Inhalt

[Abkürzungsverzeichnis A](#_Toc448831361)

[1. Einleitung 1](#_Toc448831362)

[1.1. Projektziel 1](#_Toc448831363)

[1.2. Projektbegründung 1](#_Toc448831364)

[1.3. Projektschnittstellen 1](#_Toc448831365)

[2. Projektplanung 1](#_Toc448831366)

[2.1. Projektzeitplan 1](#_Toc448831367)

[2.2. Ressourcenplanung 2](#_Toc448831368)

[2.3. Entwicklungsprozess 2](#_Toc448831369)

[2.3.1. Spiralmodell 2](#_Toc448831370)

[2.3.2. Test-Driven Development (TDD) 2](#_Toc448831371)

[3. Analysephase 2](#_Toc448831372)

[3.1. Ist-Analyse 2](#_Toc448831373)

[3.2. Lastenheft 3](#_Toc448831374)

[3.3. Zwischenstand 3](#_Toc448831375)

[4. Entwurfsphase 3](#_Toc448831376)

[4.1. Zielplattform 3](#_Toc448831377)

[4.2. Architekturdesign 3](#_Toc448831378)

[4.3. Benutzeroberfläche 3](#_Toc448831379)

[4.4. Datenbankentwurf 3](#_Toc448831380)

[4.5. Pflichtenheft 3](#_Toc448831381)

[4.6. Zwischenstand 4](#_Toc448831382)

[5. Implementierungsphase 4](#_Toc448831383)

[5.1. Implementierung der Datenstrukturen 4](#_Toc448831384)

[5.2. Implementierung der Geschäftslogik 4](#_Toc448831385)

[5.3. Implementierung der Benutzeroberfläche 4](#_Toc448831386)

[5.4. Zwischenstand 4](#_Toc448831387)

[6. Abnahmephase 4](#_Toc448831388)

[7. Dokumentation 4](#_Toc448831389)

[8. Fazit 4](#_Toc448831390)

[8.1. Soll- / Ist- Vergleich 4](#_Toc448831391)

[8.2. Aus dem Projekt Gewonnenes 4](#_Toc448831392)

[8.3. Ausblick 4](#_Toc448831393)

[A. Anhang I](#_Toc448831394)

[A.1 Detaillierter Zeitplan I](#_Toc448831395)

[A.2 Ressourcen I](#_Toc448831396)

[A.3 Lastenheft II](#_Toc448831397)

[A.4 Schichtenmodell III](#_Toc448831398)

[A.5 Oberflächenentwürfe III](#_Toc448831399)

[A.6 ER-Modell V](#_Toc448831400)

[A.7 Pflichtenheft V](#_Toc448831401)

[A.8 Screenshot der Anwendung VI](#_Toc448831402)

# **Abkürzungsverzeichnis**

**CSV** Comma Separated Value  
**ERM** Entity-Relationship-Modell  
**GUI** Graphical User Interface  
**SQL** Structured Query Language  
**TDD** Test Driven Development  
**WPF** Windows Presentation Foundation  
**XAML** Extensi ble-Application-Markup-Language

# Einleitung

Die folgende Projektdokumentation schildert den Ablauf des Datenbankprojekts, welches das Projektteam im Rahmen einer ihnen gestellten Aufgabe im Lernfeld 6 durchgeführt hat.

## Projektziel

Ziel des Projektes ist es verschiedene Hardware-Komponenten zu erfassen und zu verwalten. Dazu soll eine Datenbank erstellt werden, welche die Daten dauerhaft in einem persistenten Zustand speichert. Zudem soll eine graphische Benutzeroberfläche (GUI) erstellt werden über welche dem Nutzer eine einfache Verwaltung der Komponenten ermöglicht wird. Die Anwendung soll das Auslesen und Sortieren der Teile ermöglichen, sodass ein guter Überblick über die vorhandenen Komponenten entsteht. Durch das System können menschliche Fehler, wie zum Beispiel das doppelte Eintragen vermieden werden.

