Vorlage.png

# Programm zur Werkzeugverwaltung

## FEUER Powertrain GmbH & Co. KG


| | |
|---|---|
| Name: | Tomahogh |
| Vorname: | Jakob |
| Matrikel-Nr.: | 221201101 |
| Studiengang: | Informatik |
| E-Mail: | jakob.tomahogh@uni-rostock.de |
| Dozent: | Prof. Dr. rer. nat. Clemens H. Cap |
| Lehrstuhl: | Informations- und Kommunikationsdienste |
| Institut: | Informatik |
| Fakultät: | Informatik und Elektrotechnik |
| Modul: | Wahlpflichtbereich Extern (Berufspraktikum) |
| Fachsemester: | 5 |
| Abgabedatum: | |


Praktikumsbericht

Universität Rostock

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# I Die Praktikumsstelle - Formale Bedingungen

Mein Praktikum habe ich bei der Firma FEUER Powertrain GmbH & Co. KG (im Folgenden FPT) absolviert. Die Firma ist ein mittelständisches Unternehmen, das sich auf die Herstellung von einbaufertigen Kurbelwellen für Verbrennungsmotoren spezialisiert hat. FPT ist der größte, unabhängige Hersteller von Kurbelwellen in Europa und beliefert Firmen wie Daimler, Volkswagen Group, BMW und viele weitere. Neben Kurbelwellen für PKWs wird auch für Nutzfahrzeuge, wie LKW und Traktoren, und kleine Schiffe produziert. Die Firma hat ihren Hauptsitz in Nordhausen, Thüringen. Insgesamt beschäftigt die FPT Gruppe etwa 800 Mitarbeiter. Ein weiterer Standort befindet sich in Tunica, MS, USA. [1]

Ich wurde in Nordhausen in der Fertigungs-IT eingesetzt. Die Abteilung ist für die Entwicklung und Wartung von Software zuständig, die in der Produktion eingesetzt wird. Sie kümmert sich auch um die technische Betreuung von IT-Systemen der Produktion, häufig gemeinsam mit der Werksinstandhaltung. Die Abteilung ist relativ klein und hat nur drei feste Mitarbeiter und zwei Werksstudenten. Dort habe ich mich vor allem mit der Praktikumsaufgabe beschäftigt. Aber auch bei anderen Aufgaben, wie die neue Einrichtung von Systemen, habe ich ausgeholfen.

Für FPT habe ich mich entschieden, weil ich mich für die Automobilbranche interessiere und die Firma durch vorherige Ferienarbeit schon kennengelernt habe. Ursprünglich hatte ich mich für einen Praktikumsplatz in den USA beworben, aber aufgrund von internen Umstrukturierungen war ein Praktikum an diesem Standort nicht möglich. Daher wurde mir dieses in Nordhausen angeboten. Die Bewerbung lief über die offizielle Webseite der Firma, dort wurde ich dann an die Personalabteilung weitergeleitet.

Da es sich um ein Pflichtpraktikum handelte, war eine Vergütung nicht vorgeschrieben. Trotzdem habe ich eine Aufwandsentschädigung von 520€ pro Monat erhalten. Es war eine Wochenarbeitszeit von 38 Stunden vorgesehen. Die Kernarbeitszeit war von 9:00 bis 15:00 Uhr. Dazu gab es noch die Möglichkeit, im Homeoffice zu arbeiten, was ich selten genutzt habe. Dabei waren die Arbeitsbedingungen sehr angenehm. Mit einer minutengenauen Zeiterfassung konnte ich meine Arbeitszeiten flexibel gestalten. So konnte ich meine Zeit gut einteilen und auch mal früher gehen, wenn ich Termine hatte. Dazu konnte man Wochen auch mit Überstunden oder zu wenig gearbeiteten Stunden ausgleichen.

[1] Quelle: https://feuer-pt.de/unternehmen/

# II Aufgaben und Tätigkeiten - Der Ablauf des Praktikums

## 2.1 Rahmen des Praktikums

Durch mein Bachelor-Studium Informatik an der Universität Rostock war ein Pflichtpraktikum zur Berufs- und Forschungsorientierung vorgesehen. Während meines Praktikums wurde ich hauptsächlich in der Fertigungs-IT eingesetzt, wo ich an der Entwicklung einer neuen Werkzeugverwaltungssoftware gearbeitet habe. Um diese Software zu realisieren habe ich Hilfestellungen von allen Mitarbeitern des Büros erhalten. Vor allem konnte ich mich immer an meinen Praktikumsbetreuer Dipl. Inf. (FH) Christian Rommel wenden, der als Softwareentwickler in der Firma angestellt ist. Die Spezifikationen zu meinem Programm wurden mir von dem Leiter des Toolmanagement mitgeteilt. Dieser hat die Software auch in regelmäßigen Zwischenschritten getestet und geprüft, ob sie den Anforderungen entspricht. Häufig mussten noch Details verändert und Fehler behoben werden. Seltener musste ein grundlegender Fehler ausgebessert werden. Durch diese Tätigkeiten konnte ich auch hautnah miterleben, mit welchen vielfältigen Aufgaben die Mitarbeiter täglich konfrontiert werden. Zu den Aufgaben gehört die regelmäßige Wartung von Servern, die Sicherstellung der Funktionsfähigkeit von IT-Systemen in der Produktion und die Unterstützung der Benutzer bei technischen Problemen. Ich konnte gute Eindrücke sammeln, wie der Alltag in der Automobilbranche aussehen kann aus der Perspektiver der Informatiker. Neben der Betreuung der bestehenden spielt auch noch die Neuentwicklung von Programmen eine große Rolle. Hier ist die Zielsetzung stets die Lösung eines vorhandenen Problems oder die Vereinfachung von Arbeitsabläufen für die Angestellten.

