Vertraulich

Probe-IPA  
Secondhand-Plattform

autor(en) : Cassandra Corrodi

dokumentennummer : CCT-FFF-XXXX

version : 1.0

status : Draft

quelle : Atos

dokumentendatum : 24 Oktober 2018­

­

­

anzahl der seiten : 1

owner : Cassandra Corrodi

© Copyright 2018, Atos AG Alle Rechte vorbehalten. Reproduktion von Teilen oder dem Gesamten ist ohne vorherige schriftliche Genehmigung des Urhebers untersagt. Für Fragen oder Anmerkungen zu diesem Dokument wenden Sie sich bitte an Atos, 0795893817.

Inhaltsverzeichnis

[1 Teil 1: Umfeld und Ablauf 5](#_Toc529544382)

[1.1 Einleitung 5](#_Toc529544383)

[1.2 Kandidatin 5](#_Toc529544384)

[1.3 Beteiligte Personen 6](#_Toc529544385)

[1.4 Aufgabenstellung 7](#_Toc529544391)

[1.5 Projektorganisation 9](#_Toc529544395)

[1.6 Vorkenntnisse 10](#_Toc529544397)

[1.7 Vorarbeiten 10](#_Toc529544398)

[1.8 Firmenstandards 10](#_Toc529544399)

[1.9 Zeitplan 11](#_Toc529544400)

[1.10 Arbeitsprotokoll 13](#_Toc529544403)

[2 Teil 2: Projekt-Dokumentation 19](#_Toc529544404)

[2.1 Informieren 19](#_Toc529544405)

[2.2 Planen 23](#_Toc529544408)

[2.3 Entscheiden 30](#_Toc529544412)

[2.4 Realisieren 31](#_Toc529544413)

[2.5 Kontrollieren 38](#_Toc529544418)

[2.6 Auswerten 40](#_Toc529544419)

[3 Teil 3: Anhang 42](#_Toc529544422)

[3.1 Quellenverzeichnis 42](#_Toc529544423)

[3.2 Tabellenverzeichnis 43](#_Toc529544424)

[3.3 Glossar 44](#_Toc529544425)

[3.4 Programmcode 45](#_Toc529544426)

Änderungshistorie

| Version | Datum | Beschreibung | Autor(en) |
| --- | --- | --- | --- |
| 0.1 | 30.10.2018 | Dokumentstruktur erstellt. | Cassandra Corrodi |
| 0.2 | 31.10.2018 | Einleitung geschrieben, Kapitel Informieren fertiggestellt. | Cassandra Corrodi |
| 0.3 | 01.11.2018 | Planung wurde geschrieben, Entscheidung wurde teilweise dokumentiert. | Cassandra Corrodi |
| 0.4 | 02.11.2018 | Grafische Darstellungen für die Planung wurden erstellt und eingefügt, Entscheidung wurde fertig dokumentiert. | Cassandra Corrodi |
| 0.5 | 07.11.2018 | Es wurde realisiert und die Schritte der Realisation wurden dokumentiert. | Cassandra Corrodi |
| 1.0 | 09.11.2018 | Kontrolle und Auswertung wurde so ausführlich dokumentiert wie möglich. | Cassandra Corrodi |

Tabelle 1: Änderungshistorie

# Teil 1: Umfeld und Ablauf

## Einleitung

Im Rahmen des Moduls 223 wird eine Multiuser Applikation realisiert. Als Vorbereitung für die bald anstehende IPA wird dieses Projekt so realisiert, dass es wie eine mini IPA ist. Das Projekt wird nach den Kriterien und Anforderungen einer echten IPA bewertet. Nachdem das IPA Projekt inklusive Dokumentation angeschlossen wurde, wird eine Präsentation und Demonstration des Programmes stattfinden. Anschliessend folgt ein Expertengespräch.

## Kandidatin

Im Sommer 2015 habe ich meine Lehre zur Informatikerin Fachrichtung Applikationsentwicklung bei Siemens Schweiz gestartet. Im ersten Lehrjahr habe ich dort die Basisausbildung absolviert. Alle Informatiklehrlinge vom gleichen Lehrgang waren während dieses Jahres zusammen, und mussten theoretische wie auch praktische Arbeiten lösen.

Im zweiten Lehrjahr wurden alle in die Abteilungen verteilt. Ab dann habe ich bei Atos gearbeitet und konnte in den Abteilungen praktische Erfahrungen im Bereich C# und Angular sammeln. Zudem durfte ich auch eine Woche im First-Level-Support arbeiten, bei dem ich auch nützliche Erfahrungen mit Kunden sammeln und verschiedene Probleme lösen konnte.

Im Moment arbeite ich im SWP Team, und implementiere mit ASP.NET MVC.

## Beteiligte Personen

### Durchführung der Probe IPA

Lehrbetrieb: Siemens Schweiz AG  
 Berufsbildung  
Adresse: Freilagerstrasse 40  
PLZ/Ort: 8047 Zürich

### Kandidatin

Name: Cassandra Corrodi  
Adresse: Alte Bergstrasse 7  
PLZ/Ort: 8707 Uetikon am See  
Telefon: 079 517 38 89

### Fachvorgesetzter

Name: Patrick Maurer  
Adresse: Freilagerstrasse 28  
PLZ/Ort: 8047 Zürich  
E-Mail: patrick.maurer@atos.net

### Ausbildungsverantwortlicher

Name: Jonas Knoll  
Adresse: Freilagerstrasse 40  
PLZ/Ort: 8047 Zürich  
Telefon: 058 558 38 58  
E-Mail: jonas.knoll@siemens.com

### Experte

Name: Remo Steinmann  
Adresse: Freilagerstrasse 40  
PLZ/Ort: 8047 Zürich  
E-Mail: remo.steinmann@siemens.com

## Aufgabenstellung

Im eingereichten Projektbeschrieb ist beschrieben was das Programm schlussendlich können muss.

Ziel dieses Projektes ist es, einen einfachen Onlineshop zu erstellen. Auf diesem Onlineshop sollte es möglich sein, einen User Account zu erstellen. Jeder User sollte in der Lage sein, Waren welche er besitzt und verkaufen möchte, auf diesen Onlineshop hochzuladen und einen fixen Kaufpreis zu setzen. Andere User sollten dann in der Lage sein, die Produkte anderer kaufen zu können. Die User sollten nicht nur kaufen und hochladen können, sondern auch ihre eigens hochgeladenen Produkte wieder löschen können.

### Projektantrag

Der Projektantrag wurde schon im Voraus dem Auftraggeber zugesendet. Folgende Punkte sind im Projektantrag notiert:

**Kandidat/in**

Cassandra Corrodi

**Titel der Arbeit**

Yahtzy

**Thematik**

Es wird eine Applikation mit ASP.NET MVC realisiert. Es wird eine Applikation sein, welche mit Windows Authentication authentifiziert wird. Es wird eine SQL Datenbank erstellt, welche die Ergebnisse von jedem Spieler speichert. Die Applikation sollte auf einem Client laufen, bei dem zwei Personen gleichzeitig am spielen sind.

**Technologien**

* Multiuser-Applikation mit ASP.NET MVC
* SQL Datenbank
* C# Web API
* Git mit TortoiseGit

**Ausgangslage**

Yahtzy ist ein alter Spielklassiker, an welches sich vielleicht noch einige Leute erinnern können.

Früher hat man es meist mit einem Block gespielt, bei dem man die Ergebnisse direkt eingetragen hat. Da wir uns nun im Jahr 2018 befinden, und nicht mehr alle Lust haben, die Ergebnisse von Kopf auszurechnen und auf einem Papierblock zu schreiben, wird das Spiel nun als Webapplikation realisiert.

Mit diesem Programm können zwei Spieler zusammen an einem Rechner sitzen und spielen. Das Programm übernimmt die Rechenaufgaben und das Würfeln, während die Spieler spass beim spielen haben sollen.

**Aufgabenstellung**

Dieses Projekt wird mehrere Pages umfassen und eine Datenbank haben. Dafür sind verschiedene Anforderungen notwendig, damit man dieses Ziel erreichen kann. Hier sind man die Anforderungen für dieses Projekt:

Ich werde das Projekt nach MVC erstellen. Das heisst, es wird schlussendlich eine Visual Studio Solution bei der es eine Business Logic, eine Domain, Tests und das Webprojekt selber gibt. Zudem wird eine SQL Datenbank vorhanden sein, bei der die User und die Gegenstände gespeichert werden.  
Ich werde das Projekt so lösen, dass sich der aktuelle User mit Windows Authentication anmelden kann.

**Anforderungen**

Folgende Anforderungen sind durch das Modul an dieses Projekt gestellt:

* Objekt-orientierte Sprache
* Multiuser Applikation
* Relationale Datenbank
* Zentrale Datenbank
* Mehrere Clients sollten auf den gleichen Datenbestand zugreifen
* Transaktionssicherheit soll gewährleistet werden

**Mittel und Methoden**

* Projektmethode nach IPERKA
* Zeitplan mit Excel (Wird in die Word Dokumentation eingefügt.)
* ASP.NET MVC Webapplikation
* Versionierung mit Git (Projekt wird auf GitHub hochgeladen)

**Vorkenntnisse**

* Visual Studio mit ReSharper
* ASP.NET Grundkenntnisse
* SQL
* Angular (wird in diesem Projekt nicht benötigt)

### Anforderungen des Moduls

* Multiuser-Applikation
* Objektorientierte Programmiersprache
* Relationale Datenbank muss erstellt werden
* Viele Clients müssen auf die Datenbank zugreifen können

### Technologiestack

IDE: Visual Studio 2017  
Datenbank: SQL Server Management Studio  
Fronend: Razor  
Backend: ASP.NET MVC  
Versionierung: GitHub und TortoiseGit

## Projektorganisation

### Projektmanagement Methode

Viele Projektmanagement Methoden sind eher darauf ausgerichtet, dass man diese im Team verwenden kann. Deswegen habe ich mich entschieden die Probe IPA mit IPERKA zu realisieren, da es sich sehr gut dafür eignet, wenn man alleine arbeitet.  
Zudem kann man seine Arbeitsschritte im IPERKA genau auf jede Phase zuteilen, denn es ist ganz einfach aufgebaut.

Man muss den Auftrag verstehen und sich ein Bild des zu ereichenden Ziels machen. Dazu muss man sich ausführlich informieren.

Ziel der Planung ist es, einen Lösungsweg für das Endprodukt zu erstellen. Hier muss man die Arbeitsmittel und Arbeitsschritte einschätzen können.

Wenn man geplant hat, muss man sich entscheiden wie das Projekt genau realisiert werden muss.

