Projektdokumentation

RefugeesWelcomeApp

Autoren: Nico Remus, Philipp Staats, Stefan Ludowicy und Jean-P. Klodzinski

Bearbeitungszeitraum: 5. Semester

Semestergemeinschaft: DWI 13

Inhaltsverzeichnis

[1 Projektbeschreibung 4](#_Toc439671985)

[1.1 Ausgangsituation 4](#_Toc439671986)

[1.2 Zielsetzung 4](#_Toc439671987)

[1.2.1 Grobkonzept 4](#_Toc439671988)

[1.2.2 Zielgruppe 4](#_Toc439671989)

[1.2.3 Funktion und benötigte Rechte der Benutzergruppen 4](#_Toc439671990)

[1.2.4 Allgemeine Beschreibung der Funktionen 4](#_Toc439671991)

[1.2.5 Genutzte Plattformen 5](#_Toc439671992)

[1.2.6 Rahmenbedingungen 5](#_Toc439671993)

[1.2.7 Technische Umgebung 5](#_Toc439671994)

[1.2.8 Einsatzbedingungen 6](#_Toc439671995)

[1.2.9 Rechtliches 6](#_Toc439671996)

[1.2.10 Vorkenntnisse 6](#_Toc439671997)

[1.2.11 Einschränkungen 6](#_Toc439671998)

[*1.3* Meilensteine 6](#_Toc439671999)

[2 Organisation 6](#_Toc439672000)

[2.1 Aufbauorganisation 6](#_Toc439672001)

[2.1.1 Projekt Hierarchie 6](#_Toc439672002)

[2.1.2 Taskforces 8](#_Toc439672003)

[2.2 Skillsheet 9](#_Toc439672004)

[3 Ablaufsteuerung 10](#_Toc439672005)

[3.1 Phasenmodell 10](#_Toc439672006)

[3.1.1 Erste Planung 10](#_Toc439672007)

[3.1.2 Zweite Planung 10](#_Toc439672008)

[3.2 Scrum 11](#_Toc439672009)

[3.2.1 Scrum im Projekt 11](#_Toc439672010)

[3.3 Extreme Programming 12](#_Toc439672011)

[3.4 Risikomanagement & Stakeholderanalyse 13](#_Toc439672012)

[4 Das Softwaresystem 13](#_Toc439672013)

[4.1 Anwendungsfall Beschreibung 14](#_Toc439672014)

[4.2 Anwendungsfalldiagramm 14](#_Toc439672015)

[4.3 Aktivitätsdiagramm 15](#_Toc439672016)

[4.4 Klassendiagramm 15](#_Toc439672017)

[4.5 Sequenzdiagramm 16](#_Toc439672018)

[4.6 Zustandsdiagramm 16](#_Toc439672019)

[4.7 Auswertung der Diagramme 16](#_Toc439672020)

[4.8 Auswahl des Vorgehensmodells und Umsetzung 16](#_Toc439672021)

[4.9 Auswahl der Architektur und Umsetzung 17](#_Toc439672022)

[4.9.1 Architecture 17](#_Toc439672023)

[4.9.2 Security 17](#_Toc439672024)

[4.9.3 Allgemeines zu 3-Tier-Architektur 17](#_Toc439672025)

[4.10 Auswahl der Versionsverwaltung und Umsetzung 18](#_Toc439672026)

[4.11 Installationshandbuch 18](#_Toc439672027)

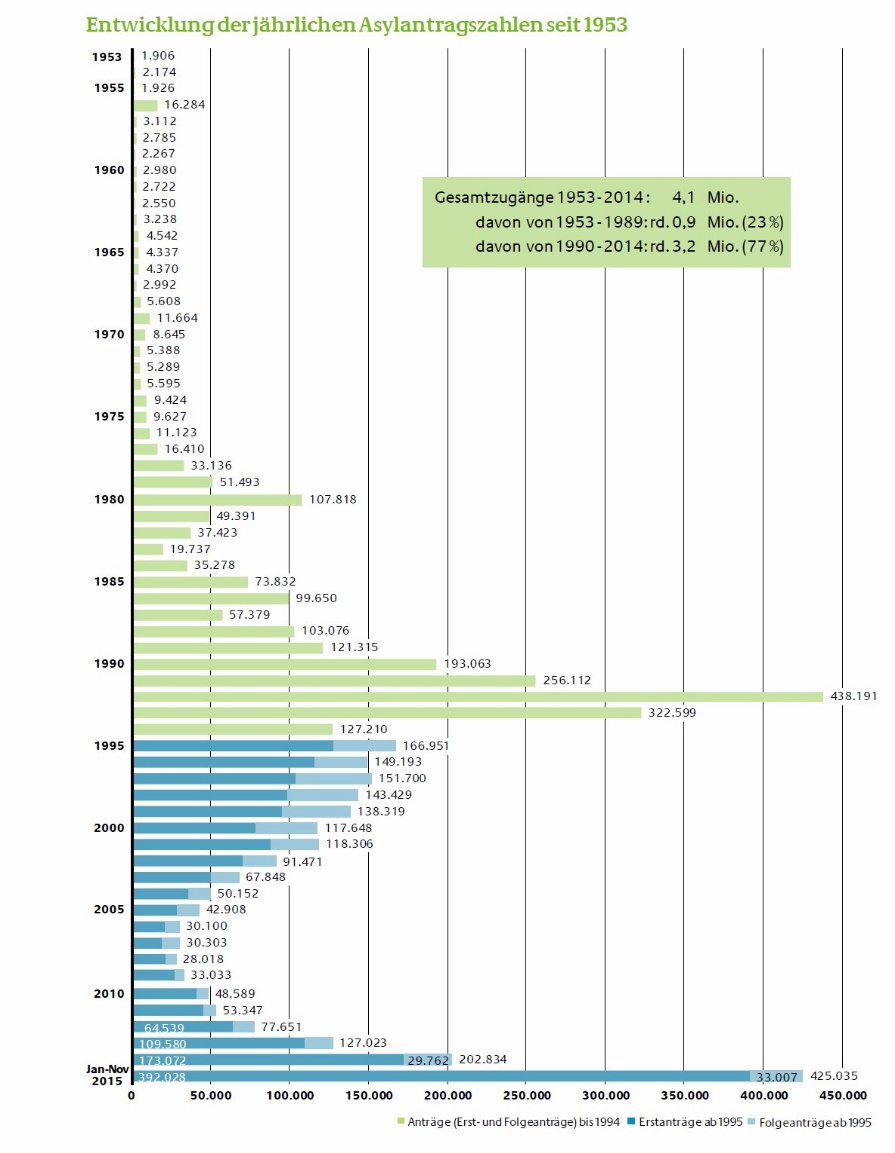
[4.12 Benutzerhandbuch 18](#_Toc439672028)