## 2.2 Aufgabe und Zielsetzung

Die übergeordnete Aufgabe des Praktikums war es einen erstmaligen realistischen Einblick in die Berufs- und Forschungswelt zu werfen. Zu einem Zeitpunkt nach dem Grundstudium, hat man schon einige Fähigkeiten erworben, sodass man einen tatsächlichen Mehrwert für die Firma darstellen konnte.

Meine konkrete Praktikumsaufgabe war die Entwicklung eines Werkzeugverwaltungsprogramms. Das sollte nicht nur zu einer besseren Übersichtlichkeit führen, sondern auch die Anwendung für die Mitarbeiter im Toolmanagement und Einkauf vereinfachen. Insbesondere sollten dabei zahlreiche alte Excel-Tabellen abgelöst werden. Während der Umsetzung dieser Aufgabe konnte ich meine Kenntnisse im Bereich Datenbankverwaltung und Softwareentwicklung anwenden und erweitern. Ein weiteres Ziel für die Software war die Erstellung von Baugruppenlisten und die Ansicht dieser als PDF-Dokument. Gleichzeitig sollte das Programm die Möglichkeit bieten, bereits bestehende Baugruppenlisten direkt aus Excel zu importieren und in das Datenbanksystem zu integrieren. Zusätzlich sollten weitere kleinere Funktionen implementiert werden, auf die im weiteren Verlauf eingegangen wird. Persönlich strebte ich an, meine Programmierfähigkeiten weiterzuentwickeln

und der Firma letztendlich einen konkreten Mehrwert zu bieten. Mein Interesse lag darin, einen Einblick in den Alltag eines Softwareentwicklers zu erhalten und abzuwägen, inwiefern dieser Beruf für mich erstrebenswert ist. Unabhängig von meiner persönlichen Neigung an diesem Beruf, wollte ich die meine grundlegenden Programmierkenntnisse erweitern, welche ich in jedem Feld der Informatik als essentiell erachte.

## 2.3 Tätigkeiten und Arbeitsergebnisse

### 2.3.1 Wahl der Technologien

Nach der Einführung in die Firma und die Abteilung, wurde mir die Aufgabe näher erklärt. Es ging zuerst um das Lesen von Daten aus verschiedenen Excel-Tabellen und die Speicherung in einer Datenbank. Diese sollten später in der Software übersichtlich dargestellt werden. Der erste Schritt dabei war die Erstellung der Datenbank. Da in der Firma größtenteils PostgreSQL verwendet wird, habe ich mich für diese Datenbank entschieden. Zum Herantasten an diese für mich neue Datenbank haben wir zuerst eine Testdatenbank erstellt, die auf dem lokalen Rechner lief. Dazu habe ich die Software pgAdmin verwendet, die eine grafische Benutzeroberfläche für PostgreSQL bietet. Nachdem ich mich in die Grundlagen eingearbeitet habe, ging es darum, zunächst Testdaten in einer Software darzustellen. Mir wurde am Anfang die Wahl zwischen ASP.NET und Windows Forms gelassen. Beide Frameworks sind in der Firma weit verbreitet und werden mit C# programmiert. C# war dabei von Vorteil, da ich aus dem Studium bereits Erfahrung mit Java hatte und diese Sprachen sich sehr ähnlich sind. Da ich noch keinen Bezug zu ASP.NET oder Windows Forms hatte, habe ich mich für anfangs für ersteres entschieden, mit dem Ziel, zumindest die Grundlagen zu erlernen. Dazu habe ich Visual Studio 2022 verwendet, da es für C#-Entwicklung viele nützliche Werkzeuge bietet. Um die Datenbankanbindung zu realisieren, habe ich Entity Framework Core verwendet. Dieses Framework bietet viele nützliche Funktionen, die die Datenbankanbindung vereinfachen. Als Frontend-Technologie habe ich Razor Pages verwendet, da diese weit verbreitet sind und ich mich mithilfe der Dokumentation schnell einarbeiten konnte.