Das ist der Zeitauswändigste Schritt von allen, denn die geplanten Arbeitsschritte werden einzeln ausgeführt. Wichtig ist, dass man die Planung wenn möglich einhält.

Wenn die Arbeit erledigt hat, muss man diese kontrollieren. Man muss schauen ob alles erledigt wurde, und die Qualität des Programmes gewährleistet ist.

Zum Schluss wertet man aus, wie die Arbeit gegangen ist. Welche Schritte sind mir gelungen? Welche eher nicht?

Abbildung 1

## Vorkenntnisse

Folgende Vorkenntnisse habe ich während der Lehre gesammelt:

* ASP.NET MVC (6 Monate)
* SQL (Erfahrungen aus dem Basislehrjahr)
* Visual Studio mit ReSharper (3 Jahre)
* Git & TortoiseGit (2,5 Jahre)
* Angular Typescript (1 Jahr)
* Visual Studio Code (1 Jahr)
* C# (1 Jahr)
* jQuery (3 Monate)

## Vorarbeiten

Bevor das Projekt offiziell begonnen hat, wurden schon kleine Vorarbeiten geleistet. Damit kann man den Stress während der Endspurtphase etwas geringer machen.

* Die Gliederung dieser Dokumentation wurde schon erstellt. Jede IPA ist verschieden, doch einige Punkte braucht es bei jeder Dokumentation
* IPERKA Abbildung, da dieses Thema bereits im ersten Lehrjahr bearbeitet wurde
* Git Repository wurde bereitgestellt, Link ist mit Remo Steinmann geteilt
* Inhalte welche schon im Voraus dokumentiert werden können, wurden erstellt

Die Struktur dieser Dokumentation wurde erstellt. Alle IPA Dokumentationen sind zwar verschieden, aber es hat ein paar Titel welche alle haben sollten. Über die wichtigsten Punkte wurde in einem Kurs bei Atos informiert und anschliessend umgesetzt.  
Ein Teil dieser Doku ist die Abbildung und Beschreibung vom IPERKA. Da dieses Thema bereits im ersten Lehrjahr behandelt wurde, konnte das Wissen benutzt werden um die Grafik schon im Voraus zu erstellen.

## Firmenstandards

Bei diesem Projekt stellt die Firma keine besonderen Standards. Tools und Technologie konnte selbst bestimmt werden.  
Atos hat ein Standard Template für Dokumentationen. Dieses wurde verwendet um die Dokumentation dieses Projektes zu schreiben.

## Zeitplan

### Meilensteine

Für das Projekt wurden Meilensteine festgelegt, welche die signifikanten Fortschritte eines Projektes darstellen. Da in diesem Projekt mit IPERKA gearbeitet wird, werden die Meilensteine nach jedem IPERKA Schritt gesetzt.

|  |  |  |
| --- | --- | --- |
| Schritt | Meilenstein | Datum/Zeit |
| I | Ich beginne die Arbeit und habe mich über verschiedene wesentliche Punkte informiert. Nach dem Informieren muss mir klar sein, was alles vorhanden ist, und was ich zu berücksichtigen habe während des ganzen Projektes. | 31.10.2018 um 12.00 Uhr |
| P | Ziel der Planung ist es, dass schon im Voraus überlegt wird, wie das Programm realisiert werden sollte. Die Planung ist ein Schritt, welcher wichtiger ist als die meisten denken. Denn hier ist das entscheidende, dass man während der Realisation nicht mehr so viel überlegen muss.  In diesem Schritt möchte ich Die Softwarearchitektur, die Views und die Datenbank geplant haben. | 01.11.2018 um 12.00 Uhr |
| E | Vor der Realisation sollte entschieden werden, wie die nicht vorgegebenen Schritte des Projektes vorgenommen werden sollten. Hier sollte dokumentiert sein, wie die Entscheidung gefallen ist. | 01.11.2018 um 14.00 Uhr |
| R | Bei der Realisation sollte das gesamte Projekt realisiert werden. Der Meilenstein ist zu Ende, wenn das Projekt ganz fertiggestellt ist. | 07.11.2018 um 12.00 Uhr |
| K | Nachdem das Projekt realisiert ist, muss es kontrolliert werden, weil die Qualität des Programmes gewährleistet werden muss. Ziel dieses Meilensteines ist es, Die geplanten Tests durchgeführt zu haben und die Ergebnissen sollten notiert werden. | 09.11.2018 um 10.00 Uhr |
| A | Ich schreibe die Reflexion und erstelle den Anhang. Somit wird die Dokumentation abgeschlossen, was auch der letzte Meilenstein dieses Projektes ist. | 09.11.2018 um 17.00 Uhr |

Tabelle 3: Meilensteine

### GANTT Diagramm



## Arbeitsprotokoll

|  |  |  |  |
| --- | --- | --- | --- |
|  | Tag 1 31.10.2018 | |  |
| Soll | | Ist | |
| Aufgabenstellung verbessert | | Erledigt | |
| Projektbeginn | | Erledigt | |
| Vorarbeiten und Vorkenntnisse Dokumentieren | | Erledigt | |
| Firmenstandards beschreiben | | Erledigt | |
| Soll-Zeit bei Zeitplan eingefügt | | Erledigt | |
| Use-Cases erstellen | | Leider ist mit dies nicht gelungen denn ich habe zuerst das Grundgerüst mit Visual Studio erstellt. | |
| Meilensteine definieren | | Erledigt | |
| Positives (Was lief gut?) | | | |
| Ich konnte die meisten Tasks welche für den heutigen Tag geplant waren erledigen. Zudem konnte ich das Grundgerüst des Projektes mit Visual Studio bereits erstellen.  Ich habe auch noch Programme, welche für die Realisation des Projektes benötigt werden auf den Rechner installiert, damit ich vorbereitet bin für die nächsten Tage. | | | |
| Negatives (Wo gab es Probleme?) | | | |
| Heute ist es mit nicht gelungen die Use-Cases zu erstellen, denn die Zeit war knapp, und ich hatte am Feierabend noch einen Termin, also konnte ich nicht länger bleiben. Da ich kurz bevor ich gegangen bin nur noch wenig Zeit hatte, und es bestimmt nicht für die Use-Cases gereicht hätte, habe ich das Grundgerüst für die Applikation in Visual Studio erstellt, da ich gewusst habe, dass dies nicht so viel Zeit in Anspruch nimmt. | | | |
| Zeitplanung | | | |
| Ich konnte die Zeitplanung nicht ganz einhalten, denn ich habe es nicht mehr auf die Reihe gebracht, die Use-Cases zu erstellen. Sonst sind alle anderen geplanten Arbeiten und noch eine vom folgenden Tag erledigt. | | | |
| Beanspruchte Hilfestellung | | | |
| Heute habe ich keine Hilfe benötigt, da ich hauptsächlich geplant und dokumentiert habe. | | | |
| Nächste Schritte | | | |
| Für den nächsten Tag habe ich geplant, gleich als erstes mit den Use-Cases zu beginnen. Ich möchte diese so schnell wie möglich erledigt haben, damit ich mit den nächsten Schritten weitermachen kann. | | | |

Tabelle 4: Arbeitsjournal Tag 1

|  |  |  |  |
| --- | --- | --- | --- |
|  | Tag 2 01.11.2018 | |  |
| Soll | | Ist | |
| Use-Cases erstellen | | Erledigt | |
| Software Architektur erstellen | | Die Software Architektur ist geplant, aber noch nicht grafisch dargestellt. | |
| ERD erstellen | | Das ERD ist ebenfalls schon geplant, aber auch noch nicht grafisch dargestellt | |
| Entscheid und Begründung für Projekt | | Erledigt | |
| ASP.NET Grundgerüst in Visual Studio erstellen | | Wurde schon am Vortag erledigt | |
| Verbesserungen am Dokument vornehmen | | Erledigt | |
| Positives (Was lief gut?) | | | |
| Ich konnte heute alle Use-Cases erstellen und diese mit Draw.io Desktop visuell darstellen. Zudem konnte ich alle Use-Cases ausführlich beschreiben.  Zudem hatte ich das erste Expertengespräch, bei dem ich viele nützliche Tipps erhalten habe, wie ich meine Dokumentation noch besser gestalten kann. Diese habe ich auch gleich umgesetzt.  Den Rest des Tages habe ich genutzt, um die restlichen Teile meines Projektes zu planen, z.B. das ERD und die Software Architektur. Ich habe mich auch am Schluss des Tages definitiv entschieden wie ich das Projekt realisieren werde. | | | |
| Negatives (Wo gab es Probleme?) | | | |
| Leider konnte ich das ERD und die Software Architektur nicht grafisch darstellen, denn die Erstellung und Darstellung der Use-Cases war sehr zeitauftreibend. Zudem hatte ich eine Stunde Besprechung mit dem Experten. Nun muss ich aufholen, damit das Projekt stressfrei laufen kann. | | | |
| Zeitplanung | | | |
| Ich konnte mich nicht an die Zeitplanung halten, denn ich habe keine Abbildungen erstellen können, weil die anderen Arbeiten sehr zeitauftreibend waren. | | | |
| Beanspruchte Hilfestellung | | | |
| Ich habe im Internet nach Use-Case Diagrammen gesucht, um auf Nummer sicher zu gehen, dass meine Use-Case Diagramme richtig aufgebaut sind. | | | |
| Nächste Schritte | | | |
| Nun werde ich mich an die Darstellung des ERDs und der Software Architektur machen, damit ich diese in die Dokumentation einfügen. Ich möchte die ersten drei Schritte des IPERKAS (Informieren, Planen und entscheiden) ganz erledigt haben. Wenn ich es schaffe widme ich mich auch schon der Entwicklung des Programmes. | | | |