[5 Anhang 18](#_Toc439672029)

[6 Literaturverzeichnis 19](#_Toc439672030)

# Projektbeschreibung

## Ausgangsituation

900.000 Flüchtlinge seit Jahresbeginn, allein 500.000 seit Anfang September: Das sind die neusten Zahlen zum Flüchtlingsstrom nach Deutschland. Es gibt viele Plattformen, Webseiten und Apps die versuchen mit verschiedenen Sprachen der Informationen für Flüchtlinge zur Verfügung zu stellen. Um diese Informationen nützlich zu machen, braucht man ein Team, die dies kontinuierlich aktualisiert, was natürlich sehr aufwändig ist und gerade nicht kostengünstig. Vor allem ein Team kann nie immer alle Informationen selbst wissen und die Fülle an Informationen bearbeiten.

Diese Menschen kommen in ein völlig Fremdes Land ohne Grundwissen wie es in Deutschland überhaupt ist.   
Damit sie mehr Informationen bekommen und große Probleme (oftmals alltäglicher Herkunft) beherrschen können, soll dieses Projekt dazu dienen, eine freie Plattform bereitzustellen.

Jeder hat eine soziale Verantwortung (sollte) und ist mit der Herausforderung mit den Flüchtlinge betroffen. Für Organisationen, Instituten und Vereine wird somit damit die Möglichkeit gegeben sehr einfach Informationen auf die Plattform zuschreiben. Von daher nehmen Informationen ständig zu und werden von verschiedenen Seiten aktualisiert und immer aktuell gehalten.

Hier in Nov. 2015 Ausgabe von BAFM (Bundesamt für Migration und Flüchtlinge) sehen Sie wie Anzahl von Flüchtlinge zunimmt.

## Zielsetzung

### Grobkonzept

Mithilfe der mehrsprachigen Web-Plattform soll es Flüchtlingen und besonders Asylsuchenden vereinfacht werden, sich in Deutschland zurecht zu finden. Um dies zu erreichen sollen möglichst alle benötigten Informationen für ein schnelles Zurechtfinden und gute Integration gesammelt und aufbereitet werden. Dazu werden weitere hilfreiche Funktionen integriert. Diese sollten möglichst genau auf die Anforderungen der Flüchtlinge zugeschnitten sein, unter anderem eine sehr einfache Benutzeroberfläche, die sofort verständlich ist und intuitiv bedient werden kann. Im Zuge der Projektarbeit nutzen wir bereits bestehende Anwendungen anderer Webseiten oder verlinken auf diese, bspw. auf eine Open Street Map oder einem Wiki.

Darüber hinaus stellt es die Prüfungsleistung der DWI 13 in dem Kurs „Softwareengineering“ dar. Ein Anspruch an die Software wie vergleichbare Projekte muss gegeben sein, um ein entsprechend gutes Ergebnis präsentieren zu können.

### Zielgruppe

Die Zielgruppe der zu erstellenden Website bzw. späteren App sind zum einen die Asylbewerber (Asylsuchende), zum anderen die Helfer.

### Funktion und benötigte Rechte der Benutzergruppen

Ein nicht angemeldeter normaler Benutzer (z.B. Asylbewerber) hat keine besonderen Rechte. Diesen Anwender kann alles angezeigt werden, jedoch sind keine Bearbeitungsrechte eingeräumt. Dieser benutzt die Website genauso wie das allbekannte Wikipedia. Für diesen Hauptprozess werden keine Daten benötigt und somit müssen auch keine gespeichert werden.

Ein angemeldeter Benutzer (Helfer) besitzt erweiterte Benutzerrechte. So kann dieser das Wiki bearbeiten und hat somit Zugriff auf die Website als auch dem Wiki. Für diesen Geschäftsfall werden die Session des Nutzers, die Benutzerrechte bzw. Rolle und der Verein für den er tätig ist (bestimmt die Bearbeitungsrechte) gespeichert.

Ein Administrator hat vollständige Rechte. Er darf alles bearbeiten, neu anfügen, löschen usw. . Ebenso benutzt er alle bereitgestellten Funktionen, auch diejenigen, die über die Website hinausgehen. So z.B. um ein neues Build auf den Application Server zu pushen. Für diesen Geschäftsfall werden die gleichen Daten gespeichert wie für einen eingeloggten User (Helfer).

### Allgemeine Beschreibung der Funktionen

Wir nutzen verschiedene Funktionen aus bereits existierenden Webseiten und Anwendungen, dabei spezifizieren wir den Umfang nach unseren Anforderungen. Hier folgen die zu implementierenden Funktionen:

* Übersichtliche, thematische, einfach zugängliche Informationssammlung
  + Behördenkontaktdaten zu den jeweiligen Standorten
  + Rechtliches Einmaleins
  + Anlaufstationen für z.B. kurzfristige Unterkünfte / Tafel / DRK
  + Infos der lokalen Refugee Law Clinics / andere lokale Vereine
  + häufig genutzte Rufnummern (Notrufnummern) & deren Aufgabengebiete
  + Gesundheitssystem, kostenlose Ärzte, „Wo finde ich einen in meiner Nähe“
  + Verknüpfungen zu anderen Hilfeseiten und Funktionalitäten
  + usw.
* Sofern möglich – Ermittlung des Standortes und Darstellung der gesuchten Informationen auf einer Karte („Wo bin ich und wo muss ich hin“)
* Ein Wikipedia – dort kann gemeinschaftlich dran gearbeitet werden. Inhaltlich dreht es sich um tiefergehende Infos zu Standorten (Freizeitaktivitäten bspw.)
* Freizeitbeschäftigung: Map/Wiki -> was kann ich in meiner Umgebung machen? Fußball etc.
* Eine Funktion die erlaubt, herauszufinden welche Materialien ich in welchem Laden finde. Was gibt es im Baumarkt, was finde ich im Supermarkt? Wie heißen die Supermärkte? Und dann direkt eine Integration „auf Karte anzeigen“ – dort werden dann die benötigten Kategorien eingeblendet. (fließender Übergang)
* Suchfunktion: Einfaches Schlagwortfiltern. Ich möchte mich nicht durchklicken? „Okay – ich such danach“
* Formular Template: Deutschland hat ein bürokratischen Dschungel der allein für Deutsche nicht immer durchschaubar ist – für Menschen die dann nicht einmal deutsch sprechen – eine noch viel größere Herausforderung. Eine standortbezogene Auflistung der benötigten Formularen mit Ausfüllhilfe: Was bedeutet das Feld oder das? Was muss ich hier eintragen?

