Ein Bild, das Text, Schrift, Screenshot, Logo enthält.

Automatisch generierte Beschreibung

**Fachinformatiker/-in Fachrichtung Anwendungsentwicklung**

**Dokumentation zur betrieblichen Projektarbeit**

Entwicklung einer Web-Anwendung zur Darstellung von vorhandenen Rohstoffmengen und Halbfertigprodukten für einen Zementproduktionsbetrieb

**von**

**Elena Solovyeva**

Prüflingsnummer: 104 03052

erstellt am: 08.12.2023

**Bildungsträger:**

CTC Lohr e.K.

Marlene-Dietrich-Str. 1

89231 Neu-Ulm

**Praktikumsbetrieb:**

S.Rothenbacher GmbH  
Zeppelinstraße 16

89160 Dornstadt

**Inhaltsverzeichnis**

1 Projektüberblick 1

1.1 Projektthema 1

1.2 Projektbeschreibung 1

1.3 Projektumfeld und Rahmenbedingungen 1

2 Zustand 2

2.1 2.1. Ist-Zustand 2

2.2 Soll-Zustand 2

3 Auswahl der Software 4

3.1 Lösungsalternativen 4

3.2 Ausgewählter Sofrware-Stack 5

4 Planung der Durchführung 6

4.1 Festlegung der Lösungsschritte 6

4.1.1 Layout 6

4.1.2 Erstellung des Projekts in Visual Studio 6

4.1.3 Initialisierung vom Git-Repository 7

4.1.4 Entwurf vom Frontend. Konfiguration 7

4.1.5 Backend 8

4.2 Zeit- und Ressourcenplanung 9

4.3 Resümee 10

5 Evaluation des Projektverlaufs 10

6 Ausblick 10

Anhang A: Listing I

A.1 Layout vom MainPage, AdminBereich und LoginPage I

A.2 Erstellung vom Projekt I

A.3. Initial Commit I

A.4. Projektverzeichnis nach dem ersten Commit II

A.5. HTML-Struktur II

A.6. SASS- , CSS und JS-Struktur III

A.7. Exemplarischer Quellcode IV

A.8 Screenshots von SASS-Konfiguration XV

A.9. Ergebnis XVI

Anhang B: XVII

Literaturverzeichnis XVII

Abbildungsverzeichnis XVII

Tabellenverzeichnis XVII

Abkürzungsverzeichnis / Glossar XVII

# Projektüberblick

## Projektthema

Entwicklung einer Web-Anwendung zur Darstellung von vorhandenen Rohstoffmengen und Halbfertigprodukten für einen Zementproduktionsbetrieb

## Projektbeschreibung

Das Projekt dient dazu, aktuelle Informationen über Rohstoffe und Halbfertige Produkte in den Produktionshallen für Büromitarbeiter in real-time darzustellen. Dafür soll eine Webanwendung erstellt werden, die zuerst als Desktop-App für Windows implementiert wird.

## Projektumfeld und Rahmenbedingungen

S. Rothenbacher GmbH in Dornstadt ist eine mittelgroße Firma, die seit 30 Jahren am Markt ist und beschäftigt sich mit der Prozessautomation, sowie mit der Entwicklung von maßgeschneiderten Softwareapplikationen für die Fertigungsindustrie. Über 350 Kunden weltweit schätzen S.Rothenbacher GmbH als einen innovativen und zuverlässigen Partner.

Die Schwerpunkte von S. Rothenbacher GmbH sind:

* die Migrationslösungen für die Modernisierung der Automatisierungstechnik von Simatic S5 auf S7,
* SPS-Programmierung für die Steuerung von Maschinen und Anlagen,
* Installation von Hard- und Software inklusive Schaltschränkebau,
* Lösungen zur Visualisierung und Auswertung von Produktionsdaten: PC-basierte Bedien- und HMI-Systeme zur Visualisierung und Kontrolle von Prozess-Steuerungen für Maschinen und Anlagen in allen Branchen,
* Wägetechnologien für unterschiedliche Branchen, von der Präzisionswaage bis hin zur Straßenfahrzeugwaage
* Softwareentwicklung für individuelle Aufgaben, unter allem für Windows und iOs. Diese Lösungen erhöhen die Effizienz der Bearbeitung von Informations- und Dokumentenflüssen, auslasten die Mitarbeiter, formalisieren Geschäftsprozesse und reduzieren Fehler.

Das Projekt wurde zur Softwareentwicklung-Abteilung zugeordnet, die vor allem die Anwendungen auf C#, SQL, .NET, ASP.Net und , ASP.Net Core entwickelt. Der Projektbetreuer ist der Geschäftsführer Andreas Rothenbacher.

# Zustand

## 2.1. Ist-Zustand

Einer ihrer Kunden ist ein Zementproduktionsbetrieb mit drei Produktionslinien. Jede Produktionslinie verfügt über Produktions-Sylos für Rohstoffe und Halbfertigprodukten

Momentan sind die Informationen über vorhandenen Mengen von Rohstoffen und Halbfertigprodukten in Produktions-Sylos für Büromitarbeiter schwer erreichbar. Diese Daten können nur von einem kleinen Monitor direkt im Produktionsraum abgelesen werden, deswegen müssen Büromitarbeiter jeden Tag die Daten telefonisch ermitteln und die Ergebnisse selbst notieren.

Die Daten über die Menge an Rohstoffen und Halbprodukten in den Produktions-Sylos werden mithilfe einer vorhandenen SPS-Anwendung in einer Microsoft Server SQL-Datenbank übertragen.

Daher existiert ein Bedarf an einer Webanwendung, die aktuelle Stände von Rohstoffen und Halbfertigprodukten in Produktions-Sylos anzeigt.