Tabelle 5: Arbeitsjournal Tag 2

|  |  |  |  |
| --- | --- | --- | --- |
|  | Tag 3 02.11.2018 | |  |
| Soll | | Ist | |
| ERD grafisch darstellen | | Erledigt | |
| Software Architektur grafisch darstellen | | Erledigt | |
| Planung fertigstellen | | Erledigt | |
| Entscheidung Dokumentieren | | Erledigt, eventuell noch Erweiterungsbedarf | |
| Datenbank implementieren (Entity Framework mit Beziehungen, Datenbank und Server erstellen, Mockdaten einlesen) | | Erledigt | |
| WEB.API implementieren | | Keine Zeit mehr | |
| Views grob erstellen | | Zwei Views fehlen noch | |
| Funktionen für Views erstellen | | Ich konnte keine Funktionen implementieren, denn ich hatte keine Zeit mehr, da ich für diesen Tag mehrere Tasks geplant für welche ich keine Zeit hatte diese zu realisieren | |
| Positives (Was lief gut?) | | | |
| In Sachen Dokumentation konnte ich heute viel erreichen, denn Ziel war es das Informieren, Planen und Entscheiden fertigzustellen. Das ist mir auch gut gelungen, denn ich musste nur noch die grafischen Darstellungen machen und Texte ergänzen. Das ist mir schon nach kurzer Zeit gelungen. Ich war sehr erleichtert als ich die geplanten Dokumentations-Tasks fertigstellen konnte.  Nachdem ich fertig war, habe ich begonnen die Datenbank zu erstellen. Dazu habe ich in der Visual Studio Solution ein neues Projekt welches nur für die Domain da sein sollte erstellt. Zuerst habe ich das Entity Framework erstellt, mit den Tabellen und allen Assoziationen. Es ist alles reibungslos gegangen, ich hatte keine Probleme. Danach habe ich das Entity Framework in die Datenbank eingelesen und Mockdaten eingefügt.  Ich habe auch damit angefangen die Views zu erstellen. Dabei habe ich alle nöti | | | |
| Negatives (Wo gab es Probleme?) | | | |
| Ich arbeite in der Abteilung auch mit Visual Studio und SQL Server Management Studio, diese sind beide auf Englisch. Jedoch habe ich nun gemerkt, dass diese Programme auf meinem Rechner auf Deutsch sind. Da es mich irritiert hat, weil alle Controls anders heissen, musste ich ein Sprachpaket runterladen. Währenddessen konnte ich nicht weiter implementieren, dafür konnte ich die Inhalte welche schon dokumentiert wurden korrigieren. Immerhin war es nicht ganz eine Zeitverschwendung, denn ich habe sie produktiv nutzen können.  Weil ich keinen Server auf meinem Rechner hatte, musste ich auch einen MS Server installieren, damit ich die Datenbank erstellen konnte.  Zudem konnte ich zwei Views nicht erstellen, weil ich schlichtweg keine Zeit mehr zur Verfügung hatte. | | | |
| Zeitplanung | | | |
| Da ich noch Vorarbeiten vom letzten Tag erledigen musste, habe ich etwas spurten müssen, denn es sind nur noch zwei Tage übrig. | | | |

|  |
| --- |
| Beanspruchte Hilfestellung |
| Ich musste im Internet nachschauen wie ich am besten Bilder in einer SQL Datenbank speichern kann. Zudem musste ich auch den Ausbildner um Rat bitten, wie man am besten einen Server auf den Rechner installiert, denn ich habe das noch nie zuvor gemacht und ich wusste auch nicht wie. |
| Nächste Schritte |
| Ich werde am nächsten Tag gleich damit beginnen die restlichen Views zu erstellen, und ich werde auch darauf achten, dass diese Benutzerfreundlich aussehen.  Danach werde ich die WEB.API erstellen, damit die View die Daten sofort aus der Datenbank holen und anzeigen kann. Alle CRUD Funktionen sollten implementiert sein. |

Tabelle 6: Arbeitsjournal Tag 3

|  |  |  |  |
| --- | --- | --- | --- |
|  | Tag 4 07.11.2018 | |  |
| Soll | | Ist | |
| Restliche Views erstellen | | Erledigt | |
| WEB.API implementieren | | Teilweise erledigt | |
| Funktionen für Views implementieren | | Nicht erledigt aufgrund des Zeitdrucks | |
| Views verbessern | | Ich lasse diesen Schritt aus, denn er ist nicht relevant für die Funktion des Programmes. Es müssen andere Prioritäten gesetzt werden, damit das Projekt fertig wird. | |
| Unit Tests schreiben | | Die Applikation wird nicht mehr mit Unit Tests getestet aufgrund Zeitmangels. | |
| Testprotokoll erstellen | | Erledigt | |
| Testresultate notieren | | Das manuelle Testen wurde noch nicht durchgeführt | |
| Positives (Was lief gut?) | | | |
| Heute konnte ich alle Views fertig machen, ich habe darauf geachtet das nicht das Design, sondern die Benutzerfreundliche Bedienung gewährleistet wird. | | | |
| Negatives (Wo gab es Probleme?) | | | |
| Heute habe ich den Druck des Endspurtes am eigenen Leibe miterlebt. Es steht nur noch wenig Zeit zur Verfügung, doch es gibt noch einiges zu tun. Deswegen habe ich mich entschieden einige Punkte aus meinem Projekt zu löschen, denn realistischerweise kriege ich es einfach nicht mehr hin. | | | |
| Zeitplanung | | | |
| Ich bin in der Zeitplanung hintendrein. Die Applikation zu erstellen ist ein grösserer Auswand als ich geplant habe. | | | |
| Beanspruchte Hilfestellung | | | |
| Ich musste zum Teil im Internet nachforschen, denn ich habe Dinge implementiert, welche ich noch nie zuvor gemacht habe. | | | |
| Nächste Schritte | | | |
| Der nächste Schritt ist, den Endspurt so gut wie möglich hinzubekommen. Ich werde mich vor allem auf die Fertigstellung der Dokumentation konzentrieren. | | | |

Tabelle 7: Arbeitsjournal Tag 4

|  |  |  |  |
| --- | --- | --- | --- |
|  | Tag 5 09.11.2018 | |  |
| Soll | | Ist | |
| Fazit schreiben | | Erledigt | |
| Reflexion schreiben | | Erledigt | |
| Schlusswort verfassen | | Erledigt | |
| Dokumentation fertigstellen | | Erledigt | |
| Positives (Was lief gut?) | | | |
| Ich habe Dokumenttechnisch allen erreicht was ich erreichen wollte. Die Applikation ist zwar noch nicht fertig, doch ich bin erleichtert dass die Doku fertig ist. | | | |
| Negatives (Wo gab es Probleme?) | | | |
| Heute gab es nichts negatives, denn ich habe für heute nur dokumentieren geplant. | | | |
| Zeitplanung | | | |
| Ich konnte mich an die Zeitplanung einhalten. | | | |
| Beanspruchte Hilfestellung | | | |
| Heute habe ich keine Hilfe benötigt. | | | |
| Nächste Schritte | | | |
| Das Projekt ist nun feritg. | | | |

Tabelle 8: Arbeitsjournal Tag 5

# Teil 2: Projekt-Dokumentation

## Informieren

### Soll-Analyse

Bei der Soll-Analyse werden die Anforderungen genau analysiert.

#### GUI

Das GUI wird in diesem Projekt mit Razor und CSS erstellt. Bei Bedarf werden einige jQuery Elemente im Frontend vorhanden sein.

Wenn die Applikation geöffnet wird, kommt man auf die Startseite eines ASP.NET Projektes. Es wurden keine Änderungen oder Vorarbeiten geleistet.

Standardmässig hat es jedoch ein gutes Grundgerüst, nämlich eine Navigation mit einer Startseite, Infoseite, Kontaktseite und eine Getting started Seite. Die getting started Seite wird bearbeitet, damit diese dann zur starseite meines Projektes wird.

Das Design des GUIs ist sehr schlicht gehalten, deswegen wird es noch einiges zum Verändern geben. Zudem sollte die Seite so Benutzerfreundlich wie möglich werden.

#### Ist-Analyse

Der Zweck der Ist-Analyse ist, herauszufinden was das vorhandene System bereits zu bieten hat.

**Hardware**

Das Entwicklungsgerät ist ein privater PC, und zwar einen Acer Aspire V 15 Nitro, mit dem Windows 10 Betriebssystem.

**Entwicklungsumgebung**

Das Projekt wird mit Visual Studio 2017 Community realisiert. In dieser Edition ist kein ReSharper vorhanden, doch sonst ist alles andere bereits konfiguriert und bereit für die Implementation des Projektes.

**SQL**

Mithilfe von Visual Studio und Microsoft MySQL Server Management Studio wird die Datenbank zum Laufen gebracht.

**Microsoft SQL Server Management Studio**

SQL Server Management Studio wurde installiert, damit man das im Visual Studio erstellte edmx ganz einfach importieren kann, und die Daten in die Datenbank einlesen kann.

**Internetbrowser**

Auf dem Gerät sind die Browser Chrome, Vivaldi, Edge und Internet Explorer installiert.

**Git**

Git ist schon seit langer Zeit auf dem Entwicklungsgerät installiert und ist bereit genutzt zu werden. Um die Arbeit mit Git zu erleichern, ist auch TortoiseGit installiert, damit Commits und Pushs schneller erledigt werden können.

### Use-Cases

Um das Programm besser darstellen zu können, wurden Use-Cases erstellt.

**Profil erstellen**

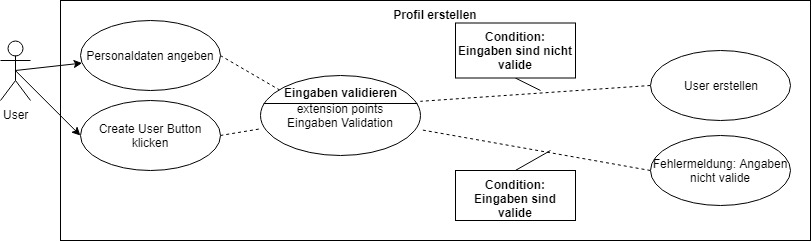


Abbildung 2

|  |  |
| --- | --- |
| Use-Case 1: Login | |
| Beschreibung | Ein neuer User muss ein Profil erstellen können |
| Personen | Neuer User |
| Eintrittsfall | Der User befindet sich auf der «User erstellen» Page. |
| Vorbedingungen | Der User ist ausgeloggt. |
| Resultat | Der User wird eingeloggt, und kann somit Aktionen ausführen, wenn er einen Gegenstand verkaufen möchte. |
| Ablauf | |  |  |  | | --- | --- | --- | | Schritt | Person | Beschreibung | | 1 | User | User öffnet die Login Page | | 2 | User | User gibt seine Personaldaten an | | 3 | System | Das System überprüft ob die eingegebenen Daten valide | | 4 | System | Erstellt den User | |