## Projektbegründung

In einem Lager gibt es oftmals viele Bauteile, welche ohne eine Struktur und die entsprechenden Dokumente nur sehr schwer zu verwalten sind. So müssen die Mitarbeiter beispielsweise Tabellen und Listen führen um einen Überblick des aktuellen Lagerbestandes zu behalten. Diese manuelle Verwaltung kostet viel Zeit und kann im schlimmsten Fall den Arbeitsfluss an mehreren Stellen unterbrechen. Aus diesen Gründen soll eine Software erstellt werden, welche die Verwaltung der Komponenten erleichtert und somit langfristig eine Kostenersparnis bewirken kann.

## Projektschnittstellen

Die Anwendung wird mit einer Datenbank kommunizieren, welche auf einem externen System betrieben werden kann, diese kann mittels einer grafischen Benutzeroberfläche auf den Client-Rechnern von den Lagerangestellten verwaltet werden. Außerdem werden Ein- und Verkaufs-Mitarbeiter das System nutzen, um einen Einblick in die aktuelle Lagersituation zu erhalten.

# Projektplanung

Während der Projektplanung wurden der Ablauf, sowie die zeitliche Durchführung des Projektes festgelegt. Zudem wurden die benötigten Ressourcen geplant.

## Projektzeitplan

Insgesamt standen dem Projektteam für die Umsetzung des Projektes 24 Stunden zur Verfügung. Die Stunden wurden auf verschiedene Phasen aufgeteilt, welche wiederrum detaillierte Aufgaben enthalten. Die definierten Aufgaben des Projektes wurden je nach Kenntnisstand und Erfahrung unter den Teammitgliedern aufgeteilt. Die Zeitplanung mit einer Unterteilung nach Phasen kann der Tabelle 1: Grober Zeitplan entnommen werden. Die detaillierte Planung findet sich im Anhang unter [A.1 Detaillierter Zeitplan](#_Detaillierter_Zeitplan).

|  |  |
| --- | --- |
| Projektphase | Geplante Zeit |
| Analysephase | 2 Stunden |
| Entwurfsphase | 4 Stunden |
| Implementierungsphase | 13 Stunden |
| Abnahmetest der Gruppe | 1 Stunde |
| Erstellen der Dokumentation | 4 Stunden |
| Gesamt | 24 Stunden |

Tabelle 1: Grober Zeitplan

## Ressourcenplanung

Während der Durchführung des Projektes wurden einige Hard- und Software-Ressourcen für verschiedenste Zwecke eingesetzt. Eine Übersicht aller verwendeten Ressourcen findet man im Anhang unter [A.2 Ressourcen](#_Ressourcen). Es wurde besonders darauf geachtet nur freie Software zu verwenden.

## Entwicklungsprozess

### Spiralmodell

Das Projektteam hat sich für ein Vorgehen mit dem Spiralmodell entschieden. Dieses ist ein iteratives Vorgehensmodell mit besonderem Augenmerk auf die Risikobetrachtung. Vor jedem Zyklus wurden die Risiken der jeweiligen Projektphase von Projektteam besprochen und Maßnahmen zur Vermeidung dieser Risiken festgelegt. Der iterative Ansatz ermöglicht ein agileres Vorgehen, als das Wasserfallmodell. Das Modell fördert eine fehlerfreie Programmierung und ermöglicht es durch die kurzfristigen Zyklen die Kontrolle über die zeitliche Durchführung des Projektes zu behalten.

### Test-Driven Development (TDD)

Bei dem Test-Driven-Development (TDD) wurden sogenannte Komponententests (Unit-Tests) genutzt, um die Entwicklung der Software zu steuern. Dabei wurde nach mit folgendem Zyklus gearbeitet:

1. Unit-Test schreiben (Schlägt fehl)
2. Code implementieren, bis Test nicht mehr fehlschlägt
3. Code refaktorisieren

Diese Art der Programmierung fördert ein sauberes Programmieren und wirkt dem Auftreten von Fehlern im Code entgegen. Das Modell lässt sich mit dem Spiralmodell kombinieren, da die Tests eine Möglichkeit darstellen, um die Risiken einer Implementierungsphase zu reduzieren.