### Genutzte Plattformen

In der ersten Version fertigen wir eine Webseite an, die man mit jedem gängigen Browser aufrufen kann. Diese Webseite wird immer weitergepflegt und erweitert. Zusätzlich soll in den zukünftigen Versionen (Version 2 und höher) auch eine App für Smartphones entwickelt werden.

### Rahmenbedingungen

* Dezentralität:  
  Unser System soll dezentral aufgestellt werden. Wir möchten damit die Probleme andere Lösungen damit beheben und geben somit den Usern (Helfer) die Möglichkeit ihre eigenen Informationen unkompliziert zu teilen.
* Sicherheit:  
  Aufgrund der Gesetzeslage in Deutschland muss auch die Datensicherheit als auch die Systemsichert für die verwendete Soft- und Hardware gewährleistet werden. Darunter sollte auch eine DDoS – Attacke abgesichert werden.
* Open Source:  
  Eine weitere Bedingung ist, dass wir nur Open Source Software nutzen möchten, um keine Kosten für das Projekt für die Gegenwart als auch Zukunft aufkommen zu lassen. Des Weiteren sollten die Open Source Software geschäftlich nutzbar sein.
* Wartbarkeit:  
  Eine weitere Bedingung die wir uns stellen ist: Einfachheit. Damit sollte eine einfache aber auch verbesserte Möglichkeit gegeben sein, die Systeme als auch Software zu warten.

### Technische Umgebung

Bei den technischen Lösungen sollen keine monetären Kosten anfallen, daher wird für das Projekt überwiegend auf Open-Source Software gesetzt. Das verwendete Modell soll eine Tier-3 Architektur sein – somit sind Skalierungsmöglichkeiten, falls ein durchschlagender Erfolg eintritt, gegeben.

Als Client kommen alle Geräte in Frage, welche Webseiten darstellen können. (Desktop PC, Laptop, Tablet, Smartphone).

### Einsatzbedingungen

Geltende Sicherheitsstandards müssen erfüllt sein. Insbesondere sobald Nutzerdaten gespeichert werden, ist dies unumgänglich.

Die Webseite sollte innerhalb üblicher Reaktionszeiten agieren (max. 3 Sekunden beim Kunden).

### Rechtliches

### Vorkenntnisse

Die einzigen Vorkenntnisse besitzen wir aus den Inhalten die im Rahmen des dualen Studiums der Wirtschaftsinformatik angeeignet wurden. Darunter fallen Programmierkenntnisse, wie das Programmieren mit u. a. Java, SQL, sowie Projektmanagementkenntnisse (unterstützt mit Microsoft Project).

### Einschränkungen

Durch fehlendes tiefergehendes Know how an vielen Stellen, ist eine 80% Lösung ausreichend. Es muss kein technisches Meisterwerk sein – vor allem da nur sehr begrenzte Zeit vorhanden ist im Rahmen des Projektes. Aufgrund begrenzter Ressourcen wird eine fertige Lösung nicht möglich sein. Die erste Version (diese wird Frau Dr. Wieland als Abschlussarbeit eingereicht) bedarf weiterer Arbeit, um alle Anwendungen zu implementieren und eine App zu erstellen. Die Weiterentwicklung nach der ersten Version ist noch nicht festgelegt.

(Klodinski, S. 1-3)[[1]](#footnote-1)

## Meilensteine

https://docs.google.com/spreadsheets/d/1Ssdn-gzXDYjQ3ldQSXaKPz8bGzdTRwECd5rJbsxSwsw/edit?usp=sharing

# Organisation

## Aufbauorganisation

### Projekt Hierarchie

Das Projekt wurde mit drei Teilprojektgruppen initialisiert. Diese sollten die unterschiedlichen Aspekte des Projekts vorbereiten und eine thematisch untergliederte Basis bilden. In den späteren Phasen (siehe 3.1) werde diese Gruppen funktional weitestgehend aufgelöst.

|  |  |  |  |
| --- | --- | --- | --- |
| Projektlead | Philipp Staats | | |
|
|
|  | Team 1 | Team 2 | Team 3 |
| Lead | Philipp Staats (134147) | Stefan Ludowicy (134131) | Nico Remus (134152) |
|  | Aras Mousavii (134242) | Philipp Kneist (134159) | ~~Tobias Karhof  (134124)[[2]](#footnote-2)~~ |
|  | Sheldon Jesudasan (134158) | Sebastian Treske (134153) | Jasmin Clos  (134108) |
|  | Christoph Ramp (134136) | Jan Bickmann  (134102) | Jean Klodzinski (134125) |
|  | Eric Skowronski  (134145) |  |  |

[[3]](#footnote-3)

#### Projektleiter - PL

Die Gesamtprojektleitung übernimmt Philipp Staats.

Die Rolle des Projektleiters dient der Gesamtkoordination des Projekts in Abstimmung mit den Teilprojektleitern. Der Schwerpunkt liegt hierbei im Controlling.   
Zu den Controllingaufgaben werden die Steuerung der Teams sowie der Gruppenmeetings, aber auch Einhaltung von Terminen zählen. Dabei wird der Projektleiter in direkter Kommunikation mit dem PO (Fr. Wieland) stehen und über projektspezifische Missstände aber auch Erfolge in Kenntnis setzen. Er ist für das Risikomanagement verantwortlich, so dass bei projektgefährden Ereignissen der Handlungsbedarf von Ihm ausgehen muss.

Fachliche Fragestellungen werden im Detail von den Know-How Trägern in den Task-Forces (siehe 2.1.2) beschlossen.

#### Teilprojektleiter – TPL

Die Unterteilung des Projekts in drei Teilprojektgruppen dient der Entlastung des Projektleiters. Jede Teilprojektgruppe hat einen thematischen Schwerpunkt, nach welchem diese dem Projekt zuarbeitet. Gemeinsam leiten sie das Gesamtprojekt.

1. Research - Philipp Staats
2. Technische Entwicklung - Stefan Ludowicy
3. Solution Design - Nico Remus

#### Projektmitglied - PM

### Taskforces

*(Aufbau nach Einsetzen der Scrum-Methodik )*

#### Anfoteam

Das komplette Anforderungsmanagement-Team bestand aus folgenden Personen:

Jasmin Clos, Stefan Ludowicy, Philipp Staats, Jean-Pierre Klodzinski, Nico Remus, Tobias Karhof (der uns leider verlassen hat).