Tabelle 9: Use-Case 1

**Produkt erstellen & löschen**

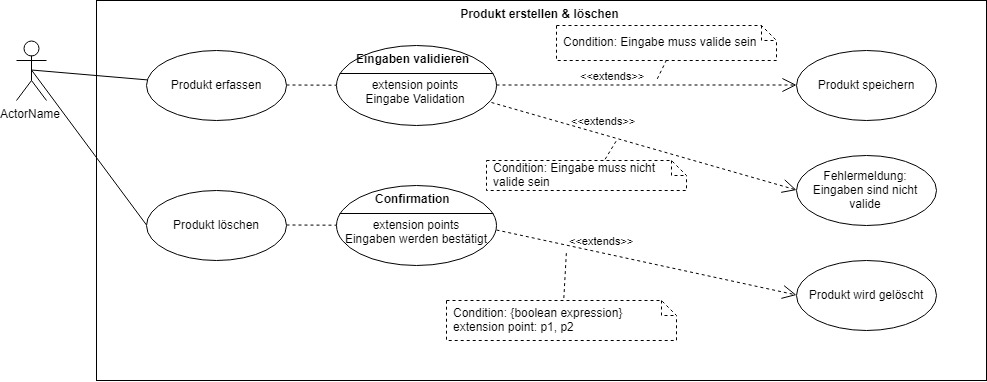


Abbildung 3

|  |  |
| --- | --- |
| Use-Case 2: Produkt erstellen und löschen | |
| Beschreibung | Ein angemeldeter User muss ein Produkt erfassen und hochladen. Zudem sollte es für ihn möglich sein, dieses Produkt zu löschen, wenn er diesen doch nicht mehr verkaufen möchte. |
| Personen | Angemeldeter User |
| Eintrittsfall | Wenn sich der User auf der «Sell Product» Seite befindet. |
| Vorbedingungen | Der User ist eingeloggt |
| Resultat | Nach dem Upload können anderen User das Produkt sehen und dieses kaufen |
| Ablauf | |  |  |  | | --- | --- | --- | | Schritt | Person | Beschreibung | | 1 | User | User öffnet die «Sell Product» Page | | 2 | User | User erfasst sein Produkt | | 3 | User | User klickt auf upload | | 4 | System | Das Produkt wird in der Datenbank gespeichert und wird auf der Product List angezeigt. | | 5 | User | User klickt auf «Delete Product» Button | | 6 | System | Das Produkt wird aus der Datenbank gelöscht | | 7 | System | Das Produkt ist nicht mehr auf der Produkteliste abgebildet. | |

Tabelle 10: Use-Case 2

**Produkt einkaufen**

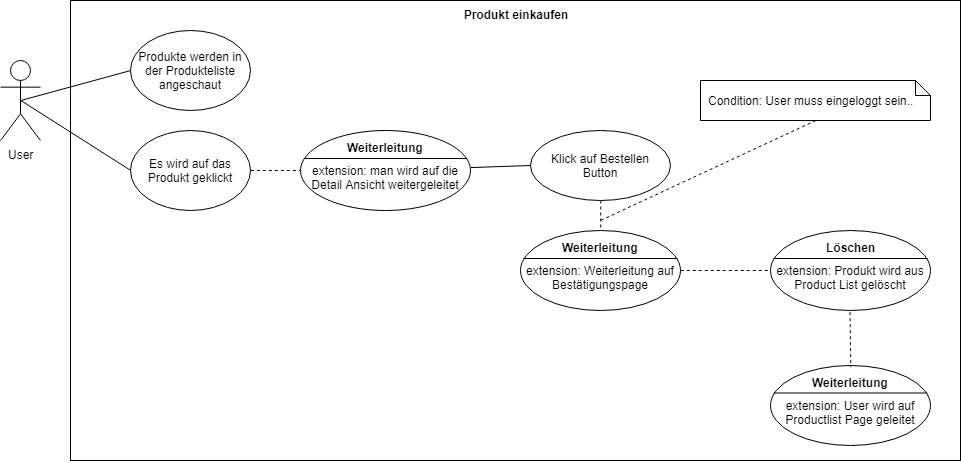


Abbildung 4

|  |  |
| --- | --- |
| Use-Case 3: Produkt einkaufen | |
| Beschreibung | Der User kann das Produkt in einer Detail Ansicht sehen, und kann von dort aus das Produkt kaufen. |
| Personen | User |
| Eintrittsfall | Der User klickt auf den «Buy» Button in der Detail View |
| Vorbedingungen | Der User muss eingeloggt sein |
| Resultat | Der User kehrt auf die Produkteliste zurück und der gekaufte Gegenstand ist nicht mehr verfügbar. |
| Ablauf | |  |  |  | | --- | --- | --- | | Schritt | Person | Beschreibung | | 1 | User | User klickt auf Gegenstand | | 2 | User | User klickt auf «Buy» Button | | 3 | System | Entfernt Gegenstand aus der Datenbank | | 4 | System | Leitet den User zurück auf die Produkteliste | |

Tabelle 11: Use-Case 3

## Planen

### Software Architektur

Die Software Architektur beschreibt, wie der technische Aufbau der Applikation ist. Auf einer Grafik wurde dargestellt, wie das Projekt schlussendlich aufgebaut werden sollte.

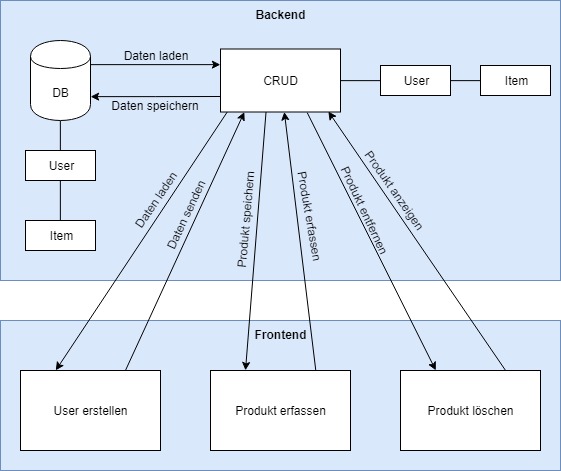


Abbildung 5

Die Architektur ist folgendermassen aufgebaut:  
Die Applikation hat eine Datenbank (oben links) welche die Tabellen User und Item haben. Diese wird per Web.API an den Client gesendet. Die Web.API enthält alle CRUD Funktionen, damit die Daten vielseitiger genutzt werden können. CRUD steht für folgende Funktionen: Create, Read, Update und Delete. Für dieses Projekt sind zwar nicht alle notwendig, doch es werden trotzdem alle Funktionen implementiert, weil es sonst nicht CRUD wäre. Das CRUD verbindet beide, die User und die Item Tabelle mit dem Frontend.

Im Frontend sind drei verschiedene Aktionen welche die Applikation ausführen muss. Alle drei Aktionen sind mit dem CRUD verknüpft, damit die Daten mit der Datenbank ausgetauscht werden können.

**User erstellen**: Wenn ein User erstellt wird muss dieser in der Datenbank gespeichert werden.

**Produkt erfassen**: Ein User sollte ein Produkt erfassen können, welches anschliessend in die Datenbank gespeichert werden kann.

**Produkt löschen**: Wenn ein User fälschlicherweise ein Produkt hochgeladen hat, kann er diesen auch Löschen. Wenn ein Produkt gelöscht wird, sollte er auch endgültig aus der Datenbank entfernt sein.

### Datenbankdesign

Die Datenbank für dieses Projekt ist eher klein. Sie besteht aus lediglich zwei Tabellen, welche mit einer 1 zu N Verbindung verknüpft sind. Es gibt eine Tabelle für die User und eine für die Gegenstände welche auf der Seite verkauft werden.

Ein User kann mehrere Items, beziehungsweise Gegenstände haben, doch ein Gegenstand kann nur einem User gehören. Wie man auf der folgenden Abbildung sieht, ist das 1 beim User und das N beim Item.

Die Datenbank wird mithilfe von Visual Studio und Visual Studio Code realisiert.

###### Entity-Relationship-Model

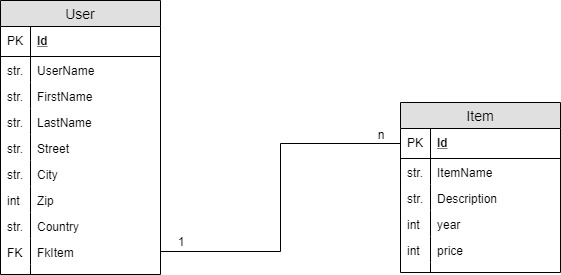


Abbildung 6

Um die Applikation noch performanter zu machen, wird es für einige Datensätze eine limitierte Zeichenlänge für die Strings haben. Auf folgender Tabelle ist ersichtlich, wie die Properties eingestellt werden sollten.

**User**

|  |  |  |
| --- | --- | --- |
| Typ | Name | Kommentar |
| String | UserName | Max. Length: 100, Nullable: False |
| String | FirstName | Max. Length: 100, Nullable: False |
| String | LastName | Max. Length: 100, Nullable: False |
| String | Street | Max. Length: Max, Nullable: False |
| String | City | Max. Length: Max, Nullable: False |
| Int | Zip | Max. Length: 15, Nullable: False |
| String | Country | Max. Length: 200, Nullable: False |
| Int | FkItem | Max. Length: 10, Nullable: False |
| Navigation Property | Items | Getter: Public, Setter: Public, Multiplicity: Many (n) |

Tabelle 12: User ERD Anforderungen

**Item**

|  |  |  |
| --- | --- | --- |
| Typ | Name | Kommentar |
| String | ItemName | Max. Length: 100, Nullable: False |
| String | Description | Max. Length: Max, Nullable: False |
| Int | Year | Max. Length: 100, Nullable: False |
| Int | Price | Max. Length: Max, Nullable: False, Currency Mode: Fixed |
| Int32 | UserId | Nullable: False |
| Navigation Property | User | Getter: Public, Setter: Public, Multiplicity: 1 |

Tabelle 13: Item ERD Anforderungen

###### Tabellen

User

In dieser Tabelle sind alle User gespeichert.

|  |  |
| --- | --- |
| Attribut | Beschreibung |
| Id | Jede Tabelle braucht eine Id, damit alle Datensätze Unique sind. |
| UserName | Jeder User braucht einen UserName, damit man diesen identifizieren kann, wenn er etwas am Verkaufen ist. |
| FirstName | Braucht man um das Produkt an die richtige Person zu adressieren. |
| LastName | Braucht man um das Produkt an die richtige Person zu adressieren. |
| Street | Braucht man um das Produkt an die richtige Person zu adressieren. |
| City | Braucht man um das Produkt an die richtige Person zu adressieren. |
| Zip | Braucht man um das Produkt an die richtige Person zu adressieren. |
| Country | Braucht man um das Produkt an die richtige Person zu adressieren. |
| FkItem | Dieser Foreign Key wird benötigt, damit die Items dem richtigen User zugewiesen werden. Ein User kann unendlich viele Foreign Keys haben. |

Tabelle 14: Beschrieb User Spalten

Item

In dieser Tabelle sind alle Gegenstände gespeichert.

