenaustausch schon teilweise oder komplett beantwortet wurde, werden hier zusätzlich auch die entsprechenden Daten des Responses eingeblendet.

Request

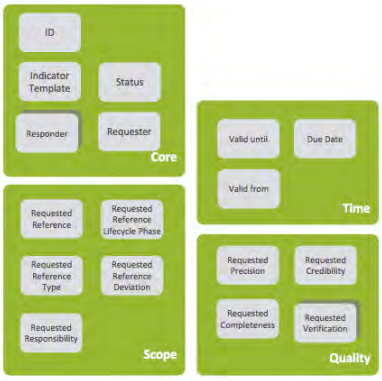


Abbildung 6.12: Entitätskategorie ”Request Data” aus dem Datenmodell

Es gibt zwei Möglichkeiten einen Request zu definieren und somit diese Daten einzugeben, den *Novizenwizard* und das *Expertenformular*. Über die Einstellungen kann der Nutzer angeben, welche Möglichkeit er nutzen will.

Novizenwizard Hier wird ähnlich der Registrierung das Formular zerteilt. Abbildung 6.12 zeigt die Entitätskategorie ”Request Data”. Die Aufteilung des Datenmodells wird dabei für die Aufteilung des Formulars übernommen. Im ersten Schritt, der in Abbildung 6.13 zu sehen ist, sind somit zwei Felder:

1. Supplier: Hierfür wird das Tokenfield, was in Kapitel 6.1.5 beschrieben wird, benutzt.
2. Indikatoreneingabe: Es existieren 66 Indikatoren. Herkömmliche UI-Komponenten unterstützen diese Eingabe nicht ausreichend, weshalb eine eigene UI-Komponente entwickelt wurde. Abbildung 6.15 zeigt einen Walkthrough durch diese. Die Indikatoren lassen sich in die 3 Hauptgruppen ”Enviromental”, ”Social” und ”Managerial” unterteilen. Mit Tabs können die Hauptgruppen ausgewählt werden. Indikatoren

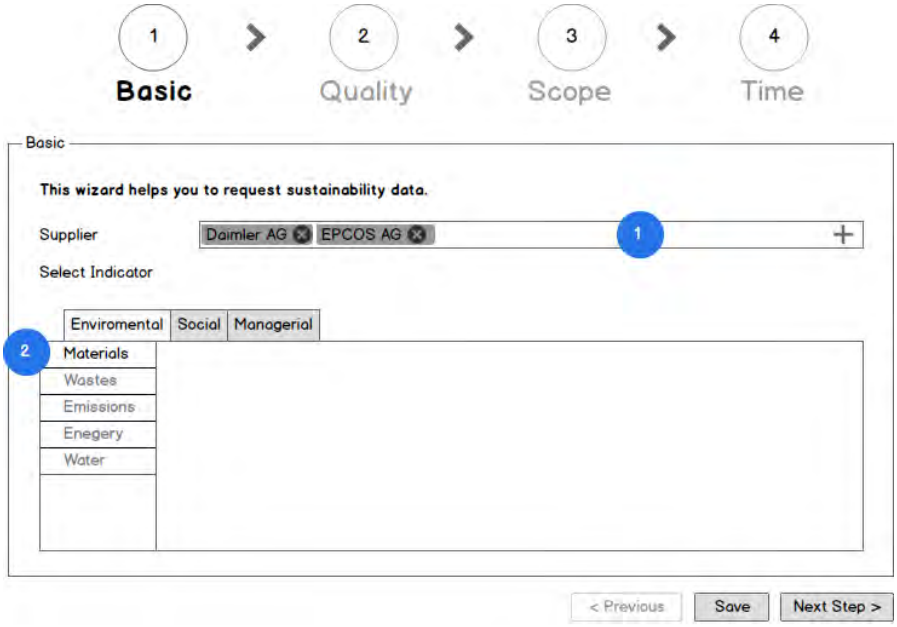


Abbildung 6.13: Low-fidelity-Mockup für Basic-Seite des Request-Wizards

lassen sich jeweils nochmal in 2 weitere Hierarchiestufen teilen. Einige Indikatoren unterscheiden sich nur in Details, z.B. der anzugebenen Einheit. Deshalb muss der Nutzer sich durch die Indikatorenliste navigieren, mehrere Indikatoren vergleichen. Hierfür wird ihm eine Beschreibungen rechts angezeigt. Handelt es sich um den gewünschten Indikator muss er zuletzt diesen mit dem ”Select”-Button auswählen.

Die Icons in Abbildung 6.14 machen diese UI-Komponente intuitiver und optisch ansprechender.

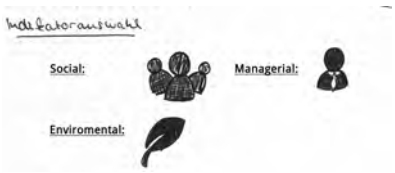


Abbildung 6.14: Iconentwurf für Indikator-UI-Element

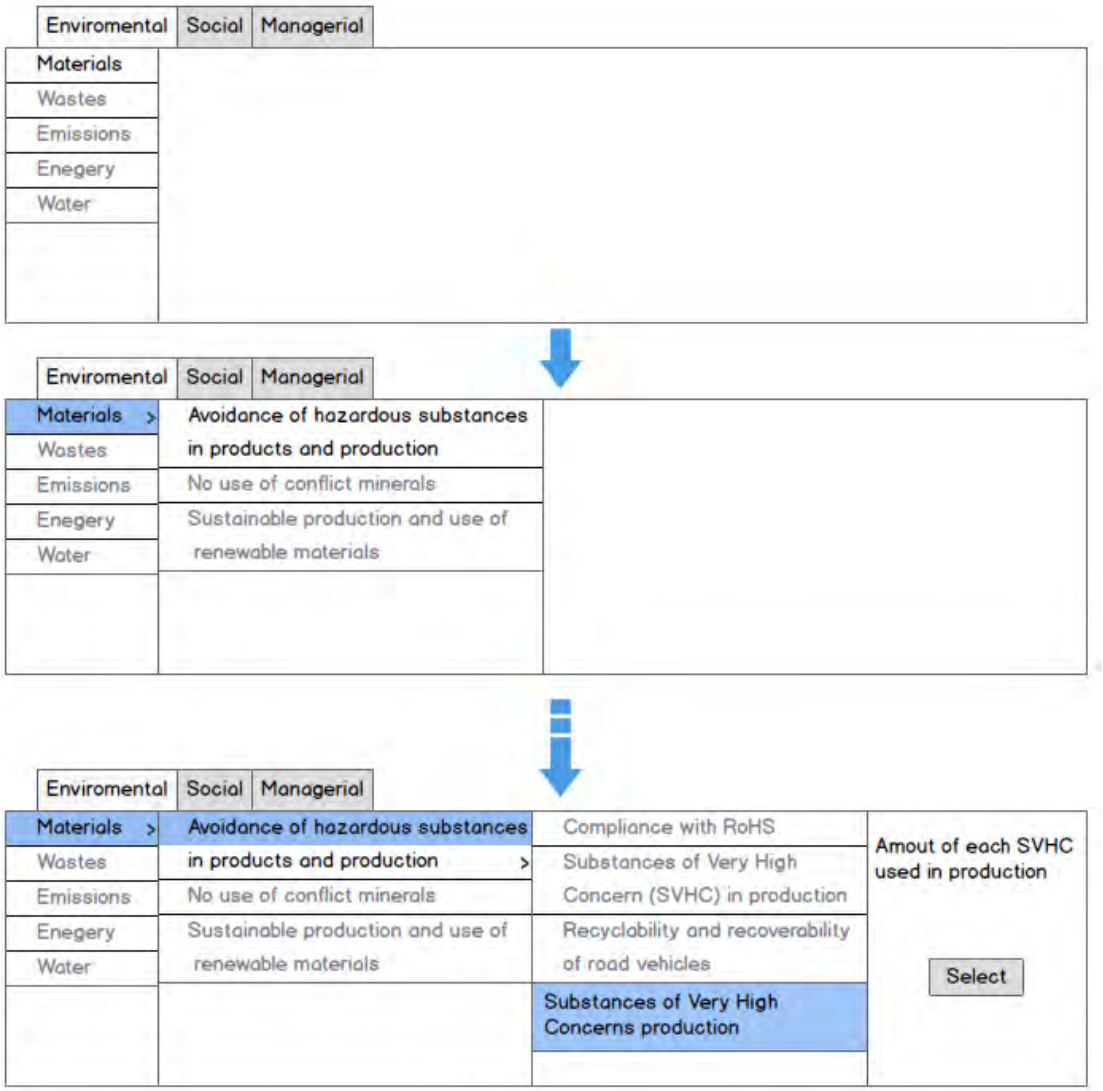


Abbildung 6.15: Low-fidelity-Mockup-Walkthrough des Indikator-UI-Element auf der Basic-Seite des Request-Wizards

Expertenformular Der Wizard führt Novizen durch den Eingabeprozess. Experten sind auf diese Hilfe nicht angewiesen. Ihm wird ein Expertenformular angeboten, welches ihm erlaubt die Daten schnell und effektiv einzugeben. Das Formular ist in Abbildung 6.16 zu sehen. Im Anschluss darauf kann der Nutzer wie im Wizard die Daten bestätigen.

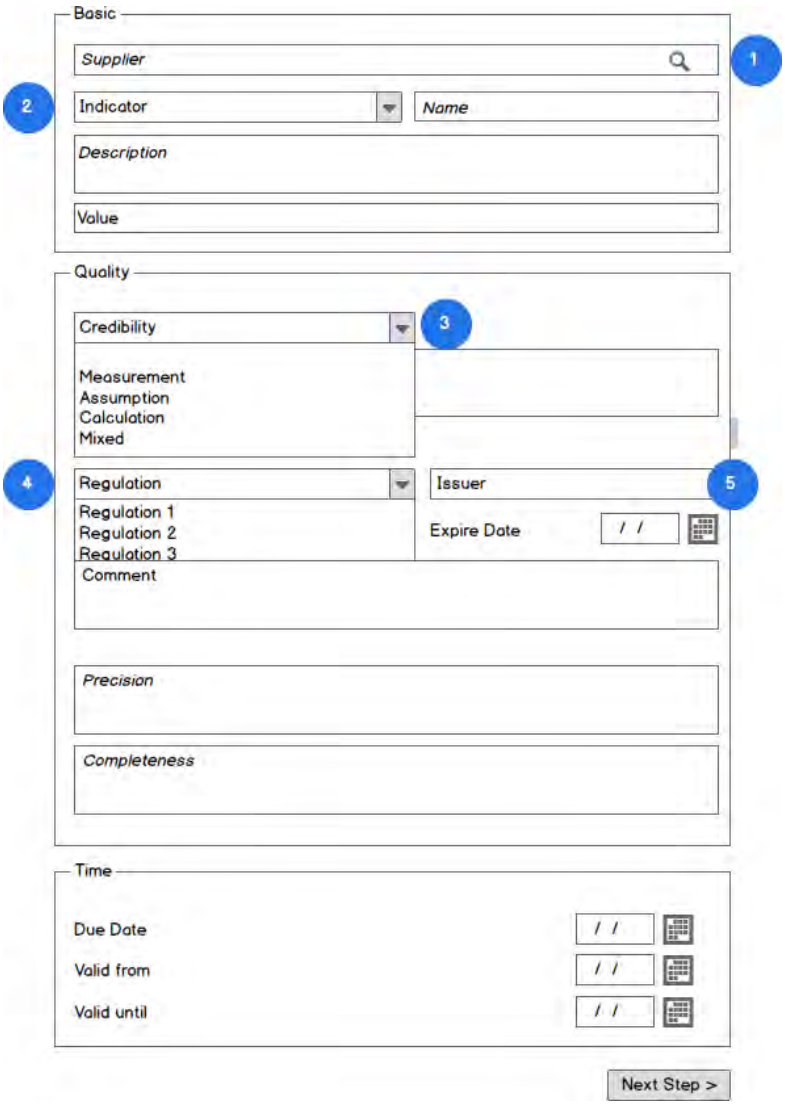


Abbildung 6.16: Low-fidelity-Mockup für Expertenrequest

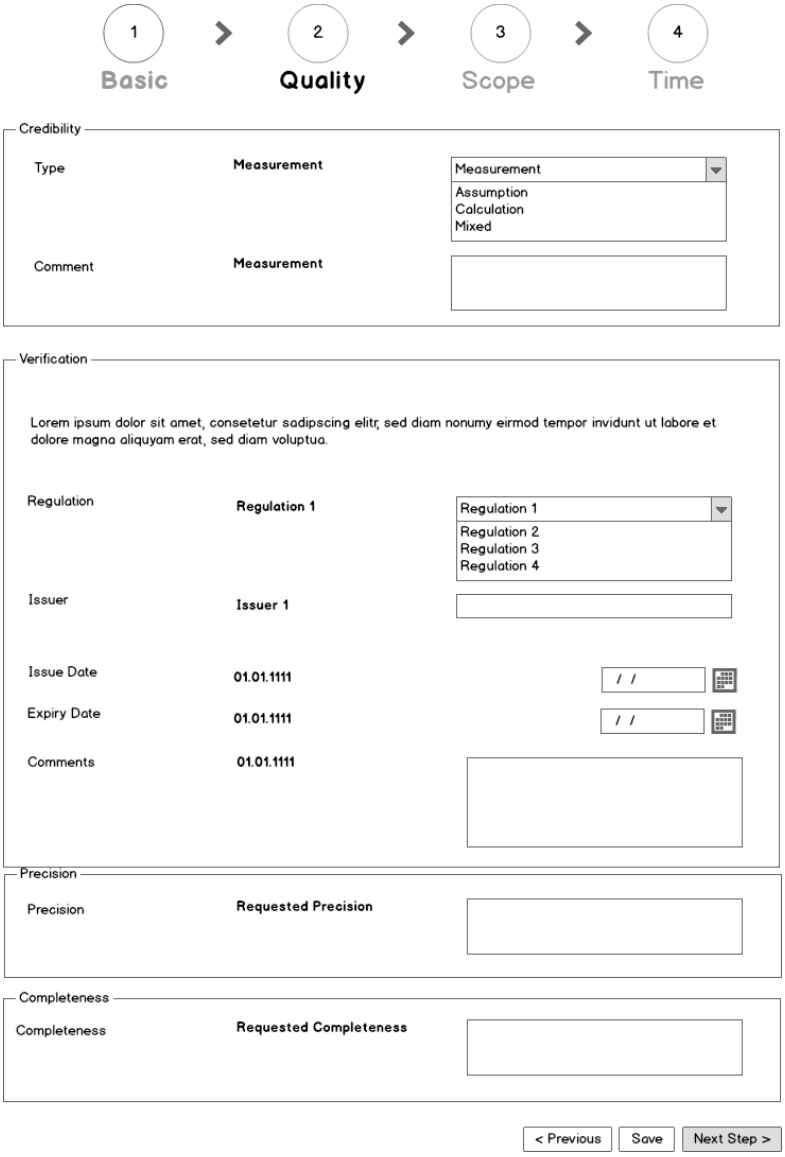


Abbildung 6.17: Low-fidelity-Mockup für Quality-Seite des Response-Wizards

Response

Die Anforderungen ähneln stark dem Wizard des Requests. Damit Fehleingaben minimiert werden, wird angegeben, was über den entsprechenden Request erfragt wurde, wie Abbildung 6.17 zeigt. Der Wizard teilt den Eingabeprozess in folgende Teile:

Basic Auf der ersten Seite werden explizit die Nachhaltigkeitsdaten angegeben. Die genaue Umsetzung erfolgt von anderen Projektteilnehmern. Im weiteren Verlauf werden die Metadaten ausgefüllt, welche die gleichen sind, wie im Request:

Quality Dies beinhaltet Präzision, Kredibilität, Vollständigkeit und Verifikation.

