



Abschlussprüfung Winter 2021/2022

Dokumentation zur betrieblichen Projektarbeit

Proof of Concept einer Wäscherei-Applikation



Berufsbezeichnung:

Fachinformatiker Systemintegration

Antragsteller:

Erik Alexis Delgado Galvez

Ausbildungsbetrieb:

INTERAP Softwareentwicklung

Kaiser-Joseph-Straße 267

79098 Freiburg im Breisgau

Termin Abschlussprüfung:

Winter 2021/ 2022

Umschulungsträger:

GFN GmbH

Unterwerkstraße 5

79115 Freiburg im Breisgau

Abgabetermin:

12.01.2022

INTERAP GmbH & Co. KG
Kaiser-Joseph-Straße 267
79098 Freiburg

Hiermit bestätige ich, Heiko Falk, in der Funktion als Ausbilder oder fachlich Verantwortlicher von Herrn Erik Delgado, Folgendes:

Der oben genannte Auszubildende hat die vorliegende Projektdokumentation eigenständig angefertigt. Die Durchführung des Projektes wurde von mir in regelmäßigen Abständen kontrolliert.

Freiburg, 11.01.2022
Ort, Datum


Unterschrift

Dieses Schreiben muss der Projektdokumentation beigelegt werden.

Inhaltsverzeichnis

1 Einleitung	1
1.1 Projektbeschreibung.....	1
1.2 Projektziel	2
1.3 Projektumfeld	2
1.4 Projektbegründung.....	2
1.5 Projektschnittstellen.....	2
1.6 Projektabgrenzung	3
2 Projektplanung	3
2.1 Projektphasen.....	3
2.2 Ressourcenplanung	4
2.3 Entwicklungsprozess.....	5
3 Analysephase	5
3.1 Ist-Analyse	5
3.2 Wirtschaftlichkeitsanalyse.....	6
3.2.1 „Make-or-Buy“-Entscheidung.....	6
3.2.2 Projektkostenkalkulation.....	6
3.3 Nicht-monetäre Vorteile	7
3.4 Aktivitätsdiagramm	8
3.5 Anforderungen des Auftragsgebers	8
3.5.1 Anforderungen an die Wäschebeutel-Ansicht	9
3.5.2 Anforderungen an die Artikelauswahl-Ansicht	9
3.5.3 Anforderungen an die Warenkorb-Ansicht	9
3.5.4 Anforderungen an die Auftrag erfolgt-Ansicht	9
3.5.5 Technologische Anforderung	9

3.6 Technologien zur Projektumsetzung	9
3.6.1 Frameworks	9
3.6.2 Datenbankmanagementsystem	10
3.6.3 Computersprachen	10
3.6.4 Weitere Technologien	11
4 Entwurfsphase	12
4.1 Zielplattform	12
4.2 Architekturdesign	12
4.2.1 Eingesetzte Frameworks	12
4.2.2 Schnittstellen	13
4.3 Entwurf der Benutzeroberfläche	13
4.4 Datenmodell	14
4.4.1 ER-Modell	14
4.4.2 Relationales Datenmodell	14
4.5 Umsetzungsleitlinien	15
4.5.1 Umsetzungsleitlinien für die Wäschebeutel-Ansicht	15
4.5.2 Umsetzungsleitlinien für die Artikelauswahl-Ansicht	16
4.5.3 Umsetzungsleitlinien für die Warenkorb-Ansicht	16
4.5.4 Umsetzungsleitlinien für die Auftrag erfolgt-Ansicht	17
4.5.5 Umsetzungsleitlinien für das Navigationsband	17
5 Implementierungsphase	18
5.1 Iterationsplanung	18
5.2 Implementierung der Datenstrukturen	18
5.3 Implementierung der Umsetzungsleitlinien	18
5.3.1 Implementierung der Wäschebeutel-Ansicht	19
5.3.2 Implementierung des Navigationsbandes	20

5.3.3 Implementierung der Artikelauswahl-Ansicht	20
5.3.4 Implementierung der Warenkorb-Ansicht	22
5.3.5 Implementierung der Auftrag erfolgt-Ansicht	23
6.Abnahme durch den Fachbereich	23
7 Fazit.....	23
7.1 Soll-/Ist-Vergleich	23
7.2 Ausblick.....	23
Tabellen	i
Detaillierte Zeitplanung.....	i
Diagramme	ii
Aktivitätsdiagramm	ii
ER-Modell	iii
Relationales Datenmodell	iii
Abbildungen.....	iv
Abb. 5.3.1-1: app.routes.ts (Auszug)	iv
Abb. 5.3.1-2: Adresszeile der Wäschebeutel-Komponenten	iv
Abb. 5.3.1-3: waeschebeutel.component.ts (Auszug)	iv
Abb. 5.3.1-4: waeschebeutel.component.ts (Auszug)	iv
Abb. 5.3.1-5: WaeschbeutelController.cs (Auszug).....	v
Abb. 5.3.1-6: waeschebeutel.component.ts (Auszug)	v
Abb. 5.3.1-7: waeschebeutel.component.ts (Auszug)	vi
Abb. 5.3.1-8: Wäschebeutelansicht bei status „belegt“	vi
Abb. 5.3.1-9: Wäschebeutelansicht bei status „exisitiert nicht“	vii
Abb. 5.3.1-10: Wäschebeutelansicht bei status „frei“	vii
Abb. 5.3.2-1: Ausgabe des Menübandes.....	viii
Abb. 5.3.2-2: header.component.html (Auszug).....	viii

Abb. 5.3.3-1: Ausgabe der Artikelauswahl-Ansicht (Auszug)	viii
Abb. 5.3.3.-2: artikelauswahl.component.html (Auszug).....	viii
Abb. 5.3.3.-3: artikelauswahl.component.ts (Auszug)	ix
Abb. 5.3.3.-3-1: artikelauswahl.component.html (Auszug)	ix
Abb. 5.3.3.-4: ArtikelauswahlController.cs (Auszug).....	ix
Abb. 5.3.3.-5: Ausgabe der Artikelauswahl-Ansicht (Auszug)	x
Abb. 5.3.3.-6: artikelauswahl.component.html (Auszug).....	x
Abb. 5.3.3.-7: artikelauswahl.component.ts (Auszug)	x
Abb. 5.3.3.-8: warenkorb.service.ts (Auszug).....	x
Abb. 5.3.3.-9: warenkorb.component.ts	x
Abb. 5.3.3.-10: warenkorb.component.html	xi
Abb. 5.3.3.-11: Ausgabe der Artikelauswahl-Ansicht (Auszug)	xi
Abb. 5.3.4-1: warenkorb.component.html (Auszug).....	xi
Abb. 5.3.4-2: warenkorb.component.ts (Auszug)	xi
Abb. 5.3.4-3: warenkorb.service.ts (Auszug).....	xi

1 Einleitung

Die folgende Projektdokumentation schildert den Ablauf des IHK-Abschlussprojektes, welches der Autor im Rahmen seiner Ausbildung zum Fachinformatiker Fachrichtung Anwendungsentwicklung durchgeführt hat. Ausbildungsbetrieb ist INTERAP, ein Betrieb, der individuelle Softwareentwicklung betreibt.

1.1 Projektbeschreibung

Die Projektarbeit ist Teil eines Proof of Concepts, das bei INTERAP durchgeführt wird. In einem Proof of Concept wird die Machbarkeit einer Applikation gezeigt. Bei INTERAP wurde die Machbarkeit einer Web-Applikation für einen Wäschereidienst getestet.

Die Web-Applikation soll dem Anwender den Gang in eine Wäscherei ersparen und ihm trotzdem die Möglichkeit geben deren Dienste in Anspruch zu nehmen.

Es wird von folgendem Szenario ausgegangen:

- Der Anwender soll an einer Wäscheannahmestelle, die bspw. in einem Supermarkt stehen könnte, selbstständig einen Wäschebeutel nehmen, der an der Wäscheannahmekontainer aushängt.
- Der Kunde soll seine Kleidung in den Wäschebeutel tun und mit seinem Handy den am Wäschebeutel befindlichen QR-Code einscannen.
- Mit dem QR-Code soll sich der Kunde mit der Web-Applikation der Wäscherei verbinden.
- Anhand der Web-Applikation soll der Kunde die Wäscherei mit dem Inhalt des Wäschebeutels beauftragen können.
- Anschließend soll der Kunde den Wäschebeutel in den Wäscheannahmecontainer einwerfen.
- Wenn der Auftrag in der Wäscherei erfolgreich bearbeitet wurde, wird der Kunde benachrichtigt. Er kann seine saubere Wäsche dann an der Wäscheannahmestelle abholen.

1.2 Projektziel

Ziel des Projektes ist es vier Ansichten des Proof of Concepts für diese Web-Anwendung umzusetzen.