# Analysephase

Während der Analysephase wurde der Ist-Zustand ermittelt. Am Ende der Analysephase steht das Lastenheft, welches die ausformulierten Anforderungen aus Anwendersicht enthält.

## Ist-Analyse

|  |  |  |
| --- | --- | --- |
| Für welche Arbeitsabläufe wird das Inventar Management System benötigt? |  |  |
| Wie wird das Management bisher erledigt? |  |  |
| Welche Probleme sind bei dem bisherigen Vorgehen Aufgetreten? |  |  |
| Wie viele Personen nutzen das Programm? |  |  |
| Wie hoch ist der Gebrauch des bisherigen Management Programms? |  |  |

## Lastenheft

Das Lastenheft wurde am Ende der Analysephase erstellt. Dieses enthält die Kriterien, die die Entwickler aus Sicht der Anwender formuliert haben. Die Anforderungen wurden der Priorität absteigend nach Muss-, Soll- und Kann-Kriterien gestaffelt. Ein Auszug des Lastenhefts findet sich im Anhang unter [A.3 Lastenheft](#_Lastenheft).

## Zwischenstand

# Entwurfsphase

Während der Entwurfsphase wurde die technische Umsetzung des Projektes geplant und das Design der Benutzeroberfläche festgelegt. Am Ende der Entwurfsphase steht das Pflichtenheft.

## Zielplattform

Als Zielplattform der Anwendung wurde ein Windows-System mit installiertem .NET-Framework gewählt. Das Aufbauen der Anwendung nach dem Schichtenmodell ermöglicht es die Datenbank auf einem separaten System zu betreiben.

## Architekturdesign

Für die Umsetzung des Projektes wurde vom Projektteam vereinbart die einzelnen Module der Anwendung in Schichten darzustellen. Dies fördert eine modulare Programmierung und somit die Wartbarkeit und Wiederverwendbarkeit der Anwendung. Eine Übersicht über die Schichten kann man dem Anhang unter [A.4 Schichtenmodell](#_Schichtenmodell) entnehmen.

## Benutzeroberfläche

Um eine ergonomische Design zu gestalten wurden bereits in der Entwicklungsphase Konzepte für eine Benutzeroberfläche erstellt (Mockups). Diese ermöglichen eine schnellere Erstellung der Oberfläche und eine strukturierte Darstellung der verwendeten Komponenten. Alle Mockups wurden mit dem Online-Tool unter <http://www.draw.io> erstellt und können im Anhang unter [A.5 Oberflächenentwürfe](#_Oberflächenentwürfe) eingesehen werden.

## Datenbankentwurf

Um den Lagerbestand in der relationalen Datenbank korrekt abzubilden wurde im Vorfeld ein Entity-Relationship-Model (ERM) erstellt. Dieses findet sich im Anhang unter [A.6 ER-Modell](#_ER-Modell). Neben den wichtigsten Komponenten werden auch zwei Stammdaten-Tabellen „Hersteller“ und „Schnittstelle“, sowie die Beziehung zu den Komponenten aufgezeigt. Das ER-Modell enthält außerdem Angaben über die Attribute der jeweiligen Entitäten.

## Pflichtenheft

Das Pflichtenheft wurde basierend auf den im Lastenheft spezifizierten Anforderungen und der in der Entwurfsphase getroffenen Entscheidungen erstellt. Es enthält die konkrete Umsetzung der im  
[3.2 Lastenheft](#_Lastenheft_1) definierten Anforderungen. Ein Auszug des Pflichtenhefts ist im Anhang unter [A.7 Pflichtenheft](#_Pflichtenheft) zu finden.