Scope Referenzen, Lebenszyklusphase, Abweichung und die Zuständigkeit.

Time Datum, ab wann und bis wann die Daten gültig sind und das Erstellungsdatum der Daten.

Provide Data

Aus Usabilitysicht ist Provide Data das gleiche wie ein *Response* mit der Ausnahme, dass keine Requestdaten angezeigt werden müssen.

6.2.5 Kontakte

Im folgende werden die abstrakten Mockups beschrieben, die Kontakte betreffen. Die dazugehörigen Anforderungen wurden in Kapitel 5.4.3 vorgestellt.

Aggregation der Kontakte

Kontakte werden in einer *Aggregation* dargestellt. Als Selektionsfunktionen werden folgende Eigenschaften herangezogen:

1. Eigene Kontakte
2. Alle Kontakte

Als Eigenschaft eines Kontakts werden der Name, der Name der Organisation und ihre Funktion innerhalb der Lieferkette im Bezug auf den eigenen Nutzer angezeigt.

In der Übersicht werden alle Daten aus ”Participant Data” und der Organisation angezeigt. Dort befindet sich ein Button, der zum Benutzerprofil führt und ein Connect-Button. Dieser zeigt ”Connected” an. Wenn die Nutzer nicht verbunden sind, zeigt er ”Connect” an. Wird dieser ausgewählt, werden die Nutzer miteinander verbunden.

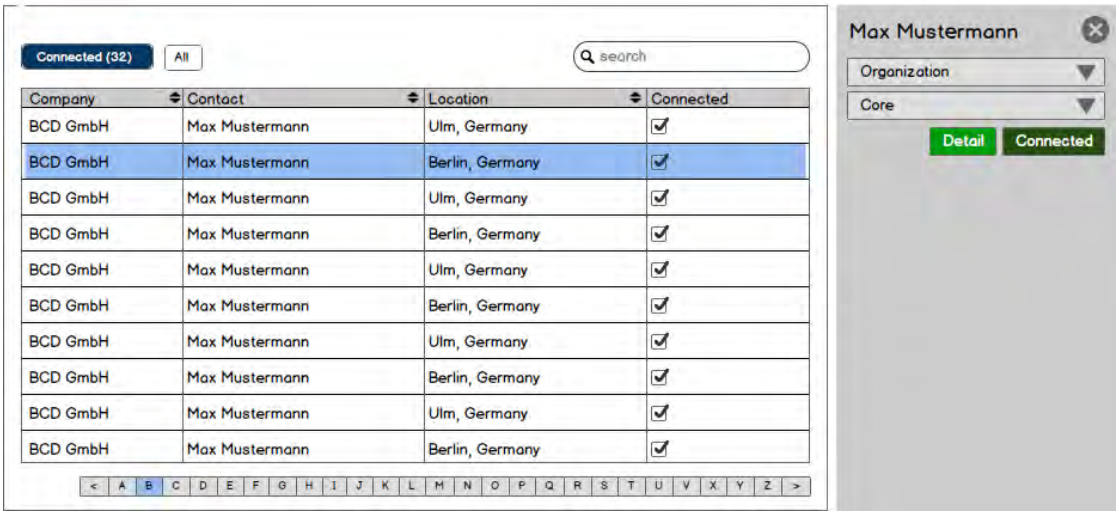


Abbildung 6.18: Low-fidelity-Mockup für Kontakte

Benutzerprofil

Benutzerprofile werden mit einer *Detailansicht* angezeigt. Die Anzeige der Firma, wird auf die entsprechende Organisationsansicht verlinkt. In Tabellen werden weitere Mitarbeiter der Organisation angezeigt und Datenaustausche, die mit diesem Nutzer bestehen. Ein Button führt zum Requestwizard, bei dem das Supplier-Feld schon mit dem Nutzer ausgefüllt ist. In der Buttonleiste befindet sich der Connect-Button.

Aggregation der Organisationen

Organisationen werden in einer *Aggregation* dargestellt. Als Selektionsfunktionen dienen folgende Eigenschaften:

1. Eigene Organisationen (Organisationen, mit denen man verbunden ist)

2. Alle Organisationen

Organisationsansicht

In der Organisationsansicht werden die Informationen in einer Detailansicht angezeigt. Datenaustausche, die mit dieser Organisation in Verbindung stehen und Mitarbeiter werden in Tabellen dargestellt.

6.2.6 Data

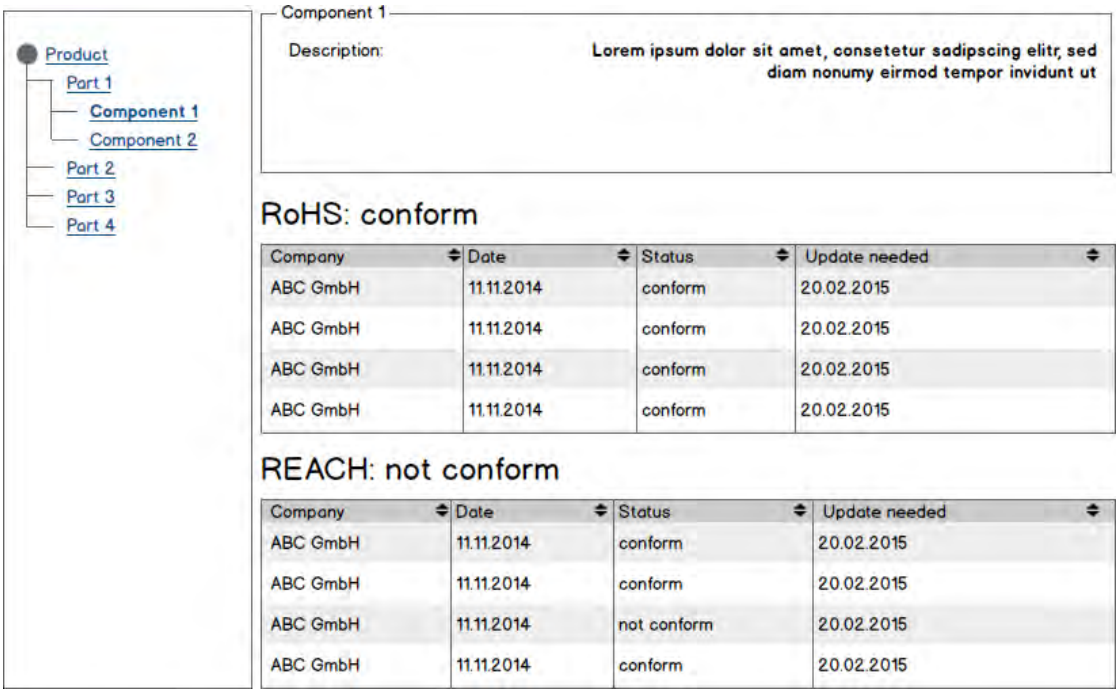


Abbildung 6.19: Low-fidelity-Mockup für Data

Ein Low-fidelity Mockup für Data wird in Abbildung 6.19 angezeigt. Mithilfe einer hierarchisierten Liste werden die Costumer Data und deren Zusammenhang angezeigt. Wird ein Item ausgewählt, erhält der Nutzer die dazugehörigen Daten in einer Detailansicht angezeigt. Zudem werden zugehörige Regulations in einer Tabelle angezeigt.

6.2.7 Dashboard

In Abbildung 6.20 ist das Dashboard zu sehen. Es werden jeweils als Tabelle die 5 aktuellsten Einträge von Datenaustauschen, Kontakten und Regulationen angezeigt.

Exchange

	Company	Indicator	Date	Status
	Daimler AG	CO2 Emmision	11.11.1111	partly
	Daimler AG	Indicator	11.11.1111	processing
	Daimler AG	Indicator	11.11.1111	open
	Daimler AG	Indicator	11.11.1111	open
	Daimler AG	Indicator	11.11.1111	open

Contacts

Company	Name	Location
BCD GmbH	Max Mustermann	Ulm, Germany
BCD GmbH	Max Mustermann	Ulm, Germany
BCD GmbH	Max Mustermann	Ulm, Germany
BCD GmbH	Max Mustermann	Ulm, Germany
BCD GmbH	Max Mustermann	Ulm, Germany

Regulations

Name	Date	Status	Type
RoHS	11.11.2011	☑	compliance
RoHS	11.11.2011	☑	compliance
RoHS	11.11.2011	☑	compliance
RoHS	11.11.2011	☑	compliance
RoHS	11.11.2011	☑	compliance

Abbildung 6.20: Low-fidelity-Mockup für Dasboard

6.3 Administration Interface

In diesem Kapitel sind die Entwürfe des Administration Interfaces dokumentiert. Neben der Möglichkeit der Manipulation der Costumer Data, können hier externe Systeme eingetragen und geändert werden.

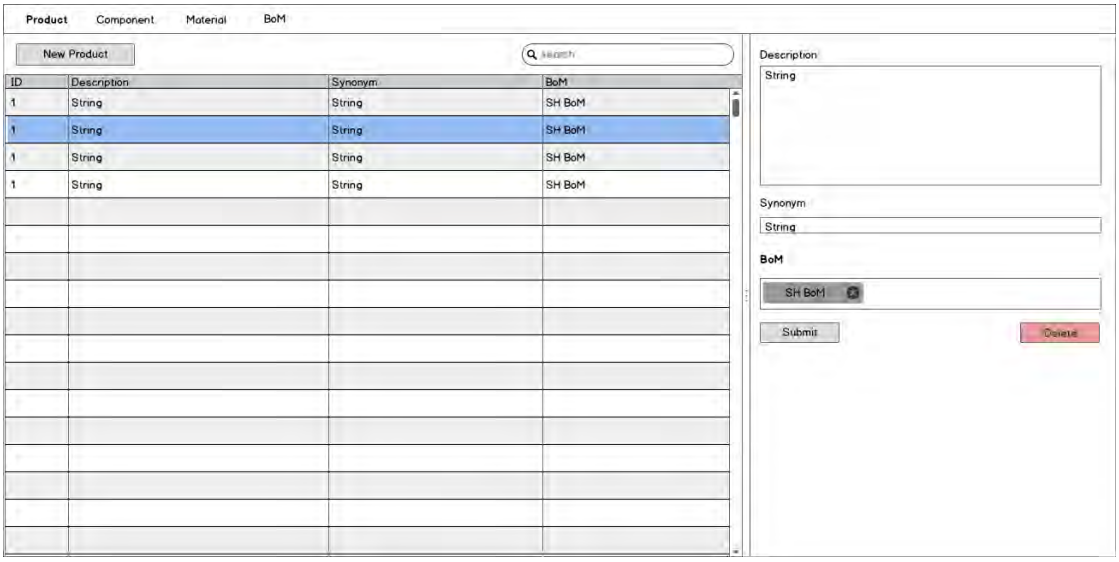


Abbildung 6.21: Low-fidelity-Mockup für Costumer-Data

6.3.1 Costumer Data

Alle Entitäten von ”Costumer Data” können gleich angezeigt, bearbeitet und gelöscht werden. Mithilfe des Mockups in Abbildung 6.21 wird gezeigt wie ein Administrator Products manipulieren kann. Analog können auch die Entitäten ”Components”, ”Material” und ”BoM” manipuliert werden. Der Dialog ist in zwei Teile gesplittet. Auf der linken Seite befindet sich ein ”New Product”-Button, ein Suchfeld und die vorhandenen Daten werden in einer Tabelle angezeigt. Wird der ”New Product”-Button geklickt, kann auf der rechten Seite ein neues Objekt instanziiert werden. Wird ein Objekt ausgewählt, kann dieses auf der rechten Seite in einem Fomular verändert werden. Mithilfe des ”Submit”-Buttons kann die Eingabe abgesendet und mithilfe des ”Delete”-Button die Entität gelöscht oder verworfen werden. Dieses User-Interface ermöglicht eine schnelle Manipulation der Daten. Das System kann noch effizienter genutzt werden durch folgende Punkte:

Direkter Fokus auf Suchfeld Der Nutzer kann somit schnell und gezielt Objekte finden.

Shortcuts Der Nutzer kann mit den Pfeiltasten durch die Tabelle navigieren. Mit dem Shortcut ”e” für **edit** kann der Nutzer das Objekt ändern. Es wird der Fokus auf

das erste Formularfeld des Objektes gesetzt. Mit ”d” für **delete** kann das Objekt gelöscht werden. Über ein Popup muss diese Löschung bestätigt werden. Mit ”n” für **new** kann der Nutzer ein neues Objekt instanziieren.