Nachdem ich es erreicht habe, die Daten in dieser Web-App darzustellen, ging es darum, Funktionen einzuführen, um die Daten zu sortieren und zu bearbeiten. Dies ging relativ einfach, weil ASP.NET bereits nützliche Funktionen bietet, die ich verwenden konnte. Jedoch war das darstellen der Daten auf verschiedenen Seiten mit Paging eine kleine Herausforderung. Dies konnte ich dann mithilfe der Microsoft-Dokumentation lösen. Nach ein paar Tagen ausprobieren und testen, habe ich die Software ein erstes Mal dem Toolmanagement vorgestellt. Die Fortschritte in der Programmierung wurden durch unerwartete technische Probleme mit der Webanwendung verzögert. Um einen Double-Wert zu verändern, gab es in dem Feld eine jquery Client-Side Validierung. Diese hat jedoch nach dem amerikanischen Standard den Punkt als Dezimaltrennzeichen verwendet. Da ASP.NET jedoch den deutschen Standard verwendet, hat es das Komma als Dezimaltrennzeichen verwendet. Dies hat dazu geführt, dass das Komma nicht akzeptiert wird und bei Zahlen mit einem Dezimalpunkt dieser Punkt entfernt wurde. Das Problem konnte ich recht schnell identifizieren aber nicht so schnell lösen. Daraufhin hat auch schon eine Besprechung gefolgt. Dort hat das Toolmanagement angemerkt, dass sie ungerne mit einer Web-App arbeiten, da sie es gewohnt sind, mit Excel-Tabellen zu ar-

beiten. Daher habe ich mich entschieden, doch auf Windows Forms zu wechseln. Auch die verbesserte Interoperabilität mit bestehenden Windows Forms Programmen war ein Grund für den Umschwung. Der Anfang war wieder etwas schwierig, aber weil ich von meinem Betreuer ein aktives Projekt und Erklärungen dazu bekommen habe, konnte ich mich einarbeiten. Um hier die Datenbankverbindung zu realisieren, habe ich Npgsql verwendet. Dieser Adapter bietet einfache Möglichkeiten zur Datenbankanbindung und ist auch in der Abteilung bekannt. Daher konnte ich bei Problemen schnell Hilfe erhalten.

Um die Daten aus den Excel-Tabellen zu lesen, habe ich das Framework EPPlus verwendet. Dieses bietet eine einfache Anwendung, um Excel-Tabellen zu lesen. Hier musste ich zuerst eine Tabelle in der Datenbank erstellen, welche die wichtigen Spalten aus allen Excel-Tabellen enthielt. Dann ich mit einem zusätzlich geschriebenem Programm die einzelnen Tabellen nacheinander ausgelesen und jede Zeile einzeln in die Datenbank eingefügt. Hier habe ich wieder mit Npgsql gearbeitet. Nachdem die Daten erfolgreich in der Datenbank gespeichert waren, konnte ich sie für meine anschließenden Arbeiten nutzen.

### 2.3.2 Erster Teil der Software

Um die Daten geordnet darzustellen, habe ich mich für die DataGridView entschieden. Dort habe ich die Daten mittels der DataSource Funktion eingefügt. Alle Funktionen der DataGridView, wie das Sortieren, das Filtern und das Bearbeiten der Daten, habe ich deaktiviert, da diese mit DataSource nicht funktioniert. Daher habe ich diese selbst implementiert. Als erstes habe ich die Bearbeitung von einzelnen Zeilen realisiert. Dazu habe ich ein neues Fenster erstellt, welches man über den „Zeile bearbeiten“ Button öffnen kann. In dieses Fenster werden die Daten aus der ausgewählten Zeile übergeben und dort in die Textfelder eingefügt. Sobald man die Bearbeitung bestätigt, werden die Daten aus den Textfeldern gelesen und gegebenfalls zu Doubles oder Integern geparst. Mithilfe der ID der Zeile, die ich in einer versteckten Spalte gespeichert habe, kann ich die Daten in der Datenbank bearbeiten. Dies habe ich mithilfe von SQL-Queries erreicht. Hier war es sehr hilfreich, dass ich im Studium SQL schon kennengelernt habe. Auch wenn ich noch nicht so viel Erfahrung hatte, kam mir das Wissen sehr zu Gute. Nachdem die Daten bearbeitet wurden, werden diese neu aus der Datenbank geladen und die DataGridView wird aktualisiert. Auch bei jeder anderen Funktion, die die Datenbank verändert, werden die Daten neu abgerufen und in die DataGridView geladen. Dies ist zwar nicht die effizienteste Methode, aber wichtig für die Anwendung, da mehrere Benutzer gleichzeitig in der Software arbeiten können müssen. Hier ist es essentiell, dass die Daten immer aktuell gehalten werden.