Das Anfo-Team ist dafür zuständig neue Anforderungen in Form von Stories zu definieren und die Ergebnisse zu überprüfen. Darüber hinaus kümmert es sich um die Steuerung des Teams, falls es intern Probleme gibt. Probleme sind unter anderem aufkommende Fragen oder wenn es einen Disput zwischen den Mitgliedern geben sollte.

#### Scrum Masters[[4]](#footnote-4)

Die beiden Scrum Master lauten:

Stefan Ludowicy und Jean-Pierre Klodzinski.

Die Scrum Master sind dafür zuständig, dass Scrum Board zu pflegen, Fragen und Probleme zu lösen, als auch den Disput zwischen Mitgliedern, fehlenden Ausarbeitungen oä. In letzter Instanz hab die Scrum Master die höchste Entscheidungsgewalt.

#### Translators

#### Development

#### Redaktion und Frontendlayout

*(Frontendtest, Usability)*

## Skillsheet

*(Ermittlung von Know-How-Trägern zur Bildung von  
Self-Teaching Teams -> 1 Erfahrener & 1 Semi-Erfahrener)*

Mit einem Gesamtprojekt übergreifenden Skillsheet werden besondere Fähigkeiten und Vorkenntnisse der einzelnen Projektmitglieder ermittelt. Daraus ergibt sich ein Kompetenzen-Pool, auf welchem die weitere Konzeption des Projekts beruht.

# Ablaufsteuerung

## Phasenmodell[[5]](#footnote-5)

*Warum Phasenmodell*

*Warum wechsel auf neues Modell*

### Erste Planung

* 1. 1.Phase (2.Wochen)
     1. Team 1 Staats Recherchiert/ Erfasst Content  
        [Erste Recherche nach Drittdienstleister]
     2. Team 2 Ludowicy Einarbeitung in die Entwicklungsplattformen
     3. Team 3 Remus AS(AnforderrungsSpezifikation) - Erstspezifikation

-> Eine Woche später FSL (Fachspezifsche Lösung) nach vorstellung des Konzepts

* 1. 2. Phase (2.Wochen)
     1. Team 1 Staats Recherche Zielgerichtet nach FSL/ Beginn Übersetzungen  
        Recherche und Abstimmung mit Team 2, wie Drittanbieter eingebunden werden können
     2. Team 2 Ludowicy Anwendungs-Gerüst
     3. Team 3 Remus FSS mit CoOp Team 2
  2. 3. Phase (2.Wochen)
     1. Team 1 Staats Hauptphase der Übersetzungen  
        Kontakt mit Drittanbietern herstellen
     2. Team 2 Ludowicy Anwendungs-Skelet
     3. Team 3 Remus Graphische Spezifikation und Elemente/ Zuarbeit zu Team 2
  3. 4. Phase...

### Zweite Planung

* 1. 1.Phase (2.Wochen)
     1. Team 1 Staats Recherchiert/ Erfasst Content  
        [Erste Recherche nach Drittdienstleister]
     2. Team 2 Ludowicy Einarbeitung in die Entwicklungsplattformen
     3. Team 3 Remus AS(AnforderrungsSpezifikation) - Erstspezifikation

-> Eine Woche später FSL (Fachspezifsche Lösung) nach vorstellung des Konzepts

* 1. 2. Phase (2.Wochen)
     1. Team 1 Staats Recherche Zielgerichtet nach FSL/ Beginn Übersetzungen  
        Recherche und Abstimmung mit Team 2, wie Drittanbieter eingebunden werden können
     2. Team 2 Ludowicy Anwendungs-Gerüst
     3. Team 3 Remus FSS mit CoOp Team 2
  2. 3. Phase (2.Wochen)
     1. Team 1 Staats Hauptphase der Übersetzungen  
        Kontakt mit Drittanbietern herstellen
     2. Team 2 Ludowicy Anwendungs-Skelet
     3. Team 3 Remus Graphische Spezifikation und Elemente/ Zuarbeit zu Team 2
  3. 4. Phase...

## Scrum

Scrum als agile Vorgehensmethode beruht auf so genannten Sprints. Jeder Sprint beginnt mit einem Planning, in dem die Aufgaben an das Team verteilt werden und endet mit dem Review, in dem die verteilten Stories vorgestellt und vom Product Owner abgenommen werden. Die Sprints dauern zwischen einer und meistens vier Wochen, basierend auf der Komplexität der Aufgaben. Mehr oder weniger ist selbstverständlich möglich.

Scrum, als sehr bekannte Form der agilen Vorgehensmethoden, kam auch für unser Projekt in Frage. Als Alternativen boten sich Extreme Programming oder das Wasserfallmodell an. Wir entschieden uns gemeinschaftlich für Scrum, weil es verschiedene Vorteile, resultierend aus dem Vorgehen, bietet. So musste nicht von Anfang an das komplette Softwaresystem ausspezifiziert worden sein. Es war uns möglich durch die Sprints nach und nach die Dinge zu durchdenken. Darüber hinaus hatten wir nach jedem Sprint ein besseres Gefühl, bis wann das Projekt etwa fertig sein wird, bzw. „wie weit“. Auch lässt sich mit Scrum besser die Last im Team verteilen, da jede Woche neu „gewählt“ wurde.

### Scrum im Projekt

Die Scrum Vorgehensweise wurde in unserem Team gemeinschaftlich beschlossen und fast vollständig umgesetzt. Wir verstehen Scrum als Richtlinie, wonach wir die mehrheitlichen Regeln befolgen. So haben wir eine Menge fester Rollen, welche ein festes Aufgabengebiet übernehmen. Jedoch kann in unserem Team eine Person mehrere Rollen gleichzeitig wahrnehmen und wirkt entsprechend anders an dem Fortschritt des Projektes mit. Darüber hinaus ist auch das Vorgehen, mit Sprint, Planning, Review, Retrospektive (war sehr wichtig für das Anfo-Team ist) dem Scrum Vorbild nachempfunden. Jedoch verzichteten wir auf Daylies und haben stattdessen eine selbstständige Eintragung in unserer Story Bord in Form von Kommentaren bevorzugt. Dies reduzierte die extreme organisationslast auf Seiten der Scrum Master ein wenig und jeder konnte sich einteilen wann er welche Aufgaben erledigt hatte. Wichtig war uns, dass die Aufgaben erledigt werden innerhalb des Zeitraums – nicht aber auf welche Art und Weise. Das Ergebnis musste stimmen. Als weiteren Unterschied wurden die Stories kategorisiert. Dies half uns bei der Verteilung den jeweilig richtigen Personen die Aufgaben zu überreichen. Selbstständiges Wählen der Aufgaben durch die Mitglieder des Teams, hatte leider nicht wie erwartet funktioniert.   
Zum Ende des Projektes, also über die Weihnachtsfeiertage, als die Zuarbeit des Teams gegen null lief, wurde das Vorgehen auf „Extreme Programming“ umgestellt. Eine kleine Gruppe setzte dann alle notwendigen Funktionen um.