- **Ansicht 1 – „Wäschebeutel“:** In dieser Ansicht soll geprüft werden, ob der Wäschebeutel, den der Kunde ausgewählt hat, schon in Benutzung ist.
- **Ansicht 2 – „Artikelauswahl“:** In dieser Ansicht soll dem Kunden die Möglichkeit gegeben werden Artikel für die Auftragserfassung auszusuchen.
- **Ansicht 3 – „Warenkorb“:** Die ausgewählten Artikel sollen in einen Warenkorb verschoben werden können, die dem Kunden erlaubt den Auftrag verbindlich aufzugeben.
- **Ansicht 4 – „Auftrag erfolgt“:** In dieser Komponente wird dem Anwender das Feedback gegeben, dass sein Auftrag erfolgreich im System eingegangen ist.

1.3 Projektumfeld

INTERAP ist ein Betrieb, der individuelle Softwareentwicklung betreibt. Zur Zeit beschäftigt INTERAP 10 Mitarbeiter. Die Kernleistungen von INTERAP sind Softwareentwicklung, Softwarekonzeption und Softwaresupport. Der Autor macht sein Praktikum und sein Praktikumsprojekt in der Entwicklerabteilung.

1.4 Projektbegründung

Im Sinne eines Proof of Concepts interessiert sich der Auftraggeber dafür in welcher Form eine Wäscherei-Applikation umgesetzt werden kann. Das aus dieser Machbarkeitsprüfung gewonnene Wissen will der Auftraggeber einsetzen, um Kunden zu rekurrieren.

1.5 Projektschnittstellen

Herr Falk: Herr Falk ist Auftraggeber des Projektes und Geschäftsführer des Praktikumbetriebes des Autors.

- Als Projektauftraggeber hat er die Anforderung an die Applikation des Autors vorgegeben (vgl. [Kapitel 3.5](#)).

- Außerdem hat er sich als Codereviewer für das Projekt des Autors zur Verfügung gestellt (vgl. [Kapitel 2.3](#)).

1.6 Projektabgrenzung

Das Projekt des Autors ist ein Teilprojekt eines Proof of Concepts, mit dem die Machbarkeit einer Wäscherei-Applikation erörtert werden soll. Das Gesamtprojekt ist ein komplett eigenständiges Projekt, ohne Bezug zu anderen Projekten.

2 Projektplanung

2.1 Projektphasen

Für die Umsetzung des Projektes standen 70 Stunden zur Verfügung. Diese Stunden wurden in verschiedene Phasen eingeteilt, die bei der Erstellung der Applikation durchlaufen werden. Die Grobe Zeitplanung der Projektphasen wird in der Tabelle unten dargestellt.

Projektphase	Geplante Zeit
Analysephase	8h
Entwurfsphase	5h
Implementierungsphase	48h
Abnahme durch die Fachabteilung	1h
Erstellen der Dokumentation	8h
Gesamt	70h

Tabelle: Grobe Zeitplanung

Eine detaillierte Projektplanung ist im Anhang zu finden (vgl. [Tabelle: Detaillierte Zeitplanung](#)).

2.2 Ressourcenplanung

Für das Projekt wurden folgende Ressourcen verwendet.

Hardware

- Home-Office mit Desktoprechner

Software

- Angular - clientseitiges JavaScript-Web-Framework
- .Net 5 – Entwicklungsframework
- Entity Framework - objektrelationaler Mapper (ORM)
- Microsoft SQL-Server - relationales Datenbankmanagementsystem (RDBMS)
- YEd – Visualisierungsdiagramm
- MS Word – Textverarbeitungsprogramm
- Visual Studio Code - Quelltext-Editor
- Visual Studio – Entwicklungsumgebung
- Balsamiq - Programm zur Erstellung von Mockups
- Fork - Git-Client
- Azure DevOps – Projektverwaltungssoftware

Computersprachen

- C# - objektorientierte Programmiersprache
- Typescript - objektorientierte Programmiersprache
- SQL – Datenbanksprache
- LINQ - Implementierung der Abfragesprache LINQ

Personal

- Heiko Falk – Auftraggeber und Codereviewer

2.3 Entwicklungsprozess

Der Entwicklungsprozess definiert die Vorgehensweise, nach der die Umsetzung des Projektes erfolgt. Der Autor hat sich für einen agilen Entwicklungsprozess bzw. für eine agile Softwareentwicklung entschieden. Agile Softwareentwicklung wird folgendermaßen beschrieben:

„Konkret geht es bei der agilen Softwareentwicklung darum, kleine Teile funktionsfähiger Software schnell bereitzustellen, um die Zufriedenheit der Kunden zu verbessern.“ ([redhat.com](https://www.redhat.com))

Auf Basis dieser Definition wurde mit dem Projektauftraggeber folgendes vereinbart:

- Sobald eine Ansicht des Projektes erstellt wird, wird diese vom Autor getestet und auf die richtige Funktionalität geprüft.
- Anschließend erfolgt ein Codereview mit Herr Falk.
- Zuletzt äußert sich Herr Falk darüber, ob er mit der Ansicht zufrieden ist oder noch Optimierungsbedarf besteht.

Diese Vorgehensweise hat den Vorteil, dass der Autor während des Projektes mit dem Auftraggeber, Herr Falk, in Verbindung bleibt und sich wichtiges Feedback einholen kann.

3 Analysephase

3.1 Ist-Analyse

Wie im Kapitel Projektbeschreibung schon erwähnt, ist das Projekt des Autors Teil eines Proof of Concepts. Das Projekt ist Teil einer betriebsinternen Machbarkeitsprüfung zur Umsetzung einer Web-Applikation für einen Wäschereidienst.

Das Projekt ist „from the scratch“, also von Anfang an ohne Vorstudien oder Vorgängerprogramme entstanden.

Wie im Kapitel [Projektbegründung](#) schon erwähnt, will der Projektauftraggeber Herr Falk das durch die Machbarkeitsprüfung gewonnene Wissen einsetzen, um Kunden zu rekurrieren.

3.2 Wirtschaftlichkeitsanalyse

3.2.1 „Make-or-Buy“-Entscheidung

Bei der „Make-or-Buy“-Entscheidung steht zur Debatte, ob ein bestimmtes Produkt, in diesem Fall eine Applikation, von einem externen Anbieter gekauft („Buy“) oder im Betrieb selbst erstellt wird („Make“) (vgl. projektmagazin.de).

Bei INTERAP hat man sich für das Selbererstellen aus zwei Gründen entschieden:

- Der erste Grund ist, dass man mit der Applikation Kunden an das Unternehmen binden will. Wenn die Applikation von einem Fremdunternehmen gekauft wird, dann bleibt dieses Fremdunternehmen in der Wertschöpfungskette eingebunden. Das heißt, Wartungen der Applikation werden vom Fremdunternehmen übernommen und nicht von INTERAP. Damit geht ein Service verloren, den INTERAP selber übernehmen will.
- Der zweite Grund ist der Autor selbst. Mit dem Projekt wird dem Autor die Möglichkeit gegeben einen Einblick in die Arbeit eines Full Stack Developers zu bekommen. Der Begriff Full Stack Developer wird im Kapitel „Nicht-monetäre Vorteile“ erläutert.

3.2.2 Projektkostenkalkulation

Die Projektkostenkalkulation erfolgt auf Grundlage von betriebsinternen Gehältern plus Lohngemeinkosten und Verwaltungsgemeinkosten.

Darstellung der Kostensätze:

Auszubildender	28€
Entwickler	50€
Senior Entwickler	75€
Lohngemeinkosten	60%
Verwaltungsgemeinkosten	10%

Tabelle: Kostensätze

- Bei den betriebsinternen Gehältern handelt es sich um Annäherungswerte, da diese dem Datenschutz unterliegen.

Proof of Concept einer Wäscherei-Applikation

- In den Lohngemeinkosten sind Kosten für die Nutzung der Hardware, Software sowie Lizenzkosten enthalten.
- In den Verwaltungsgemeinkosten sind die Kosten, die für die Verwaltung entstehen, einberechnet.

Berechnung der Personalkosten:

Vorgang	Verantwortlicher	Std.	Gesamt
Entwicklung	Auszubildender	70h	1.960 €
Review/Tests	Entwickler	4h	200 €
Abnahme	Senior Entwickler	1h	75 €
Gesamt			2.235 €

Tabelle: Personalkosten

Berechnung der Gesamtkosten des Projekts:

Auf Grundlage der Personalkosten lassen sich die Gesamtkosten des Projekts berechnen.

Personalkosten		2.235,00 €
Lohngemeinkosten	$2.235 \text{ €} \cdot 0,6$	1.341,00 €
Verwaltungsgemeinkosten	$2.235 \text{ €} \cdot 0,1$	223,50 €
Gesamt		3.799,50 €

Tabelle: Gesamtkosten des Projekts

Die Gesamtkosten des Projekts betragen 3.799,50 Euro.

3.3 Nicht-monetäre Vorteile

Neben dem wirtschaftlichen Aspekt ergeben sich ebenfalls nicht-monetäre Vorteile.