## Soll-Zustand

Es soll eine Webanwendung namens *MaterialStatus* entwickelt werden, die die aktuelle Daten über die Menge an Rohstoffen und Halbprodukten in den Produktions-Sylos aus einer vorhandenen Datenbank abliest und eine grafische Oberfläche für die Benutze zur Verfügung stellt.

Im Rahmen vom Project soll Test-Version entwickelt werden, die über Windows 10 beziehungsweise Windows 11 auf stationären PCs Lauft.

Aus Sicherheitsgründen soll die Start-Page für Login vorbereitet werden. Es wurden zwei Benutzergruppen definiert: Users, die nach dem Einloggen die Daten vom MainPage ablesen können, und Admins, die außer Users-Rechte auch den Zugang zum AdminArea haben.

Damit ist die folgende Struktur von der *MaterialStatus-*Anwendung definiert:

LoginPage

Adminbereich

MainPage

Abbildung 1. Oberflächernschema für Administrator:

MainPage

LoginPage

Abbildung 2. Oberflächernschema für Administrator:

Für die bessere Usabitity ist es möglich, Daten zu filtern.

Filter und Navigation befinden sich im oberen Bereich (header).

Im unteren Bereich (footer) befinden sich die Kontaktdaten vom Entwickler, die momentan an nicht nur im AdminBereich vorhanden sind, sondern auch für Benutzer auf MainPage, damit im Development-Modus gerade Kontakt mit End-User zu haben. Dies erlaubt mit der Rückmeldungen besser umzugehen.

Damit ist die Struktur von aller Pages standartmäßig gleich: header – main bereich -footer.

Außer technischen Eigenschaften soll auch Benutzerfreundlichen und intuitiv klaren Design entwickelt werden.

Im Rahmen vom Abschlussprojekt wird nur ein Teil von der gesamten Anwendung implementiert, da der Aufwand größer ist, als die Abschlussprojektanforderungen erlauben.

Das Projekt umfasst die Frontend- und Backend-Funktionalität für die Users bzw. Benutzer im Form von MainPage, Navigation zwischen Seiten und die gemeinsame Elemente für alle Oberflächern. Im ersten Schritt wird die Anwendung lokal ausgeführt und auf dem lokalen Server gespeichert.

Aus Zeitlichen Gründen erfolgt die Testphase von MainPage erst nach dem Abschlussprojekt.

Danach wird meine Aufgabe die Authentifizierung und Autorisierung mittels ASP.NET Core Identity zu implementieren. Entwurf vom AdminBereich und die Emplementierung vom Layout für Tabletts gehören dazu.

Die Sicherheitsmaßnahmen wie zun Beispiel die Überprüfung vom Context der Datatsets, die von der Datenbank abgelesen werden, gehören zu meinen Aufgaben nicht, da die benötigte Kenntnisse durch die entsprechende Schulungen erworben werden sollen.

Der Schwerpunkt vom Abschlussprojekt liegt an der Frontendentwicklung und den Aufbau von der Verbindung mit der existierenden Microsoft SQL Server – Datenbank, von deren die benötigte Daten abgelesen werden.

Datenaustausch min dem Auftragsgeber erfolgt über TeamViewer.

Während der Projektarbeit entstehen folgende tatsächliche Kosten:

Tabelle 1. Kostenkalkulation:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Stunden | Stundensatz, Eur | Kosten, Eur |
| Projektleiter | 3 | 120 | 360 |
| Praktikantin[[1]](#footnote-1) | 80 | 5 | 400 |
| **Gesamt** | **760** | | |

Im Project werden Community-Editions von Entwicklungssoftware verwendet, was keine zusätzliche Kosten verursacht.

Einer der Hauptvorteile des Projekts ist modulare Design, das die Erweiterung und die anschließende Wartung erleichtert. Die Benutzerfreundlichkeit und die Bedeckung vom Auftragsgeberanforderungen stehen im Mittelpunkt und spielen in diesem Projekt die wichtigste Rolle. Deswegen wurde jedes Detail sorgfältig ausgearbeitet.

# Auswahl der Software

## Lösungsalternativen

Einen alternativen Lösungsweg würde die Verwendung der reine CSS3 und HTML5 anstatt SASS und Razor-Pages für Frontend. In diesem Fall könnten nicht die Variablen CSS-Dateien verwendet werden. Im Unterschied zu HTML erlauben Razor-Pages mehr Möglichkeiten für die Modulare Aufbau vom Code und, was auch wichtig ist, die C#-Anweisungen in HTML-Code zu integrieren.

Als Frontend-Computersprache wurde reine JavaScript benutzt, da die besser für die kleine Code-Abschnitte (PopUp-Fenster und Filter) passt als jQuery Framework. Die JavaScript-Grundlagen waren ein Teil von der Umschulung. Im Gegenteil dazu braucht man riesig viel Zeit, um mit jQuery einzusteigen.

Als Backend-Computersprache könnten zum Beispiel Java oder PHP verwendet werden. Für C# spricht die Tatsache, dass diese Sprache im Betriebsunternehmen gängig ist und auch die Andere ähnliche Anwendungen für den Auftragsgeber auch auf C# implementiert sind. Außerdem verwendet der Auftragsgeber die Microsoft SQL Server – Datenbanken, die mit Entity Framework als Datenprovider sehr gut mit .Net – Frameworks wie ASP .NET oder ASP .NET Core einen gängigen Software-Stack bilden.

Die Anwendung vom ASP.NET Core Framework, der erlaubt die Anwendungsveröffentlichung nicht nur für Windows, sondern auch für die andere gängige Betriebssysteme wie iOS und Linux.

Als Pattern wurde Model-View-Controller(MVC) ausgewählt, was für die Projektarbeit als überflüssig angesehen werden kann, aber am Endeffekt für die gesamte Anwendung genau gut passt.