Hier eine kurze Zusammenfassung unserer Rahmenwerte:

#### Agil

* Sprintlänge: 1 Woche

Story - Creation Team

siehe 2.1.1 Anfo Team.

#### Kategorisierung der Stories

Unterscheidung der Stories in:

* + Entwicklung
  + Research
  + Design
  + Visual Design
  + Pre-Dev

Storypoints

= Arbeitstunden (60min reale Arbeit)

#### Scrum-Storyboard[[6]](#footnote-6)

Das Story Board ist das zentrale Fortschritts-Messinstrument gewesen. Auf dem Story Bord sind alle Aufgaben hinterlegt. Es zeigt den aktuellen Status (Fertig, in Bearbeitung und offen), sowie welche Personen zuständig sind und wie komplex die Aufgabe scheint. Die Idee zum Story Bord kam während des ersten Sprints zur Erfassung des Fortschritts und um die Motivation bei den Gruppenmitgliedern zu fördern. Auf dem Board sieht man sehr gut wie viel Arbeit noch getan werden muss. Hauptverantwortlich für die Anforderungsdefinition war Jean-Pierre Klodzinski (in Zusammenarbeit mit Stefan Ludowicy und Philipp Staats) und letztendlich umgesetzt wurde es von Nico Remus (in Kooperation mit Philipp Staats).

## Extreme Programming

*Funktionalität der CoOp Arbeit von 2-3 Entwicklern + Backup*

*[[7]](#footnote-7)*

## Risikomanagement & Stakeholderanalyse

Bei der Risikoanalyse wurden überwiegend alle monetären Stakeholder und Risiken ignoriert, da diese zur Zeit des Projektstarts als auch der derzeitigen Entwicklung keine Rolle spielten. Ob diese nach dem DWI13 Modul SWE bestehen könnten, müssten nachträglich geprüft werden. Grundsätzlich gab es für de Projekt keine großen ernsthaften Risiken, die Thematiken, die vorhanden sein könnten, sind mit den Eintrittswahrscheinlichkeiten kaum bedrohlich.

Detaillierte Informationen kann dem Exelsheet im Anhang entnommen werden.

# Das Softwaresystem

In diesem Kapitel werden die für die Prüfungsleistung benötigten Dokumente und UML Diagramme hinterlegt. Sofern ein bestimmter Anwendungsfall gewünscht war, wird im Folgenden die Funktionalität „Wo Gibt Es Was“ benutzt. Dazu eine wörtliche Beschreibung aus dem Solution Design-Dokument:

*Mit der Funktion "Wo gibt es was" soll dem Benutzer ermöglicht werden auf einfache Art und Weise herauszufinden, in welchem Laden er welche Dinge kaufen kann.*

*Um das möglichst einfach zu gestalten, wählt der Nutzer als erstes ein Bild einer Produktkategorie aus (Beispielsweise Nahrung). Neben dem Bild findet sich noch der Kategoriename.*

*Anschließend hat er die Möglichkeit eine Unterkategorie zu wählen, oder direkt sich die Shops anzeigen zu lassen.*

*Falls er eine Unterkategorie auswählt, wird er zu einer Übersichtsseite weitergeleitet. Auf dieser wird die Shopkategorie angezeigt (z.Bsp. Supermarkt) und die dazu gehörenden Shopnamen (Lidl, Rewe, Aldi), sowie eine Übersicht von verfügbaren Produkten (begrenzt auf x Werte).*

*Zu den Shops gibt es dann noch eine grobe Richtung in welcher Preisklasse, Angebotsvielfalt und Qualität dieser liegt: Beispiel: Aldi 1, Lidl 2, Rewe und Edeka 3.*

*Falls er jedoch direkt wissen möchte, wo er beispielsweise Nahrung bekommt, dann klickt er auf einen Button wie z.Bsp. "Wo bekomme ich das?" und bekommt gleich die Übersichtseite incl. Shopliste angezeigt als auch eine Menge von Beispielprodukten (Nudeln, Reis, Tomaten, ...).*

*Darüber hinaus hat er auf der Übersichtsseite die Möglichkeit eine Verlinkung zur Karte zu öffnen über einen Klick auf den Button "In meiner Nähe"/"der nächste in meiner Nähe"/"Wo finde ich diese". Damit öffnet sich ein Fenster mit einer Karte gefiltert nach der ausgewählten Shop-Kategorie. Er kann dort nun seine Adresse eingeben und ein Ziel auswählen und die Navigation starten.*

*Alternativ ist es dem Benutzer möglich, direkt nach Produkten zu suchen (Bspw. Reis). Falls ein Eintrag gefunden wird, wird die Übersichtsseite angezeigt, falls nicht eine Fehlermeldung: "Hupps - das gewünschte Produkt konnte keiner Kategorie zugeordnet werden. Klicke hier um einen erneuten Suchvorgang zu starten“. Wenn mehrere Einträge gefunden werden, dann gibt es eine Übersicht mit den Ergebnissen, welche gefunden wurden. Daraus kann der Benutzer dann auswählen und gelangt dann zur Übersichtsseite.*

*Die suche soll partiell, vollständig und familiär sein. Und doch alles irgendwie anders. Hier zu ein Beispiel:*