Nutzen der Applikation für den Anwender:

Anwender der Applikation ist der Kunde der Wäscherei. Der Anwender hat durch die Applikation den Vorteil, dass für ihn der Prozess der Auftragserfassung erleichtert wird. Außerdem muss der Anwender nicht zusätzlich in eine Wäscherei gehen, um seine Wäsche abzugeben oder abzuholen.

Nutzen der Applikation für den Kunden (die Wäscherei):

Der potenzielle Käufer dieser Applikation ist eine Wäscherei, die einen Online-Wäschereidienst anbieten will. Die Projektarbeit soll dazu dienen dem Kunden einen ersten Einblick darauf zu geben wie die Applikation funktionieren kann.

Nutzen der Applikation für den Ausbildungsbetrieb:

Durch die Projektarbeit hat der Ausbildungsbetrieb einen ersten Workflow einer Applikation für einen Wäschereidienst. Dieser Workflow beinhaltet ein User-Interface für den Anwender und die Datenverarbeitung über eine Datenbank.

Nutzen für den Auszubildenden:

Die Projektarbeit gibt dem Autor die Chance eine Applikation zu entwickeln, die einen Frontend- und Backendbereich besitzt. Damit bekommt der Autor einen Einblick in die Arbeit von Full Stack Developern. Full Stack Developer zeichnen sich durch Kenntnisse im Frontend- und Backendbereich der Erstellung von Applikationen aus.

3.4 Aktivitätsdiagramm

Anhand eines Aktivitätsdiagramms wird der Ablauf der Aktionen innerhalb der Applikation erläutert (vgl. [Aktivitätsdiagramm](#) im Anhang). Im Aktivitätsdiagramm werden die fünf Kernaktionen der Wäscherei-Applikation herausgearbeitet:

- Der Status des Wäschebeutels soll geprüft werden.
- Artikel sollen zur Verfügung gestellt werden, die der Kunde auswählen kann.
- Der Kunde soll die Möglichkeit haben einen Auftrag zu erstellen.
- Der Kunde soll eine Bestätigung kriegen, sobald sein Auftrag im System eingegangen ist.

3.5 Anforderungen des Auftragsgebers

In diesem Kapitel sollen die Anforderungen des Auftraggebers an das Projekt wiedergegeben werden. Die Anforderungen werden nach folgenden Ansichten gegliedert:

- Wäschebeutel-Ansicht
- Artikelauswahl-Ansicht
- Warenkorb-Ansicht
- Auftrag erfolgt-Ansicht

3.5.1 Anforderungen an die Wäschebeutel-Ansicht

Herr Falk erwartet, dass in dieser Ansicht geprüft wird, ob der Wäschebeutel, den der Kunde einscannt, schon in Benutzung ist. Diese Prüfung soll verhindern, dass ein Wäschebeutel zur selben Zeit aus Versehen doppelt beauftragt wird.

3.5.2 Anforderungen an die Artikelauswahl-Ansicht

In dieser Komponente sollen dem Kunden die Wäschestücke/ Artikel zur Verfügung gestellt werden, mit denen er die Wäscherei beauftragen kann.

3.5.3 Anforderungen an die Warenkorb-Ansicht

Die, in der Artikelauswahl-Ansicht ausgewählten, Artikel sollen in einen Warenkorb verschoben werden können, die dem Kunden erlaubt den Auftrag verbindlich aufzugeben.

3.5.4 Anforderungen an die Auftrag erfolgt-Ansicht

Wenn der Auftrag erfolgreich ins System eingegangen ist, dann soll der Kunden das im Browser sehen können.

3.5.5 Technologische Anforderung

Der Projektauftraggeber besteht darauf, dass die Frameworks Angular und .Net 5 zur Umsetzung der Applikation eingesetzt werden.

3.6 Technologien zur Projektumsetzung

In diesem Kapitel werden die Technologien präsentiert, die zur Hilfe genommen werden, um die Web-Applikation zu erzeugen.

3.6.1 Frameworks

Angular

Angular ist ein clientseitiges JavaScript-Web-Framework zur Erstellung von Single-Page-Webanwendungen (vgl. angular.de). Angular eignet sich zur clientseitigen Programmierung. Das bedeutet, dass dieses Framework Programmiersprachen einsetzt, die vom Webbrowser des Benutzers ausgeführt werden (vgl. ibm.com). Anhand von Angular lassen sich clientseitige Anwendungen erstellen. Eine clientseitige Anwendung läuft auf dem Endgerät

eines Netzwerks und kommuniziert mit einem Server (vgl. [wikipedia.de](https://de.wikipedia.org/)). Angular ist ein Frontend-Framework (vgl. [entwickler.de](https://de.wikipedia.org/)).

.Net 5

.Net 5 ist eine Entwicklungsframework der Firma Microsoft, mit dem verschiedene Arten von Apps wie Desktop-Apps oder Mobile-Apps erstellt werden können. .Net 5 unterstützt Programmiersprachen wie C++ oder C#. .Net 5 ist ein Backend-Framework (vgl. [entwickler.de](https://de.wikipedia.org/)).

Entity Framework

Das Entity Framework ist Teil von .NET. Es ist ein objektrelationaler Mapper (ORM). Objektrelationales Mapping bedeutet, dass die Datenbanktabellen des Microsoft SQL Servers als Objekte abgebildet werden können.

3.6.2 Datenbankmanagementsystem

Microsoft SQL-Server

Microsoft SQL-Server ist ein relationales Datenbankmanagementsystem (RDBMS) der Firma Microsoft. Ein Datenbankmanagementsystem (DBMS) ist eine Software, die das Modell eines Datenbanksystems festlegt. Mit einem DBMS kann man eine Datenbank aufbauen, verwalten und nutzen.

3.6.3 Computersprachen

C#

C# ist eine objektorientierte Allzweck-Programmier Sprache (vgl. [wikipedia.de](https://de.wikipedia.org/)). In diesem Projekt wird sie innerhalb des Frameworks .Net 5 eingesetzt.

Typescript

Typescript ist eine, von der Firma Microsoft entwickelten, objektorientierte Programmiersprache, mit der JavaScript typisiert und klassenbasiert programmiert werden kann (vgl. [chip.de](https://de.wikipedia.org/)). In diesem Projekt wird sie innerhalb des Frameworks Angular eingesetzt.

SQL

SQL ist eine Datenbanksprache. Mit ihr können Datenstrukturen in relationalen Datenbanken bearbeitet und abgefragt werden. In diesem Projekt wird sie im relationalen Datenbankmanagementsystem Microsoft SQL-Server eingesetzt (vgl. [wikipedia.de](https://www.wikipedia.de)).

LINQ to SQL

LINQ ist eine Abfragesprache auf Datenstrukturen von der Firma Microsoft. LINQ to SQL ist eine konkrete Implementierung der Abfragesprache LINQ, die die LINQ-Abfrage in die Datenbanksprache SQL übersetzt. LINQ to SQL wird in diesem Projekt innerhalb des Frameworks .NET 5 eingesetzt.

3.6.4 Weitere Technologien

YEd

Das Visualisierungsdiagramm YEd der Firma yWorks wird zum Erstellen und Bearbeiten von Diagrammen benutzt.

MS Word

Das Textverarbeitungsprogramm Microsoft Word wird zum Erstellen der Projektdokumentation verwendet.

Visual Studio Code

Visual Studio Code ist ein Quelltext-Editor der Firma Microsoft. Er wird in der Projektarbeit eingesetzt, um den Frontend-Bereich des Projektes zu erstellen.

Visual Studio

Visual Studio ist eine Entwicklungsumgebung der Firma Microsoft. Er wird in der Projektarbeit eingesetzt, um den Frontend-Bereich des Projektes zu erstellen. Eine Entwicklungsumgebung enthält Texteditor, Compiler (Interpreter), Linker, Debugger und Quelltextformatierungsfunktionen. Visual Studio wird im Backend-Bereich des Projektes eingesetzt.

4 Entwurfsphase

4.1 Zielplattform

Das Projekt soll als eigenständige Desktopanwendung mit eigener Datenbank umgesetzt werden. Microsoft SQL-Server ist das Datenbanksystem, das eingesetzt wird.

Die Computersprachen, die zum Einsatz kommen sind Typescript, C#, SQL und LINQ to SQL. Die Sprachen wurden in Kapitel Computersprachen erläutert.

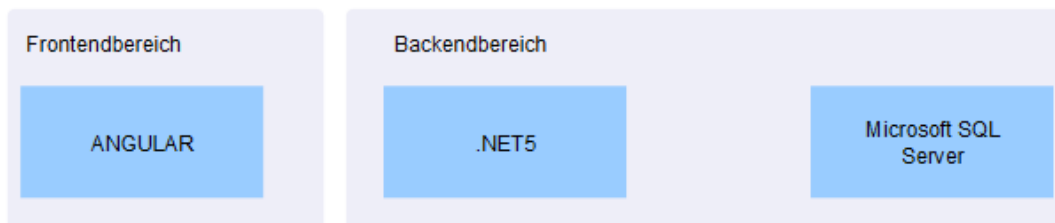
Außerdem wird mit dem JavaScript-Web-Framework Angular und dem Entwicklungsframework .Net 5 gearbeitet.