Es würden keine selbstentwickelten Lösungen für die Authentifizierung und Autorisierung vorgeschlagen, da die Verwendung von standartmäßigen ASP .NET Core Identity Framework mehr professionell aussieht und macht die entwickelte Anwendung von den Entwickler unabhängig.

## Ausgewählter Sofrware-Stack

* Komputersprachen: C#, JavaScript
* Frameworks: ASP .NET Core 3.1., Entity Framework 3.1.32
* Databank Management System: Microsoft SQL Server Express 2019
* Stylesheet-Sprache / Preprozessor: SASS
* IDE: VisualStudio 2019, Community Edition
* Versionsverwaltung: git.

Für Visual Studio wurden folgende Erweiterungen installiert:

* + Bundler & Minifier - nach dem Download erscheint im Projekt-Verzeichnis die Konfigurations-Datei Bundleconfig.json, die angepasst werden soll.
  + WebCompiler - erlaubt die SASS- Dateien zu compilieren
  + ZehnCoding, CSS Tools 2019 – erleichtern das Schreiben vom CSS- und HTML-Code.

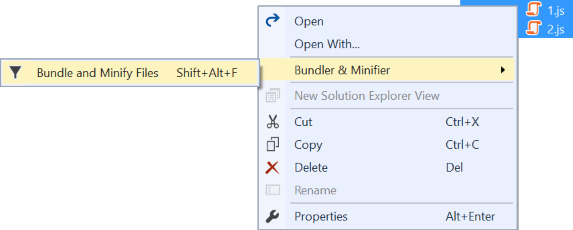
[](https://alekseev74.ru/files/upload/images/edt4m1we.jpg)

Abbildung 3. Die Anwendung vom Bundler & Minifier

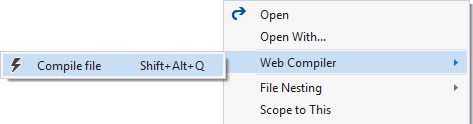
[](https://alekseev74.ru/files/upload/images/m0lhlskr.jpg)

Abbildung 4. Die Anwendung vom WebCompiler

# Planung der Durchführung

## Festlegung der Lösungsschritte

### Layout

Am Anfang wurden die Anforderungen zum Projekt analysiert und die wichtigsten davon als erster Teil des Projekts (entspricht dem Abschlussprojekt) festgelegt.

Dafür wurde ein Layout mit drei Web-Seiten erstellt um sicher zu sein, dass die Anforderungen richtig verstanden wurden. Nach der Abstimmung wurde das Layout angepasst. Dafür wurde die kostenlose Testversion von Moqups[[2]](#footnote-2) verwendet.

Als Schriftart wurde ‚Roboto‘, eine kostenlose Alternative zu Helvetica, ausgewählt und von GoogleFonts als .ttf-Dateien hochgeladen[[3]](#footnote-3); danach mittels Online-Konverter Convertio[[4]](#footnote-4) in .woff und .woff2 Formaten gespeichert, um weiter in SASS-Datei einzubinden.

Icons wurden auf SVG-Repo[[5]](#footnote-5) und CleanPng[[6]](#footnote-6) gefunden und hochgeladen.

### Erstellung des Projekts in Visual Studio

Das Projekt wurde als eine leer ASP .Net Core Anwendung erzeugt (Anhang A.2.).

Die Implementierung vom MVC-Pattern wurde manuell durch die Erzeugung von den entsprechenden Verzeichnissen implementiert:

* wwwroot für statische Dateien mit Unterverzechnisse css/, fonts/, images/, js/ für javascript-Dateien und sass/ für Style;
* Controllers/, Models/ und Views/ ür die Implementierung vom MVC-Pattern;
* Service/ für die Konfigurationsdatei Config.cs

Ein Bild, das Text, Screenshot, Software, Computersymbol enthält.

Automatisch generierte Beschreibung

Abbildung 5. Die erste Verzeichnisse

### Initialisierung vom Git-Repository

Als Versionsverwaltungstool wurde git 2.42.1 installiert, danach die .gitignore-Datei für die zusätzliche Dateien, die nicht in Commits erscheinen sollen, erstellt.

Im Projektverzeichnis wurde mittels *Git Bash* ein Repository initialisiert und der Initial-Commit durchgeführt (Anhang A.3.).

### Entwurf vom Frontend. Konfiguration

Zuerst wurde die HTML- Seiten laut Layout implementiert. Die HTML-Dateien wurden auf mehrere Teile zerlegt, um den Code klar und wartbar zu machen (Anhang A.5.). Die Index.cshtml-Datei befindet sich standartmäßig im Views/Home/ - Verzeichnis, MainPage.cshtml und Partial-Dateien – im Views/Shared/. AdminBereich.cshtml in der 1. Phase bleibt auch im Views/Shared/.

Im MVC-Pattern erden solche Klassen Views genannt.

Um die Fehler zu vermeiden wurde die Anwendung mehrmals gestartet und mit dem Layout verglichen. Die Validation wurde mittels [W3C Markup Validation Service durchgeführt. Leider unterstützt dieser Validator die Razor-Pages nicht, deswegen wurden die Code-Abschnitte manuell durch copy-paste eingefügt und geprüft[[7]](#footnote-7).](https://validator.w3.org/)

Danach wurden Style mittels .scss-Dateien implementiert. Dafür wurde folgende Struktur entworfen: main.scss als Hauptdatei, an der die andere Teile, die den Klassen und Blöcken entsprechen, importiert wurden. (Anhang A.6.)