6.3.2 IHS verwalten

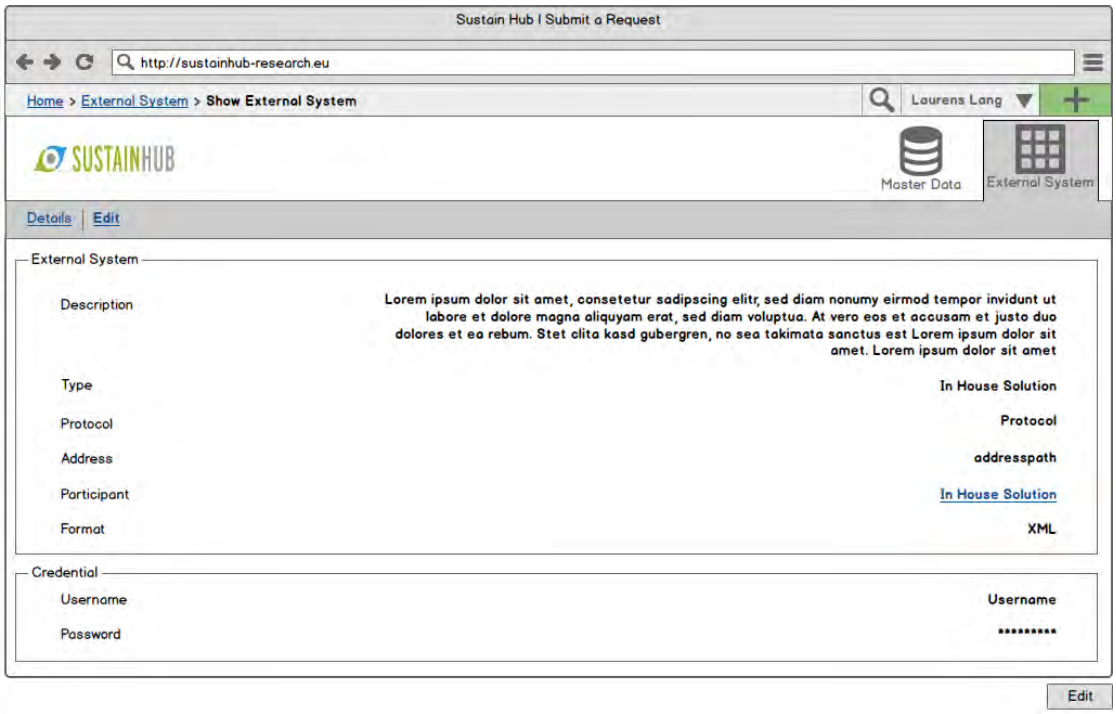


Abbildung 6.22: Low-fidelity-Mockup für IHS anzeigen

Aus Datenmodellsicht handelt es sich um die Daten der Entität ”External System”. Diese müssen angezeigt und bearbeitet werden können. Dies wird im Folgenden beschrieben.

IHS anzeigen

In einer Detailansicht werden die Daten des External Systems ausgegeben (Vgl. Abb. 6.22). Ein Button führt zu *IHS bearbeiten*. Die Nutzerdaten, Nutzername und Passwort, werden getrennt angezeigt.

IHS bearbeiten

In einem einfachen Formular können die Daten manipuliert werden, wie Abbildung 6.23 zeigt. Eine Bestätigung findet über eine Detailansicht mit einem ”Verify”-Button statt.

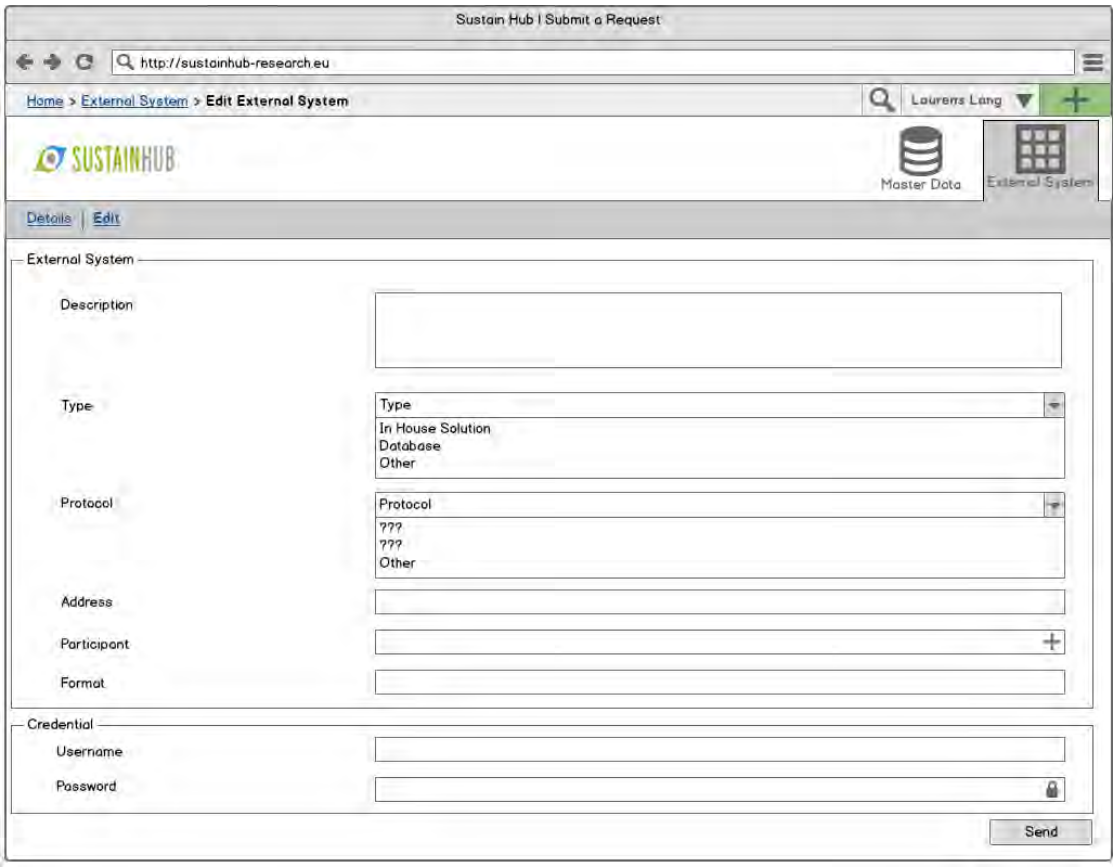


Abbildung 6.23: Low-fidelity-Mockup für IHS bearbeiten

7

Styleguide

Auf Basis der vorangengangen Anforderunganalyse aus Kapitel 5 und der Usability-Entwürfe aus Kapitel 6 wurde ein Styleguide für *SustainHub* ausgearbeitet. Die Ergebnisse dieses Entwicklungsprozesses wird als Vorlage für die Implementierung dienen. Im ersten Kapitel werden generelle Gestaltrichtlinien vorgestellt, die global im System gelten. Im Projekt treten bestimmte Probleme mehrfach auf. Die Lösung zu diesen werden im darauf folgenden Kapitel ”Wiederkehrende Oberflächenelemente” definiert.

7.1 Generelle Gestaltrichtlinien

Viel wichtiger als eine Aneinanderreihung von Entwurfsideen ist eine Selektion einzelner Ideen, die an mehreren Stellen im System zum Einsatz kommen. So wird Konsistenz erreicht, die Nutzern ein neues Erkennen und Erlernen von Mustern erspart und somit

die Konzentration auf Inhalte beschleunigt. Um Konsistenz zu erreichen, bedarf es an globalen Gestaltungsrichtlinien, die für das gesamte System gelten. Diese werden in diesem Kapitel vorgestellt [Ram13].

7.1.1 Typografie

Im gesamten System wird nur eine Schriftart genutzt: Droid Sans. Droid Sans wurde von Steve Matteson, Type Director bei Ascender Corp, erstellt und ist eine humanistische und serifenlose Schriftart, die mit ihren offenen Formen zu einem neutralen, offenem Erscheinungsbild führt. Sie wurde erstellt, um das Lesen sowohl in Menüs, Web-Browsern und anderen Bildschirmtexten zu erleichtern und eignet sich somit speziell für User Interfaces in Mobile- und Webapplikationen. Sie enthält einen umfangreichen Zeichensatz, der alle 24 Amts- und Arbeitssprachen der Europäischen Union abdeckt und bietet sich somit optimal für SustainHub, einem Projekt der Europäischen Union, an [Goo13].

Die Schriftart sollte serverseitig gespeichert werden, damit sie plattformunabhängig von jedem Gerät angezeigt werden kann. Kann dies nicht gewährleistet werden, sollte eine andere serifenlose Schrift, wie Arial oder Tahoma, als Fallback dienen. Die einzige Ausnahme stellt die Wortmarke im Header dar, da diese bereits vor der Erstellung des Usability-Konzepts eingesetzt wurde.

Größen

Überschrift: 18 pt regular

Unterüberschrift: 14 pt bold

Text: 14 pt regular

Abbildung 7.1: Typographiegrößen im Überblick

Die Textgrößenbestimmung lassen sich auf einige wenige Regeln reduzieren, die in Abbildung 7.1 zu sehen sind. Der geringe Unterschied zwischen Unterüberschrift und Text beruht darauf, dass Unterüberschriften meist noch farblich hinterlegt sind.

7.1.2 Farbkonzept

Ein Coporate Design und somit auch ein Farbkonzept existieren bereits für SustainHub. Das Farbkonzept des SustainHub Systems orientiert sich daran. Die 3 Basisfarben #4d9bdf (Blau) #8dae33 (Grün) und #6d6d6d (Grau) werden 7.2 durch Abstufungen ergänzt (vgl. 7.2).

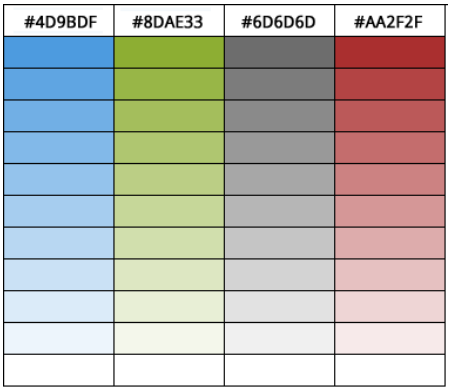


Abbildung 7.2: Farbkonzept von SustainHub: Abstufungen der 3 Grundfarben

Um Warnungen und Errors entsprechend anzuzeigen, wird eine Warnfarbe benötigt: #aa2f2f (Rot) Sowohl Schriften, Gestaltelemente und Icons basieren ausschließlich auf diesen Farben.

7.1.3 Favicon

Abbildung 7.3: Favicon: Links die Rastergrafik, rechts in Originalgröße im Browser

Ein Favicon ist ein Icon, welches 16 x 16 Pixel groß ist. Es wird im Browser links neben der Adresszeile, in Bookmarks etc. angezeigt und dient als Erkennungszeichen der SustainHub-Webapplikation. In Abbildung 7.3 wird das SustainHub-Favicon links in einer vergrößerten Rastergrafik und rechts im Browserkontext angezeigt.

7.2 Wiederkehrende Oberflächenelemente

In SustainHub werden einige Oberflächenelemente an mehreren Stellen verwendet. Diese werden in diesem Kapitel vorgestellt.

7.2.1 User-Interface-Komponenten

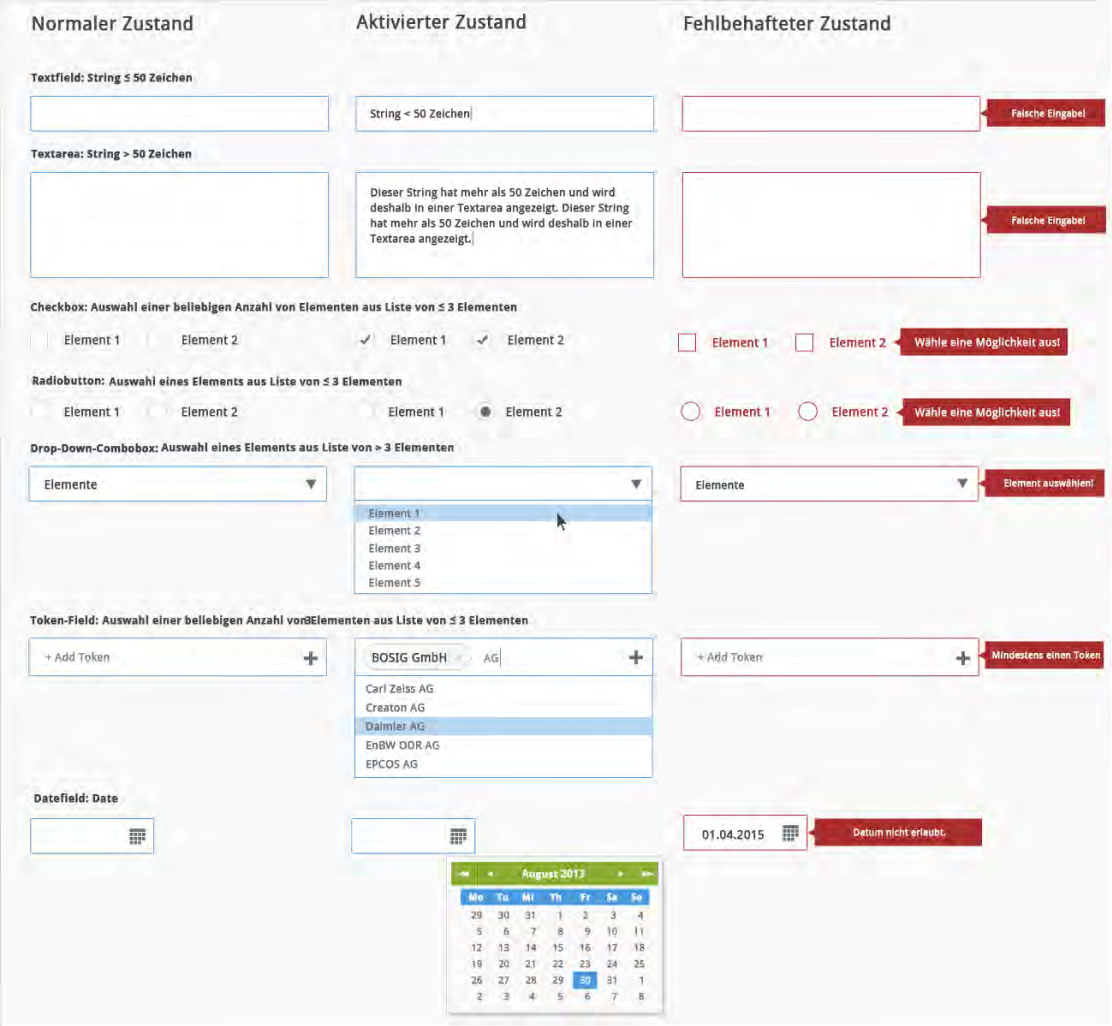


Abbildung 7.4: User-Interface-Komponenten Kit

Innerhalb SustainHub sind User-Interface-Komponenten konsistent gestaltet. Abbildung 7.4 zeigt das SustainHub User-Interface-Kit. Müssen Strings, Zahlenwerte (Doubles oder Integers) eingegeben, werden abhängig ihrer erwarteten Länge entweder Textfields oder Textareas genommen. Diese Unterscheidung ist bei 50 Zeichen angesetzt und räumt dem Nutzer bei längeren Zeichenketten Freiheit ein und dient als Orientierung. Müssen aus Listen Strings ausgewählt werden, die Wertetypen referenzieren, wird auch nach der Anzahl der Möglichkeiten unterschieden. Für Listen, die mehrere Eingaben erlauben, werden Checkboxen für weniger als vier Möglichkeiten benutzt. Für vier oder mehr Möglichkeiten wird ein Tokenfield benutzt. Für Listen, die nur eine Eingabe erlauben, werden bei weniger als vier Möglichkeiten Radiobuttons benutzt. Für vier oder mehr Möglichkeiten, wird eine Combobox benutzt. Werden Dates erwartet, kommt ein Datefield zum Einsatz. Die Abbildung zeigt die Komponenten einmal im normalen Zustand, einmal im aktiven Zustand und wenn Fehlereingaben getätigt wurden.