4.2 Architekturdesign

In diesem Kapitel wird die Architektur des Wäscherei-Applikation vorgestellt.

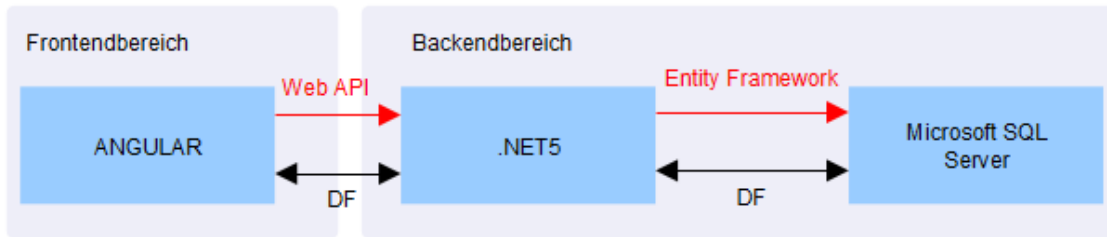
4.2.1 Eingesetzte Frameworks

Wie schon in [Kapitel 4.1](#) erwähnt wurde, ist das Projekt als eigenständige Desktopanwendung geplant. Zur Veranschaulichung werden Diagramme eingesetzt.



- Frontend ist das, was der Benutzer einer Software oder einer Webseite sieht, berührt und erlebt. In diesem Bereich wird das Framework Angular eingesetzt.
- Der Backendbereich ist der Teil einer Applikation, den der Anwender nicht sieht. Hier werden das Framework .NET 5 und das Datenbanksystem Microsoft SQL Server eingesetzt.

4.2.2 Schnittstellen



Die Web API ermöglicht einen Datenfluss (DF) zwischen Frontend- und Backendbereich. Eine API (Application Programming Interface) ist eine Programmierschnittstelle, die der Kommunikation dient.

Das Entity Framework regelt die Kommunikation bzw. den Zugriff auf die Datenbank mittels objektrelationalen Mappings. Das Entity Framework ist Teil des Frameworks .NET und ist ein objektrelationaler Mapper (ORM). Objektrelationales Mapping bedeutet, dass die Datenbanktabellen des Microsoft SQL Servers als Objekte abgebildet werden können.

4.3 Entwurf der Benutzeroberfläche

Die Gestaltung der Benutzeroberfläche soll dazu dienen die Wäscherei-Applikation benutzerfreundlich zu gestalten. Für den Benutzer ist das Sehen von folgenden Komponenten möglich:

- Wäschebeutel-Ansicht
- Artikelauswahl-Ansicht
- Warenkorb-Ansicht
- Auftrag erfolgt-Ansicht
- Navigationsleiste

Für die **Wäschebeutel-Ansicht** gibt es drei optionale Ausgaben, die vom Status des Wäschebeutels abhängen. Optional erscheint ein Button, der den Zugang zur Artikelauswahl-Ansicht erlaubt.

In der **Artikelauswahl-Ansicht** sind die Artikel tabellarisch aufgelistet. In jeder Tabellenzeile gibt es ein Hinzufügen-Button.

In der **Warenkorb-Ansicht** werden die ausgewählten Artikel ebenfalls tabellarisch aufgelistet. In jeder Tabellenzeile gibt es ein „Entfernen“-Button. Zusätzlich gibt es ein „Warenkorb leeren“-Button und ein „Auftrag absenden-Button“.

Die **Auftrag erfolgt-Ansicht** gibt dem Kunden aus, dass sein Auftrag ins System eingegangen ist.

In der **Navigationsleiste** befinden sich folgende Buttons:

- Button 1 verlinkt zur Wäschebeutel-Ansicht.
- Button 2 verlinkt zur Artikelauswahl-Ansicht.
- Button 3 verlinkt zur Warenkorban sicht.

4.4 Datenmodell

In diesem Kapitel wird die Strukturierung der Datenbank dargelegt. Die Datenbank strukturiert sich aus folgenden Entitätstypen:

- Waeschebeutel
- Auftrag
- Auftragsposition
- Status
- Artikel

4.4.1 ER-Modell

Mit Hilfe des Entity-Relationship-Modells (ER-Modells) wird der logische Aufbau der Datenbank dargestellt (vgl. Anhang: [Er-Modell](#)). Die Rechtecke repräsentieren die Entitäten und in den Ovalen stehen die Attribute der jeweiligen Entitäten.

4.4.2 Relationales Datenmodell

Das Relationale Datenmodell stellt eine mathematische Beschreibung einer Tabelle und ihre Beziehung zu anderen möglichen Tabellen dar (vgl. [datenbanken-verstehen.de](https://www.datenbanken-verstehen.de)). Das Datenmodell ist im Anhang zu finden (vgl. [Relationales Datenmodell](#)).

Jede Entität hat ein Primary Key (PK). Die Entität Auftrag und Auftragsposition enthalten darüber hinaus Foreign Keys (FK). Anhand der Kardinalitäten (den Zahlen und Buchstaben auf den Verbindungslinien) lässt sich die Beziehung zwischen den Entitäten ableiten:

- Einem Auftrag kann immer nur ein Wäschebeutel zugewiesen werden (1:1-Beziehung).
- Eine Auftragsposition kann nur einen Auftrag haben, aber in Auftrag kann mehrere Auftragspositionen haben (1:n-Beziehung).
- Eine Auftragsposition kann nur ein Artikel beherbergen, während ein Artikel in mehreren Auftragspositionen auftauchen kann (1:n-Beziehung).
- Ein Auftrag kann nur einen bestimmten Status haben, während ein Status mehreren Aufträgen zugewiesen werden kann (1:n-Beziehung).

4.5 Umsetzungsleitlinien

In diesem Kapitel werden die Umsetzungsleitlinien für das Projekt festgelegt. Die Umsetzungsleitlinien entsprechen den Anforderungen des Auftraggebers an das Projekt (vgl. [Kapitel 3.5](#)).

4.5.1 Umsetzungsleitlinien für die Wäschebeutel-Ansicht

Die Anforderung an diese Ansicht lässt sich folgendermaßen zusammenfassen: Der Wäschebeutel soll danach geprüft werden, ob er in Benutzung oder frei ist (vgl. [Kapitel 3.5.1](#)). Die Umsetzungsleitlinien (UL) sehen folgendermaßen aus:

- **UL-1 – Unique-Key-Generierung und Verknüpfung:** Jeder Wäschebeutel bekommt einen Unique-Key als eindeutiges Identifikationsmerkmal zugeordnet.
- **UL-2 – Unique-Key als Teil der URL:** Jeder Wäschebeutel bekommt eine eigene URL, in der der Unique-Key vorkommt.
- **UL-3 – Unique-Key aus URL auslesen:** Der Unique-Key soll aus der URL ausgelesen werden können.
- **UL-4 – Status-Prüfung:** Mit dem ausgelesenen Unique-Key findet eine Datenbankabfrage statt, die den Status des Wäschebeutels prüft.
- **UL-5 – Ausgabe von Status abhängig:**

- Ist der Status des Wäschebeutels auf „belegt“, dann findet eine Ausgabe im Browser statt, die den Kunden auffordert einen anderen Wäschebeutel einzuscannen.
- Ist der Status des Wäschebeutels auf „frei“, dann findet eine Ausgabe im Browser statt, die den Kunden via Button eine Weiterleitung zur Artikelauswahl-Ansicht erlaubt.
- Ist der Status des Wäschebeutels auf „existiert nicht“, dann findet eine Ausgabe im Browser statt, die den Kunden darauf hinweist einen anderen Wäschebeutel zu nehmen.

4.5.2 Umsetzungsleitlinien für die Artikelauswahl-Ansicht

Die Anforderung an diese Ansicht lässt sich folgendermaßen zusammenfassen: Es sollen Artikel zur Verfügung gestellt werden, mit denen der Kunde die Wäscherei beauftragen kann (vgl. [Kapitel 3.5.2](#)). Die Umsetzungsleitlinien sehen folgendermaßen aus:

- **UL-1 – Datenbankabfrage:** Anhand einer Datenbankabfrage werden aus der Tabelle Artikel die Artikelnamen, Preise und Oberkategorien aufgerufen und in tabellarischer Form ausgegeben.
- **UL-2 – Warenkorbservice:** Die Artikel können anhand eines Buttons dem Warenkorb hinzugefügt werden.