Als nächstes habe ich die Suche implementiert. Das war recht herausfordernd, da ich nach jeder einzelnen oder mehreren Spalten gleichzeitig suchen wollte. Zusätzlich sollte die Suche auch nach Teilen der Zeichenkette funktionieren. Auch hatte ich geplant, dass die derzeitige Sortierung nach einer bestimmten Spalte erhalten bleiben soll. Um das zu

erreichen habe ich mit einer einfachen SELECT-Query gearbeitet. Diese habe ich dynamisch erweitert, je nachdem nach welcher Spalte gefiltert werden soll. Dazu habe ich für jede Spalte eine eigene Textbox erstellt, in der der Benutzer den Suchbegriff eingeben kann. Nach dem Klick auf den „Suchen“ Button werden alle Inhalte der Textboxen in einer Datenklasse gespeichert. Diese wird dann an die Funktion übergeben, welche die Datenbankanfrage erstellt. Dort wird für jeden Eintrag in der Datenklasse geprüft, ob ein Suchbegriff vorhanden ist. Wenn ja, wird die Query um den entsprechenden Teil erweitert. Am Ende der Query wird noch die Sortierung angehängt. Diese wird aus der aus den Variablen „string currentSortedColumn“ und „bool ascendingSort“ gelesen. Diese Variablen werden bei jedem Klick auf eine Spaltenüberschrift aktualisiert und sind als globale Variablen deklariert. Nach der Anfrage werden die Daten ausgelesen und wieder in die DataGridView geladen. Später wurde noch der Wunsch geäußert, dass die Suche auch mit Enter bestätigt werden kann. Das habe ich mithilfe von KeyDown-Events erreicht. Dabei habe ich geprüft, ob die Enter-Taste gedrückt wurde und dann die Suchen-Funktion aufgerufen. Ein weiterer Wunsch war, dass in den Spalten von Zahlenwerten auch „<“ und „>“ als Teil der Suche eingegeben werden kann. Nach einiger Recherche habe ich mit regex-Ausdrücken gearbeitet. Die waren zwar am Anfang etwas schwierig, aber mit der Hilfe von regex101.com konnte ich die Ausdrücke recht schnell erstellen und testen. Mit den Ausdrücken habe ich die Suchbegriffe geprüft und die Zahlenwerte aus der Anfrage geparst.

Im weiteren Schritt habe ich habe ich die Fenster erstellt um die Zeilen zu bearbeiten und zu löschen. Das Löschen war recht einfach, da man nur die ID der Zeile benötigt. Trotzdem habe ich die Daten der Zeile in die Datenklasse gespeichert, an das Fenster übergeben und dort dargestellt. Das Löschen wird erst nach einer Bestätigung ausgeführt. Die Bearbeitung war recht ähnlich, nur dass die Textboxen im Fenster bearbeitbar ist. Mit der Bestätigung werden die Daten aus den Textboxen gelesen, gegebenenfalls geparst und mit einer UPDATE Query in die Datenbank geschrieben.

Nach einer weiteren Besprechung mit dem Toolmanagement kam der Wunsch, dass jede Änderung der Daten in einer beliebigen Form gespeichert werden soll. Dazu habe ich eine neue Tabelle in der Datenbank erstellt, in der die veränderte Tabelle, der durchführende Nutzer, die Zeit, die Art der Änderung und den alten und neuen Wert der Zeile erfasst werden. Die Spalten habe ich mit dem Zeichen Thorn (þ) getrennt, weil dieses ein Zeichen aus dem erweiterten ASCII-Code ist und normalerweise nicht in Nutzereingaben vorgefunden wird. Dazu habe ich eine Funktion erstellt, welche die Daten aus der DataGridView ausliest und zu einem String konkateniert mit dem Thorn-Zeichen jeweils getrennt. Danach wird zuerst die vom Nutzer gewünschte Operation durchgeführt. Es werden die neuen Daten aus der Datenbank abgefragt und auch zu einem String konkateniert. Diese Strings dienen als alter und neuer Wert in der Log-Tabelle. Der Name der Operation wird jeweils übergeben, der Nutzer wird aus dem Windows-Login ausgelesen und als Zeit wird die aktuelle Systemzeit verwendet.

Später habe ich weitere Excel-Tabellen erhalten, die ich noch in die Datenbank einfügen sollte. Dazu musste ich die Spalten der Datenbank erweitern und auf die Wünsche des Toolmanagements anpassen. Des Weiteren mussten im Programm auch die Spalten angepasst werden, sowie die Datenklasse. Das war recht aufwendig, da unter anderem die Datenbankanfragen der Funktionen angepasst werden mussten. Auch die Suche musste angepasst werden und zusätzliche Textboxen für die neuen Spalten erstellt werden. Die Log-Tabelle war von den Änderungen nicht betroffen, da ich die Spalten dynamisch auslesen konnte.