|  |  |
| --- | --- |
| Attribut | Beschreibung |
| Id | Jede Tabelle braucht eine Id, damit alle Datensätze Unique sind. |
| ItemName | Jedes Item braucht einen Namen, damit die Kunden wissen was es ist. |
| Description | Jedes Item braucht eine Description, damit die Kunden genau wissen was es ist. |
| Year | Die Produkte sollten gekennzeichnet sein, aus welchem Jahrgang sie sind. |
| Price | Natürlich ist es wichtig, wenn die Kunden wissen wie teuer der Gegenstand ist. |

Tabelle 15: Beschrieb Item Spalten

### Testprotokoll

Nach der Implementierung des Projektes, muss man die Applikation auf Herz und Nieren prüfen. Vor dem Testen wurden Testfälle verfasst. Unten sieht man die Protokolle dieser Tests und das Resultat.

|  |  |
| --- | --- |
| Testfall 1: Navigation funktioniert | |
| Vorbedingungen: | Navigation mit allen nötigen Bullets sollte vorhanden sein. |
| Testablauf: | |  |  | | --- | --- | | Schritt | Beschreibung | | 1 | Klick auf einen beliebigen Bullet Point | | 2 | Weiterleitung auf die korrekte Page | |
| Testmittel: | Vivaldi Browser |
| Erwartetes Resultat | Die Navigation sollte den Benutzer bei Klick auf die Bullets auf die richtige Page leiten. Testablauf sollte bei allen Bullet Points ausgeführt werden und ein erfolgreiches Resultat aufweisen. |

Tabelle 16: Testfall 1

|  |  |
| --- | --- |
| Testfall 2: Validierung bei User erstellen | |
| Vorbedingungen: | User hat keinen Account |
| Testablauf: | |  |  | | --- | --- | | Schritt | Beschreibung | | 1 | User Daten werden eingegeben | | 2 | Klick auf «Create User» Button | | 3 | Eingabe wird validiert | | 4 | Bei korrekter Validation wird der User kreiert | |
| Testmittel: | Vivaldi Browser |
| Erwartetes Resultat | Bei Erstellung eines Accounts, wird der User aufgefordert Informationen über sich einzutippen. Diese Informationen müssen korrekt validiert werden. Bei einer fehlerhaften Eingabe, kann der User nicht erstellt werden und der User wird aufgefordert seine Angaben zu überarbeiten. Wenn die Inputs korrekt ausgefüllt wurden, wird der User erstellt. |

Tabelle 17: Testfall 2

|  |  |
| --- | --- |
| Testfall 3: Produkt hochladen, Validation | |
| Vorbedingungen: | User ist eingeloggt und hat Produkt erfasst |
| Testablauf: | |  |  | | --- | --- | | Schritt | Beschreibung | | 1 | User klickt auf «Upload Item» | | 2 | Validation überprüft Eingabe | |
| Testmittel: | Vivaldi Browser |
| Erwartetes Resultat | Bei fehlerhafter Eingabe kommt eine Fehlermeldung. Bei korrekter Eingabe kann das Produkt hochgeladen werden. |

Tabelle 18: Testfall 3

|  |  |
| --- | --- |
| Testfall 4: Produkt hochladen | |
| Vorbedingungen: | User muss eingeloggt sein |
| Testablauf: | |  |  | | --- | --- | | Schritt | Beschreibung | | 1 | User befindet sich auf der «Upload Item» Page | | 2 | User erfasst Informationen für das Produkt. | | 3 | Klick auf «Upload Item» | |
| Testmittel: | Vivaldi Browser |
| Erwartetes Resultat | Item wird hochgeladen und ist auf der Produkteliste ersichtlich. |

Tabelle 19: Testfall 4

|  |  |
| --- | --- |
| Testfall 5: Produkt verkaufen | |
| Vorbedingungen: | User muss eingeloggt sein |
| Testablauf: | |  |  | | --- | --- | | Schritt | Beschreibung | | 1 | User Ist auf Detailansicht eines Produktes. | | 2 | User klickt auf «Buy» | | 3 | Handlung wird verarbeitet | | 4 | Weiterleitung auf die Produkteliste | |
| Testmittel: | Vivaldi Browser |
| Erwartetes Resultat | Wenn ein User ein Produkt kauft, wird dieses aus der Produkteliste und der Datenbank entfernt. |

Tabelle 20: : Testfall 5

|  |  |
| --- | --- |
| Testfall 6: Produkt löschen | |
| Vorbedingungen: | User muss ein Produkt hochgeladen haben |
| Testablauf: | |  |  | | --- | --- | | Schritt | Beschreibung | | 1 | Bei der Produkteliste auf «Delete Item» klicken | |
| Testmittel: | Vivaldi Browser |
| Erwartetes Resultat | Wenn der User, welcher das Produkt hochgeladen hat auf den «Delete Item» Button klickt, sollte das Produkt aus der Datenbank und somit auch von der Produkteliste gelöscht werden. |

Tabelle 21: : Testfall 6

## Entscheiden

In der Entscheidungsphase ist es das Ziel, sich für jene Sachen zu entscheiden, welche nicht vorgegeben wurden, das heisst, alles wofür der Entwickler selber verantwortlich ist.

**Entwicklungsumgebung**

**Das Projekt wird mit Visual Studio 2017 realisiert, weil ich schon viele praktische Erfahrungen damit sammeln konnte. Zudem ist es praktisch, denn man kann Client, Business Logic, Datenbank etc. hier realisieren. Diesen Vorteil kommt mir zu Gute, denn es ist zum Teil irreführend, wenn man mit zu vielen Tools gleichzeitig arbeiten muss.**

**Das Entity Framework realisiere ich zwar auch mit Visual Studio, doch zur Hilfe werde ich auch Microsoft SQL Server Management Studio verwenden, um die Datenbank für die Applikation zu erstellen. Dieses Tool ist sehr praktisch, denn man kann Datenbanken in den Servern erstellen, um dann das, im Visual Studio erstellte Entity Framework einzulesen. Mit SQL Server Management Studio ist es zudem einfach Daten in die Datenbank einzufügen. Somit kann die Applikation einfacher getestet werden.**

**Bei der Solution achte ich darauf, dass ich**

**Versionierung**

**Damit die Sicherstellung des Projektes gewährleistet ist, habe ich mich dazu entschieden das Projekt nicht nur auf einem USB Stick zu sichern, sondern es auch auf ein Git Repository zu laden. Dieses Git Repository wird auf GitHub verfügbar sein. Der Experte wird Zugriff auf dieses Repository haben.**

**Um meine aktuellen Stände auf das Repository zu laden werde ich TortoiseGit verwenden. Ich benutze dieses Tool, denn es ist einfach damit einen Commit und einen Push zu machen. Es ist auch gut, um Änderungen der Files auf älteren Commits zu sehen. Zudem ist es einfach einen Revert zu machen, falls die Applikation während der Implementation plötzlich abstürzt.**

**Vorgehensweise**

**Als erstes wird bei der Implementation das Grundgerüst der Webapplikation erstellt. Danach wird die Datenbank implementiert.**

**Nachdem die Datenbank fertig implementiert ist, wird das Webprojekt implementiert. Wenn das Projekt fertig ist, wird alles getestet, um die Qualität des Programmes zu gewährleisten.**

**Ich habe mich für diesen Vorgang entschieden, weil ich bei Atos immer so vorgegangen bin. Ich habe mich an diese Vorgehensweise gewöhnt und es funktioniert gut für mich.**

**Testing**

**Für das Testing wird in der Visual Studio Solution ein Unit Test Projekt erstellt. Hier werden Grundfunktionen des Programmes getestet. Das sind zum Beispiel Tests, bei denen man die Kommunikation mit dem Server überprüft.**

**Doch hauptsächlich werde ich die Applikation mit manuellen Tests testen. Diese Tests sind praktisch, denn man kann die Use-Cases zu Tests umschreiben. Man kann dann alle Schritte überprüfen, um zu sehen ob das Projekt gut implementiert wurde.**

## Realisieren

Bei der Realisation wird das ausführlich geplante Projekt umgesetzt. In diesem Kapitel sollte ersichtlich sein was realisiert wurde, und wie die Vorgehensweise ist.

Wie schon erwähnt, wird in der Solution der Applikation mehrere Projekte erstellt, damit alles getrennt ist, und der Entwickler eine gute Übersicht hat.

### Secondhand.BusinessLogic

Die Business Logic ist da, damit die Datenbank weiss wie Daten gespeichert, erstellt, verändert und gelöscht werden sollten. Im ASP.NET macht man für dies eine WEB.API.

Die Business Logic wurde nicht implementiert, denn der Stand der Applikation trägt dazu bei, dass eine Business Logic nicht brauchbar wäre. Jedoch ist das Grundgerüst der Business Logic verfügbar.

### Secondhand.Domain

In der Domain wird alles erstellt was mit der Datenbank zu tun hat.

**Entity Framework**

Damit die Datenbank erstellt werden kann muss ein Entity Framework erstellt werden. Man kann dies manuell oder mit dem Designer machen. In diesem Projekt wurde die Struktur mit dem Designer erstellt, denn die Zeit für dieses Projekt war sehr beschränkt also wollte ich es so schnell wie möglich erledigt haben. Nachdem das Entity Framework fertig war, habe ich das dazugehörige SQL File in der Datenbank ausgeführt.

Den Server und die Datenbank wurden bereits im Voraus im SQL Server Management Studio erstellt. Der Name der Datenbank ist: SecondhandDatabase.

Auf folgender Abbildung sieht man die beiden Tabellen welche nach der Ausführung des SQL Files erstellt wurde.

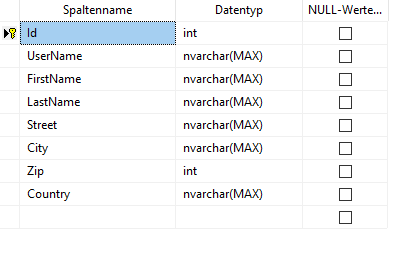


Abbildung 7: Tabelle User

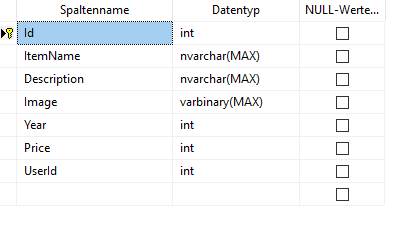


Abbildung 8: Tabelle Item

Nachdem die beiden Tabellen erfolgreich in die Datenbank eingelesen wurden, und alles nochmals säuberlich überprüft wurde, wurde die Datenbank mit Testdaten befüllt. Dies wurde mithilfe eines SQL Files gemacht, welches ebenfalls im SQL Server Management Studio erstellt wurde.

Die Datenbank wurde mit einem SQL Script befüllt, weil eine Spalte vom Datentyp Binary ist. In dieser Spalte sollten die Bilder der Produkte gespeichert werden. Da es schwer ist, Binärdaten manuell zu erfassen, wurden diese eben mit dem Script eingefügt.