Da Tokenfields nicht zu dem Standardrepertoire von User-Interface-Komponenten gehört, werden diese im folgenden Kapitel erklärt.

Tokenfield

In Abbildung 7.5 ist der High-fidelity Mockup Walkthrough eines Tokenfield abgebildet. Die Funktionsweise wurde bereits in Kapitel 6.1.5 anhand der Low-fidelity-Mockups erklärt.

Indikatorenauswahl-Komponente

Die Abbildung 7.6 zeigt die umgesetzte Indikatorauswahl-Komponente. Mit den Tabs, in denen sich Icons und Überschriften befinden, können die Hauptkategorien ausgewählt werden. Der Nutzer kann sich durch die verschiedenen Unterkategorien navigieren. Wählt er ein Indikator aus, wird im rechten Feld eine Beschreibung angezeigt. Mit ”Select” kann er diesen Indikator eingeben.

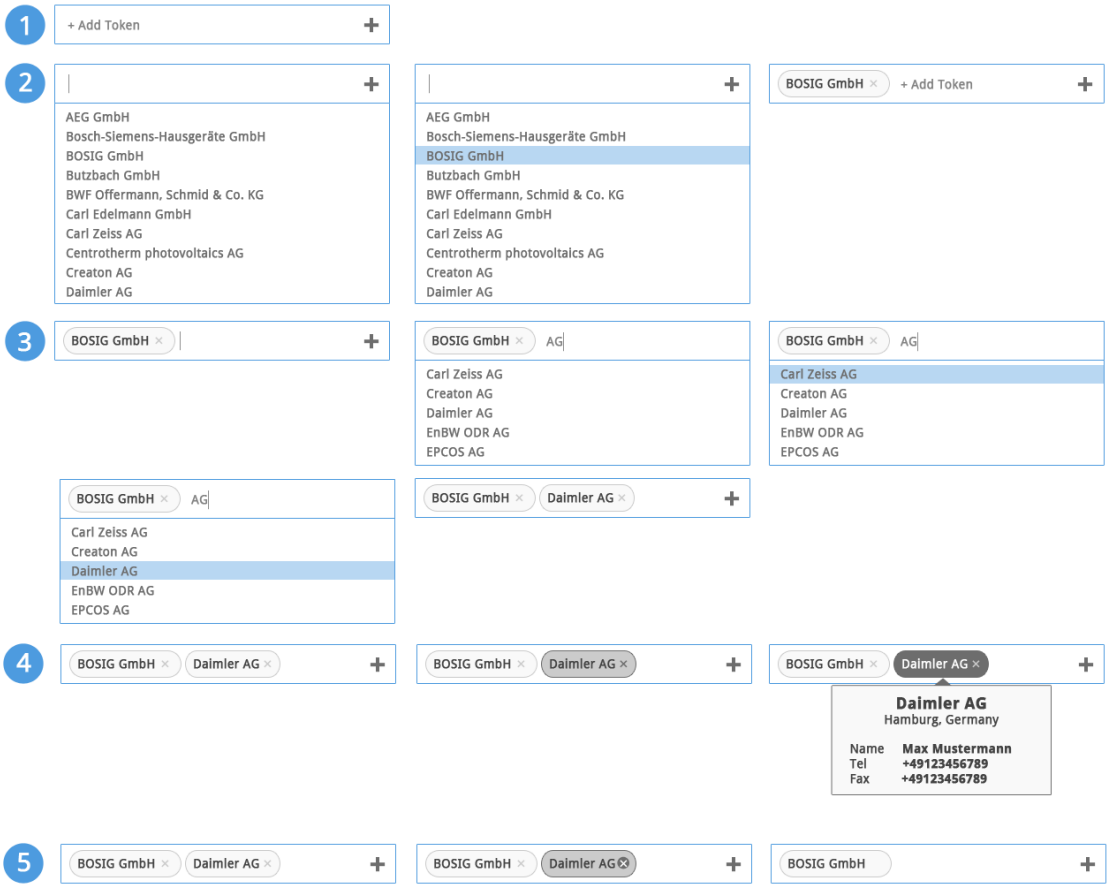


Abbildung 7.5: High-fidelity Mockup Walkthrough für Tokenfield

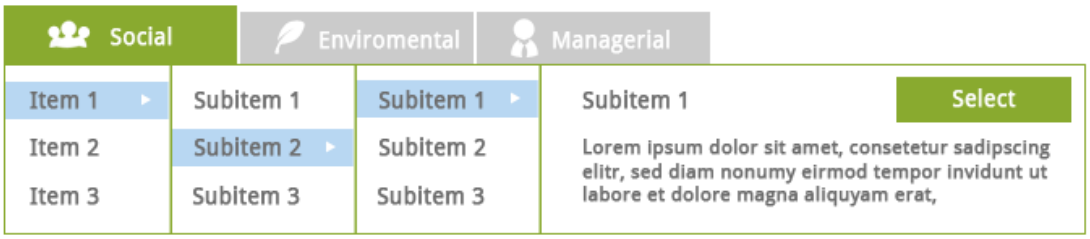


Abbildung 7.6: High-fidelity der Indikatorauswahl-Komponente

7.2.2 Gruppierung von Formularfeldern

In vielen Fällen ist es sinnvoll mehrere Formularfelder bzw. 7.2.1 zusammenzufassen, da sie inhaltlich zusammengehörig sind. Abbildung 7.7 zeigt eine solche Gruppe. Es können sich mehrere Gruppen auf einer Seite befinden.

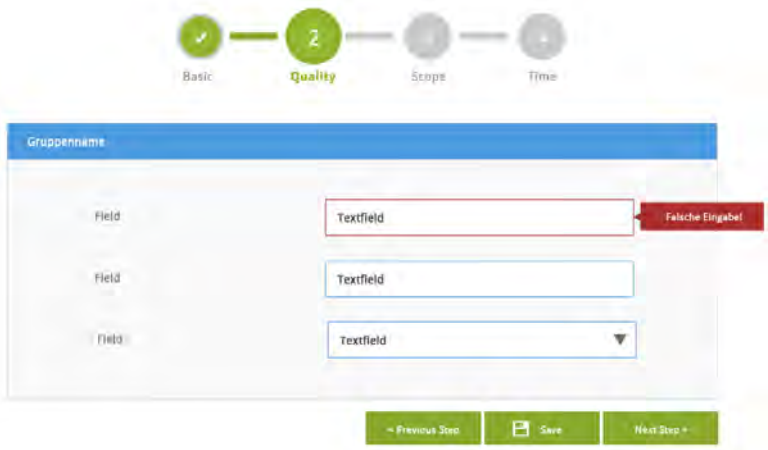


Abbildung 7.7: High fidelity Mockup einer Gruppierung von Formularfeldern

7.2.3 Pagination

Pagination beschreibt die Aufteilung von Informationen auf verschiedene Seiten. Dies ist für Listen, Tabellen oder Suchergebnisse erforderlich, da diese häufig die Anzeige auf einem Screen bei weitem übersteigt. Außerdem bietet sie dem Nutzer eine Orientierung, die er aus der realen Welt kennt. Die Seitenaufteilung entlastet das Kurzzeitgedächtnis und führt zu schnelleren Laden der Website. Neuere Webapplikationen benutzen sogenanntes ”Infinite Scrolling”, was bedeutet, dass der Inhalt weitergeladen wird, wenn man das Ende der Seite erreicht hat. Dieses lässt sich schwer realisieren, da Vaadin keine Scrollevents verarbeiten kann. Abbildung 7.8 zeigt die Pagination-Komponente. Der Pfeil nach links führt zur vorherigen, der rechte zur nächsten. Die aktuelle Seite wird durch ein größeres Rechteck hervorgehoben.

Abbildung 7.8: High fidelity Mockup der Pagination-Komponente

7.3 Aufbau des User-Interfaces

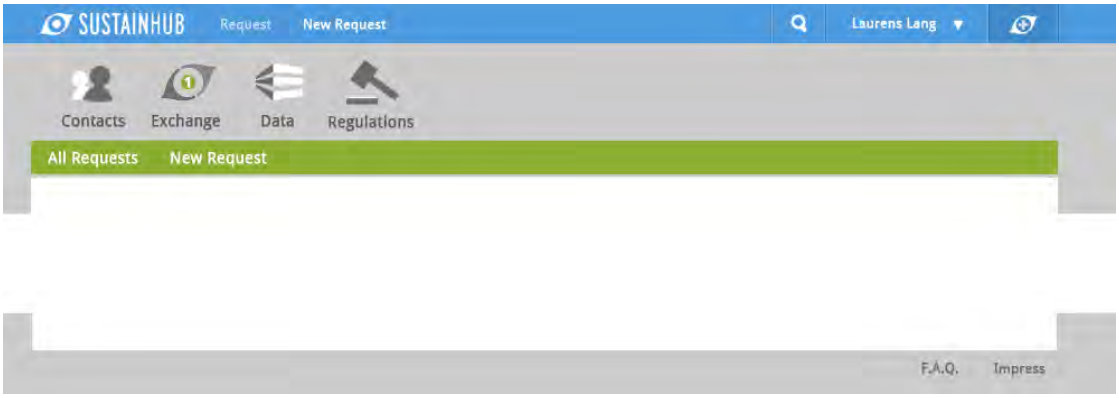


Abbildung 7.9: High-fidelity-Mockup des User-Interface-Aufbaus

Abbildung 7.9 zeigt den Aufbau des User-Interfaces. Die blaue Toolbar und die Navigation werden in den darauffolgenden Kapiteln genauer vorgestellt. Die grauen Flächen schließen den individuellen Seiteninhalt ein, obwohl diese links und rechts nicht durchgängig sind (vgl. Kapitel 4.3.1: Gesetz der Geschlossenheit). Darunter befindet sich der Footer.

7.3.1 Toolbar

In der Toolbar befinden sich neben dem Logo, das Breadcrumb, die Suche, das Benutzermenü und Deeplinks. Abbildung 7.10 zeigt die verschiedenen Zustände:

1. **Logo.** Das Logo verweist auf das Dashboard.
2. **Breadcrumb.** Das Breadcrumb zeigt an, in welcher Aufgabe der Nutzer sich innerhalb des Systems befindet. Hierfür wird der Name der Aufgabe angezeigt und leicht ausgegraut die Kategorie gemäß der Aufgabenhierarchie.

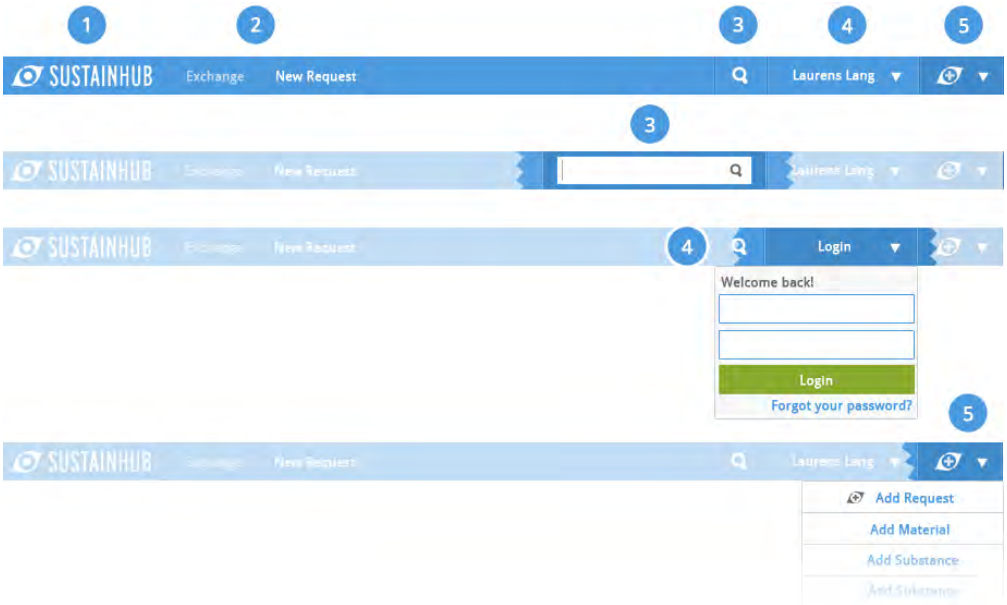


Abbildung 7.10: High-fidelity-Mockup der Toolbar

3. Suche. Ein Klick auf den Suchbutton aktiviert das Suchfeld. Die Ergebnisse werden mithilfe einer Tabelle angezeigt.

4. Benutzermenü. Je nach Zustand wird ein Login-Formular oder das Benutzermenü angezeigt. Hierfür wird der Name des Nutzers angezeigt. Das Dreieck impliziert das Menü. Hier befinden sich Links zum Profil, Einstellungen und Logout.

5. Deeplink. Im Participant Interface wird nur "New Request" verlinkt. Im Administration Interface können hierüber auch die Formulare zur Erstellung von Costumer Daten erreicht werden.

7.3.2 Navigation und Subnavigation

In der Navigation werden wie in Abbildung 7.9 zu sehen ist, die Icons angezeigt. In der Subnavigation werden gemäß der Aufgabenhierarchie aus Kapitel 6.2.1 die entsprechenden Unterpunkte angezeigt, wenn ein Navigationspunkt ausgewählt wird. Über dem Icon für Exchange wird bei

7.3.3 Dialogarten

Die in Kapitel 6.1 beschriebenen Dialogarten werden hier detailliert vorgestellt. Im SustainHub-Projekt gibt es Wizards, einfache Formulare, Detailseiten, Tabellen und Aggregationen. Im Aschluss werden die Seiten aufgezeigt, die nicht in eine dieser Kategorien passen.