Im nächsten Schritt habe ich Hyperlinks aus den Excel-Tabellen mit in die Datenbank geschrieben. Diese Links sollten auf PDF-Dokumente zu den einzelnen Datensätzen führen. Es stellte sich heraus, dass die Datengrundlage der Links fehlerhaft war. Die Links verwiesen auf lokale Dateien, die nicht mehr existierten oder nicht zugänglich waren. Es fehlten Zugriffsberechtigungen auf die Netzlaufwerke, auf denen die Dateien gespeichert waren. Deswegen war die Überprüfung nicht möglich. Später wurden alle Dateien in ein Verzeichnis auf einem anderen Netzlaufwerk aufgrund von einem Serverumzug verschoben. Dort habe ich von der Büro-IT die Berechtigung erhalten, auf das Verzeichnis zuzugreifen. Allerdings waren nun alle Hyperlinks aus den Excel-Tabellen ungültig. Daher habe ich beim Auslesen für jedes Material geprüft, ob ein gleichnamiges Dokument in dem vorgegebene Verzeichnis existiert. War dies der Fall, habe ich den Pfad ausgelesen und in eine neue Spalte in der Datenbank geschrieben. Um nun dem Nutzer erkenntlich zu machen, dass zu einem Material ein Dokument existiert, habe ich das Ereignis „CellFormatting“ der DataGridView verwendet. Bei verfügbarem, verlinktem Dokument, habe ich die Schriftfarbe des Materials blau gefärbt und die Schrift unterstrichen. Damit der Nutzer die PDF öffnet, muss dieser doppelt auf die Zelle des Materials klicken. Dazu habe ich das Ereignis „CellDoubleClick“ verwendet. Dort habe ich den Pfad aus der versteckten Spalte ausgelesen und das Dokument geöffnet. In einer Vorführung des Programms kam der Wunsch auf, eine Vorschau von PDF-Dateien zu haben, ähnlich wie im Windows Datei-Explorer. Um das zu erreichen habe ich ein „WebBrowser“-Element in das Fenster eingefügt. Bei einem einfachen Klick auf die Zeile wird die Vorschau des Dokuments im WebBrowser-Element angezeigt. Dazu habe ich das Ereignis „CellClick“ verwendet.

Ein größeres Problem der Software war, dass das Scrollen in der DataGridView aufgrund der großen Datenmenge sehr langsam war. Um das zu beheben habe ich Double Buffering aktiviert. Dies ist eine Technik zur Vermeidung vom Flackern der Anzeige in grafischen Benutzeroberflächen. Dazu kann es die Leistung und die visuelle Stabilität verbessern. Bei dem Double Buffering werden jegliche Änderungen an der Anzeige nicht direkt auf dem Bildschirm, sondern in einem Zwischenspeicher, dem Buffer, vorgenommen. Anschließend werden die Änderungen gesammelt auf einmal auf den Bildschirm übertragen. Dadurch war das Scrollen visuell flüssiger. Nun kam die Anfrage, dass Materialnummern einzigartig sein sollen. Dazu habe ich eine Funktion erstellt, welche überprüft, ob diese Materialnummer bereits existiert. War dies der Fall, wird eine Fehlermeldung angezeigt

und die Daten werden nicht in die Datenbank geschrieben. Diese Funktion rufe ich auf, bevor eine neue Zeile erstellt oder eine bestehende bearbeitet wird.

Eine der wichtigsten Teile war es, die Daten der Materialien mit den Bestandsdaten aus dem SAP zu verknüpfen. Um das zu realisieren, habe ich zuerst eine eigene Tabelle für die SAP-Daten erstellt. Ein Problem bestand aber bei den SAP-Daten, da die Materialnummer teilweise auch in der Spalte für die SAP-Nummer stand. Da ich die Daten aus dem SAP nur lesen kann, musste ich beim Auslesen die Daten in die richtigen Spalten schreiben. Am Anfang habe ich das in der Weise probiert, dass für jeden Eintrag die Materialnummer oder die SAP-Nummer geprüft mit der Materialnummer mit meiner vorhandenen Datenbank abgeglichen wird. Ich habe schnell bemerkt, dass diese Methode sehr langsam ist. Da ich auch die Daten aktuell halten muss, werden diese bei jedem Programmstart neu eingelesen. Also musste ich eine andere Methode finden. Dies habe ich mithilfe von LinQ erreicht. Dazu habe ich alle Daten aus dem SAP in eine Liste geladen. Gleichzeitig habe ich die meine vorhandenen Daten auch in einer Liste gespeichert und diese dann miteinander verglichen und eine neue Liste erstellt. Hat die Materialnummer der SAP-Daten meiner vorhandenen Materialnummer entsprochen, habe ich den Eintrag nicht geändert. Hat die SAP-Nummer meiner Materialnummer entsprochen, habe die SAP-Nummer in die Spalte „Materialnummer“ geschrieben. War keine Übereinstimmung vorhanden, wurde der Eintrag übersprungen. Bei dem Übertragen der Daten in die Datenbank, habe ich gemerkt, dass es bei Schleifen mit vielen Einträgen sehr langsam ist, die Daten einzeln zu schreiben. Deswegen habe ich mich für eine Massenaktualisierung entschieden mit dem NpsgqlCopyHelper. Damit habe ich einen erheblichen Geschwindigkeitsvorteil erreicht. Zusätzlich habe ich die Initialisierung meiner SAP-Daten mit einem background-Worker realisiert. Dieser läuft im Hintergrund beim Laden des Menüs. Sobald der Prozess abgeschlossen ist, darf der Nutzer auf die verschiedenen Tabellen zugreifen. Um nun in der DataGridView die Bestandsdaten anzuzeigen, habe ich die SQL-Query mit einem LEFT-JOIN erweitert. In dieser Form werden alle Daten zu den Materialien angezeigt und zusätzlich die Bestandsdaten, falls vorhanden.