*DB: Paprika und Nüsse*

*Suche: Paprikaschoten -> Ergebnis: Paprika*

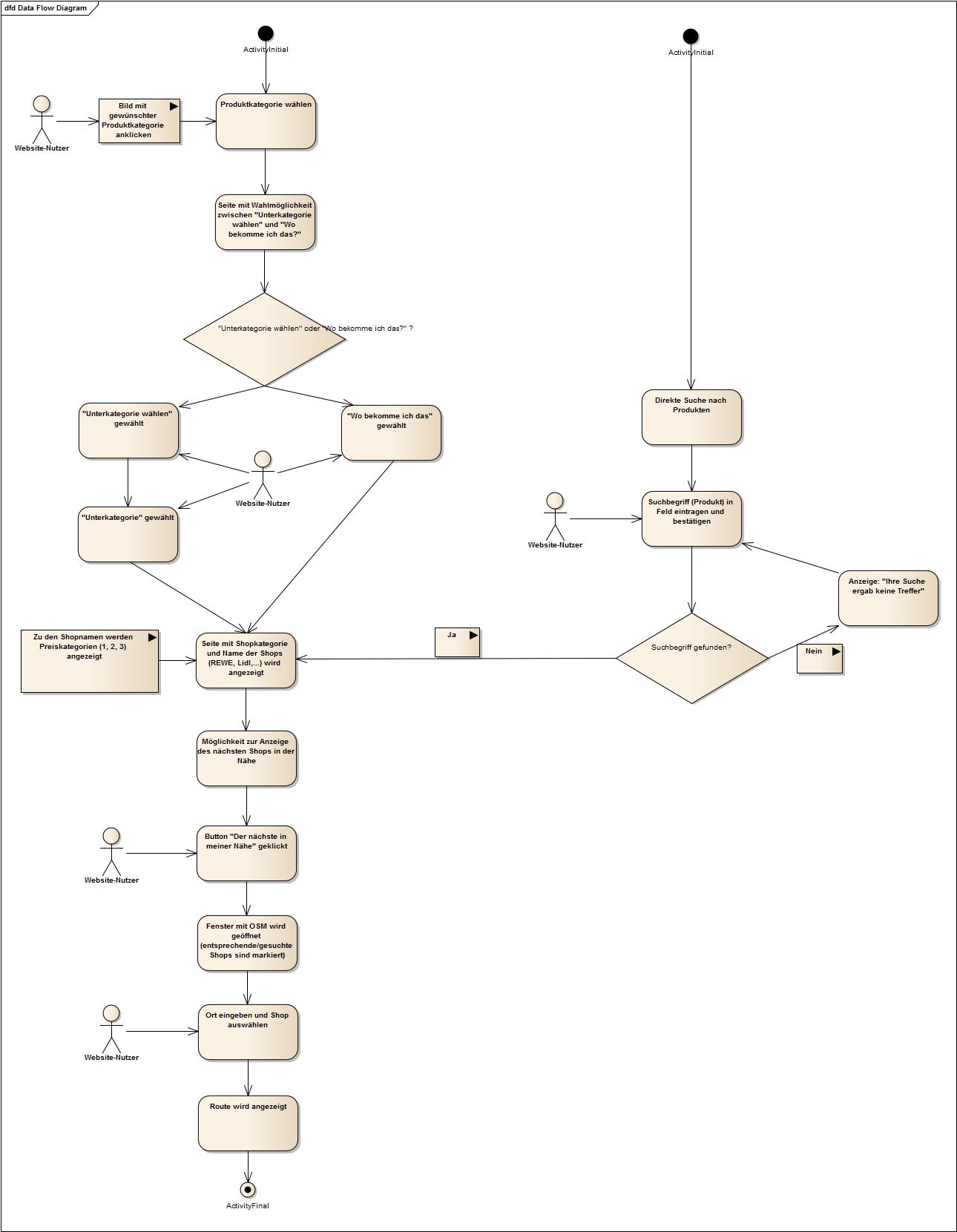
*Suche: Erdnüsse -> Nüsse[[8]](#footnote-8)*

## Anwendungsfall Beschreibung

|  |  |
| --- | --- |
| Use Case | Wo gibt es was |
| Actors | Webseiten Besucher |
| Pre Conditions | * Die Webseite ist online * Der Datenbankserver ist online und gepflegt |
| Main flow | 1. Benutzer besucht die Webseite 2. Benutzer wählt „Wo finde ich was?“ in der Menuleiste aus 3. Der Benutzer wählt eine Produkthauptkategorie aus 4. Der Benutzer wählt eine Produktunterkategorie aus 5. Der Benutzer sieht Produktbeispiele der Produktunterkategorie und in welcher Shopkategorie er das Produkt kaufen kann.   Auf der angezeigten Seite wird darüber hinaus eine eingebundene Karte von Open Street Map angezeigt. |
| Alternative flow (noch nicht implementiert) | 1. Benutzer besucht die Webseite 2. Benutzer wählt „Wo finde ich was?“ in der Menuleiste aus 3. Der Benutzer trägt in das Suchfeld einen Suchbegriff ein 4. Der Benutzer erhält alle gefundenen Ergebnisse übersichtlich angezeigt |
| Post conditions | * Der Benutzer muss auf die Ausgangsseite zurückkehren können |
| end | Wo gibt es was |