INSERT INTO dbo.Items(ItemName, Description, Image, Year, Price, UserId)

Values ('Bratpfanne',

'Eine Pfanne',

(SELECT \* FROM OPENROWSET(BULK N'F:\ProbeIPA\Src\Secondhand.Domain\BeispielBilder\bratpfanne.jpg',

SINGLE\_BLOB) AS Image), 1990, 80, 1);

INSERT INTO dbo.Items(ItemName, Description, Image, Year, Price, UserId)

Values ('Antiker Sessel',

'Ein ganz spezieller antiker Sessel',

(SELECT \* FROM OPENROWSET(BULK N'F:\ProbeIPA\Src\Secondhand.Domain\BeispielBilder\sessel.jpg',

SINGLE\_BLOB) AS Image), 1850, 5000, 3);

INSERT INTO dbo.Items(ItemName, Description, Image, Year, Price, UserId)

Values ('Gieskanne',

'Gieskanne für Ihren Garten. Nie gebraucht.',

(SELECT \* FROM OPENROWSET(BULK N'F:\ProbeIPA\Src\Secondhand.Domain\BeispielBilder\giesskanne.jpg',

SINGLE\_BLOB) AS Image), 2018, 7, 2);

Wie man sieht wurden alle Values standgemäss eingefügt, doch die Bilder haben folgende Zeile benötigt um eingefügt zu werden:

(SELECT \* FROM OPENROWSET(BULK N'F:\ProbeIPA\Src\Secondhand.Domain\BeispielBilder\giesskanne.jpg',

SINGLE\_BLOB) AS Image)

Der Pfad muss noch zusätzlich in einer SELECT Abfrage angegeben werden. Zusätzlich muss man auch noch angeben, dass das Bild ein BLOB ist, welcher als Bild in der Datenbank gespeichert werden sollte.

### Secondhand.Tests

Ich hatte in der Planungsphase geplant, die Tests mithilfe von Unit Tests und manuellen Tests durchgeführt werden. Da die Zeit sehr knapp war, um alles zu machen, wurden nur die manuellen Tests gemacht.

Das heisst das Test-Projekt ist leer und es kann nichts Weiteres dazu dokumentiert werden.

### Secondhand.Web

Das Webprojekt ist das Frontend mit den ganzen Klassen und Funktionen welche der Benutzer auch sieht, wenn er die Applikation anwendet.

Das Webprojekt wurde mit MVC realisiert.



Abbildung 9: MVC

MVC ist ein Architektur-Pattern. Das Gute daran ist, dass das Projekt sauber in die verschiedenen Bereiche aufgeteilt werden. Hierbei spielen bei MVC folgende Dinge eine Rolle:

Der User sieht die View und führt dort sämtliche Aktionen aus. Diese Aktionen werden vom Controller übernommen, den dieser hat alle Funktionen implementiert, welcher der View braucht. Der Controller manipuliert dann das Model, und das Model aktualisiert die View.

In der obigen Abbildung sieht man den Kreislauf, welcher das MVC bei Ausführung jeder Aktion macht.

Im Projekt hat es je einen Ordner für das MVC. Als erstes wurden die Views erstellt, denn wenn die View schon steht ist es einfacher die dazugehörige Implementation anzufügen. Es wurde eine View für alle notwendigen Pages erstellt. Auf den Screenshots sieht man, wie die View aussieht.

Das Design der Applikation wurde eher schlicht gehalten, weil nicht besonders viel Zeit zur Verfügung steht. Jedoch wurde darauf geachtet, dass das Design trotzdem einigermassen anschaulich ist.

**Homapage**

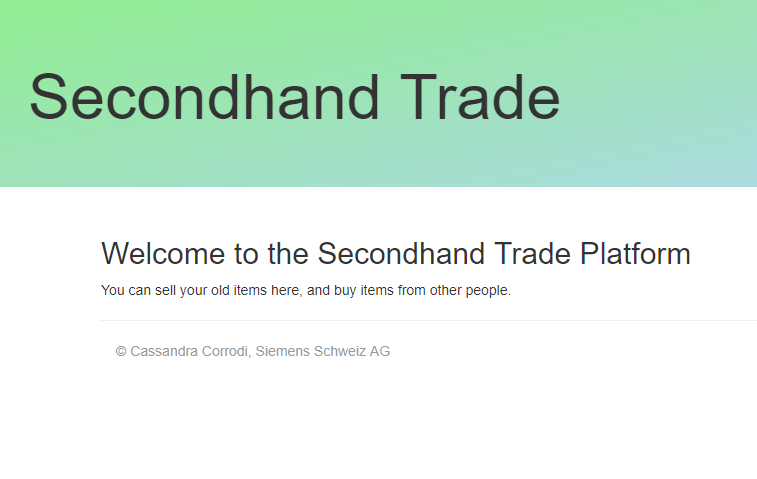


Abbildung 10: Screenshot Homepage

**Produkteliste**

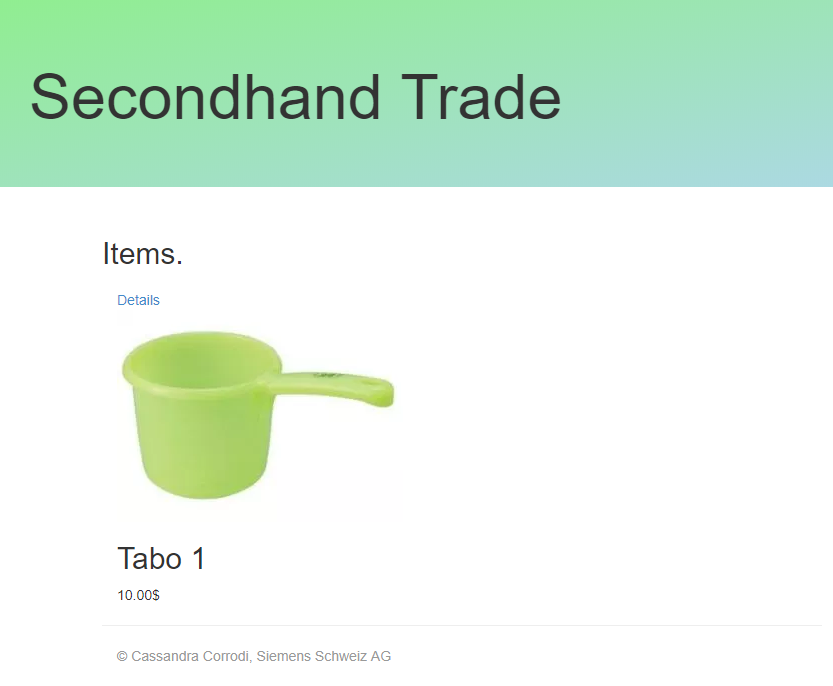


Abbildung 11: Screenshot Produkteliste

**Detailansicht der Produkte**

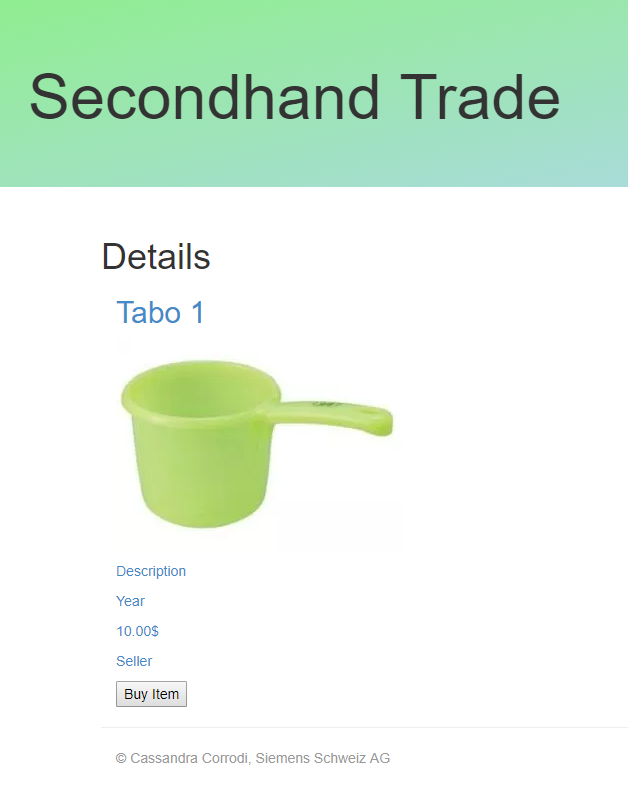


Abbildung 12: Screenshot Detailansicht

**User erstellen**

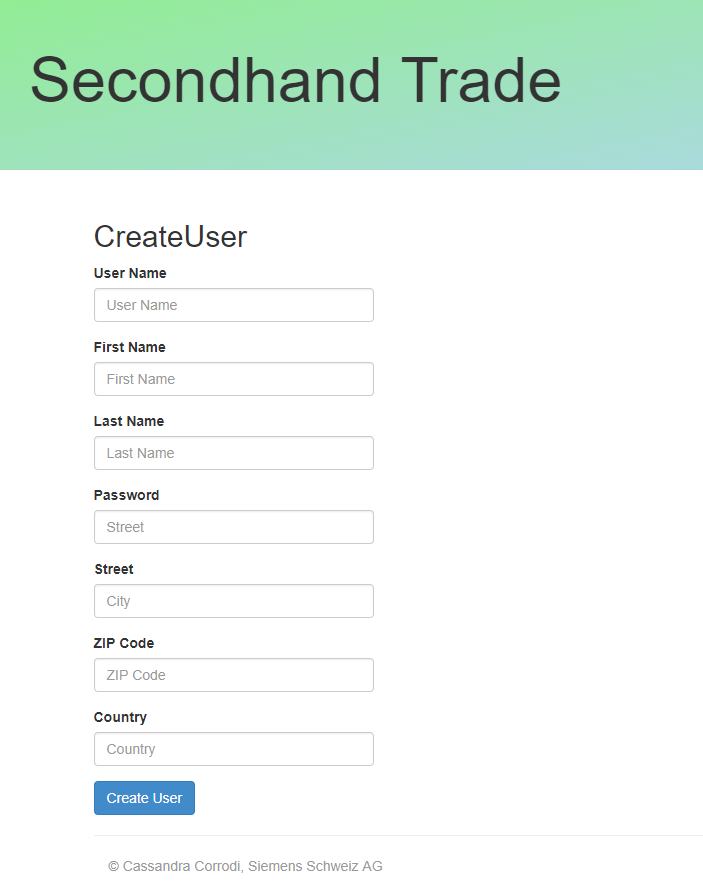


Abbildung 13: Screenshot User erstellen

Nachdem alle Views erstellt wurden, wurde an den Controllern gearbeitet.