Als Letztes musste noch eine Benutzerverwaltung implementiert werden. Dazu habe ich eine neue Tabelle in der Datenbank erstellt, in der die Benutzer und ihre Rechte gespeichert werden. Die jeweiligen Rechte werden als Bool gespeichert und jeweils beim Start von der Software ausgelesen. Damit auch Nutzer die Rechte für die Benutzerverwaltung haben, habe ich ein neues Fenster erstellt. In diesem ist eine DataGridView mit allen Nutzern und deren Rechten. Nur Benutzer mit der nötigen Berechtigung können dies einsehen und bearbeiten.

Damit war die erste Version fertig. Nach einer Vorführung der Software wurde mir mitgeteilt, dass diese in die Testphase übergehen kann. Dafür habe ich die Datenbank auf einen Server der Firma übertragen und die Software in ein öffentliches Verzeichnis auf dem Server veröffentlicht. Insgesamt hat dieser Teil der Entwicklung etwa zwei und einen halben Monat gedauert. Anschließend an die Veröffentlichung kam es noch gehäuft

zu Feedback und Fehlermeldungen. Nach einigen Tagen an Fehlerbehebungen und Anpassungen kam es zu kaum weiteren Meldungen. Die Software war nun bereit für den produktiven Einsatz.

### 2.3.3 Erweiterung der Software

Das nächste Ziel der Software war es, Baugruppenlisten in darzustellen. Diese Listen bestehen aus einzelnen Werkzeugen, mit zugeordneten Materialien. Diese Materialien werden benötigt, um ein Werkzeug zusammenzubauen. Eine Baugruppe stellt ein Werkzeug mit den benötigten Materialien dar. Die zusammengefassten Baugruppen stellen eine Liste von Werkzeugen dar, welche für eine Maschine an einem Bearbeitungsschritt benötigt werden. Die Baugruppenlisten werden in Excel-Tabellen geführt und sollten für die bessere Bedienbarkeit in die Software übertragen werden. Die alte Variante kann in der Abbildung **??** gesehen werden. Dort hat man lediglich mit den möglichen Mitteln aus Excel versucht, die Daten übersichtlich darzustellen. Um nun die Darstellung zu verbessern, hatte ich anfangs den Plan, das Layout der Tabellen zu kopieren und in DataGridViews umzusetzen. Dazu ein treeView, welches das Dateisystem darstellt. So müssten die Nutzer keine große Umstellung erfahren, sondern können im gewohnten Stil in der neuen Software weiter arbeiten. Um das Dateisystem zu speichern habe ich eine eigene Tabelle angelegt mit den Werten ID, parentNode, nodeText und dem bool „file“. Dieser Bool bestimmt, ob es sich hierbei um ein Ordner handelt oder um eine Datei. Dies war später wichtig, damit von Baugruppenlisten keine weiteren Kind-Elemente erzeugt werden konnten. Um nun den treeView aus der Datenbank zu laden, habe ich mit Rekursion gearbeitet. Dabei habe ich zuerst mit den rootNodes angefangen und jeweils alle Knoten mit dem rootNode als parentNode geladen. Als ich alle Knoten aus der Datenbank abgerufen habe, wurden alle in das treeView geschrieben. Für die bessere Übersichtlichkeit habe ich alle Knoten, die einen Ordner darstellen ein Ordner-Icon hinzugefügt. Dies war auch wichtig, um zu bestimmen, ob eine Datei oder ein Ordner geklickt wurde. Bei einer Datei wurde versucht, eine Baugruppenliste mit dem zugehörigen Namen zu öffnen. Dies würde funktionieren, da als Vorraussetzung gegeben war, dass alle Baugruppenlisten einen einzigartigen Namen haben. Doch das wäre zu unübersichtlich für den Nutzer mit solch langen Dateinamen. Dazu würde es einige NULL-Werte in der Datenbank geben, wenn die Tabelle für die Baugruppenliste aus der Excel im gleichen Schema übernommen würde. Daher habe ich durch Rücksprache mit meinem Betreuer eine neue Lösung gefunden.