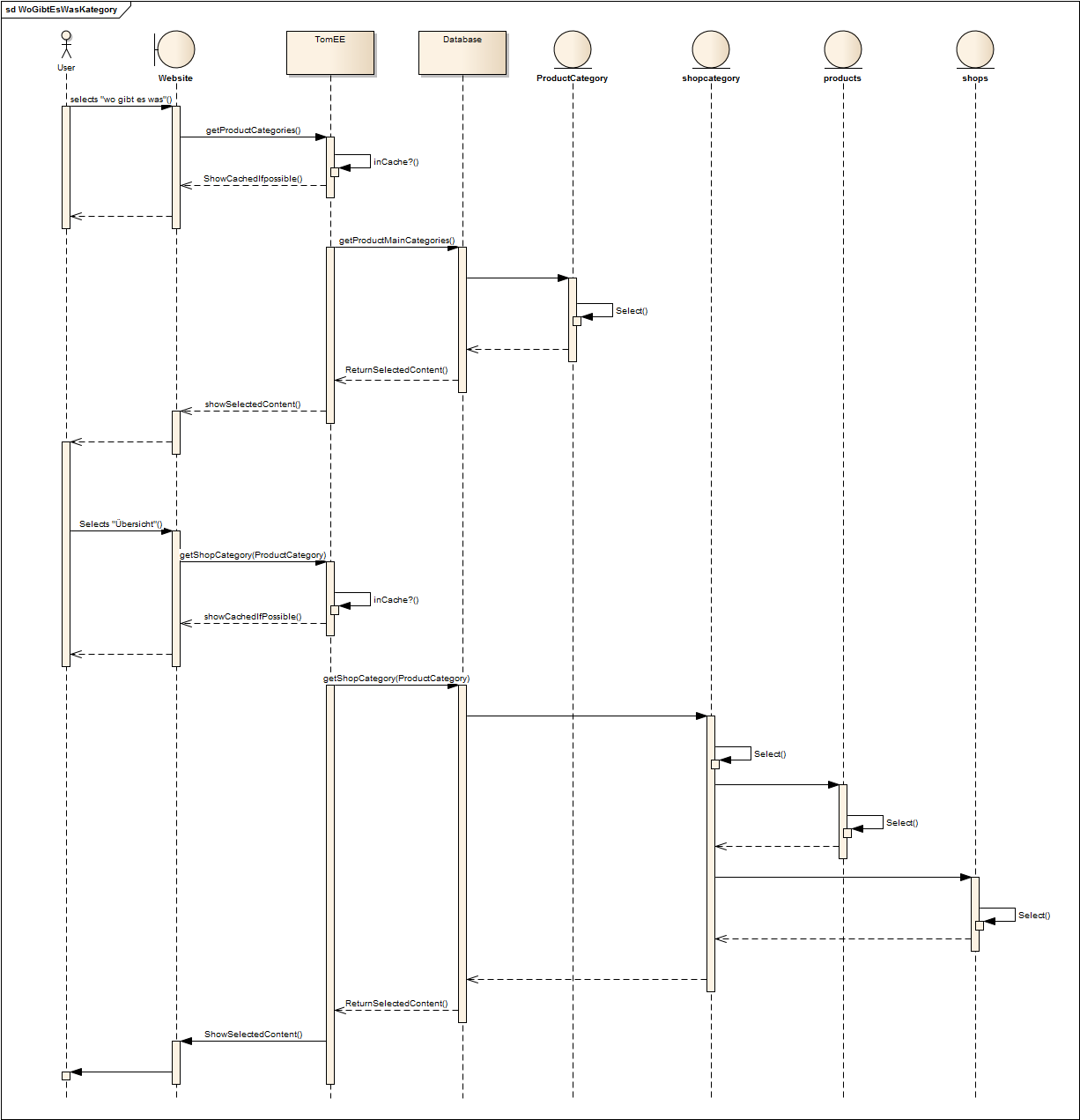
## Anwendungsfalldiagramm

## Aktivitätsdiagramm



## Klassendiagramm

## Sequenzdiagramm



## Zustandsdiagramm

## Auswertung der Diagramme

Jedes der Diagramme zeigt eine spezifische Sichtweise auf das Softwareprojekt. Doch beschreiben die meisten spezifischen Diagramme ein und den gleichen Sachverhalt. Das eine sehr detailliert, das andere recht grob. Das Anwendungsfalldiagramm ist relevant für eine frühe Phase in der Softwareentwicklung, wohingegen z.B. das Sequenzdiagramm eher später relevant wird.

Tabellarische Übersicht: s. Anhang

Für den Designprozess ist das Klassendiagramm die Basis einer jeden Software. Hierbei werden die Akteure genau definiert, wie auch ihre Rechte, Attribute, Beziehungen etc.  
Im Laufe des Designprozesses müssen die Eigenschaften dieser Akteure immer wieder neu angepasst und definiert werden; da beispielsweise ein neues Attribut benötigt wird.  
Auf Grund dessen muss das Klassendiagramm immer weiter konkretisiert, ausgebaut und angepasst werden.

Zum Erstellen der Diagramme wird EA benutzt da der EA lässt keine unzulässigen Symbole zu. Somit kann man sich nach der Wahl des Diagrammtyps nur noch der richtigen Symbole bedienen.  
Des Weiteren vereint der EA viele UML Diagramme. Man kann also verschiedene Diagrammtypen erstellen und behält trotzdem dem Überblick über das jeweilige Projekt, da auch Verknüpfungen untereinander möglich sind.   
Auch die Mouse Over Hilfen waren sehr hilfreich, da man dadurch nochmal eine kompakte Hilfestellung zu den einzelnen Diagrammtypen erlangen kann.

## Auswahl des Vorgehensmodells und Umsetzung

Im Projekt wurde sich gemeinschaftlich für Scrum entschieden. Eine ausführliche Beschreibung welche Vorteile wir in Scrum sehen und wie gut das funktioniert hat, entnehmen sie bitte Punkt 3.

## Auswahl der Architektur und Umsetzung

### Architecture

Das Softwaresystem basiert auf einer Tier 3 Architektur.

* Einen HTML kompatiblen Webbrowser als Präsentationsschicht. Der Foundation Framework und die AngularJS werden eine Vereinfachung der Entwicklung auf Frontendseite herbeiführen.
* Als zweite Schicht, die Application Server Schicht, wird ein “TomEE web profile” (basierend auf dem Apache Tomcat webserver/ servlet container, Java EE featuresets und OpenEJB) fest gelegt.
* Als dritte und letzte Schicht, die so genannte Datenschicht wird die MariaDB benutzt. Darüber hinaus wird diese Datenbank für einfache Konsistenz und Autorisierungschecks sowie Validierungschecks für den Zugriff auf die gespeicherten Daten genutzt.

### Security

The finished application will comply to the Deutsche Telekom Group [privacy and security assessment process](https://www.telekom.com/psa-process) and thus fulfill the security requirements available at [DE](https://www.telekom.com/static/-/155996/18/technische-sicherheitsanforderungen-si) [EN](https://www.telekom.com/static/-/246042/6/security-requirements-si).

### Allgemeines zu Tier-3-Architektur

Tier-3-Architektur, oder auch Schichtenarchitektur genannt, ist ein Strukturierungsprinzip für die Architektur von Softwaresystemen. Den einzelnen Schichten (Layern) werden einzelne Aspekte des Softwaresystems konzeptionell zugeordnet, wobei zu beachten ist, dass Aspekte einer Schicht nur durch höhere Schichten verwendet werden können.

Vorteile:

Komplexitäten der Abhängigkeiten innerhalb des Systems werden reduziert und vereinfacht.

Zyklen im Abhängigkeitsgraphen werden vermieden (leichteres Verständnis und leichtere Wartung).

Einzelne Schichten können ausgetauscht werden ohne das gesamte Softwaresystem ändern zu müssen.

Nachteile:

Ausführgeschwindigkeit der Anwendung kann auf Grund der Datenverarbeitung in den einzelnen Schichten verschlechtert werden.

Die drei Schichten der Tier-3-Architektur sind:

Präsentationsschicht (Front-End): Schnittstelle zwischen Benutzer und System, Datenein- und ausgabe.

Logikschicht (Middle-Tier): Verarbeitungsmechanismen.

Datenhaltungsschicht (Back-End): enthält die Datenbank und ist verantwortlich für das Laden und das Speichern der Daten.