Wizards

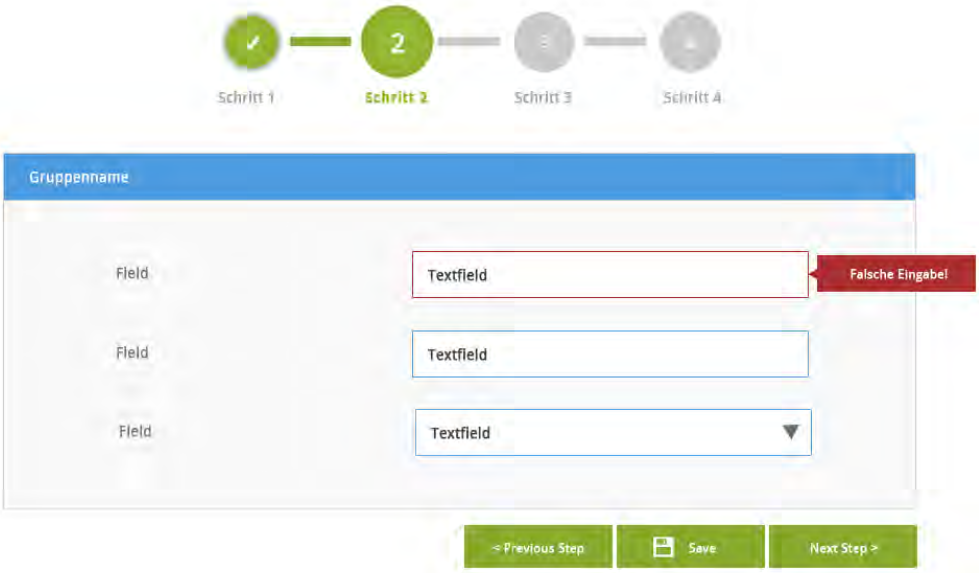


Abbildung 7.11: High-fidelity-Mockup eines Wizards

Abbildung 7.11 zeigt einen hypothetischen Wizard. Ein Wizard besteht aus drei Teilen:

Fortschrittsindikator. Dieser zeigt die Teile an, in die der Eingabeprozess aufgeteilt ist. Der aktuelle Schritt ist hervorgehoben. Die Schritte, die bereits erfolgt sind, sind mit einem Haken versehen. Leicht ausgegraut sind die bevorstehenden Schritte dargestellt.

Formular. Die Formularfelder sind in mindestens einem Panel (vgl. 7.2.2).

Buttons. Wie die Abbildung zeigt können mithilfe des ”Previous Step”- und ”Next Step”-Buttons die Richtung des Wizard gelenkt werden. Ein Save-Button inkl. Icon ermöglich die Speicherung der Eingaben. Im ersten Schritt gibt es keinen ”Previous Step”-Button.

Wird der Wizard abgebrochen, zum Beispiel durch Schließen des Browser-Fensters, erscheint ein Popup, das den Nutzer explizit fragt, ob der Wizard abgebrochen werden soll. Es gibt drei Buttons: ”Continue”, der den Nutzer zurück zum Wizard führt, ”Save and close”, der die Eingabe speichert und den Wizard beendet und ”Cancel Wizard”, bei dem der Wizard abgebrochen wird.

Tabellen und Aggregationen

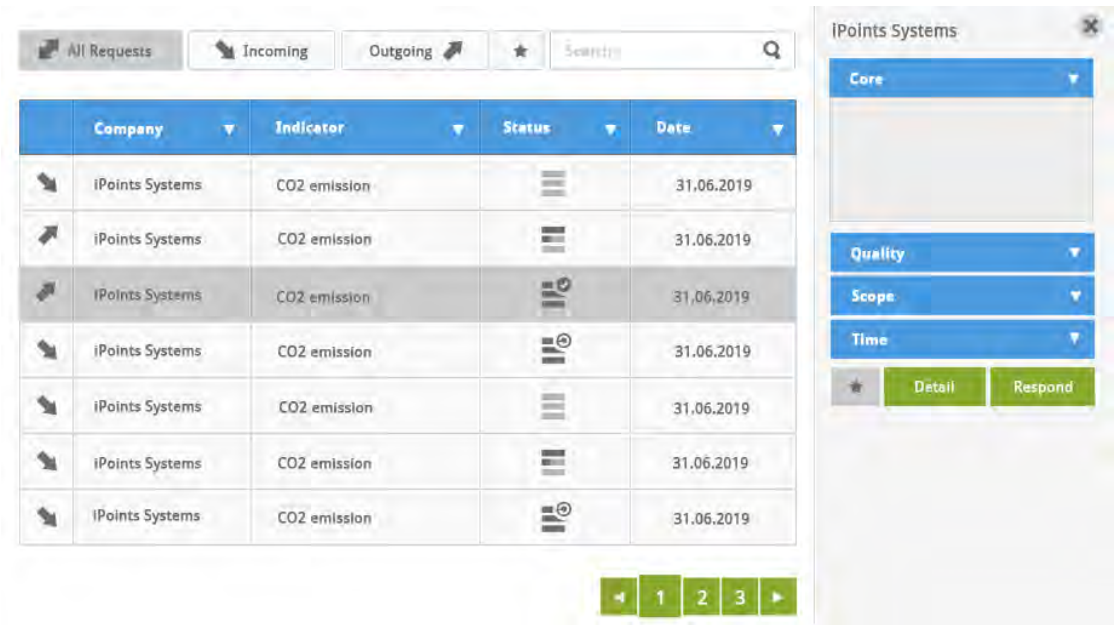


Abbildung 7.12: High-fidelity-Mockup einer Aggregation inkl. Tabelle

Tabellen besitzen einen blauen Kopf. Der Spaltenkopftext ist weiß. Ein Dreieck impliziert, dass eine Sortierung möglich ist. Abbildung 7.12 zeigt eine Tabelle innerhalb einer Aggregation. Hier gibt es zusätzlich die Möglichkeit, Objekte zu selektieren sowie eine Übersicht. Unter der Tabelle befindet sich das Pagination-Element.

Detailansicht

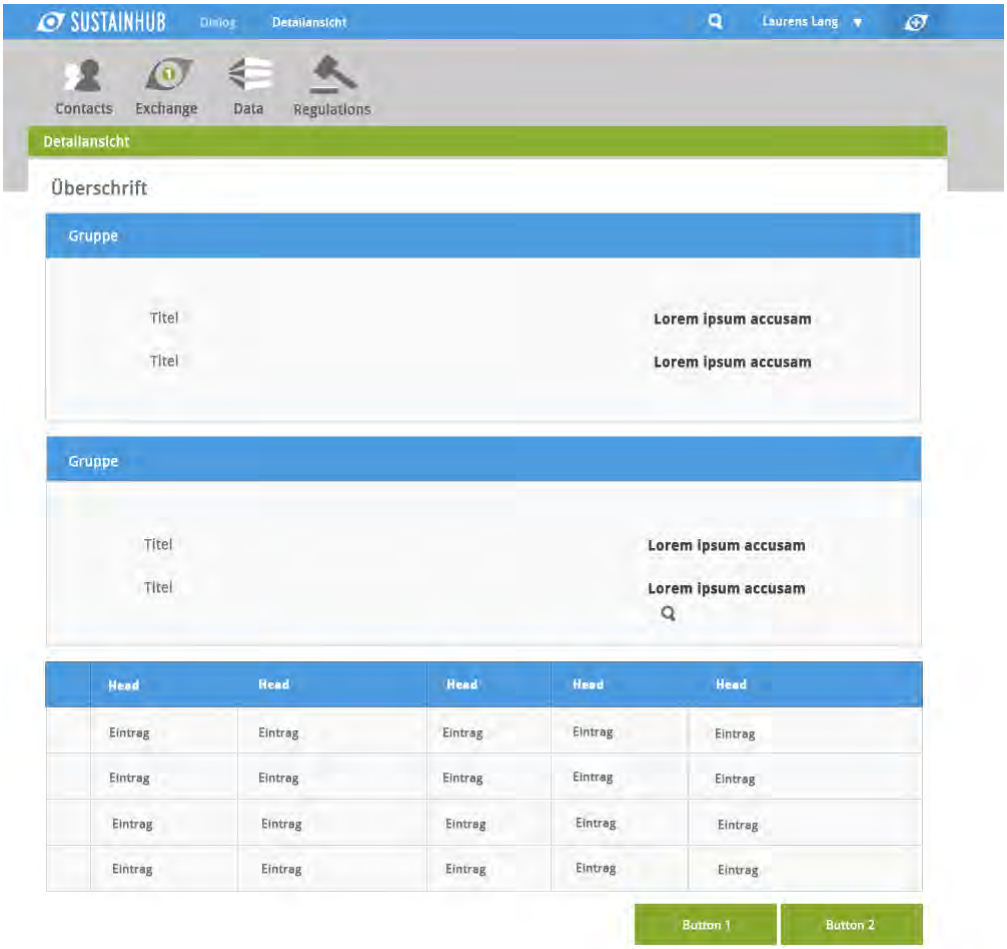


Abbildung 7.13: High-fidelity-Mockup einer Detailansicht

Abbildung 7.13 zeigt den High-fidelity Mockup einer Detailansicht. Auf die Überschrift folgen die gruppierten Informationen. Darunter können Tabellen, wie sie im vorherigen Kapitel beschrieben wurden, z.B. für assoziative Objekte und Buttons platziert sein. Darunter können sich Buttons befinden. Im Falle eines Datenaustauschs werden beispielsweise alle Daten strukturiert in Gruppen ausgegeben. In den Tabellen befinden sich assoziative Requests. Über die unteren Button kann zum Beispiel der entsprechende Response eingeleitet werden (vgl. Kapitel 6.2.4).

Organisationsmodell

Abbildung 7.14 zeigt das Organisationsmodell. Die Seite ist unterteilt in die Übersicht und rechts daneben das Informations- und Bearbeitungsfeld. Es können zuerst Departments (OrgUnits) erstellt werden. Hierfür öffnet sich nach dem Klick auf den grünen Button ein Formular auf der rechten Seite. Organisations-Units, -Agents, -Roles und -Abilities können erstellt werden. Das Formular zur Erstellung einer OrgPosition wird über den blauen "Add Position"-Button geöffnet. Ist diese erstellt, erscheint sie unter ihrem Department blau untermalt. Daneben können in ein Tokenfield OrgAgents eingetragen werden. Werden Departments, Agents oder Roles ausgewählt erscheinen im rechten Feld Informationen zum jeweiligen Objekt.

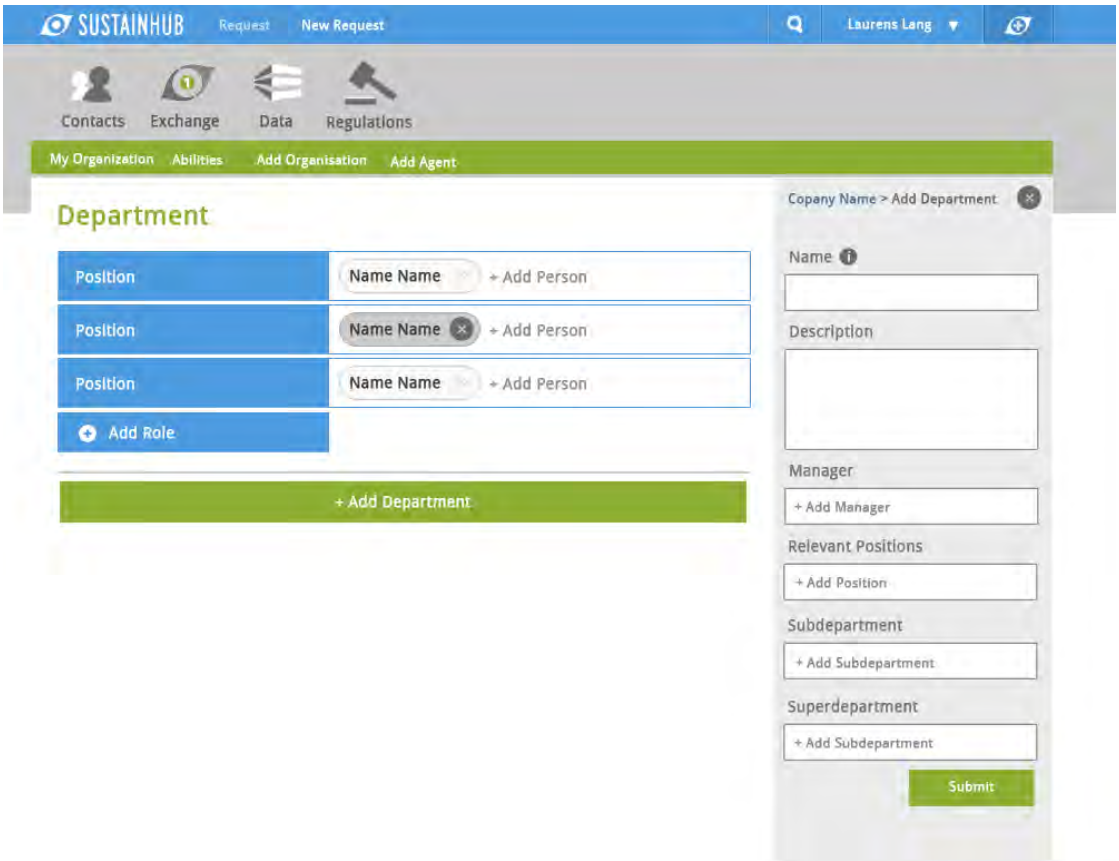


Abbildung 7.14: High-fidelity-Mockup des Organisationsmodells

8

Alternative Konzepte

Im Laufe der Entwicklung des Usability-Konzepts sind verschiedene Entwürfe entstanden. Durch das iterative Vorgehen wurden einige Konzepte überdacht, optimiert oder geändert. Dabei entstanden oftmals mehrere Entwürfe, von denen manche nicht in den endgültigen Styleguide mit eingeflossen sind. Während sich verschiedene Entwürfe zu einer Aufgabe aus früheren Phasen meist grundsätzlich unterscheiden, sind aufgrund des iterativen Vorgehens in späteren Phasen meist nur ästhetische Unterschiede entscheidend. Dieses Kapitel stellt die alternativen Konzepte vor und gibt somit einen Einblick in den Entwicklungsvorgang.

8.1 Seitenaufbau

Besonders entscheidend für die Qualität des Benutzerschnittstelle ist der Seitenaufbau. Abbildung 8.1 zeigt einen Ausschnitt aus den Mockups, die entstanden sind.

1. Von Anfang an bestand die Idee einer Toolbar. Die grundlegende Idee war, die Suche und die Navigation links und die Subnavigation nach oben zu platzieren. Zu diesem Zeitpunkt war noch nicht endgültig klar, welche Funktionen SustainHub anbieten wird. Die Idee wurde verworfen, weil die Navigation zu viel Platz einnimmt, wenn noch weitere Navigationspunkte hinzugekommen wären. Der Nutzer hätte eventuell scrollen müssen, um die unteren Navigationspunkte zu erreichen. Außerdem hätte der Zusammenhang zwischen

2. Daraufhin wurde die Toolbar herausgenommen und die Navigation dorthin versetzt. Diese Verschiebung räumt mehr Platz für die Navigation ein. Das Benutzermenü wurden unter die Suche platziert. Diese Aufteilung bietet allerdings keinen geeigneten Platz für einen Breadcrumb. Außerdem ist der Zusammenhang zwischen Navigation und Subnavigation durch die räumliche Trennung nicht gegeben.