Da die Rehenfolge von CSS-Style sehr wichtig ist, hat main.scss die folgende Aufbau:

* Einbindung von Shriften,
* Imports von der Normalize-Datei[[8]](#footnote-8), Variablen und Container-Style für <main>-Bereich,
* Style für Tags <main>, <footer>, <a>, <img>, <p> und <ul>, die die default-Style von Brausern überschreiben,
* Style für Klassen, die den Verhalten von Buttons und Liste überschreiben,
* Style für visually-hidden Klass, der ein Block auszublenden erlaubt,
* Imports von Style für Klassen, die in Blöcke verteilt wurden.

Die Verwendung von Style wurde mittels Klassen anstatt Tags-Style implementiert.

Um die SASS- Style erstes Mal anzuwenden, wurde WebCompiler angewendet (recht click auf wwwroot/sass/main.scss 🡪 web compiler 🡪 compile file). Damit werden die folgende Dateien automatisch generiert (Anhang A.8.): main.css, main.min.css, compilerconfig.json, compilerconfig.json.defaults.

Die Dateien main.css, main.min.css sind zu löschen, da durch die Anpassung von compilerconfig.json werden diese css-Dateien beim nächsten Start im Verzeichnis css/ gespeichert:

Ein Bild, das Text, Schrift, Screenshot, Reihe enthält.

Automatisch generierte Beschreibung

Abbildung 6. Anpassung von compilerconfig.json

Um das Projekt starten zu können, wurde ein minimalistishes Homecontroller mit der Action (Metode) Index erzeugt.

Danach wurden die Konfigurations-Infos in der Datei appsettings.json angelegt. Dazu gehören ConnectionString, Kontaktdaten und Konstatnten (Quellcode ist im Anhang A.7. vorhanden).

Die Datei appsettings.json wurde im Laufe des Projekts erweitert (Quellcode ist im Anhang A.7. eingetragen). Um diese Daten im Projekt besser anwenden zu können wurde die Spigel-Datei Config.cs erstellt. Damit können die wichtige Informationen ein Mal in der appsettigs.json geändert und weiter im ganzen Project benutzt werden.

Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

Automatisch generierte Beschreibung

Abbildung 7. Config.cs und appsettings.json

Danach wurde die Methode Configure der Datei sturtup.cs angepasst:

- die statische Dateien (die Dateien aus wwwroot) angeschlossen,

- ein endpoint für HomeController definirt.

In der Methode ConfigureServices wurden Controllers und Viewes angeschlossen und die Verbindung zwischen die Dateien appsettings.json und Config.cs erstellt.

### Backend

Für die Abbildung von Daten, die aus der Datenbank gelesen wurden, wurden die Klassen mit entsprechenden Klassenvariablen (Properties) implementiert: jede aus der Datenbank abgelesene Spalte entspricht einer Klassenvariable. Im MVC-Pattern erden solche Klassen Models genannt.

Jede Model-Klasse entspricht der Abbildung einer Databank-Tabelle. In diesem Projekt wurden zwei solche Tabellen verwendet, jede davon enthält die Daten aus einer von zwei Produktionshallen (Halle B und Halle G).

Die zwei Models (DispoB.cs und DispoG.cs) wurden durch die Vererbung implementiert, da beide teilweise die gleiche Klassenvariablen haben. Die gemeinsame Klassenvariablen sind in der Eltern-Klasse Dispo.cs definiert.

Die Klasse AppBbContext.cs dient zum Ablesen vom Datenbank mittels Entity Framework und enthält die Listen von beiden Models.

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Abbildung 8. MainPageController

Die abgelesenen Daten sind im Controller (MainPageController.cs für MainPage) in zwei Datasets ViewData["DataB"] und ViewData["DataG"] vorhanden.

Die Überprüfung vom Inhalt vom Datacontext ist in diesem Projekt nicht vorgesehen und ist damit eine Schwachstelle.

Die abgelesenen Daten werden im PartialView SyloStatus.cshtml durch die foreach-Schleifen in den Tabellen eingefügt (Quellcode ist im Anhang A.7. vorhanden).

Ein Beispiel von dem Anwendungsablauf ist im Anhang A.9. vorhanden.

## Zeit- und Ressourcenplanung

Der Vergleich mit der im Antrag abgegebenen Planung zeigt, dass der Entwurf von der GUI und die Implementierung haben länger gedauert, als es geplant wurde. Ein Versuch der Implementierung der Verbindung mit der Datenbank mittels Migration hat wegen Versionskompatibilität nicht gelungen. Anstatt dessen sollte eine Alternative gefunden werden, was viel Zeit gekostet hat. Dadurch musste die Zeit für die Dokumentation reduziert und die Autorisierungsfunktionalität aufgegeben werden.

Tabelle 2. Zeitplannung

|  |  |  |  |
| --- | --- | --- | --- |
| Merkmale | Geplannt,  Stunden | Tatsächlich, Stunden | Abweichung |
| **Anforderungsanalyse** | **4** | **2** | **-2** |
| **Konzeptionsphase** | **20** | **20** | **0** |
| Entwurf von Klassen | 8 | 7 | -1 |
| Entwurf von Grafischen Oberflächen | 8 | 12 | +4 |
| Definition von Rollen und Anmeldungs-Optionen | 4 | 1 | -3 |
| **Implementierung** | **40** | **47** | **+7** |
| **Test, Fehlerkorrektur** | **6** | **3** | **-3** |
| **Dokumentation** | **10** | **8** | **-2** |
| **Gesamt** | **80** | **80** | **0** |

## Resümee

Das Projekt wurde aus Zeitmangel in minimalem Umfang umgesetzt.

# Evaluation des Projektverlaufs

Durch das Projekt habe ich erste Erfahrungen von reeller Softwareentwicklung bekommen. Es ist ganz anders als die Theorie: die Sachen, die ganz klar scheinen, funktionieren in der Realität nicht immer. Um manche Probleme zu lösen, braucht man die Flächigkeit zu recherchieren, um selbständig die Lösung zu finden.