Zuerst wurden Klassen implementiert, welche dafür das sind, dass die Views angezeigt werden. Zudem wurde auch das Routing für jede View erstellt, damit man sich schonmal durch die Applikation probeklicken kann.

Das Routing muss konfiguriert werden. Unten sieht man wie es in diesem Projekt gemacht wurde.

routes.MapRoute(

name: "Default",

url: "{controller}/{action}/{id}",

defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }

);

Je nachdem was benötigt wird, können Routings bei allen verschieden aussehen. Doch hier ist es folgendermassen gemacht:

* url: wenn man im Browser eine URL eintippt, muss nach dem Namen der Seite (in diesem Fall: localhost: 51011) zuerst eintippen. Danach muss man den Namen des Controllers, der Action, und wenn vorhanden von der ID eingeben.
* Defaults: bei Defaults werden Ausnahmen definiert. Oben sieht man das «home» nicht in der URL angezeigt wird, wenn man sich auf der Startseite befindet.  
  Zudem wird auch Index nicht angezeigt, wenn man sich auf ein Index HTML File leiten lässt. Bei ID sieht man, wie schon oben erwähnt wurde, dass diese optional ist. Das ist so, weil nicht alle Pages eine ID haben. Jedoch ist die ID Beispielsweise sinnvoll, wenn sich der User auf der Detailansicht eines Produktes befindet.

Dieser Code zeigt, wie von der Homepage auf die Item-Page das Routing implementiert wurde.

@Html.ActionLink("Items", "Index", "Item")

In Razor wird mit @

Für dieses Projekt wurde ein simples Routing implementiert, um Komplikationen und Probleme aufgrund mangelnder Zeit zu verhindern.

Für das Webprojekt wurden auch Controllers implementiert. Die View benötigen Controllers, da die Aktionen ohne gar nicht implementiert werden können. Die Controller wurden in einen separaten Ordner geladen und es hat einen für Home, Items und User.

In den Controllern wurde implementiert, dass die Views angezeigt werden sollten. Auf folgendem Code ist ersichtlich, wie es in diesem Projekt realisiert wurde.

public ActionResult Index()

{

return View();

}

Der Index ist für die HTML Ansicht. Und das return View() ist da, damit das Programm das HTML File als User Interface zurückgibt.

## Kontrollieren

Ziel der Kontrolle ist es, die Applikation auf Herz und Nieren zu prüfen. Die im Kapitel «Planen» erfassten Testfälle werden durchgeführt und die Resultate notiert.

Ziel der Kontrolle sollte es sein, die Applikation auf Herz und Nieren zu prüfen. Zudem ist hier für den Entwickler ersichtlich, ob er das Programm gut implementiert hat.

|  |  |
| --- | --- |
| Testfall 1: Navigation funktioniert | |
| Erwartetes Ergebnis | Wenn der Benutzer auf die Navigation klickt, dann sollte er auf die entsprechenden Pages kommen. Es sollten keine Error 404 erscheinen. |
| Testergebnis | Positives Testergebnis. |

Tabelle 22: Testergebnis 1

|  |  |
| --- | --- |
| Testfall 2: Validierung bei User erstellen | |
| Erwartetes Ergebnis | Wenn der User sein Profil erstellt, wird validiert ob seine Angaben korrekt sind. |
| Testergebnis | Keine Zeit für die Fertigstellung dieses Features. |

Tabelle 23: Testergebnis 2

|  |  |
| --- | --- |
| Testfall 3: Produkt hochladen, Validation | |
| Erwartetes Ergebnis | Die Eingaben bei dem «Upload Product» Tab sollten validiert werden wenn der User etwas eingegeben hat. |
| Testergebnis | Keine Zeit für die Fertigstellung dieses Features. |

Tabelle 24: Testergebnis 3

|  |  |
| --- | --- |
| Testfall 4: Produkt hochladen | |
| Erwartetes Ergebnis | Ein User sollte ein Produkt auf die Plattform laden können. Dieses Produkt sollte dann in der Produkteliste angezeigt werden. |
| Testergebnis | Keine Zeit für die Fertigstellung dieses Features. |

Tabelle 25: Testergebnis 4

|  |  |
| --- | --- |
| Testfall 5: Produkt verkaufen | |
| Erwartetes Ergebnis | Wenn ein User ein Produkt kaufen möchte. Sollte er über die Detail Ansicht per Button das Produkt kaufen können. Das Produkt sollte danach aus der Produkteliste entfernt werden. |
| Testergebnis | Keine Zeit für die Fertigstellung dieses Features. |

Tabelle 26: Testergebnis 5

|  |  |
| --- | --- |
| Testfall 6: Produkt löschen | |
| Erwartetes Ergebnis | Wenn ein User eines Seiner Produkte löschen möchte, sollte er dies über Button-Klick machen können. Das Produkt sollte danach aus der Produkteliste entfernt werden. |
| Testergebnis | Keine Zeit für die Fertigstellung dieses Features. |

Tabelle 27: Testergebnis 6

## Auswerten

### Reflexion

Ich habe während der Probe IPA versucht, mein Bestes zu geben. Jedoch habe ich bemerkt, dass es ein riesiger Aufwand ist eine IPA zu schreiben.

Ich habe den Aufwand ziemlich unterschätzt, vor allem in Sachen Dokumentation. Die ersten drei Tage habe ich fast hauptsächlich mit der Dokumentation verbracht, denn es ist ein riesen Aufwand das Informieren und die Planung zu machen. Zudem habe ich bei der Entscheidung mehr dokumentiert als ich anfangs geplant habe. Ich habe die IPA von anderen Leuten schonmal gesehen, doch ich habe gedacht, dass diese Dinge schnell dokumentiert werden können, weil ich nicht damit gerechnet habe, dass man so viel überlegen muss beim Dokumentieren. Zudem muss man an alle Anforderungen denken, welche im Kriterienkatalog definiert sind.

Was mich auch ziemlich ins Schwitzen gebracht hat, war die Erstellung der grafischen Darstellungen. Ich habe zwar schon oft mit Draw.io gearbeitet, doch noch nie so intensiv wie bei der Probe IPA.

Ich muss aber auch zugeben, dass mir immerhin die Dokumentation ziemlich gut gelungen ist, denn ich habe sehr viel Arbeit in diese gesteckt, weil ich wollte, dass alle Punkte säuberlich dokumentiert wurden. Es hat sehr viel Zeit und Nerven in Anspruch genommen, doch ich bin mit dem Endresultat ziemlich zufrieden. Ich habe mehr geschrieben als ich anfangs erwartet habe.

Das grosse Problem bei meiner Applikation ist nicht, dass ich wenig implementiert habe. Es ist, dass die verschiedenen Elemente nicht miteinander verknüpft sind. Somit meint der User, wenn er die Applikation bedient das es ziemlich mickrig ist, jedoch hat es viel Code im Hintergrund. Ich finde es schade das es nicht gezeigt wird, wenn man die Applikation gezeigt wird, denn ich habe mir ziemlich viel Mühe gegeben bei der Implementation.

### Schlusswort

Ich habe gelernt, dass es wichtig ist, dass man vor allem am Anfang so effizient wie möglich zu arbeiten, denn sonst kommt man in einen grossen Zeitdruck und kann die Applikation nicht fertig implementieren. Zudem sollte ich mich auch sputen, wenn ich meine implementierten Files miteinander verknüpfen muss damit die Funktionalitäten für den User ersichtlich sind.

Ich werde das gelernte dieser Probe IPA im Frühjahr 2019 für die echte IPA anwenden, und diese hoffentlich mit einem guten Ergebnis bestehen.

# Teil 3: Anhang

## Quellenverzeichnis

|  |  |
| --- | --- |
| Was? | Quelle |
| Erklärung wie man ein Formular mit Bootstrap erstellt. | https://www.tutorialrepublic.com/twitter-bootstrap-tutorial/bootstrap-forms.php |
| Hilfe bei Visual Studio & C# Erklärungen | https://msdn.microsoft.com/de-de/ |

Tabelle 28: Quellenverzeichnis

Abbildungsverzeichnis

|  |  |
| --- | --- |
| Begriff | Erklärung/Quelle |
| Abbildung 1 | IPERKA Abbildung |
| Abbildung 2 | Use-Case 1: Profil erstellen |
| Abbildung 3 | Use-Case 2: Produkt erstellen und löschen |
| Abbildung 4 | Use-Case 3: Produkt einkaufen |
| Abbildung 5 | Software Architektur |
| Abbildung 6 | Entity Relationship Diagram |
| Abbildung 7 | Screenshot: Tabelle User bei SQL Server Management Studio |
| Abbildung 8 | Screenshot: Tabelle Item bei SQL Server Management Studio |
| Abbildung 9 | https://upload.wikimedia.org/wikipedia/commons/a/a0/MVC-Process.svg |
| Abbildung 10 | Homepage: Screenshot vom Programm |
| Abbildung 11 | Produkteliste: Screenshot vom Programm |
| Abbildung 12 | Detailansicht: Screenshot vom Programm |
| Abbildung 13 | User erstellen: Screenshot vom Programm |

Tabelle 29: Abbildungsverzeichnis

https://upload.wikimedia.org/wikipedia/commons/thumb/a/a0/MVC-Process.svg/500px-MVC-Process.svg.png

## Tabellenverzeichnis

|  |  |
| --- | --- |
| Begriff | Erklärung |
| Tabelle 1 | Änderungshistorie |
| Tabelle 2 | Meilensteine |
| Tabelle 3 | Arbeitsprotokoll Tag 1 |
| Tabelle 4 | Arbeitsprotokoll Tag 2 |
| Tabelle 5 | Arbeitsprotokoll Tag 3 |
| Tabelle 6 | Arbeitsprotokoll Tag 4 |
| Tabelle 7 | Arbeitsprotokoll Tag 5 |
| Tabelle 8 | Use-Case 1: Profil erstellen |
| Tabelle 9 | Use-Case 2: Produkt erstellen und löschen |
| Tabelle 10 | Use-Case 3: Produkt einkaufen |
| Tabelle 11 | User ERD Anforderungen |
| Tabelle 12 | Item ERD Anforderungen |
| Tabelle 13 | Beschrieb User Spalten |
| Tabelle 14 | Beschrieb Item Spalten |
| Tabelle 15 | Testfall 1: Navigation funktioniert |
| Tabelle 16 | Testfall 2: Validierung bei User erstellen |
| Tabelle 17 | Testfall 3: Produkt hochladen, Validation |
| Tabelle 18 | Testfall 4: Produkt hochladen |
| Tabelle 19 | Testfall 5: Produkt verkaufen |
| Tabelle 20 | Testfall 6: Produkt löschen |
| Tabelle 21 | Testergebnis von Testfall 1 |
| Tabelle 22 | Testergebnis von Testfall 2 |
| Tabelle 23 | Testergebnis von Testfall 3 |
| Tabelle 24 | Testergebnis von Testfall 4 |
| Tabelle 25 | Testergebnis von Testfall 5 |
| Tabelle 26 | Testergebnis von Testfall 6 |
| Tabelle 27 | Quellenverzeichnis |
| Tabelle 28 | Abbildungsverzeichnis |
| Tabelle 29 | Tabellenverzeichnis |
| Tabelle 30 | Glossar |