3. Die Toolbar wurde wieder eingebaut, auf der diesmal auch das Breadcrumb positioniert wurde. Das Problem der Größe der Navigation wurde gelöst, indem die Navigationselemente mit Text anstatt Icons umgesetzt werden. Darunter leidet die Intuitivität. Die Suche wird durch die Platzierung direkt unter dem Logo eine unbgegründet in den Fokus gerückt.

4. Somit wurde die Suche nach rechts oben verschoben. Dort wurde ebenfalls die Navigation und direkt darunter die Subnavigation platziert. Dieser Aufbau wurde als Grundlage für den endgültigen genutzt.

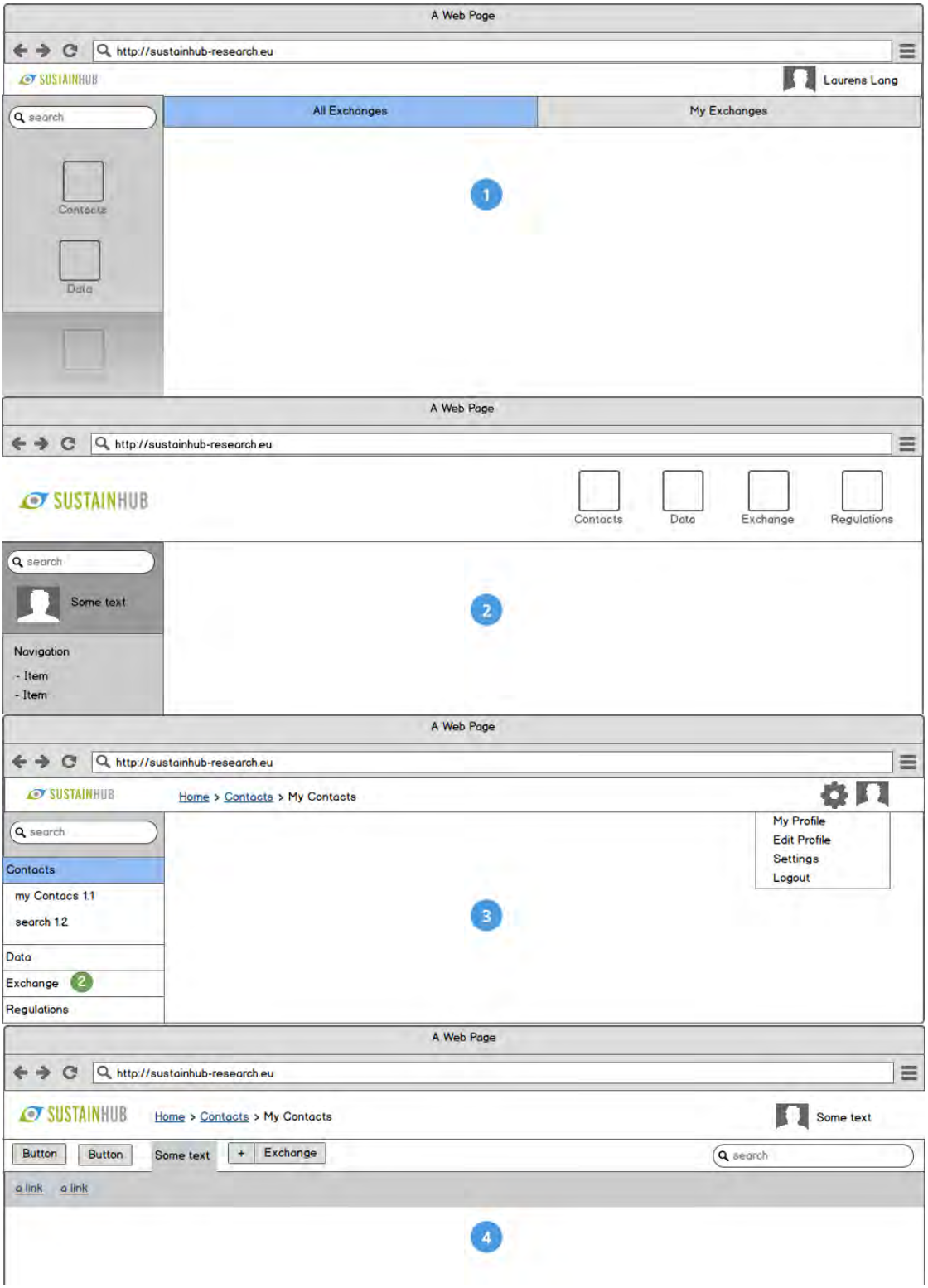


Abbildung 8.1: Mockups der alternativen Konzepte im Bezug auf den Seitenaufbau

8.2 Wizards

Auch die Wizards wurden anfangs anders entworfen. Der Fortschrittsindikator des Wizards wurde als einfache Buttonbar dargestellt. Der aktuelle Teilschritt wurde farblich hervorgehoben. Hierbei können Unklarheiten entstehen, denn es wird nicht angezeigt, welche Schritte bereits erfolgreich absolviert sind. Weiterhin leidet unter der Simplizität die Ästhetik.

Abbildung 8.2: Mockup des alternativen Wizards

8.3 Organisationsmodell

Abbildung 8.3 zeigt einen Mockup des alternativen Organisationsmodells in einer frühen Entwicklungsphase. Es wurde zuerst versucht die Abteilungen als rechteckige Gruppierungen darzustellen, in der sich die Positionen befinden. Diese Umsetzung wurde

verworfen, weil sich zwar Positionen anlegen lassen, jedoch sich die Personen erst in einer getrennten Ansicht erstellen bzw. ändern lassen. Diese Ansicht wäre über "Personnel" in den oberen Tabs erreichbar. Die Lösung macht das Organisationsmodell unübersichtlich und die Interaktion unnötig kompliziert.

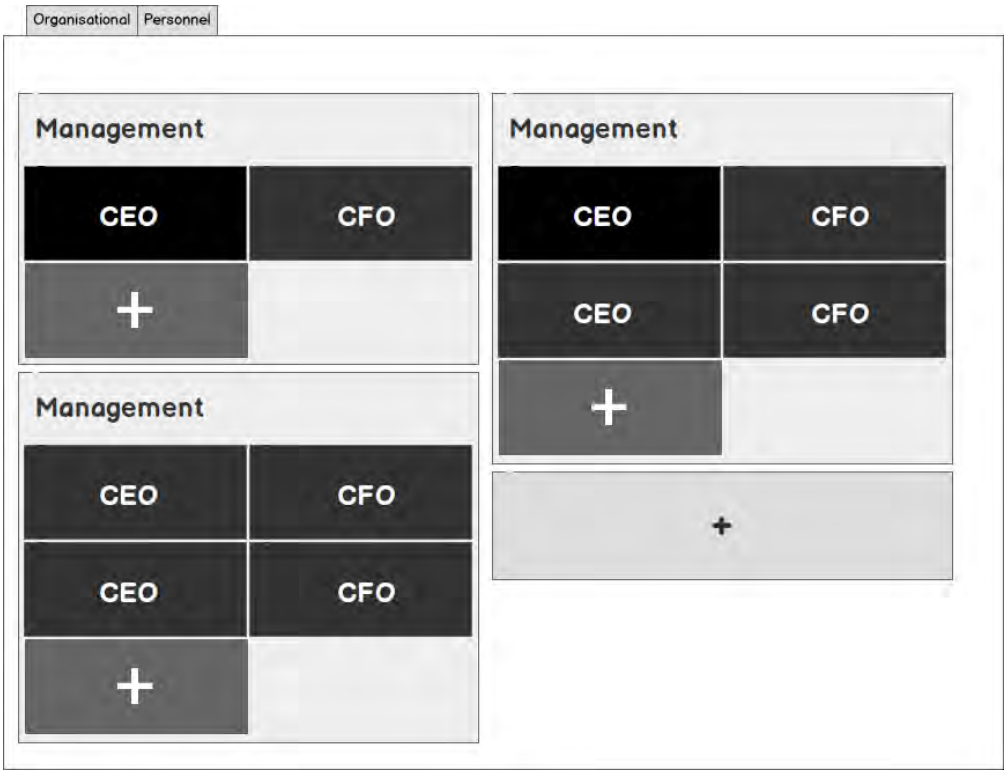


Abbildung 8.3: Mockup des alternativen Organisationsmodells in einer frühen Entwicklungsphase

Es wurden auch einige Umsetzungen in späteren Phasen der Entwicklung verworfen. Da in diesen Phasen das Konzept bereits optimiert wurde, unterscheiden sich die Mockups nur in wenigen Punkten. Abbildung 8.4 zeigt ein alternatives Organisationsmodells in einer späten Entwicklungsphase. Um die Abteilungen (Departments) abzugrenzen wurden die Gruppierungselemente benutzt. Wenn der Nutzer allerdings eine Unterabteilung erstellen will, müsste folglich darin ein weiteres Gruppierungselement platziert werden. Bei einer feinstrukturierten Organisation müssten mehrere Gruppen ineinander

verschachtelt angezeigt werden, was sehr unübersichtlich ist. Die Benutzeroberfläche wirkt dann überladen ohne einen Verständnisgewinn zu erzeugen.

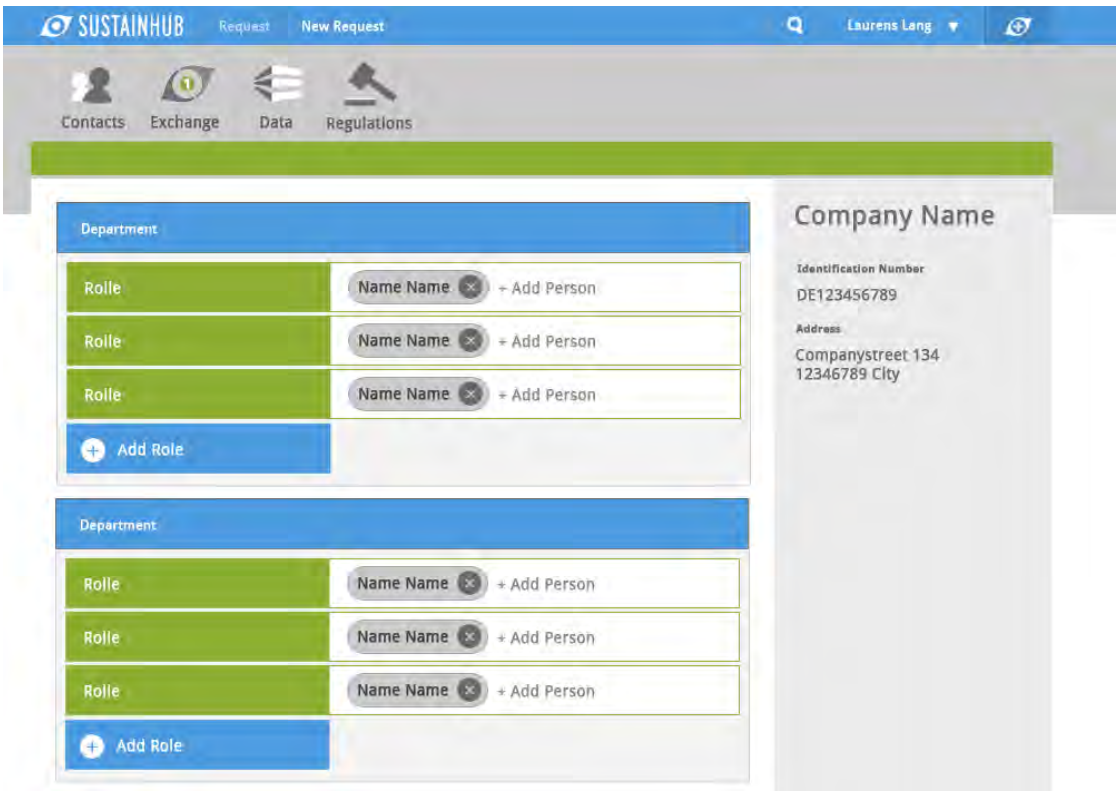


Abbildung 8.4: Mockup des alternativen Organisationsmodells in einer späten Entwicklungsphase

9

Implementierung eines Usability-Konzepts

Beim Implementieren von Usability-Konzepten, kommt es oft zu Problemen. Entwickler sind sich der Wichtigkeit von Usability nicht bewusst und arbeiten häufig an dem Konzept vorbei. Im Falle von SustainHub wird die Benutzeroberfläche getrennt vom Theme entwickelt. Zwar gibt es die Möglichkeit in der Anwendungslogik den einzelnen Komponenten Stylenamen, die im Theme auf Selektoren referenzieren, zu definieren. Dies sollte allerdings auf ein Minimum reduziert werden, um die Anwendungsentwicklung nicht zu komplizieren.

Im folgenden Kapitel wird eine systematische Vorgehensweise für das Implementieren von Usability-Konzepten vorgestellt. Dieses eignet sich besonders für UI-Konzepte, die mithilfe SCSS realisiert werden. Mit wenig Mehraufwand kann diese Vorgehensweise auch für Webapplikationen, die nur CSS zur Gestaltung verwenden, genutzt werden.

Daraufhin wird anhand des SustainHub-Wizards ein Proof of Concept durchgeführt.