4.5.3 Umsetzungsleitlinien für die Warenkorb-Ansicht

Die Anforderung an diese Ansicht lässt sich folgendermaßen zusammenfassen: Im Warenkorb soll der Kunde die Wäscherei verbindlich beauftragen (vgl. [Kapitel 3.5.3](#)). Die Umsetzungsleitlinien sehen folgendermaßen aus:

- **UL-1 – Tabellarische Ausgabe:** Die Artikel im Warenkorb werden in tabellarischer Form dargestellt. Es wird der Artikelname, Preis und Anzahl der Artikel ausgegeben.
- **UL-2 – Artikel individuell entfernen:** Jeder Artikel soll durch einen Button individuell entfernt werden können.
- **UL-3 – Alle Artikel entfernen:** Ein Button ermöglicht es den gesamten Warenkorb zu leeren.

- **UL-4 – Auftrag erstellen:** Ein Button ermöglicht es die ausgewählten Artikel als Auftrag abzusenden. Durch ein Datenbankaufruf wird der Auftrag in die Tabellen „Auftrag“ und „Auftragsposition“ eingetragen.

4.5.4 Umsetzungsleitlinien für die Auftrag erfolgt-Ansicht

Die Anforderung an diese Ansicht lässt sich folgendermaßen zusammenfassen: Wenn der Auftrag erfolgreich ins System eingegangen ist, dann soll der Kunde darüber ein Feedback bekommen (vgl. [Kapitel 3.5.4](#)). Die Umsetzungsleitlinien sehen folgendermaßen aus:

- **UL-1 – Auftrag erfolgt-Feedback:** Sobald der Auftrag aus der Ansicht Warenkorb in die Tabellen „Auftrag“ und „Auftragsposition“ eingetragen ist, findet im Browser eine Ausgabe statt, die den Kunden darüber informiert, dass sein Auftrag in das System eingegangen ist.

4.5.5 Umsetzungsleitlinien für das Navigationsband

- **UL-1 – Header erstellen:** Das Navigationsband ist in der Applikation allgegenwärtig.
- **UL-2 – Verlinkungen erstellen:** Es besteht aus drei Buttons:
 - Button 1 verlinkt zur Wäschebeutel-Ansicht.
 - Button 2 verlinkt zur Artikelauswahl-Ansicht.
 - Button 3 verlinkt zur Warenkorbansicht.

5 Implementierungsphase

5.1 Iterationsplanung

Bevor die auf die Implementierung der Applikation eingegangen wird, soll noch erläutert in welchen Iterationen die Applikation umgesetzt wird.

- Iteration 1: Erstellung der Datenbank.
- Iteration 2: Implementierung der Wäscherei-Ansicht.
- Iteration 3: Implementierung des Navigationsbandes.
- Iteration 4: Implementierung der Artikelauswahl-Ansicht.
- Iteration 5: Implementierung der Warenkorb-Ansicht.
- Iteration 6: Implementierung der Auftrag erfolgt-Ansicht.

Nach jeder abgeschlossenen Iteration findet eine Prüfung, im Sinne des agilen Entwicklungsprozesses, statt wie es in [Kapitel 2.3](#) beschrieben wurde.

5.2 Implementierung der Datenstrukturen

Die Datenbank wurde auf Basis der Datenstrukturen in [Kapitel 4.4](#) implementiert. Zur Implementierung der Datenbank wurde das relationale Datenbankmanagementsystem (RDBMS) Microsoft SQL-Server genutzt (vgl. [Kapitel 3.6.2](#)).

- Im Object Explorer wurde eine Datenbank mit dem Namen „waeschereiDB“ angelegt.
- Die Tabellen mit ihren Foreign Keys wurden, im Sinne des Relationalen Datenmodells in [Kapitel 4.4.2](#), mit der Datenbanksprache SQL angelegt.
- Für Funktionalitätstests der Applikation waren Testdaten notwendig. Diese wurden mit Hilfe des Tabellen-Editors angelegt.

5.3 Implementierung der Umsetzungsleitlinien

Zur Implementierung der Umsetzungsleitlinien (vgl. [Kapitel 4.5](#)) hat der Autor im Frontendbereich der Applikation mit dem Quelltext-Editor „Visual Studio Code“ gearbeitet. Im Backendbereich kam die Entwicklungsumgebung „Visual Studio“ zum Einsatz.

Es kamen zwei Frameworks zum Einsatz. Im Frontendbereich wurde das clientseitige JavaScript-Web-Framework Angular eingesetzt. Im Backendbereich wurde das .Net 5 Entwicklungsframework eingesetzt.

Zu Beginn der Implementierung wurden mit Angular die Komponente „waeschebeutel“ erstellt. Als nächstes wurde mit Visual Studio ein „ASP.NET Core-Web-API“-Projekt erstellt.

5.3.1 Implementierung der Wäschebeutel-Ansicht

Zur Implementierung dieser Ansicht wurde zunächst mit Angular eine Komponente mit dem Namen „Waeschebeutel“ erstellt. Anschließend wurden die Umsetzungsleitlinien aus [Kapitel 4.5.1](#) umgesetzt.

UL-1 – Unique-Key-Generierung und Verknüpfung: Zunächst wurden 10 Unique-Keys mit einem Online-Unique-Key-Generator erstellt. Mit den Keys wurden 10 Einträge in der Waeschebeutel-Tabelle der Datenbank erstellt. Somit existieren in der Datenbank 10 unterschiedliche Wäschebeutel, die zu Testzwecken aufgerufen werden können.

UL-2 – Unique-Key als Teil der URL: Die Anforderung dieser Umsetzungsleitlinie ist, dass jeder Wäschebeutel, bzw. sein Unique-Key, über die URL der Wäschebeutel-Ansicht abrufbar ist. Implementiert wird diese Funktionalität, indem in der app.routes.ts dem path der Wäschebeutel-Komponenten der Parameter „uniqueKey“ angehängt wird (vgl. [Abbildung 5.3.1-1](#)).

Nun kann ein Wert wie bspw. ein Unique-Key als Parameter an die URL der Wäschebeutel-Komponenten angehängt werden wird (vgl. [Abbildung 5.3.1-2](#)).

UL-3 – Unique-Key aus URL auslesen: Der Unique-Key wird mit Hilfe der lokalen Variablen router in der waeschebeutel.component.ts ausgelesen. router ist eine Variable der Klasse ActivatedRoute, mit deren Hilfe die Properties snapshot und params aufgerufen werden können. Der Unique-Key wird in der lokalen Variablen uniqueKey gespeichert und ist somit auslesbar (vgl. [Abbildung 5.3.1-3](#)).

UL-4 – Status-Prüfung: Mit dem ausgelesenen Unique-Key soll nun der Status des Wäschebeutels geprüft werden. Der lokalen Variablen der Klasse http wird eine get-Methode angehängt, die im Backendbereich die Methode GetStatus des WaeschebeutelControllers aufruft. Der ausgelesene Unique-Key

„this.router.snapshot.params.uniqueKey“ wird als Parameter angehängt (vgl. [Abbildung 5.3.1-4](#)).

Die Methode `GetStatus()` im Backendbereich der Applikation gibt den Status als Rückgabewert. Anhand der Variablen `waeschebeutelExistiert` wird in der Datenbanktabelle `Waeschebeutel` geprüft, ob der im Frontendbereich angegebene `UniqueKey` existiert. Wenn der `UniqueKey` nicht existiert dann wird der Status „exisitiert nicht“ widergegeben. Wenn der `UniqueKey` existiert, wird eine `if`-Verzweigung aktiv, die in der Datenbanktabelle `Auftrag` prüft, ob der Status des Wäschebeutels auf „aufgegeben“ gesetzt ist. Ist das der Fall, dann wird der Status „belegt“ widergegeben. Trifft der Fall nicht zu, dann wird der Status „frei“ widergegeben (vgl. [Abbildung 5.3.1-5](#)).

Im Frontendbereich der Wäschebeutel-Ansicht wird der Status durch eine `subscribe`-Methode der Klasse `HttpClient` eingefangen und in der lokalen Variablen `status` gespeichert (vgl. [Abbildung 5.3.1-6](#)).

UL-5 – Ausgabe von Status abhängig: Im Template der Wäschebeutel-Komponente wurden drei `ng-templates` erstellt, die vom Wert der Variablen `status` abhängig sind (vgl. [Abbildung 5.3.1-7](#)).

Durch die Implementierung der Angular-spezifischen `[ngIf]`-Verzweigung unterscheiden sich die Ausgaben der Wäschebeutelansicht nach dem Status-Wert (vgl. [Abbildung 5.3.1-8](#), [5.3.1-9](#), [5.3.1-10](#)).

5.3.2 Implementierung des Navigationsbandes

Zunächst wurde mit Angular eine header-Komponente erstellt. Im Template, also in der HTML-Datei der header-Komponente, wurden drei Buttons implementiert und mit Icons versehen (vgl. [Abbildung 5.3.2-1](#)).

Durch `routerLink` sind die Buttons mit der Wäschebeutel-, Artikelauswahl- und Warenkorb-Komponente verbunden (vgl. [Abbildung 5.3.2-2](#)).

5.3.3 Implementierung der Artikelauswahl-Ansicht

Zur Implementierung der Artikelauswahl-Ansicht wurde zunächst mit Angular eine `artikelauswahl`-Komponente erstellt.