## Zwischenstand

# Implementierungsphase

## Iterationsplan

Zu Beginn der Implementierungsphase wurde ein Iterationsplan erstellt, dieser bietet eine Übersicht über die zu implementierenden Bestandteile der Software. Der Iterationsplan diente während der Durchführung als Anhaltspunkt für den Fortschritt der Implementierungsphase. Der Implementierungsplan ist im Anhang unter [A.8 Iterationsplan](#_Iterationsplan) zu finden.

## Implementierung der Datenstrukturen

Die in der Entwurfsphase definierte Struktur der Datenbank, welche unter [4.4 Datenbankentwurf](#_Datenbankentwurf) genauer beschrieben wurde, wurde mittels SQL in Tabellen und Attribute in einer relationalen Datenbank übersetzt. Die Integrität der Daten wurde mit Hilfe von Datenbank-Constraints umgesetzt.

## Implementierung der Geschäftslogik

Die Entitäten der Datenbank wurden als eigenständige Klassen mit den jeweiligen Attributen implementiert. Auf der Schicht der Datenbankverbindung wurden sogenannte Datenzugriffsklassen (Data Access) geschrieben, welche die Kommunikation mit der Datenbank gewährleisten. Um die Konsistenz der Daten zu sichern wurden zusätzlich zu den Datenbank-Constraints auch Validatoren implementiert.

## Implementierung der Benutzeroberfläche

Die grafische Benutzeroberfläche wurde auf Basis der in der Entwurfsphase erstellten Mockups mit Hilfe der *Windows Presentation Foundation (WPF)* erstellt. Der Code für die grafische Oberfläche wurde im sogenannten *Extensi ble-Application-Markup-Language-Format (XAML)* geschrieben. Dabei wurde auch der in Visual Studio integrierte Designer verwendet. Einen Screenshot der Anwendung findet sich im Anhang unter [A.9 Screenshot der Anwendung](#_Screenshot_der_Anwendung).

## Zwischenstand

# Abnahmephase

In der Abnahmephase wurde von dem Projektteam ein Oberflächentest durchgeführt, um die grafische Benutzeroberfläche unter dem Gesichtspunkt der Benutzerfreundlichkeit und Ergonomie zu überprüfen. Im Zuge dieses Tests wurde auch ein Datenkonsistenztest durchgeführt, um die erstellten Datenbank-Regeln und die implementierten Validatoren auf ihre Korrektheit zu testen.

# Dokumentation

# Fazit

## Soll- / Ist- Vergleich

## Aus dem Projekt Gewonnenes

## Ausblick

# Anhang

## Detaillierter Zeitplan

|  |  |  |  |
| --- | --- | --- | --- |
| Phasen | Stunden | | |
| Analysephase |  |  | 2 |
| 1. Ist-Analyse |  | 1 |  |
| 1. Erstellen des Lastenhefts |  | 1 |  |
| Entwurfsphase |  |  | 4 |
| 1. Datenbank Entwurf |  | 2 |  |
| * 1. Festlegen der Entitäten | 1 |  |  |
| * 1. Datenbank-Modellierung | 1 |  |  |
| 1. Benutzeroberfläche entwerfen und abstimmen |  | 1 |  |
| 1. Erstellen des Pflichtenhefts |  | 1 |  |
| Implementierungsphase |  |  | 13 |
| 1. Erstellen des Datenbankschemas |  | 2 |  |
| 1. Schreiben von Unit-Tests |  | 2 |  |
| 1. Implementieren des C#-Codes |  | 7 |  |
| 3.1 Implementieren der Entitäts-Klassen | 1 |  |  |
| 3.2 Implementieren der Datenzugriffs-Klassen | 2 |  |  |
| 3.3 Implementieren der Validatoren | 1 |  |  |
| 3.4 Implementieren der Geschäftslogik | 2 |  |  |
| 1. Umsetzung der Oberfläche |  | 2 |  |
| Abnahmetest |  |  | 1 |
| 1. Abnahmetest durch das Projektteam |  | 1 |  |
| Dokumentation |  |  | 4 |
| 1. Erstellen der Projektdokumentation |  | 4 |  |
| Gesamt |  |  | 24 |