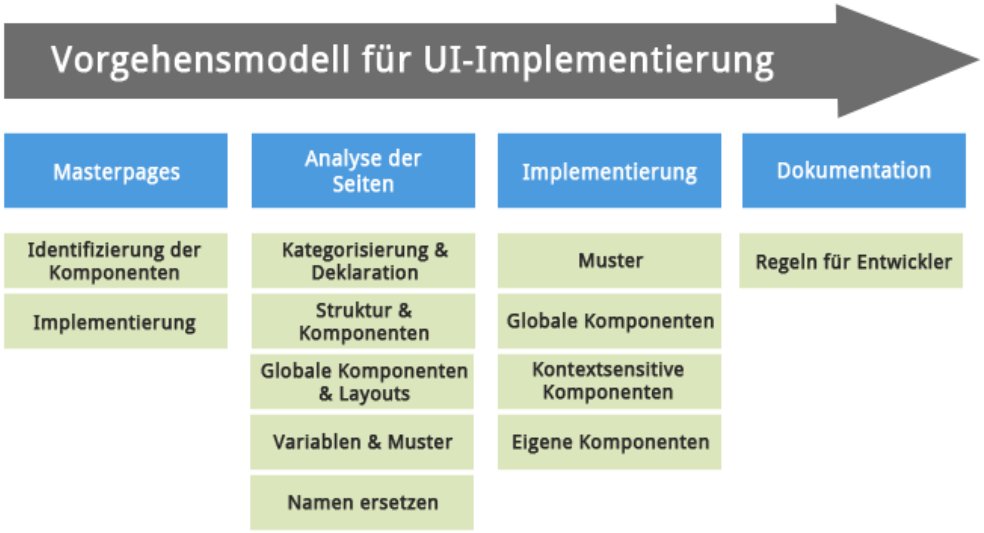


Abbildung 9.1: Vorgehensweise für die Implementierung von Usability-Konzepten

9.1 Vorgehensmodell zur Implementierung von UI-Konzepten

Ziel ist es, das Usability-Konzept für Entwickler auf einige wenige Regeln zu reduzieren. Um dies zu erreichen, kann die Verschachtelung von SCSS-Selektoren genutzt werden (vgl. Kapitel 3.1.5). Hierfür wird jeder Seite ein SCSS-Selektor zugewiesen. In der dazugehörigen Klasse werden alle Komponenten, die sich auf der Seite befinden, die Gestaltung definiert. Dies führt dazu, dass sich die Regeln auf ein Minimum reduzieren, weil in dem Code der Benutzeroberfläche nur einmal dieser Selektor angegeben werden muss. Ein weiteres Ziel ist es, Wiederholungen innerhalb des Codes zu minimieren. Dazu werden die Seiten, die gleich aufgebaut sind und die gleichen Komponenten beinhalten, in eine SCSS-Klasse zusammengefasst. Weiterhin können wiederholende Eigenschaften und Muster identifiziert werden. Mithilfe Mixins und Variablen (vgl. Kapitel 3.1.5) werden diese einmalig definiert und können beliebig oft in anderen SCSS-Klassen aufgerufen werden. Um diese Methoden systematisch zu unterstützen, wurde das Vorgehensmodell, das in Abbildung 9.1 zu sehen ist, entwickelt. Die einzelnen Schritte werden in den folgenden Kapiteln vorgestellt.

9.1.1 Masterpage

Webapplikationen haben meist einen einheitlichen Rahmen, der auf allen Seiten gleich ist. Meist befinden sich darin die Navigation und Funktionen wie z.B. die Suche. In der Phase *Masterpage* werden die Masterpages analysiert und umgesetzt. In der Ebene *Identifizierung der Komponenten* werden hierfür die Komponenten, aus denen eine Masterpage besteht, gesammelt. Da sich die Komponenten hier meist von den gleichen Komponenten auf den individuellen Seiten unterscheiden, sollte ein Stylename für das Layout der Masterpage gesetzt werden. Die Listenelemente werden durch die entsprechenden SCSS-Selektoren ersetzt. Daraufhin folgt in *Implementierung* die Umsetzung.

9.1.2 Analyse der Seiten

Im UI-Konzept lassen sich einige Seiten finden, die gleich aufgebaut sind und die gleichen Komponenten beinhalten. In der Phase *Kategorisierung und Deklaration* werden diese klassifiziert und ein Name für jede Kategorie gefunden. Seiten, die sich nicht gruppieren lassen, werden individuell benannt. In *Struktur und Komponenten* wird jede Kategorie jeweils analysiert. Zuerst muss der Aufbau geklärt werden und welche Komponenten sich darin befinden. Als Ergebnis dieses Prozessschrittes sollte eine hierarchisierte Liste der Komponenten und Layouts jeder Kategorie erstellt werden. Sollten sich gleiche Komponenten einer Hierarchiestufe optisch unterscheiden, sollte diesen jeweils ein Name zugewiesen werden. Komponenten, die nicht dem Standardrepertoire des Frameworks angehören, sollten gesondert gesammelt werden. In der Phase der *Eliminierung* wird versucht Codewiederholungen frühzeitig zu erkennen und zu eliminieren. Ist eine Komponente in einer tieferen Hierarchiestufe in der gleichen Form vorhanden wie in einer höheren, kann auf die tiefere verzichtet werden. Während der Phase *Globale Komponenten und Layouts* gleiche Komponenten und Layouts aus allen Kategorien miteinander verglichen. Unterscheiden sich diese nicht, werden diese aus der Liste gestrichen und auf die Liste *Globale Komponenten und Layouts* gestellt. Zudem gibt es Eigenschaften und Muster, d.h. Ansammlungen von Eigenschaften, die bei unterschiedlichen Komponenten auftauchen. Diese Muster werden in der Phase der

Muster und Variablen identifiziert, nummeriert und benannt. Die Layouts und Komponenten, die betroffen sind, sollten auf die Muster referenzieren. Das Ergebnis der Analyse sind fünf Listen:

- Muster
- Variablen
- Globale Komponenten
- Hierarchisierte Liste der Komponenten und Layouts
- Eigene Komponenten

In *Namen ersetzen* werden die Standardkomponentennamen daraufhin durch ihre SCSS-Selektoren ersetzt. Diese gibt die Implementierungstechnologie vor. Im Falle von Vaadin lautet der Selektor für ein Panel beispielsweise ”v-panel”.

9.1.3 Implementierung

Die vier Listen werden in dieser Reihenfolge im Prozessschritt *Implementierung* realisiert. Sie dienen als Grundlage der Themedatei. Einzelne Eigenschaften aus der Liste *Muster und Variablen* werden als Variablen und mehrere Eigenschaften als Mixins deklariert und umgesetzt. Daraufhin werden die *globalen Komponenten* als Selektoren in der Themedatei angelegt und realisiert. Diese Reihenfolge ist wichtig, um Wiederholungen im SCSS-Code und somit den Aufwand zu minimieren, da globale Komponenten, kontextsensitive Komponenten, Layouts und eigene Komponenten solche Muster enthalten können.

9.2 Proof of Concept

Unter Proof of Concept versteht man eine Analyse in der die prinzipielle Machbarkeit einer technisch komplexen Problemstellung innerhalb eines Projektes analysiert wird. Somit werden Entscheidungen vereinfacht, Risiken minimiert und kritische Anforderungen an das System enttarnt [Wik13]. In diesem Kapitel wird ein Prototyp eines Teils der

Benutzeroberfläche durchgeführt. Jeder Schritt des Vorgehensmodell wird mindestens einmal ausgeführt.

9.2.1 Masterpages

In diesem Kapitel wird beschrieben, wie die Masterpage von SustainHub aus Usability-sicht umgesetzt wurde. Die Masterpage besteht aus folgenden Komponenten:

- VerticalLayout
  - HorizontalLayout
    - Image (Logo)
    - Button (Suchbutton)
    - TextField (Suchfeld)
    - MenuBar (Benutzermenü und Deeplinkmenü)
      - MenuItem (Benutzermenüitems, Deeplinks)
    - Label (Ausgeloggt)

Dem VerticalLayout wird ein Stylename zugewiesen, die Elemente in der Liste werden durch die entsprechenden SCSS-Selektoren ersetzt und definiert.

9.2.2 Analyse der Seiten

In diesem Kapitel folgt das Proof of Concept des Schrittes ”Analyse der Seiten”. Dieses beinhaltet die Teilschritte *Kategorisierung und Deklaration*, *Struktur und Komponenten*, *Globale Komponenten und Layouts*, *Variablen und Muster* und *Namen ersetzen*, die jeweils in den folgenden Kapitel ausgeführt werden.

Kategorisierung und Deklaration

Die Seiten der Aufgaben *New Request*, *Registrierung*, *Push Data* und *Response* ähneln sich stark in ihrem Aufbau und den beinhalteten Komponenten und Layouts. Sie werden somit in eine Kategorie namens *wizard* zusammengefasst.

Struktur und Komponenten

Alle Layouts bestehen aus einem VerticalLayout, indem der Fortschrittsindikator, mehrere Panels und ein HorizontalLayout mit Buttons befinden. Im Panel befinden sich Formularfelder. Im Falle von *New Request*, *Registrierung* und *Push Data* handelt es sich um ein zweispaltiges Layout, in der links das Label und rechts die Formularfelder angezeigt werden. Hierfür eigenet sich ein FormLayout. Auf den Seiten von *Response* wird für die Anzeige der Requestdaten (vgl. Kapitel 6.2.4) eine weitere Spalte benötigt, was mit einem normalen FormLayout nicht möglich ist. Ein solches Layout wird mit einem VerticalLayout für die Spalten und einem HorizontalLayout für die Zeilen realisiert. In diesen Layouts befinden sich die Labels und die Formularfelder.

Folgende Hierarchie ergibt sich für die Kategorie *wizard*:

VerticalLayout

- ProgressIndicator
- Panel
  - Label
  - FormLayout
    - Label
    - Textfield
    - TextArea
    - DateField
    - CheckBox

        * PasswordField
        * ComboBox
        * TokenField
    - VerticalLayout
        * HorizontalLayout
            · Label
            · Textfield
            · TextArea
            · DateField
            · CheckBox
            · PasswordField
            · ComboBox
            · TokenField
- HorizontalLayout
    - Button

Eliminierung

In diesem Fall ist zwar eine Eliminierung aller Formularkomponenten in Response möglich, indem diese auf die eine höhere Ebene verschoben werden. Allerdings ist sie nicht sinnvoll, da sich einige Komponenten in ihrer Breite unterscheiden. Ausgenommen davon ist die Checkbox und das DateField. Diese werden nun innerhalb des Panels definiert.

Muster und Variablen

Für die anderen Formularkomponenten ist sinnvoller diese als Muster zu definieren und die Breite als Parameter zu setzen. Als Variablen werden die drei Hauptfarben blau, grün und grau gesetzt.

Globale Komponenten und Layouts

Die Buttons sind bis auf einige spezielle Buttons im gesamten SustainHub gleich. Diese können somit ebenfalls aus der Liste gestrichen werden und außerhalb ihres Kontextes als globale Komponenten deklariert werden.

Es ergeben sich folgende Listen:

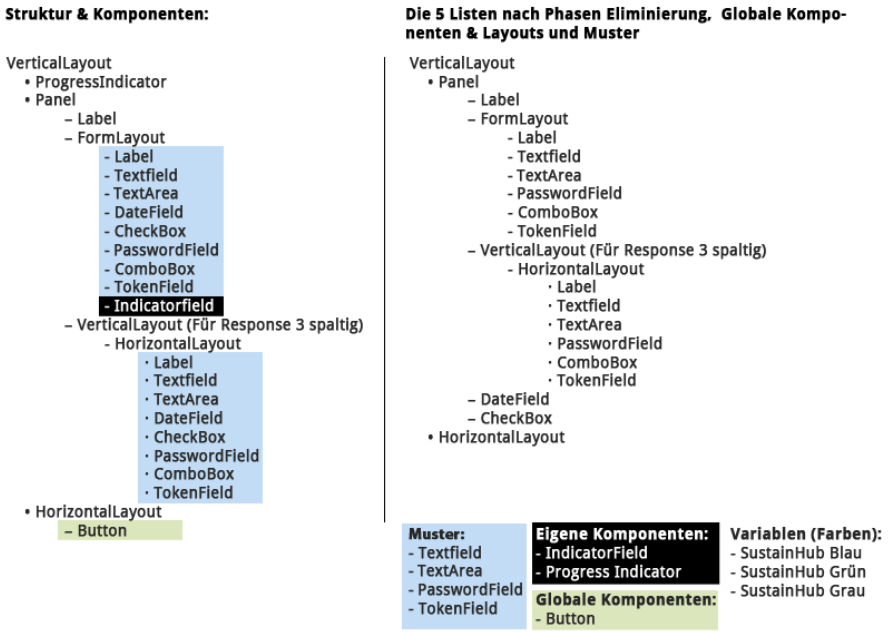


Abbildung 9.2: Listen nach Analyse

Namen ersetzen

Die Komponenten haben spezifische SCSS-Selektoren, die von Vaadin vorgegeben werden. Die Liste wird durch diese ersetzt. Dadurch ergibt sich die Struktur der Theme-Datei (vgl. 3.2.1).

9.2.3 Implementierung

In dieser Phase folgt die Implementierung. Es werden hier zu jedem einzelnen Punkt Code-Snippets vorgestellt und erklärt. Die Codes dienen nur als Beispiel und sind nicht vollständig.

Muster und Variablen

Als Variablen werden drei wiederkehrenden Hauptfarben definiert. Wie in Kapitel 9.2.2 bereits erwähnt, taucht das TextField in gleicher Darstellung auf, allerdings mit zwei unterschiedlichen Größen. Im Beispiel wird deshalb dafür ein Mixin erstellt, welches die Größe als Parameter setzt.

```
/* Variablen */
$susBlue: #4d9bdf;
$susGreen: #8dae33;
$susGreyText: #434343;

/* Muster fuer TextField */
@mixin textfieldmixin($breite){
        border:1px solid $susblue;
        color: $susGreyText;
        line-height:40px;
        padding:20px 20px 20px 15px;
        width: $breite;
}
```

Globale Komponenten

Im folgenden werden die als globale Komponenten identifizierten Buttons implementiert. Ein Button besitzt neben den Größen und der grünen Farbe eine weitere Klasse, in der die weiße Schrift definiert wird. Der Mouse-over-Effekt ändert nur den Schriftstil,

weshalb mittels @extend die Eigenschaften des normalen Buttons geerbt werden und zusätzlich die Schrift fett ausgegeben wird:

```
.v-button {
        background-color:$susGreen;
        width:150px;
        height:40px;
        line-height:40px;

        .v-button-caption {
        color:white;
        font-size: 14px;
  }
}

/* Mouse-over-Effekt des Buttons */
.v-button:hover {
        @extend .v-button;

        .v-button-caption {
        font-weight: bold;
  }
}
```

Kontextsensitive Komponenten

In diesem Schritt werden nun auch die *kontextsensitiven Komponenten* gestaltet. Hierbei wird die Möglichkeit in SCSS Selektoren zu verschachteln, genutzt. An oberster Stelle muss der Selektor *wizard* sein. In Kapitel 9.2.2 wurde gezeigt, dass die TextFields innerhalb verschiedener Layouts unterschiedliche Größe haben. Im Beispiel werden die kontextsensitiven Textfelder, die vorher als Muster bereits definiert wurden, umgesetzt:

```
.wizard{
        /*  Weitere Verschachtelungen werden hier nicht aufgefuehrt
                Texfeld in FormLayout */
        .v-formlayout{
                v-textfield{
                        @include textfieldmixin(400px);
                }
        }
        /* Texfeld in HorizontalLayout */
        .v-verticallayout{
                .v-horizontallayout{
                        v-textfield{
                                @include textfieldmixin(250px);
                        }
                }
        }
}
```