Außerdem ist die Unterstützung von Erfahrenden Kollegen sehr wichtig.

Nach diesem Projekt wurde ich besser mit dem ausgewählten Software-Stack umgehen frohe ich mich auf die Möglichkeit an solchen Projekten teilzunehmen.

# Ausblick

Es ist vorgesehen, die Implementierung von Authentifikation und Autorisation umzusetzen.

Dazu soll die Adminumgebung entwickelt werden mit der Funktionalität wie Verwaltung von Benutzern und Rollen.

Für Benutzer soll ein persönliches Bereich und mehrere Filtern für mainPage entwickelt werden.

Außerdem ist die Version für Tabletts vorgesehen.

# Anhang A: Listing

## A.1 Layout vom MainPage, AdminBereich und LoginPage

Ein Bild, das Screenshot, Text, Software, Computersymbol enthält.

Automatisch generierte Beschreibung

Ein Bild, das Screenshot, Software, Multimedia-Software, Grafiksoftware enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Software, Computersymbol enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Software, Computersymbol enthält.

Automatisch generierte Beschreibung

## A.2 Erstellung vom Projekt

Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

Automatisch generierte Beschreibung

## A.3. Initial Commit

Ein Bild, das Text, Elektronik, Screenshot, Software enthält.

Automatisch generierte Beschreibung

## A.4. Projektverzeichnis nach dem ersten Commit

Ein Bild, das Text, Screenshot, Software, Display enthält.

Automatisch generierte Beschreibung

## A.5. HTML-Struktur

Ein Bild, das Text, Screenshot, Software, Schrift enthält.

Automatisch generierte Beschreibung

## A.6. SASS- , CSS und JS-Struktur

Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Software enthält.

Automatisch generierte Beschreibung

## A.7. Exemplarischer Quellcode

Index.cshtml

@using MaterialStatus.Service

@using MaterialStatus.Models

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Login</title>

<link rel="icon" type="image/x-icon" href="/images/favicon\_material\_status.ico">

<link type="text/css" rel="stylesheet" href="~/css/main.min.css" asp-append-version="true"/>

</head>

<body>

@Html.Partial("HeaderLoginPartial")

<main class="container container-login">

<p class="tmp">Diese Seite befindet sich im Development-Modus</p>

<p class="tmp">Login und navigation ohne Passwort</p>

<a class="tmp-link" href="MainPage">to main page ></a>

<a class="tmp-link" href="AdminArea">to admin area ></a>

</main>

@Html.Partial("FooterPartial")

<div class="modal-window" id="modal-contact"> <!-- modal window -->

@Html.Partial("ModalContactPartial")

@Html.Partial("ModalLocationPartial")

<button class="reset-button modal-close" id="contact-close"></button>

</div>

</body>

</html>

MainPage.cshtml

@using MaterialStatus.Service

@using MaterialStatus.Models

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>MainPage</title>

<link rel="icon" type="image/x-icon" href="/images/favicon\_material\_status.ico">

<link type="text/css" rel="stylesheet" href="~/css/main.min.css" asp-append-version="true" />

<script src="/js/script.min.js" async asp-append-version="true"></script>

</head>

<body>

@Html.Partial("HeaderMainPagePartial")

<main class="container">

<aside class="filter-main">

<!--todo visually-hidden-->

@Html.Partial("ModalFilterMainPartial")

</aside>

<section class="content-main-page">

<div><a name="top"></a></div>

@Html.Partial("SyloStatus") <!-- tables -->

</section>

</main>

@Html.Partial("FooterPartial")

<div class="modal-window" id="modal-contact">

<!-- modal window -->

@Html.Partial("ModalContactPartial")

@Html.Partial("ModalLocationPartial")

<button class="reset-button modal-close" id="contact-close"></button>

</div>

</body>

</html>

ModalContactPartial.cshtml

@using MaterialStatus.Service

<secion class="modal-contact">

<h2 class="visually-hidden">Firmen-Kontakte von Rothenbacher GmbH</h2>

<ul class="contact-list">

<li class="contact-item contact-header">

Wir sind für Sie da

</li>

<li class="contact-item">

<a class="contact-link" href=@Config.CompanyWebSite target="\_blank">@Config.CompanyName</a>

</li>

<li class="contact-item ">

<address class="address-wrapper">

<a class="contact-link adress" href=@Config.CompanyGoogleMapLink target="\_blank">

@Config.CompanyStreet, <br /> @Config.CompanyCity ﻿

</a>

</address>

</li>

<li class="contact-item">

<a class="contact-link phone" target="\_blank" href="tel:@Config.CompanyPhoneShort">@Config.CompanyPhone</a>

</li>

<li class="contact-item">

<a class="contact-link fax" target="\_blank" href="fax:@Config.CompanyFaxShort">@Config.CompanyFax</a>

</li>

<li class="contact-item">

<a class="contact-link email" href="mailto:@Config.CompanyEmail">@Config.CompanyEmail</a>

</li>

<li class="contact-item">

<a class="contact-link web-address" href=@Config.CompanyWebSite target="\_blank">@Config.CompanyWebSiteShort</a>

</li>

</ul>

</secion>

HeaderMainPagePartial.cshtml

<header class="header">

<nav class="main-nav">

<div class="main-nav-block main-nav-left-block">

@Html.Partial("ProjectLogoPartial")

@Html.Partial("NavigationElementsPartial")

</div>

<div class="main-nav-block main-nav-center-block">

<p class="nav-item user-name">John Snow</p>

</div>

<div class="main-nav-right-block">

<ul class="functional-nav main-nav-block">

<li class="functional-nav-item">

<a class="admin-area-link main-page-link" href="AdminArea">Adminbereich</a>

</li>

<li class="functional-nav-item">

<a class="help-link main-page-link">Hilfe</a>

</li>

<li class="functional-nav-item">

<a class="logout-link main-page-link" href="https://localhost:44340/">Ausloggen</a>