## Ressourcen

**Hardware:**

* Arbeitsplatz mit Rechner und Bildschirm

**Software:**

* Visual Studio Community 2015 (Entwicklungsumgebung)
* Windows 7 Professional (Betriebssystem)
* Windows 10 (Betriebssystem)
* GitHub (Versionsverwaltung)
* GitHub-Plugin für Visual Studio
* draw.io (Tool für Mockups und ER-Modelle)
* Windows Presentation Foundation (Framework für GUIs unter C#)
* Notepad++ (Editor)
* .NET MySQL-Treiber (Treiber für die Datenbankverbindung)

## Lastenheft

**Muss-Kriterien:**

* Dauerhaftes Speichern von Komponenten mit allen definierten Merkmalen (Arbeitsspeicher, Festplatte, Grafikkarte, Hauptplatine, Monitor, Prozessor)
* Anzeigen von Komponenten in einer Liste
* Bearbeiten von bereits gespeicherten Komponenten
* Löschen von gespeicherten Komponenten
* Grafische Oberfläche

**Soll-Kriterien:**

* Filtern von Komponenten nach definierten Merkmalen
* Umrechnung in andere Einheiten (z.B. Byte -> GB)
* Löschen von mehreren Komponenten gleichzeitig

**Kann-Kriterien:**

* Ansprechende grafische Oberfläche
* Konfigurieren der Datenbankverbindung über die Oberfläche
* Exportieren einer gefilterten Tabelle als CSV-Datei
* Importieren von Tabellen aus einer CSV-Datei