Dazu hab ich zuerst das Datenbankschema überarbeitet:
Hier habe ich, wie im Schema **??** zu sehen, drei Tabellen erstellt. In der Baugruppenliste wird der Pfad abgespeichert, unterteilt in Kunde, Produkt, Afo-Bezeichnung, WerkStep und Maschine. Der Pfad für das treeView wird als Name der Baugruppenliste dann automatisch zusammengesetzt. Zusätzlich wird alles hier gespeichert, was für die gesamt Liste gütlig ist, wie Bemerkung, Antragsteller oder Genehmigung. Die Tabelle der Werkzeuge

hat als Fremdschlüssel die BaugruppenlistenID. In der gleichen Form hat die Materialtabelle die WerkzeugID als Fremdschlüssel. Um das neue System ordentlich darzustellen, habe ich mit zwei DataGridViews gearbeitet. Das erste für die Werkzeuge. Bei einem Klick auf ein Werkzeug werden die dazugehörigen Materialien in der zweiten DataGridView geladen. So ist eine kompakte Übersicht gegeben und die NULL-Werte werden erheblich reduziert. Nachdem die Ansicht erstellt wurde, habe ich wieder die Funktionen zum bearbeiten der Listen implementiert. Da ich das vorher angeeignete Wissen nur übertragen musste, ging das recht schnell. Beim Abrufen der Materialliste hab ich mit einer JOIN-Query gearbeitet, um die Bestandsdaten aus dem SAP mit in der Liste anzuzeigen. Mit dieser Datenbankstrukture konnte ich die Baugruppenlisten übersichtlich darstellen und bearbeiten. In Abbildung **??** ist die neue Ansicht zu sehen. Hier ist zu erkennen, dass die Materialien direkt den Werkzeugen zugeordnet sind und keine Leeren Zeilen mehr benötigt werden. Dadurch ist die Übersichtlichkeit deutlich verbessert.

Circa zu diesem Zeitpunkt hatte ich eine ungewollte Pause durch eine Krankheit. Hier war es die Schwierigkeit, sich wieder in das Programm einzuarbeiten. Aber da das Programm vollständig von mir geschrieben wurde, war das recht zügig. Die nächste größere Aufgabe die ich mir gesetzt habe, war ein Rechtsklick-Menü für das treeView zu implementieren. Die Funktionen sollten „Umbenennen“, „Löschen“, „Kopieren“ und „Einfügen“ sein. Mit dem Umbenennen von Knoten habe ich angefangen. Ich habe die Fälle voneinander getrennt, auf welchem Level des treeViews der Knoten ausgewählt war. Zusammen mit der Angabe des alten Pfades konnte ich genau den ausgewählten Knoten umbenennen. Da bei es mehrere Listen mit beispielsweise dem Kunden „Daimler“ gibt, aber nur einmal dieser Knoten angezeigt werden, ist es wichtig die Knoten mit Angabe des Pfades zu verändern. Die weiteren Funktionen sind komplexer, aber recht ähnlich aufgebaut. Um einen Ordner oder eine Datei zu löschen, ist es wichtig, dass auch alles zugehörigen Materialien und Werkzeuge gelöscht werden. Ich habe hier jeweils mit foreach-Schleifen gearbeitet. Zuerst habe ich alle zugehörigen Baugruppenliste zu dem ausgewählten Knoten aus der Datenbank abgerufen. Für jede Baugruppe dann jedes Werkzeug und für jedes Werkzeug habe ich jedes Material gelöscht. Danach habe ich das Werkzeug gelöscht und schließlich die Baugruppenliste. Das Kopieren war in der Hinsicht ähnlich, dass ich auch mit foreach-Schleifen alle Baugruppenlisten, Werkzeuge und Materialien aus der Datenbank geladen habe. Diese habe ich dann aber als Listen in globalen Variablen gespeichert, auf die beim Einfügen wieder zugegriffen werden kann. Für das Einfügen mussten zuerst die Baugruppenliste in die Datenbank geschrieben werden. Als Ergebnis wurde hier die neue ID zurückgegeben. Diese wird bei den Werkzeugen als Fremdschlüssel verwendet und daher ist es wichtig, dass diese auch verändert wird. Das gleiche Prozedere läuft mit dem Einfügen der Werkzeuge und den dazugehörigen Materialien ab. Hier war ein größeres Problem, dass ich Anfangs den Namen der Listen nicht verändert habe. Somit konnte es vorkommen, dass der Pfad exakt gleich mit den originalen Listen war und daher die Kopierten nicht angezeigt wurden. Deshalb hab ich präventiv bei jeder kopierten Liste an das

Ende des Pfades, also Maschine, ein „- Kopie“ angefügt. Dennoch konnte es vorkommen das zwei Listen die gleichzeitig kopiert werden auf den gleichen Namen enden. Der Grund ist, dass so lange ein „- Kopie“ angefügt wird, bis keine Kollision mehr erkannt wird. Allerdings werden die kopierten Listen nicht gegenseitig beachtet. Um das zu lösen, habe ich einen Zähler deklariert, der jeder kopierten Liste zusätzlich eine eigene Zahl gibt. Somit kann es keine Kollisionen mehr unter den kopierten Listen oder zwischen bestehenden und kopierten Listen entstehen.