UL-1 – Datenbankabfrage: Wie in [Kapitel 5.2](#) erläutert wurden in die Tabelle Artikel Testdaten eingepflegt, damit eine Datenbankabfrage stattfinden kann. Die Ausgabe der Artikelauswahl-Ansicht ist in [Abbildung 5.3.3-1](#) dargestellt.

Zur Generierung der Ausgabe wurde zunächst eine HTML-Tabelle erstellt. In der Tabelle ist die Variable artikelliste in der Variablen artikel gespeichert (vgl. [Abbildung 5.3.3-2](#)).

In der artikelauswahl.component.ts wird über die Variable http, der Klasse HttpClient, eine get-Methode aufgerufen, die über „this.baseUrl + 'api/Artikelauswahl/GetArtikel/'“ die GetArtikel-Methode des ArtikelauswahlController.cs im Backend aufruft (vgl. [Abbildung 5.3.3-3](#)).

In der GetArtikel-Methode befindet sich eine select-Abfrage der Datenbanktabelle Artikel. Es werden die Attribute Id, Name, Preis, Bildname und Oberkategorie abfragt. Die GetArtikel-Methode hat eine Artikelliste als Return-Wert (vgl. [Abbildung 5.3.3-4](#)).

Dieser Return-Wert wird in der artikelauswahl.component.ts über die subscribe-Methode aufgefangen und in der lokalen Variable artikelliste gespeichert (vgl. [Abbildung 5.3.3-3](#)).

Im Template (artikelauswahl.component.html) wird die Variable artikelliste in der Variablen artikel gespeichert und mit *ngFor werden die Daten, die in artikel gespeichert sind, in den Tabellenzeilen ausgegeben (vgl. [Abbildung 5.3.3-1](#)).

Auf diese Weise gelangen die Artikel aus der Datenbank in die Artikelauswahl-Ansicht.

UL-2 – Warenkorbservice: Die Umsetzungsleitlinie gibt hier vor, dass die Artikel in der Artikelauswahl-Ansicht anhand eines Buttons der Warenkorb-Ansicht hinzugefügt werden können (vgl. [Abbildung 5.3.3-5](#)).

Zunächst wurde dafür mit Angular die Service-Komponente warenkorb.service.ts erstellt.

Im Template wurde ein Button erstellt, der die Methode addToCart() aufruft. Diese Funktion übergibt den ausgesuchten Artikel als Parameter (vgl. [Abbildung 5.3.3-6](#)).

In der artikelauswahl.component.ts wird, durch die Methode addToCart, die Methode addToCart() des warenkorb.service.ts aufgerufen. (vgl. [Abbildung 5.3.3-7](#)).

Die Methode addToCart() der warenkorb.service.ts sorgt dafür, dass die ausgesuchten Artikel in der Variablen warenkorbitems gespeichert werden. Außerdem erhöht sie die

Artikelanzahl. In der Methode wird eine if-else-Verzweigung genutzt. Wenn (if) es in `warenkorbitems` schon einen Artikel mit derselben ID gibt, dann wird nur die Anzahl des Artikels erhöht, ansonsten (else) wird die Anzahl auf 1 eingesetzt. Durch die Methode `push()` wird `warenkorbitems` mit dem ausgesuchten Artikel aktualisiert auf (vgl. [Abbildung 5.3.3-8](#)).

Der hinzugefügte Artikel findet in die in die Warenkorb-Ansicht Eingang, indem in der lokalen Variablen `warenkorbitems` die Variable `warenkorbitems` der `warenkorb.component.ts` gespeichert wird (vgl. [Abbildung 5.3.3-9](#)).

Die Variable `warenkorbitems` wird im Template der Warenkorb-Ansicht in der Variablen `item` gespeichert. Über die Variable `item` werden Artikelname, Artikelpreis und Artikelanzahl ausgegeben (vgl. [Abbildung 5.3.3-10](#)).

Es werden Artikelname, Artikelpreis und die Anzahl der Artikel Warenkorb-Ansicht ausgegeben (vgl. [Abbildung 5.3.3-11](#)).

5.3.4 Implementierung der Warenkorb-Ansicht

Zur Implementierung der Warenkorb-Ansicht wird mit Angular eine `warenkorb`-Komponente erstellt.

UL-1 – Tabellarische Ausgabe: Die tabellarische Ausgabe der Warenkorb-Ansicht wurde schon in [Kapitel 5.3.3](#)/ UL-2: `Warenkorbservice` dargestellt.

UL-2 – Artikel individuell entfernen: Die Umsetzungsleitlinie gibt hier vor, dass jeder Artikel der Warenkorb-Ansicht durch einen Button individuell entfernt werden soll.

Zunächst wurde im Template der Warenkorb-Komponenten ein Button erstellt, der die Methode `removeItem()` aufruft. Als Parameter wird der Artikel übergeben, der vom Warenkorb gelöscht werden soll (vgl. [Abbildung 5.3.4-1](#)).

Die Methode `removeItem()` befindet in der `warenkorb.component.ts`. Die Funktion der Methode besteht darin die Methode `removeWarenkorbItem()` von `warenkorb.service.ts` aufzurufen. Als Parameter wird der Artikel übergeben, der vom Warenkorb gelöscht werden soll (vgl. [Abbildung 5.3.4-2](#)).

Die Methode `removeWarenkorbItem()` verändert die Variable `warenkorbitems`. In dieser Variablen befinden sich die Artikel, die in der Warenkorb-Ansicht sind. Es wird die Methode

`indexOf()` an der Variablen `warenkorbitems` angewendet, um den Index des zu löschenden Artikels herauszufinden. Dann wird die Methode `splice()` angewendet, um den zu löschenden Artikel zu entfernen (vgl. [Abbildung 5.3.4-3](#)).

Die Umsetzungsleitlinie 3 (alle Artikel der Warenkorb-Ansicht durch einen Button entfernen), und die Umsetzungsleitlinie 4 (einen Auftrag durch einen Button erstellen), konnten aus Zeitgründen nicht mehr umgesetzt werden.

5.3.5 Implementierung der Auftrag erfolgt-Ansicht

Die Implementierung der Auftrag erfolgt-Ansicht ist ebenfalls aus Zeitgründen nicht mehr möglich gewesen. In [Kapitel 7.1](#) wird auf das zeitliche Problem Bezug genommen.

6.Abnahme durch den Fachbereich

Da die Wäscherei-Applikation noch nicht fertig erstellt wurde, gab es noch keine Abnahme durch die Entwicklungsabteilung. Es wurde mit dem Auftraggeber vereinbart, dass die Applikation bis zum Ende der Praktikumszeit fertig erstellt wird.

7 Fazit

7.1 Soll-/Ist-Vergleich

Die Wäscherei-Applikation konnte in der geplanten Zeit nicht fertig erstellt werden. Die 47 Stunden für die Implementierung der Ansichten erwiesen sich als zu gering.

7.2 Ausblick

Die Applikation in der vorgegebenen Zeit nicht zu Ende bekommen zu haben bewertet der Autor nicht als negativ. Die Projektarbeit hat dem Autor die Chance eine Applikation zu entwickeln, die einen Frontend- und Backendbereich besitzt. Der Autor hat einen Einblick in unterschiedliche Programmiersprachen und Frameworks bekommen und viel dazugelernt.