Tabelle 30: Tabellenverzeichnis

## Glossar

|  |  |
| --- | --- |
| Begriff | Erklärung |
| Glossar | Im Glossar werden Begriffe erklärt, welche für den Leser vielleicht nicht ganz verständlich sein können. |
| CRUD | Create, Read, Update und Delete.  Wird bei Web.APIs verwendet. |
| ERD | Entity Relationship Diagram.  Das ist ein Diagramm, bei dem der Aufbau der Datenbank ersichtlich sein sollte. |
| BLOB | Binary Large Object.  In einem BLOB können Binäre Daten abgespeichert werden. Dazu können Bilder, Audio oder andere Multimedia Dateien sein. |
| MVC | Model-View-Controller.  Das ist eine Architektur-Pattern welches man meistens benötigt, wenn man ein User Interface implementiert. |
| Routing | Das ist die Implementation welche man benötigt, wenn man von einer Page auf die andere weitergeleitet wird. Routing kann genutzt werden bei einem Klick des Users, oder wenn etwas neu geladen wird. |
| Razor | Sprache mit der die Views implementiert werden. Razor ist sozusagen eine Kombination von HTML und C#. |

Tabelle 31: Glossar

## Programmcode

Hier sieht man den Programmcode welcher während den fünf Tagen realisiert wurde. In den Kapiteln wurden diese nach Projekt getrennt.

### Secondhand.BusinessLogic

**Items/ItemService.cs**

using Secondhand.Domain.Model;

using System.Collections.Generic;

namespace Secondhand.BusinessLogic.Items

{

class IItemService

{

IEnumerable<Item> GetItems();

}

}

**Items/IItemService.cs**

using System.Collections.Generic;

using System.Linq;

using Secondhand.Domain.Model;

using Secondhand.Domain.Services;

namespace Secondhand.BusinessLogic.Items

{

class ItemService : IItemService

{

private readonly IItemRepository \_itemRepository;

public ItemService(IItemRepository itemRepository)

{

\_itemRepository = itemRepository;

}

public ItemService(IItemRepository, repository)

{

}

public IEnumerable<Item> GetItems()

{

return \_itemRepository.GetAll().OrderBy(item => item.Name);

}

}

}

### Secondhand.Domain

**SecondhandModel.edmx.sql**

SET QUOTED\_IDENTIFIER OFF;

GO

USE [SecondhandDatabase];

GO

IF SCHEMA\_ID(N'dbo') IS NULL EXECUTE(N'CREATE SCHEMA [dbo]');

GO

IF OBJECT\_ID(N'[dbo].[FK\_UserItem]', 'F') IS NOT NULL

ALTER TABLE [dbo].[Items] DROP CONSTRAINT [FK\_UserItem];

GO

IF OBJECT\_ID(N'[dbo].[Users]', 'U') IS NOT NULL

DROP TABLE [dbo].[Users];

GO

IF OBJECT\_ID(N'[dbo].[Items]', 'U') IS NOT NULL

DROP TABLE [dbo].[Items];

GO

CREATE TABLE [dbo].[Users] (

[Id] int IDENTITY(1,1) NOT NULL,

[UserName] nvarchar(max) NOT NULL,

[FirstName] nvarchar(max) NOT NULL,

[LastName] nvarchar(max) NOT NULL,

[Street] nvarchar(max) NOT NULL,

[City] nvarchar(max) NOT NULL,

[Zip] int NOT NULL,

[Country] nvarchar(max) NOT NULL

);

GO

CREATE TABLE [dbo].[Items] (

[Id] int IDENTITY(1,1) NOT NULL,

[ItemName] nvarchar(max) NOT NULL,

[Description] nvarchar(max) NOT NULL,

[Image] varbinary(max) NOT NULL,

[Year] int NOT NULL,

[Price] int NOT NULL,

[UserId] int NOT NULL

);

GO

ALTER TABLE [dbo].[Users]

ADD CONSTRAINT [PK\_Users]

PRIMARY KEY CLUSTERED ([Id] ASC);

GO

ALTER TABLE [dbo].[Items]

ADD CONSTRAINT [PK\_Items]

PRIMARY KEY CLUSTERED ([Id] ASC);

GO

ALTER TABLE [dbo].[Items]

ADD CONSTRAINT [FK\_UserItem]

FOREIGN KEY ([UserId])

REFERENCES [dbo].[Users]

([Id])

ON DELETE NO ACTION ON UPDATE NO ACTION;

GO

CREATE INDEX [IX\_FK\_UserItem]

ON [dbo].[Items]

([UserId]);

GO

**ISecondhandContext.cs**

using System.Data.Entity;

namespace Secondhand.Domain.Model

{

public interface IWiwaContext

{

DbSet<Item> Items { get; set; }

DbSet<User> Users { get; set; }

int SaveChanges();

}

}

**IItemRepository.cs**

using System.Collections.Generic;

using System.Linq;

using Secondhand.Domain.Model;

namespace Secondhand.Domain.Services

{

public interface IItemRepository

{

IQueryable<Item> GetAll();

}

}

### Secondhand.Tests

Aufgrund mangelnder Zeit, sind keine Unit Tests entstanden.

### Secondhand.Web

#### Views

**\_ViewStart.cshtml**

@{

Layout = "~/Views/Shared/\_Layout.cshtml";

}

**\_Layout.cshtml**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>@ViewBag.Title - My ASP.NET Application</title>

@Styles.Render("~/Content/css")

@Scripts.Render("~/bundles/modernizr")

</head>

<body>

<div class="navbar navbar-inverse navbar-fixed-top">

<div class="container">

<div class="navbar-header">

<button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">

<span class="icon-bar"></span>

<span class="icon-bar"></span>

<span class="icon-bar"></span>

</button>

</div>

<div class="navbar-collapse collapse">

<ul class="nav navbar-nav">

<li>@Html.ActionLink("Home", "Index", "Home")</li>

<li>@Html.ActionLink("Items", "Index", "Item")</li>

<li>@Html.ActionLink("Create User", "Index", "User")</li>

</ul>

<p class="nav navbar-text navbar-right">Hello, @User.Identity.Name!</p>

</div>

</div>

</div>

<div class="jumbotron">

<h1>Secondhand Trade</h1>

</div>

<div class="container body-content">

@RenderBody()

<hr />

<footer class="footer">

<div class="container">

<span class="text-muted">&copy; Cassandra Corrodi, Siemens Schweiz AG</span>

</div>

</footer>

</div>

@Scripts.Render("~/bundles/jquery")

@Scripts.Render("~/bundles/bootstrap")

@RenderSection("scripts", required: false)

</body>

</html>

**Error.cshtml**

@model System.Web.Mvc.HandleErrorInfo

@{

ViewBag.Title = "Error";

}

<h1 class="text-danger">Error.</h1>

<h2 class="text-danger">An error occurred while processing your request.</h2>

**Home/Index.cshtml**

@{

ViewBag.Title = "Home Page";

}

<div class="row">

<div class="col-md-8">

<h2>Welcome to the Secondhand Trade Platform</h2>

</div>

<div class="col-md-8">

You can sell your old items here, and buy items from other people.s

</div>

</div>

**Item/Index.cshtml**

@using Secondhand.Domain.Model

@{

ViewBag.Title = "Items";

}

<h2>@ViewBag.Title.</h2>

<h3>@ViewBag.Message</h3>

<a>

<div class="col-md-4">

@Html.ActionLink("Details", "Detail", "Item")

<p>

<img src="https://wanderlustandsmoke.files.wordpress.com/2017/08/filipino-tabo-hinoki-plastic-ladle-1.jpg?w=286&h=212" />

</p>

<h2>Tabo 1</h2>

<p>10.00$</p>

</div>

</a>

**Item/Detail.cshtml**

<h1>Details</h1>

@foreach (Item item in Items)

{

<a>

<div class="col-md-12">

<h2>Tabo 1</h2>

<p>

<img @item.image />

</p>

<p>@item.description</p>

<p>@item.year</p>

<p>@item.price</p>

<p>@item.userId</p>

<button>

Buy Item

</button>

</div>

</a>

}

**User/Index.cshtml**

<h2>CreateUser</h2>

<form>

<div class="form-group">

<label for="inputEmail">User Name</label>

<input type="text" class="form-control" id="inputEmail" placeholder="User Name">

</div>

<div class="form-group">

<label for="inputPassword">First Name</label>

<input type="text" class="form-control" id="inputPassword" placeholder="First Name">

</div>

<div class="form-group">

<label for="inputPassword">Last Name</label>

<input type="text" class="form-control" id="inputPassword" placeholder="Last Name">

</div>

<div class="form-group">

<label for="inputPassword">Password</label>

<input type="text" class="form-control" id="inputPassword" placeholder="Street">

</div>

<div class="form-group">

<label for="inputPassword">Street</label>

<input type="text" class="form-control" id="inputPassword" placeholder="City">

</div>

<div class="form-group">

<label for="inputPassword">ZIP Code</label>

<input type="text" class="form-control" id="inputPassword" placeholder="ZIP Code">

</div>

<div class="form-group">

<label for="inputPassword">Country</label>

<input type="text" class="form-control" id="inputPassword" placeholder="Country">

</div>

<button type="submit" class="btn btn-primary">Create User</button>

</form>

#### Controllers

**HomeController.cs**

using System.Web.Mvc;

namespace SecondhandTrade.Controllers

{

public class HomeController : Controller

{

public ActionResult Index()

{

return View();

}

public ActionResult Contact()

{

return View();

}

}

}

**ItemController.cs**

using System.Web.Mvc;

using Secondhand.BusinessLogic.Teams;

using Secondhand.Domain.Model;

using Secondhand.Web.Models;

namespace SecondhandTrade.Controllers

{

public class ItemController : Controller

{

public ActionResult Index()

{

return View();

}

public ActionResult Detail()

{

return View();

}

}

}

**UserController.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace SecondhandTrade.Controllers

{

public class UserController : Controller

{

public ActionResult Index()

{

return View();

}

}

}