## Auswahl der Versionsverwaltung und Umsetzung

Für unser Projekt benutzen wir Github als eine dezentrale Versionsverwaltung. Diese Dezentralität mit Github hat den Vorteil, dass ohne Wartezeit gleichzeitig gearbeitet werden kann und keine Absprache unter den Teammitgliedern notwendig ist.

In Github werden für unser Projekt drei Ordner angelegt. Im Ordner Projekt Management sind alle Dateien bezüglich des Projektmanagements hinterlegt (z. Bsp. Portfolio, Ablaufplanung usw.). Der Ordner Application Operation beinhaltet die Konfigurationen und Informationen zu den TomEE Server sowie sämtliche Algorithmen zur Informationsbeschaffung und Schnittstellen zu all unseren Systemen. Im Application Development Ordner sind alle Stories und deren Lösungen, die im Laufe des Projektes bearbeitet wurden, abgelegt.

## Installationshandbuch

Die Installationshinweise finden sie vollständig als Dokument im Anhang. (Installationshandbuch.docx)

## Benutzerhandbuch

Die Benutzung der Webseite ist denkbar einfach. Wie auf jeder anderen Webseite auch, kann man mit linksklick einzelne Menüpunkte anwählen und somit auf diese Seite gelangen. Zurück gelangt man über den Browser. Mehr besondere Funktionen gibt es nicht – alles bringt ein herkömmlicher Internet Browser mit sich.  
Wir empfehlen dennoch den Chrome Browser von Google, da bestimmte Funktionalitäten (die Einbindung der Karte) bisher lediglich in diesem funktioniert.

Eine wesentlich detaillierte Beschreibung finden sie im Anhang unter „User Manual.docx“.

# Anhang

<Hier einfügen: Benutzerhandbuch, Installationsanleitung,…>

RA-60 Dokumentation Sicherheitsarchitektur (PSA Verfahren der Telekom):



Installationshandbuch:



JavaDocs – Klassendokumentation aus dem TomEE



Risikoplanung – Excelsheet



Benutzerhandbuch



Tabellarische Übersicht: Auswertung Diagramme

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ***Anwendungsfalldiagramm*** | ***Aktivitätsdiagramm*** | ***Klassendiagramm*** | ***Sequenzdiagramm*** | ***Zustandsdiagramm*** |
| Hauptaufgabe | ***Dient zur Grobplanung der Anwendung. Hier wird dargestellt, was die Anwendung aus der Usersicht leisten soll.*** | ***Hier wird das Verhalten bzw der Ablaufplan von Aktionen beschrieben.*** | ***Definition der Akteure,deren Beziehung zueinanderund ihre Attribute*** | ***Darstellung der Informationen, die zwischen den Elementen ausgetauscht werden.*** | ***Hierbei ist der Zustand und der Zustandsübergang eines Objektes im Zentrum.*** |
| Reihenfolge/ Ablauf | Reihenfolge ist unwichtig und nicht ersichtlich | Reihenfolge einer Aktivität ist angedeutet. Einzelne Schritte sind ersichtlich. Jedoch werden nur elementare Aktionen aufgezeigt. | Reihenfolge ist unwichtig und nicht ersichtlich | das Zusammenspiel einzelner Objekte in einer Aktivität wird in chronologisch exakter Reihenfolge dargestellt | Reihenfolge der Zustandsübergänge sind dargestellt. |
| Darstellung der Akteure | Das Zusammenspiel zwischen Akteuren und Anwendungsfällen wird ausschnittweise dargestellt. | Akteure sind nicht so wichtig. Lediglich ihr Verhalten steht im Vordergrund. | Ausführliche Darstellung der Akteure | Akteure sind ersichtlich. | Akteure sind nicht ersichtlich |
|  |  | eignet sich sehr gut um Use Case Abbildungen darzustellen | Definiert die Klassen und Objekte und deren Beziehung zueinander | informationen die sie austauschen sind ersichtlich |  |
|  |  | Kontrollflüsse können hier abgebildet werden | dient zur stukturierung des systems | Lifeline eines Objekts wird dargestellt |  |
|  | nicht ersichtlich sind die detaillierten Zusammenhänge zwischen den Akteuren, deren Rollen und Möglichkeiten. Auch die ganze Bandbreite an Funktionalitäten des programms bleiben unberücksichtigt. Dieses Diagramm könnte man auch als inhaltsverzeichnis der Anwendung bezeichnen. Der Benutzer bekommt zwar einen kurzen Überblick über das Programm, aber auf Details wird nicht eingegangen. | Nicht ersichtlich sind hier die Akteure, die Informationen, die ausgetauscht werden, die Zustände und die Abhängigkeiten zwischen den Elementen. | Abläufe werden nicht dargestellt. | Nicht ersichtlich ist der Kontext. Bei diesem Diagramm wird lediglich ein elementarer Aspekt der Anwendung gezeigt. | Auch hier ist der Kontext nicht ersichtlich. |
| In unserem Projekt: |  | wir haben es genutzt, um den Klick weg des Users unter dem Punkt "wo bekomme ich was" darzustellen |  | in unserem sequenzdiagramm haben wir dargestellt, welche informationen zwischen user, server und datenbank ausgetauscht werden in dem Fall der Produktsuche durch den user |  |

# Literaturverzeichnis

Klodinski, J. (kein Datum). Konzept 1.0 SoftwareEngineering.

1. Konzept1.0 Software Engineering [↑](#footnote-ref-1)
2. Herr Karhof war bis zum Sprint 4 Teil des Teams. Jedoch stieg er aus dem Projekt aus, da er das Unternehmen verließ. [↑](#footnote-ref-2)
3. [Aufbauorganisation g+](https://docs.google.com/spreadsheets/d/1W5Od9s6aarCx1sFPQoNwrKGn5htl_ddexCt8joMCvyU/edit?usp=sharing) [↑](#footnote-ref-3)
4. RA-17 [↑](#footnote-ref-4)
5. Meetingprotokoll vom 20.10.2015 [↑](#footnote-ref-5)
6. [Scrumboard](https://docs.google.com/spreadsheets/d/16N5eu0yM38uJgUU9UVnepE90Qff0_ujf9t0Hzrlt9jo/edit?usp=sharing) [↑](#footnote-ref-6)
7. Extreme Programming Präsentation von Sheldon Jesudasan & Nico Remus 28.10.2015 [↑](#footnote-ref-7)
8. (Jean-Pierre Klodzinski, 2015) [↑](#footnote-ref-8)