</li>

</ul>

</div>

</nav>

</header>

FooterPartial.cshtml

<footer class="footer">

<div class="rothenbacher-logo-wrapper">

<a class="footer-link" href="http://www.rothenbacher-gmbh.de/" target="\_blank">

<img class="rothenbacher-logo-image" src="/images/logo\_rothenbacher.png"

alt="Logo von Rothenbacher GmbH und Link zur Website rothenbacher.com"/>

</a>

</div>

<div class="footer-item-wrapper">

<button class="reset-button contact-button footer-link" id="contact-open">Kontakt</button>

</div>

<div class="footer-item-wrapper emergency-service-wrapper">

<img class="emergency-service-icon" src="~/images/icon\_24\_7.png"

alt="Symbol vom Computernotdienst"/>

<div class="emergency-service-text-block">

<p class="emergency-service-titel">Emergency Service</p>

<a class="footer-link" href="tel:+4917620063724">(+49) 176 200 63 724</a>

</div>

</div>

<div class="footer-item-wrapper">

<p>

Copyright &#169; 2023

<a class="footer-link" href="http://www.rothenbacher-gmbh.de/" target="\_blank">Rothenbacher GmbH</a>.

All rights reserved.

</p>

</div>

<div class="footer-item-wrapper">

<p>

Graphic design tools:

<a class="footer-link" href="https://moqups.com/" target="\_blank">moqups</a>,

<a class="footer-link" href="https://www.svgrepo.com" target="\_blank">svgrepo</a>,

<a class="footer-link" href="https://www.cleanpng.com/" target="\_blank">cleanpng</a>

</p>

</div>

</footer>

Main.scss

@font-face {

font-family: 'Roboto';

src: url('/fonts/roboto-regular.woff2') format('woff2'),

url('/fonts/roboto-regular.woff') format('woff');

}

@font-face {

font-family: 'Roboto-bold';

src: url('/fonts/roboto-bold.woff2') format('woff2'), url('/fonts/roboto-bold.woff') format('woff');

}

@import "blocks/normalize.scss";

@import "blocks/variables.scss";

@import "blocks/container.scss";

html {

box-sizing: border-box; // um die paddings und margins bei width: 100%; nicht aus dem Block fallen

}

\*,

\*::before,

\*::after {

box-sizing: inherit;

}

body {

position: relative;

display: flex;

flex-direction: column;

min-width: $hd-width;

min-height: 100vh;

margin: 0;

padding: 0;

color: $main-color;

font-family: 'Roboto', 'Arial', sans-serif;

}

main {

box-sizing: border-box;

}

footer {

flex-shrink: 0;

}

a {

margin: 0;

padding: 0;

color: $main-color;

text-decoration: none;

}

img {

max-width: 100%;

height: auto;

}

p {

margin: 0;

}

ul {

margin: 0;

padding: 0;

list-style-type: none;

}

.reset-button {

margin: 0;

padding: 0;

background: initial;

border: none;

font-family: inherit;

font-size: inherit;

}

.reset-list {

position: relative;

list-style: none;

padding: 0;

}

.visually-hidden {

position: absolute;

width: 1px;

height: 1px;

margin: -1px -1px;

border: 0;

padding: 0;

white-space: nowrap; /\* Пробелы не учитываются, переносы строк в коде HTML игнорируются, весь текст отображается одной строкой;\*/

clip: rect(0, 0, 0, 0);

clip-path: inset(100%);

overflow: hidden;

}

//Imports

//--Header (main page)

@import "blocks/header.scss";

@import "blocks/filter-button.scss";

@import "blocks/burger-button.scss";

@import "blocks/project-logo.scss";

@import "blocks/main-nav.scss";

@import "blocks/nav-items.scss";

@import "blocks/functional-nav.scss";

@import "blocks/main-page-link.scss";

@import "blocks/user-name.scss";

@import "blocks/admin-area-link.scss";

@import "blocks/help-link.scss";

@import "blocks/logout-link.scss";

//--Footer

@import "blocks/footer.scss";

@import "blocks/emergency-service.scss";

@import "blocks/rothenbacher-logo.scss";

//--Modal window

@import "blocks/modal.scss";

//--Table

@import "blocks/table-material.scss";

//--Contetnt (main-page)

@import "blocks/contetnt-main-page.scss";

@import "blocks/sylo.scss";

//--Filter (main-page)

@import "blocks/filter.scss";

@import "blocks/filter-button.scss";

//--Tmp-styles for development modus

@import "blocks/tmp.scss";

Startup.cs

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using MaterialStatus.Models;

using Microsoft.AspNetCore.Identity;

using MaterialStatus.Service;

namespace MaterialStatus

{

public class Startup

{

public IConfiguration Configuration { get; }

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public void ConfigureServices(IServiceCollection services)

{

Configuration.Bind("Project", new Config());

services.AddDbContext<AppDbContext>(item => item.UseSqlServer(Config.ConnectionStringTest));

//--

services.ConfigureApplicationCookie(options =>

{

options.Cookie.Name = "myCompanyAuth";

options.Cookie.HttpOnly = true;

options.LoginPath = "/account/login";

options.AccessDeniedPath = "/account/accessdenied";

options.SlidingExpiration = true;

});

services.AddControllersWithViews()

.SetCompatibilityVersion(CompatibilityVersion.Version\_3\_0).AddSessionStateTempDataProvider();

}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseStaticFiles();

app.UseRouting();