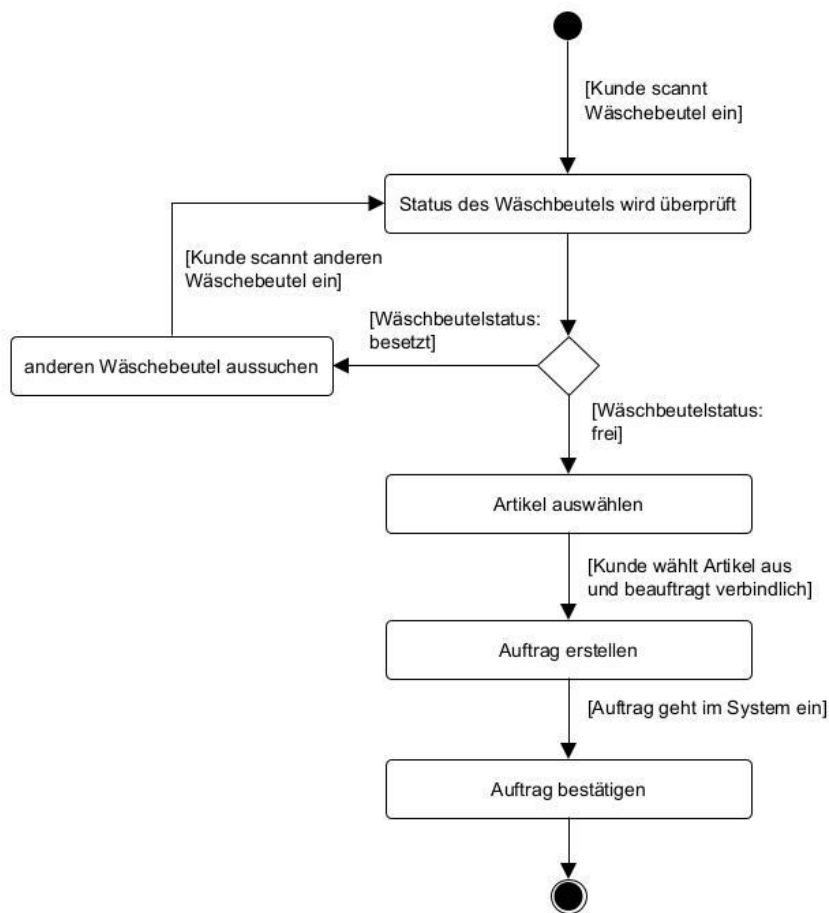
Tabellen

Detaillierte Zeitplanung

Analysephase	8h
1 Aufnahme des Ist-Zustandes	4h
2 Erstellung einer Gewinn- und Umsatzrentabilität	2h
3 Projektanforderungen vom Auftraggeber einholen	2h
Entwurfsphase	5h
1 Benutzeroberfläche entwerfen	1h
2 Datenbank entwerfen (ERM, Datenmodell)	1h
3 Umsetzungsleitlinien erstellen	3h
Implementierungsphase	48h
1 Anlegen der Datenbank	1h
2 Implementierung der Ansichten	47h
2.1 Implementierung der Ansicht Wäschekorb	12h
2.1.1 Erstellung eines Frontend-Dienstes	6h
2.1.2 Erstellung eines Backend-Dienstes	6h
2.2 Implementierung der Ansicht Artikelauswahl	17h
2.2.1 Erstellung eines Frontend-Dienstes für die Ansicht	9h
2.1.2 Erstellung eines Backend-Dienstes für die Ansicht	8h
2.3 Implementierung der Ansicht Warenkorb	15h
2.3.1 Erstellung eines Frontend-Dienstes für die Ansicht	8h
2.3.2 Erstellung eines Backend-Dienstes für die Ansicht	6h
2.4 Implementierung der Ansicht Auftrag erfolgt	2h
2.5 Qualitätstest	1h
Abnahme durch die Fachabteilung	1h
3 Übergabe an den Projektverantwortlichen	1h
Erstellen der Dokumentation	8h
4 Erstellen der Projektdokumentation	8h
Gesamt	70h

Diagramme

Aktivitätsdiagramm



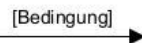
Legende



Startknoten



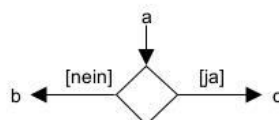
Endknoten



Beschreibt den Einfluss zwischen den Aktionen



Legt das Verhalten fest, das eine Veränderung herbeiführt.

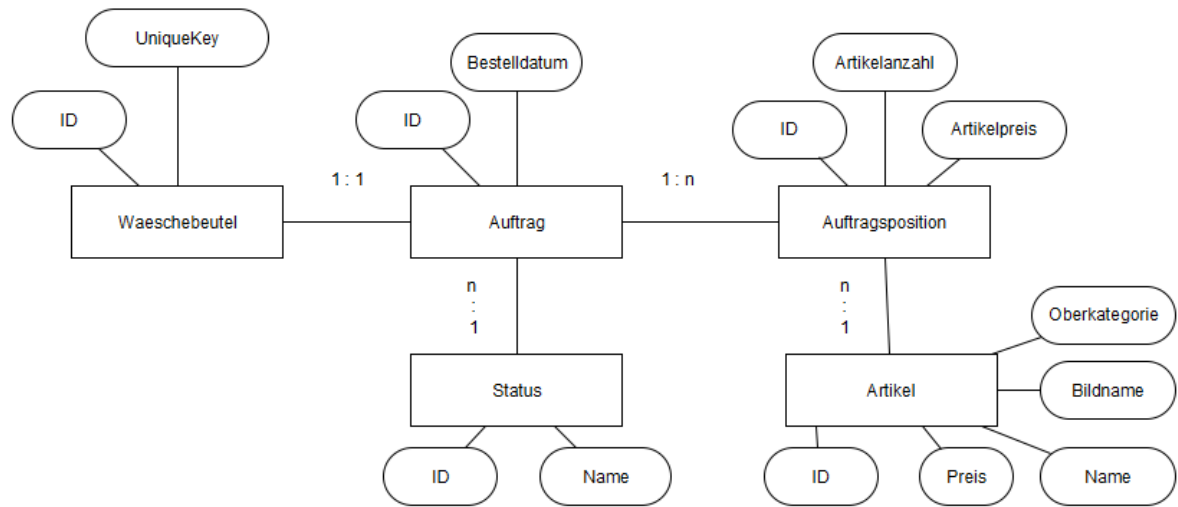


Entscheidung

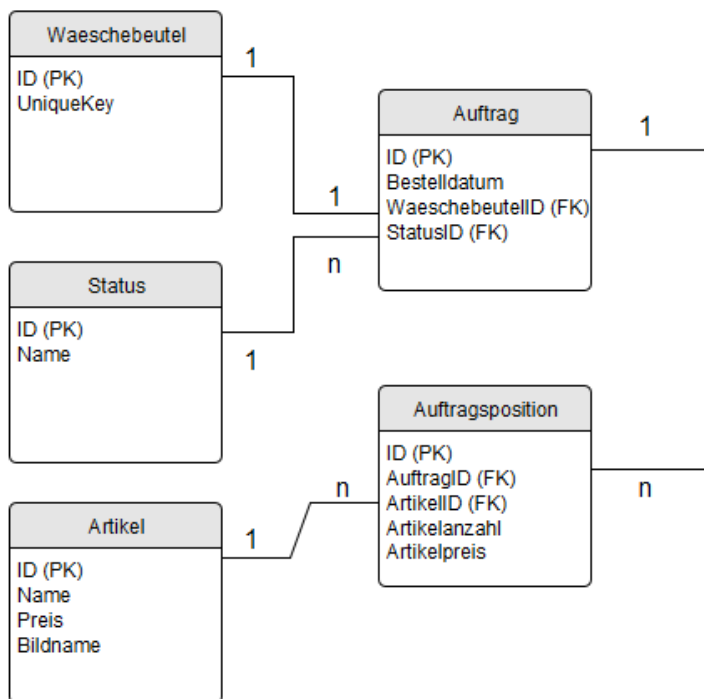
Nach Ende Aktion a wird entweder Aktion b oder Aktion c ausgeführt.

Proof of Concept einer Wäscherei-Applikation

ER-Modell



Relationales Datenmodell

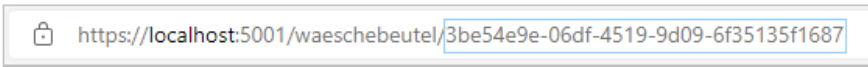


Abbildungen

Abb. 5.3.1-1: app.routes.ts (Auszug)

```
export const meineRouten: Routes = [  
  { path: 'waeschebeutel/:uniqueKey', component: WaeschebeutelComponent },  
  { path: 'artikelauswahl', component: ArtikelauswahlComponent },  
  { path: 'warenkorb', component: WarenkorbComponent },  
];
```

Abb. 5.3.1-2: Adresszeile der Wäschebeutel-Komponenten



https://localhost:5001/waeschebeutel/3be54e9e-06df-4519-9d09-6f35135f1687

Abb. 5.3.1-3: waeschebeutel.component.ts (Auszug)

```
this.uniqueKey = this.router.snapshot.params.uniqueKey;
```

Abb. 5.3.1-4: waeschebeutel.component.ts (Auszug)

```
ngOnInit() {  
  this.http  
    .get(this.baseUrl + 'api/Waeschebeutel/GetStatus/' + this.router.snapshot.params.uniqueKey,  
        { responseType: 'text' }  
    )  
    .subscribe((result) => {this.status = result;});  
}
```

Abb. 5.3.1-5: WaeschbeutelController.cs (Auszug)

```
public IActionResult GetStatus(string uniqueKey)
{
    string status;
    WaeschereiDatabaseContext db = new WaeschereiDatabaseContext();

    bool waeschebeutelExistiert = db.Waeschebeutelcs.Where(w => w.UniqueKey == uniqueKey).Any();

    if(waeschebeutelExistiert)
    {
        bool isbelegt = db.Auftrags
            .Where(a => a.Waeschebeutel.UniqueKey == uniqueKey
                && a.StatusId == (int)Status.StatusEnum.Aufgegeben).Any();

        if (isbelegt)
        {
            status = "belegt";
        }
        else
        {
            status = "frei";
        }
    }
    else
    {
        status = "existiert nicht";
    }

    return Ok(status);
}
```

Abb. 5.3.1-6: waeschebeutel.component.ts (Auszug)

```
ngOnInit() {
    this.http
        .get(this.baseUrl + 'api/Waeschebeutel/GetStatus/' + this.router.snapshot.params.uniqueKey,
            { responseType: 'text' })
        .subscribe((result) => {this.status = result;});
}
```