## Schichtenmodell

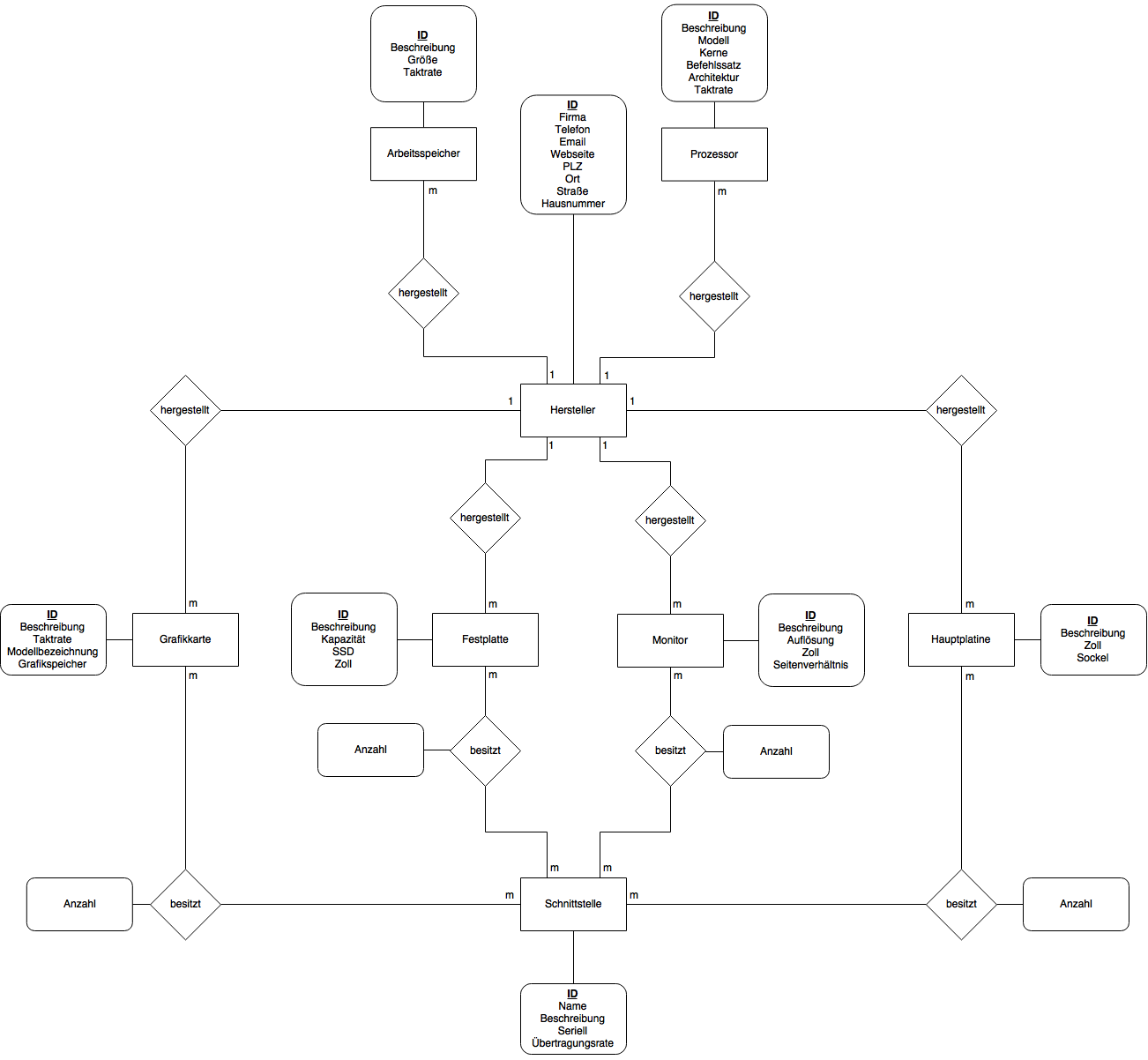


## Oberflächenentwürfe





## ER-Modell



## Pflichtenheft

**Muss-Kriterien:**

* Speichern vom Komponenten in einer relationalen Datenbank (MariaDB)
* Datenbankverbindung mittels MySQL-Treiber für .NET (C#)
* Selektieren von Datensätzen aus der Datenbank mittels SQL
* Anzeigen von Datensätzen in einer Tabelle auf der GUI
* Löschen von gespeicherten Komponenten aus der Datenbank mittels SQL
* Gestalten der grafischen Oberfläche (GUI) mit dem Windows Presentation Foundation Framework (WPF)
* Die Konsistenz der Daten wird durch Validatoren gewährleistet

**Soll-Kriterien:**

* Selektieren von gefilterten Listen aus der relationalen Datenbank mittels SQL
* Schreiben eines Algorithmus für die Umrechnung von Einheiten in C#
* Selektieren von mehreren Einträgen mittels Auswahlboxen auf der WPF-Oberfläche

**Kann-Kriterien:**

* Entwicklung einer ergonomischen GUI mittels Mockups
* Konfigurieren der Datenbankverbindung über ein Konfigurationsfenster in C# mittels WPF
* Exportieren einer gefilterten Tabelle als CSV-Datei mittels File Stream

## Iterationsplan

1. Erstellen des IMS-Projektes
   * Anlegen eines Visual Studio Projektes
   * Erstellen eines Repositories auf GitHub
   * Anlegen einer Verzeichnisstruktur
2. Anlegen einer relationalen Datenbank
   * Anlegen von Tabellen
   * Anlegen von Attributen und Festlegen von Datentypen
   * Erstellen von Datenbank-Regeln
3. Implementieren von Entitätsklassen in C#
   * Implementieren von Klassen für jede Entität
4. Implementieren des Datenbankzugriffes in C#
   * Schreiben von Unit-Tests für die Verbindung mit der Datenbank
   * Implementieren einer allgemeinen Datenbank-Klasse
   * Implementieren von Datenbank-Klassen für jede Entität
5. Implementieren der Validatoren in C#
   * Erstellen von Validierungsregeln
   * Schreiben von Unit-Tests für die Validierung
   * Implementieren von Validatoren für jede Entität
6. Implementieren der Benutzeroberfläche in C# und XAML
   * Erstellen von WPF-Fenstern
   * Implementieren der Hintergrundlogik

## Screenshot der Anwendung