//--

app.UseCookiePolicy();

app.UseAuthentication();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute("default", "{controller=Home}/{action=Index}/{Id?}");

endpoints.MapControllerRoute("main", "{controller=MainPage}/{action=MainPage}/{Id?}");

endpoints.MapControllerRoute("admin", "{controller=AdminArea}/{action=AdminArea}/{Id?}");

});

}

}

}

Appsettings.json

{

"Project": {

"Version": "1.0",

"ConnectionStringTest": "Data Source=BEELINK\_ELENA\\SQLEXPRESS;Database=SWB2; Persist Security Info=false; User ID='sa'; Password='sa'; MultipleActiveResultSets=True; Trusted\_Connection=False;",

"ConnectionStringSWBv2": "Data Source=BEELINK\_ELENA\\SQLEXPRESS; Database=Rodos\_SWB\_v2; Persist Security Info=false; User ID='sa'; Password='sa'; MultipleActiveResultSets=True; Trusted\_Connection=False;",

"ConnectionStringSWGv2": "Data Source=BEELINK\_ELENA\\SQLEXPRESS; Database=Rodos\_SWG; Persist Security Info=false; User ID='sa'; Password='sa'; MultipleActiveResultSets=True; Trusted\_Connection=False;",

"HalleB\_Linie1\_id": 0,

"HalleB\_Linie1\_SyloNr": "B-1",

"HalleB\_Linie3\_id": 2,

"HalleB\_Linie3\_SyloNr": "B-3",

"HalleG\_Linie2\_id": 1,

"HalleG\_SyloNr": "G-2",

"CompanyName": "Rothenbacher GmbH",

"CompanyPhone": "(07348) 20 12 08",

"CompanyPhoneShort": "+497348201208",

"CompanyFax": "(07348) 20 13 82",

"CompanyFaxShort": "+497348201382",

"CompanyEmail": "info@rothenbacher-gmbh.com",

"CompanyStreet": "Zeppelinstr. 16",

"CompanyCity": "89160 Dornstadt",

"CompanyWebSite": "http://www.rothenbacher-gmbh.de/",

"CompanyWebSiteShort": "www.rothenbacher-gmbh.de",

"CompanyGoogleMapLink": "https://maps.google.com/maps?q=S.+Rothenbacher+GmbH"

}

}

AppDbContext.cs

using Microsoft.EntityFrameworkCore;

namespace MaterialStatus.Models

{

public class AppDbContext : DbContext

{

public DbSet<DispoB> DisposB { get; set; }

public DbSet<DispoG> DisposG { get; set; }

public AppDbContext(DbContextOptions<AppDbContext> options)

: base(options)

{

Database.EnsureCreated(); // without migration

}

}

}

SyloStatus.cshtml  
@using MaterialStatus.Models

@using MaterialStatus.Service

@using System

<article class="sylo sylo-halle-b"> <!--Halle B, Line 1 -->

<div class="sylo-left-block">

<h2 class="sylo-header">Sylo <span class="sylo-number">@Config.HalleB\_Linie1\_SyloNr</span></h2>

<div class="sylo-image-wrapper">

<img class="sylo-image" src="/images/img\_silo.png"

alt="Sylo icon" />

</div>

</div>

<div class="sylo-right-block">

<table class="table-material">

<tr class="table-material-row">

<th class="table-material-header-item">Material</th>

<th class="table-material-header-item">Soll C.</th>

<th class="table-material-header-item">Ist C.</th>

<th class="table-material-header-item">Soll, Kg</th>

<th class="table-material-header-item">Ist, Kg </th>

<th class="table-material-header-item">AuftragsNr</th>

<th class="table-material-header-item">LineId</th>

<th class="table-material-header-item">AnfangsDatum</th>

<th class="table-material-header-item">EndDatum</th>

<th class="table-material-header-item">RezeptNr</th>

<th class="table-material-header-item">RezeptVersion</th>

</tr>

<!-- foreach -->

@foreach (var item in ViewData["DataB"] as List<DispoB>)

{

if (item.Line\_id == Config.HalleB\_Linie1\_id & item.Start\_time >= new DateTime(2023, 12,01))

{

<tr class="table-material-row">

<td class="table-material-item">@item.Material\_Bez</td>

<td class="table-material-item">@item.Chargenanzahl\_soll</td>

<td class="table-material-item">@item.Chargenanzahl\_ist</td>

<td class="table-material-item">@Math.Round(item.Auftrags\_gewicht, 2)</td>

<td class="table-material-item">@Math.Round(item.Auftrags\_gewicht\_ist, 2)</td>

<td class="table-material-item">@item.Auftragsnr</td>

<td class="table-material-item">@item.Line\_id</td>

<td class="table-material-item">@item.Start\_time</td>

<td class="table-material-item">@item.End\_time</td>

<td class="table-material-item">@item.Rezept\_id</td>

<td class="table-material-item">@item.Rezept\_version\_id</td>

</tr>

}

}

</table>

</div>

</article>

<article class="sylo sylo-halle-b"> <!--Halle B, Line 3 -->

<div class="sylo-left-block">

<h2 class="sylo-header">Sylo <span class="sylo-number">@Config.HalleB\_Linie3\_SyloNr</span></h2>

<div class="sylo-image-wrapper">

<img class="sylo-image" src="/images/img\_silo.png"

alt="Sylo icon" />

</div>

</div>

<div class="sylo-right-block">

<table class="table-material">

<tr class="table-material-row">

<th class="table-material-header-item">Material</th>

<th class="table-material-header-item">Soll C.</th>

<th class="table-material-header-item">Ist C.</th>

<th class="table-material-header-item">Soll, Kg</th>

<th class="table-material-header-item">Ist, Kg </th>

<th class="table-material-header-item">AuftragsNr</th>

<th class="table-material-header-item">LineId</th>

<th class="table-material-header-item">AnfangsDatum</th>

<th class="table-material-header-item">EndDatum</th>

<th class="table-material-header-item">RezeptNr</th>

<th class="table-material-header-item">RezeptVersion</th>

</tr>

@foreach (var item in ViewData["DataB"] as List<DispoB>)