Abb. 5.3.1-7: waeschebeutel.component.ts (Auszug)

```
<ng-template [ngIf]="status == 'belegt'">
  <div id="status-belegt">
    <p>Der Wäschebeutel</p>
    <p>"{{ uniqueKey }}"</p>
    <p>ist {{ status }}.</p>
    <p>Bitte scannen Sie einen</p>
    <p>anderen Wäschebeutel ein.</p>
  </div>
</ng-template>

<ng-template [ngIf]="status == 'frei'">
  <div id="status-frei">
    <p>Der Wäschebeutel</p>
    <p>"{{ uniqueKey }}"</p>
    <p>ist {{ status }}.</p>
    <button routerLink="/artikelauswahl" id="button-link-artikel">
      <b>Hier geht's zu den Artikeln</b>
    </button>
  </div>
</ng-template>

<ng-template [ngIf]="status == 'existiert nicht'">
  <div id="falsche-beutel-id">
    <p>Dieser Wäschebeutel</p>
    <p>{{ status }}.</p>
    <p>Bitte scannen Sie einen</p>
    <p>anderen Wäschebeutel ein.</p>
  </div>
</ng-template>
```

Abb. 5.3.1-8: Wäschebeutelansicht bei status „belegt“

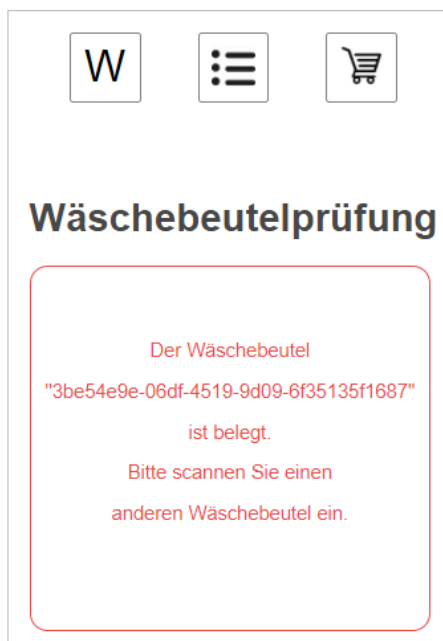


Abb. 5.3.1-9: Wäschebeutelansicht bei status „exisitiert nicht“



Abb. 5.3.1-10: Wäschebeutelansicht bei status „frei“

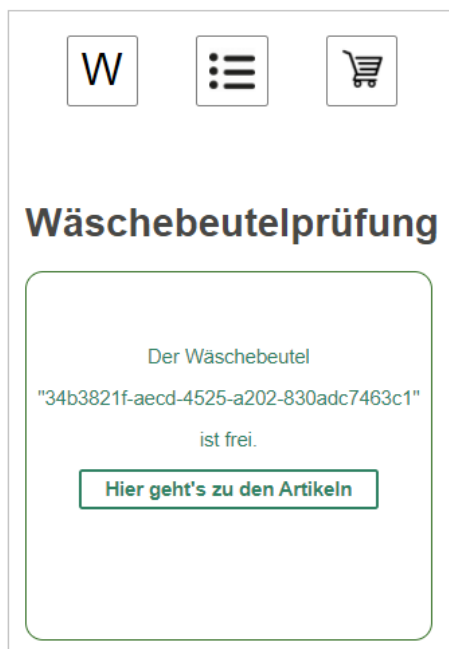


Abb. 5.3.2-1: Ausgabe des Menübandes



Abb. 5.3.2-2: header.component.html (Auszug)

```
<button
  id="button-artikelauswahl"
  class="header-button"
  routerLink="artikelauswahl"
>
  
</button>
```

Abb. 5.3.3-1: Ausgabe der Artikelauswahl-Ansicht (Auszug)

Artikelliste				
Artikelname	Preis	Oberkategorie		
Anzughose	6.9 EUR	Business		<button>HINZUFÜGEN</button>
Anzugsweste	4.9 EUR	Business		<button>HINZUFÜGEN</button>

Abb. 5.3.3-2: artikelauswahl.component.html (Auszug)

```
<table class="table">
  <tr id="table-head">
    <th class="table-head-th">Artikelname</th>
    <th class="table-head-th">Preis</th>
    <th class="table-head-th">Oberkategorie</th>
    <th class="table-head-th"></th>
    <th class="table-head-th"></th>
  </tr>
  <tr *ngFor="let artikel of artikelliste" id="table-corp">
    <td class="table-corp-td">{{ artikel.name }}</td>
    <td class="table-corp-td">{{ artikel.preis }} EUR</td>
    <td class="table-corp-td">{{ artikel.oberkategorie }}</td>
    <td class="table-corp-td">
      
    </td>
  </tr>
```

Abb. 5.3.3.-3: artikelauswahl.component.ts (Auszug)

```
ngOnInit() {  
  this.http  
    .get<Artikel[]>(this.baseUrl + 'api/Artikelauswahl/GetArtikel/')  
    .subscribe((data) => (this.artikelliste = data));  
}
```

Abb. 5.3.3.-3-1: artikelauswahl.component.html (Auszug)

```
<tr *ngFor="let artikel of artikelliste" id="table-corp">  
  <td class="table-corp-td">{{ artikel.name }}</td>  
  <td class="table-corp-td">{{ artikel.preis }} EUR</td>  
  <td class="table-corp-td">{{ artikel.oberkategorie }}</td>  
  <td class="table-corp-td">  
      
  </td>
```

Abb. 5.3.3.-4: ArtikelauswahlController.cs (Auszug)

```
public IList<ArtikelDto> GetArtikel()  
{  
  using (var db = new WaeschereiDatabaseContext())  
  {  
    var artikel =  
      from Artikel a in db.Artikels  
      select new ArtikelDto  
      {  
        ID = a.Id,  
        Name = a.Name,  
        Preis = a.Preis,  
        Bildname = a.Bildname,  
        Oberkategorie = a.Oberkategorie  
      };  
    return artikel.ToList();  
  }  
}
```

Abb. 5.3.3.-5: Ausgabe der Artikelauswahl-Ansicht (Auszug)

Artikelliste				
Artikelname	Preis	Oberkategorie		
Anzughose	6,9 EUR	Business		<button>HINZUFÜGEN</button>

Abb. 5.3.3.-6: artikelauswahl.component.html (Auszug)

```
<button
  (click)="addToCart(artikel)"
  id="button-hinzufuegen"
>
  <b>HINZUFÜGEN</b>
</button>
```

Abb. 5.3.3.-7: artikelauswahl.component.ts (Auszug)

```
addToCart(artikel: Artikel){
  this.warenkorbService.addToCart(artikel);
}
```

Abb. 5.3.3.-8: warenkorb.service.ts (Auszug)

```
addToCart(artikel: Artikel) {
  console.log(this.warenkorbitems);
  var existing = this.warenkorbitems.find((item) => item.artikel.id === artikel.id);

  if (existing){
    existing.anzahl++;
  }

  else{
    var item: Warenkorbitem = { artikel: artikel, anzahl: 1 };
    this.warenkorbitems.push(item);
  }
}
```

Abb. 5.3.3.-9: warenkorb.component.ts

```
ngOnInit(): void {
  this.warenkorbitems = this.warenkorbService.warenkorbitems;
}
```

Abb. 5.3.3.-10: warenkorb.component.html

```
<div id="table-warenkorb">
  <table id="table">
    <tr id="table-head">
      <th class="table-head-th">Artikelname</th>
      <th class="table-head-th">Preis</th>
      <th class="table-head-th">Anzahl</th>
      <th class="table-head-th"></th>
      <th class="table-head-th"></th>
    </tr>
    <tr *ngFor="let item of warenkorbitems" id="table-corp">
      <td class="table-corp-td">{{ item.artikel.name }}</td>
      <td class="table-corp-td">{{ item.artikel.preis }} EUR</td>
      <td class="table-corp-td">{{ item.anzahl }}</td>
```

Abb. 5.3.3.-11: Ausgabe der Artikelauswahl-Ansicht (Auszug)

Warenkorb				
Artikelname	Preis	Anzahl		
Anzughose	8.9 EUR	1		ENTFERNEN

Abb. 5.3.4-1: warenkorb.component.html (Auszug)

```
<button
  (click)="removeItem(item)"
  class="button"
  id="button-entfernen"
>
  <b>ENTFERNEN</b>
</button>
```

Abb. 5.3.4-2: warenkorb.component.ts (Auszug)

```
removeItem(item: Warenkorbitem){
  this.warenkorbService.removeWarenkorbItem(item);
}
```

Abb. 5.3.4-3: warenkorb.service.ts (Auszug)

```
removeWarenkorbItem(product: Warenkorbitem){
  var index = this.warenkorbitems.indexOf(product);
  this.warenkorbitems.splice(index, 1);
}
```