Eigene Komponenten

Als Beispiel einer eigenen Komponente wird der Fortschrittsindikator herangezogen. Für diesen muss mit Java die Komponente konstruiert werden. Die Komponente wird mit ProgressIndicator progressindicator = new ProgressIndicator(2); aufgerufen. Dadurch werden sieben Bilder, die die Teilschritte und die Trennstriche beinhalten, in einem HorizontalLayout angeordnet. Bevor diese als UI-Komponenten (Images) mit eingebunden werden können, müssen deren Pfade (FileResources) erst definiert werden. Der Code-Snippet beschränkt sich auf den zweiten Schritt des Fortschrittsindikator (i = 2). In diesem wird der erste abgehakt, der zweite aktiv und die beiden anderen Schritte ausgegraut angezeigt. Der erste wird grün und die anderen Trennstriche grau angezeigt. Die weiteren Schritte lassen sich analog dazu erstellen.

```
1  class ProgressIndicator extends CustomComponent {
2      public ProgressIndicator(int i) {
3
4          HorizontalLayout progressIndicator = new HorizontalLayout();
5
6          // Hiermit wird ein SCSS-Selektor der Komponente zugewiesen:
7          progressIndicator.setStyleName("progressIndicator");
8
9          /*
10          * Der Progressindicator besteht aus folgenden Images:
11          * Die Stepbilder sind die Kreise, die between Bilder, die betweenbilder die Striche dazwischen
12          */
13         Image step1, between1, step2, between2, step3, between3, step4;
14
15         /** Images als FileResources:
16          *      gruener und blauer Strich zwischen den Kreisen
17          */
18
19         FileResource barGreyResource = new FileResource(new File(folder + "/sustheme/img/pi/barGrey.png"));
20         FileResource barGreenResource = new FileResource(new File(older + "/sustheme/img/pi/barGreen.png"));
21
22         // Kreise, die in verschiedenen Zustaenden verschiedene Resourcen haben und die Striche dazwischen
23                 FileResource step1res, between1res, step2res, between2res, step3res, between3res, step4res;
24
25                 switch(i){
26                         case 1:
27                                 /*
28                                  *       Dieser Case wird analog zum zweiten implementiert
29                                  */
30
31                 case 2:
32                         System.out.println("Step 2");
33
34                                 // Step 1 ist erledigt, Step 2 ist aktiviert, die anderen ausgegraut
35                         step1res = new FileResource(new File(folder + "/sustheme/img/pi/basic_check.png"));
36                         step2res = new FileResource(new File(folder + "/sustheme/img/pi/quality_active.png"));
37                         step3res = new FileResource(new File(folder + "/sustheme/img/pi/scope_grey.png"));
38                         step4res = new FileResource(new File(folder + "/sustheme/img/pi/basic_grey.png"));
```

```
39
40                      // In diesem Fall sind alle Striche grau
41
42                      between1res = barGreyResource;
43                      between2res = barGreyResource;
44                      between3res = barGreyResource;
45
46
47
48                  break;
49                      /*
50                       *       Weitere Cases werden analog zum zweiten Case programmiert.
51                       */
52
53                      step1 = new Image("", step1res);
54                      between1 = new Image("", between1res);
55                      step2 = new Image("", step2res);
56                      between2 = new Image("", between2res);
57                      step3 = new Image("", step3res);
58                      between3 = new Image("", between1res);
59                      step4 = new Image("", step4res);
60
61                      between1.setStyleName("wizardbetween");
62                      between2.setStyleName("wizardbetween");
63                      between3.setStyleName("wizardbetween");
64
65                      // der Komponenten-Root muss gesetzt sein
66                      setCompositionRoot(progressIndicator);
```

Mithilfe der SCSS-Klasse *wizardbetween* werden die Trennstriche mittig angezeigt.

9.2.4 Dokumentation

Durch gezielte Nutzung von Verschachtlung in SCSS muss der Entwickler der Benutzeroberfläche nur 2 Regeln einhalten.

Regeln für Entwickler

1. Die Benutzeroberflächen müssen gemäß der Hierarchielisten zusammengesetzt werden.
2. Der Stylename des VerticalLayout muss mit verticallayout.setStyleName("wizard"); auf *wizard* gesetzt werden.

9.3 Ergebnis

Wird nun ein Wizard regelkonform zusammengesetzt, werden die Komponenten richtig angeordnet und gestaltet. Abbildung 9.3 zeigt einen Screenshot eines realisierten Beispielwizards.

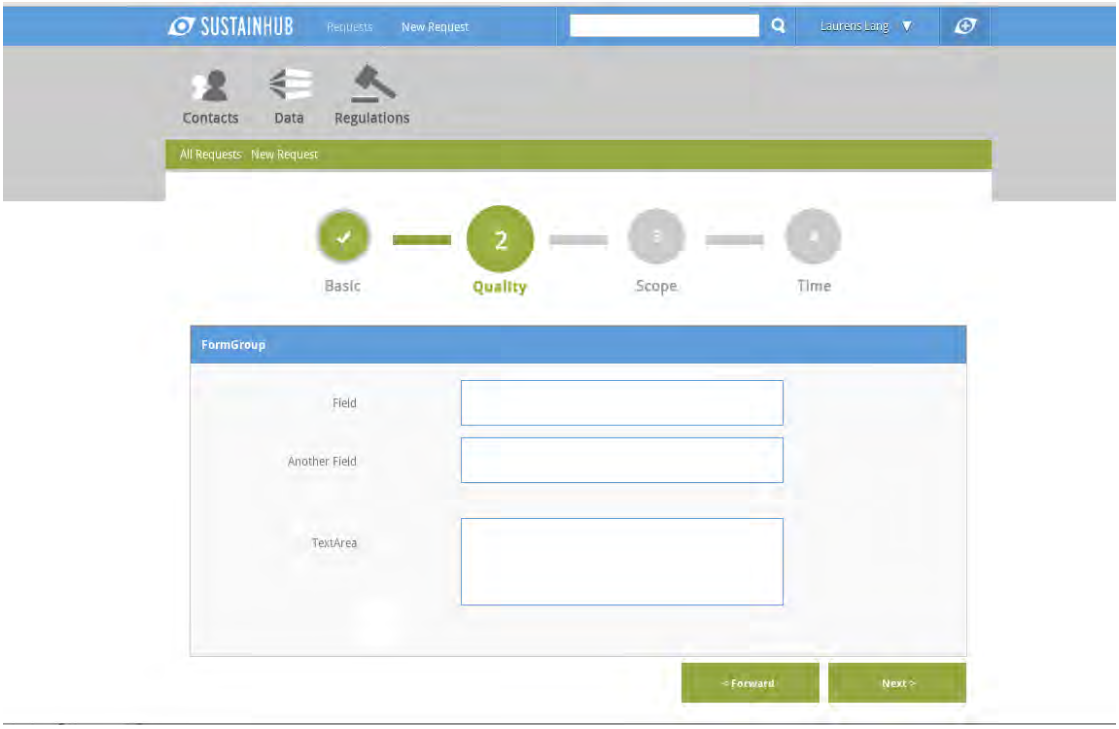


Abbildung 9.3: Screenshot eines Beispielwizards

Abbildungsverzeichnis

2.1 Architektur von SustainHub: Komponenten und Subkomponenten (Ausgegraute Komponenten sind ”out-of-scope”) 8
2.2 Datenmodell von SustainHub) 10

3.1 Variablen in SCSS 16
3.2 Verschachtelung von Selectors und Properties in SCSS 16
3.3 Mixins in SCSS 17
3.4 Vererbung in SCSS 17

4.1 Usabilitykriterien und deren Relation 20
4.2 IFIP Modell 22
4.3 Gestaltgesetze [Dah05] 23
4.4 Referenzmodell nach Daimler AG 26

5.1 Use-Case-Diagramm: Administration und System 36
5.2 Use-Case-Diagramm: Data Exchange 38
5.3 Use-Case-Diagramm: Kontakte 41
5.4 Use-Case-Diagramm für das Administration Interface 43
5.5 Verteilung von Bildschirmauflösungen der Monitore im Januar 2013 laut w3school 45
5.6 Browserverteilung im August 2013 laut w3school 46
5.7 Farbkonzept anhand SustainHub Logo 47

6.1 Low-fidelity-Mockup einer Detailseite 50

6.2 Low-fidelity-Mockup eines Wizards 51
6.3 Low-fidelity Mockup für Formularkomponenten 53
6.4 Low-fidelity Mockup Walkthrough für Tokenfield 54
6.5 Grober Benutzeroberflächenaufbau 57
6.6 Robustnessdiagramm: Registrierung 58
6.7 Entitätskategorie ”Participant Data” aus dem Datenmodell 59
6.8 Low-fidelity Mockup Walkthrough für Aggregation der Dataexchanges . . 60
6.9 Iconentwurf für Selektionsoperationen 61
6.10 Iconentwurf für Status eines Request 62
6.11 Low-fidelity Mockup für Quick-Overview eines offenen Requests 62
6.12 Entitätskategorie ”Request Data” aus dem Datenmodell 63
6.13 Low-fidelity-Mockup für Basic-Seite des Request-Wizards 64
6.14 Iconentwurf für Indikator-UI-Element 64
6.15 Low-fidelity-Mockup-Walkthrough des Indikator-UI-Element auf der Basic-Seite des Request-Wizards 65
6.16 Low-fidelity-Mockup für Expertenrequest 66
6.17 Low-fidelity-Mockup für Quality-Seite des Response-Wizards 67
6.18 Low-fidelity-Mockup für Kontakte 69
6.19 Low-fidelity-Mockup für Data 70
6.20 Low-fidelity-Mockup für Dasboard 71
6.21 Low-fidelity-Mockup für Costumer-Data 72
6.22 Low-fidelity-Mockup für IHS anzeigen 73
6.23 Low-fidelity-Mockup für IHS bearbeiten 74

7.1 Typographiegrößen im Überblick 76
7.2 Farbkonzept von SustainHub: Abstufungen der 3 Grundfarben 77
7.3 Favicon: Links die Rastergrafik, rechts in Originalgröße im Browser 77
7.4 User-Interface-Komponenten Kit 78
7.5 High-fidelity Mockup Walkthrough für Tokenfield 80
7.6 High-fidelity der Indikatorauswahl-Komponente 80
7.7 High fidelity Mockup einer Gruppierung von Formularfeldern 81
7.8 High fidelity Mockup der Pagination-Komponente 82

7.9 High-fidelity-Mockup des User-Interface-Aufbaus 82
7.10 High-fidelity-Mockup der Toolbar 83
7.11 High-fidelity-Mockup eines Wizards 84
7.12 High-fidelity-Mockup einer Aggregation inkl. Tabelle 85
7.13 High-fidelity-Mockup einer Detailansicht 86
7.14 High-fidelity-Mockup des Organisationsmodells 87

8.1 Mockups der alternativen Konzepte im Bezug auf den Seitenaufbau . . . 91
8.2 Mockup des alternativen Wizards 92
8.3 Mockup des alternativen Organisationsmodells in einer frühen Entwicklungsphase 93
8.4 Mockup des alternativen Organisationsmodells in einer späten Entwicklungsphase 94

9.1 Vorgehensweise für die Implementierung von Usability-Konzepten 96
9.2 Listen nach Analyse 102
9.3 Screenshot eines Beispielwizards 107

Literaturverzeichnis

[Ale13] ALEXA: *Alexa Top 500 Global Sites*. http://www.alexa.com/topsites. Version: 2013, Abruf: 06.09.2013

[BK09] BUSCH, Marianne ; KOCH, Nora: Rich Internet Applications - State-of-the-Art. (2009)

[CWE13] CATLIN, Hampton ; WEIZENBAUM, Nathan ; EPPSTEIN, Chris: *Sass - Syntactically Awesome Stylesheets*. http://sass-lang.com/guide. Version: 2013, Abruf: 27.07.2013

[Dah05] DAHM, Markus: *Grundlagen der Mensch-Computer-Interaktion*. 2005

[GMS+13] GRAMBOW, Gregor ; MUNDBROD, Nicolas ; STELLER, Vivian ; REICHERT, Manfred ; SCHIFFLEITNER, Andreas ; BLEY, Thomas ; FEICK, Christian: State-of-the-Art and Requirements for Collecting and Managing Sustainability Data Along Today's Supply Chains. In: *6th Int'l Conference on Life Cycle Management (LCM'13)* (2013)

[GMSR13] GRAMBOW, Gregor ; MUNDBROD, Nicolas ; STELLER, Vivian ; REICHERT, Manfred: Challenges of Applying Adaptive Processes to Enable Variability in Sustainability Data Collection. In: *3rd Int'l Symposium on Data-Driven Process Discovery and Analysis (SIMPDA'13)* (2013)

[Goo13] GOOGLE: Droid Fonts. (2013). http://www.droidfonts.com/, Abruf: 08.08.2013

[Gro10] GROENROOS, Marko: *Book of Vaadin - Vaadin 7*. https://vaadin.com/book. Version: 2010, Abruf: 20.10.2013

[ISO06] ISO: Ergonomics of human-system interaction – Part 110: Dialogue principles. (2006)

[Jan01] JANSON, Andre: *Usability-Engineering als Instrument des Managements informationstechnolocher Veränderungsprozesse in Unternehmen*, Otto-Friedrich-Universität Bamberg, Diss., 2001

[LWW13] LEHERGERAT, Phillipe ; WHITMER, Ray ; WOOD, Laurens: Document Object Model (DOM). (2013). `http://www.w3.org/DOM/#what`, Abruf: 19.10.2013

[Nie04] NIELSEN, Jacob: Usability engineering. (2004). ISBN 978–0125184069

[Off12] OFFERGELD, Michael: *Usability Engineering Skript WS2012/13*. Universität Ulm, Germany, 2012

[Pru13] PRUSCH, Thomas: 10 Usability Heuristics. (2013). `http://www-cg-hci.informatik.uni-oldenburg.de/~usabili/Ausarbeitung/it/10_Heuristiken_von_Nielsen.htm`, Abruf: 02.10.2013

[Ram13] RAMPL, Hansjörg: *Erwartungskonformität*. `http://www.handbuch-usability.de/erwartungskonformitaet.html`. Version: 2013, Abruf: 14.9.2013

[Ref13a] REFSNESDATA: OS Statistik. (2013). `http://www.w3schools.com/browsers/browsers_os.asp`, Abruf: 06.09.2013

[Ref13b] REFSNESDATA: Screen Statistics Stand: 06.09.2013. (2013). `http://www.w3schools.com/browsers/browsers_display.asp`, Abruf: 06.09.2013

[Vaa12] VAADIN: *Compare vaadin.com*. `http://vaadin.com/comparison`. Version: 2012, Abruf: 02.08.2013

[Wik13] WIKIMEDIA, Foundation: *Proof of Concept - Wikipedia*. `http://de.wikipedia.org/wiki/Proof_of_Concept`. Version: 2013, Abruf: 19.09.2013

[WN12] WEBER, Michael ; NEUMANN, Heiko: *Skript: Grundlagen der Interaktiven Systeme*. Universität Ulm, Germany, 2012

Name: Laurens Lang Matrikelnummer: 726269

Erklärung

Ich erkläre, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Laurens Lang