{

if (item.Line\_id == Config.HalleB\_Linie3\_id & item.Start\_time >= new DateTime(2023, 12, 01))

{

<tr class="table-material-row">

<td class="table-material-item">@item.Material\_Bez</td>

<td class="table-material-item">@item.Chargenanzahl\_soll</td>

<td class="table-material-item">@item.Chargenanzahl\_ist</td>

<td class="table-material-item">@Math.Round(item.Auftrags\_gewicht, 2)</td>

<td class="table-material-item">@Math.Round(item.Auftrags\_gewicht\_ist, 2)</td>

<td class="table-material-item">@item.Auftragsnr</td>

<td class="table-material-item">@item.Line\_id</td>

<td class="table-material-item">@item.Start\_time</td>

<td class="table-material-item">@item.End\_time</td>

<td class="table-material-item">@item.Rezept\_id</td>

<td class="table-material-item">@item.Rezept\_version\_id</td>

</tr>

}

}

</table>

</div>

</article>

<article class="sylo sylo-halle-g"> <!--Halle G, Linie 2-->

<div class="sylo-left-block">

<h2 class="sylo-header">Sylo <span class="sylo-number">@Config.HalleG\_SyloNr</span></h2>

<div class="sylo-image-wrapper">

<img class="sylo-image" src="/images/img\_silo.png"

alt="Sylo icon" />

</div>

</div>

<div class="sylo-right-block">

<table class="table-material">

<tr class="table-material-row">

<th class="table-material-header-item">Material</th>

<th class="table-material-header-item">Soll C.</th>

<th class="table-material-header-item">Ist C.</th>

<th class="table-material-header-item">Soll, Kg</th>

<th class="table-material-header-item">Ist, Kg </th>

<th class="table-material-header-item">AuftragsNr</th>

<th class="table-material-header-item">LineId</th>

<th class="table-material-header-item">AnfangsDatum</th>

<th class="table-material-header-item">EndDatum</th>

<th class="table-material-header-item">RezeptNr</th>

<th class="table-material-header-item">RezeptVersion</th>

</tr>

@foreach (var item in ViewData["DataG"] as List<DispoG>)

{

if (item.Line\_id == Config.HalleG\_Linie2\_id && item.Start\_time >= new DateTime(2023, 12, 01))

{

<tr class="table-material-row">

<td class="table-material-item">@item.Material\_Bez</td>

<td class="table-material-item">@item.Chargenanzahl\_soll</td>

<td class="table-material-item">@item.Chargenanzahl\_ist</td>

<td class="table-material-item">@Math.Round(item.Auftrags\_gewicht, 2)</td>

<td class="table-material-item">@Math.Round(item.Auftrags\_gewicht\_ist, 2)</td>

<td class="table-material-item">@item.Auftragsnr</td>

<td class="table-material-item">@item.Line\_id</td>

<td class="table-material-item">@item.Start\_time</td>

<td class="table-material-item">@item.End\_time</td>

<td class="table-material-item">@item.Rezept\_id</td>

<td class="table-material-item">@item.Rezept\_version\_id</td>

</tr>

}

}

</table>

</div>

</article>

## A.8 Screenshots von SASS-Konfiguration

Ein Bild, das Text, Screenshot, Schrift, Design enthält.

Automatisch generierte Beschreibung

## A.9. Ergebnis

Ein Bild, das Text, Elektronik, Screenshot, Software enthält.

Automatisch generierte Beschreibung

# Anhang B:

# Literaturverzeichnis

Online-Kurs ASP.NET CORE DEVELOPER von ITVDN  
<https://itvdn.com/ru/specialities/aspnet-core>

# Abbildungsverzeichnis

Abbildung 1. Oberflächernschema für Administrator…………………………………..2

Abbildung 2. Oberflächernschema für User…………………………………………….3

Abbildung 3. Die Anwendung vom Bundler & Minifier…………………………………5

Abbildung 4.Die Anwendung vom WebCompiler……………………………………….5

Abbildung 5. Die erste Verzeichnisse……………………………………………………6

Abbildung 6. Anpassung von compilerconfig.json………..…………………………….8

Abbildung 7. Config.cs und appsettings.json ………………………………………..…8

Abbildung 8. MainPageController ………………………………………………………..9

……………..

# Tabellenverzeichnis

Tabelle 1: Kostenkalkulation 4

Tabelle 2. Zeitplannung……………………………………………………..…….10

# Abkürzungsverzeichnis / Glossar

MVC Model-View-Controller

SQL Structured Query Language

JSON JavaScript Object Notation

SASS Syntactically Awesome Stylesheets

ASP Application Service Provider

.ttf TrueType Font

.woff Web Open Font Format

1. Inklusive Kosten für Arbeitsplatz [↑](#footnote-ref-1)
2. <https://moqups.com/> - stand 08.12.2023 [↑](#footnote-ref-2)
3. <https://fonts.google.com/specimen/Roboto> - stand 08.12.2023 [↑](#footnote-ref-3)
4. <https://convertio.co/de/ttf-woff/> - stand 08.12.2023 [↑](#footnote-ref-4)
5. <https://www.svgrepo.com/> - stand 08.12.2023 [↑](#footnote-ref-5)
6. <https://www.cleanpng.com/> - stand 08.12.2023 [↑](#footnote-ref-6)
7. <https://validator.w3.org/#validate_by_input> – stand 08.12.2023 [↑](#footnote-ref-7)
8. <https://necolas.github.io/normalize.css/> - Stand 08.12.2023 [↑](#footnote-ref-8)