Später hab ich ein Logo für das Programm erstellt, damit dies in einem bestehenden Programm eingebunden werden kann. Dabei hat mir die Bildgenerierung vom Bing-Chat geholfen. Das generierte Bild habe ich mit GIMP an meine Vorstellungen angepasst.

Nun musste noch die Möglichkeit geschaffen werden, Baugruppenlisten zu genehmigen. Das habe ich mit einer neuen Berechtigung erreicht. Dazu hatte ich in der Tabelle für die Baugruppenliste auch schon Spalten für den Nutzer und das Datum der Freigabe. Um dem Vorgesetzten die Arbeit zu erleichtern, kann dieser die Liste nach Baugruppelisten filtern, welche noch nicht genehmigt wurden. Diese können dann mit einem Klick auf einen für normale Nutzer nicht sichtbaren Button freigegeben werden.

In diesem Teil der Software war es wichtig, dass die Nutzer auch die Möglichkeit haben, bestehende Excel-Tabellen in die Datenbank zu importieren. Hier ist es wichtig, dass die Excel-Tabellen nach einer bestimmten Struktur aufgebaut sind. Diese Struktur habe ich vom Toolmanagement erhalten und diese war auch schon großteils angewandt. Das Einlesen der Tabellen habe ich mit dem EPPlus-Framework realisiert. Zuerst habe ich die wichtigen Daten aus den Kopfzeilen ausgelesen. Diese habe ich in die Tabelle für die Baugruppenlisten geschrieben. Als nächstes durchläuft das Programm durch die einzelnen Spalten und prüft nach bekannten Spaltenbezeichnungen. Der Spaltenindex wird in einer Datenklasse gespeichert und beim Einlesen der Daten aus der Excel-Datei verwendet. Die Daten werden nach der Vorlage aus der Excel-Datei ausgelesen und in die Datenbank geschrieben. Es ist wichtig zu beachten, dass die Werkzeuge und Materialien immer den richtigen Fremdschlüssel erhalten. Ich habe das so erreicht, dass ich beim Einfügen der Baugruppenliste und der Werkzeuge jeweils die automatisch generierte ID zurückgeben lasse. Diese wird dann beim Einfügen der jeweils Werkzeuge und Materialien als Fremdschlüssel verwendet. Damit der Nutzer möglichst genaue Fehlermeldungen erhält, habe ich ausführlich getestet, welche Fehler auftreten können und diese nach Möglichkeit genau angezeigt. Die Fehler beinhalten beispielsweise, dass die Excel-Datei nicht geöffnet werden konnte oder dass die Bezeichnung der Baugruppenliste nicht nach den Vorgaben ist oder bereits existiert.

Ein weiterer wichtiger Teil war es auch, die Baugruppenlisten als PDF zu exportieren. Dafür habe ich die iText7-Bibliothek verwendet. Angefangen habe ich damit die Kopfzeilen zu erstellen. Diese habe ich mit einer Tabelle realisiert, wobei ich die Zeilen einzeln erstellt und formatiert habe. Dazu habe ich die Zellenränder ausgeblendet. Generell habe ich mich bei dem Design an den PDF Ausdrucken der Excel-Tabellen orientiert. Die Daten

habe ich wieder mit foreach-Schleifen ausgelesen und entsprechend als Zeile in die Tabelle geschrieben. Sobald die Tabelle fertig war, habe ich noch die Felder für die Unterschriften der Nutzer und das Datum hinzugefügt, sowie Bemerkungen. Als Fußzeile habe ich die Seitenzahl hinzugefügt. Das stellte ein Problem dar, weil man die Seitenzahl vor dem erstellen der PDF nicht kennt. Daher habe ich die PDF einmal erstellt und die Seitenzahl ausgelesen. Dann habe ich die PDF nochmal erstellt und die Seitenzahl hinzugefügt. Das ist zwar nicht die effizienteste Methode, aber sie ist für die Anwendung ausreichend, da die PDFs nicht in großen Mengen erstellt werden. Die PDFs werden in unter C:/temp/ gespeichert. Falls dieser Ordner nicht existiert, wird er erstellt. Wenn eine PDF mit dem gleichen Namen bereits existiert, wird diese überschrieben, es sei denn diese ist geöffnet. Dann wird eine Fehlermeldung angezeigt.

Nun sollten für eine geplante Produktionszahl die noch benötigten Materialien benötigt werden. Dazu mussten Bestandszahlen aus dem SAP vorhanden sein. Sofern auch andere Zahlen wie die Stückzahl des Material pro Werkzeug und die Anzahl der Schneiden und eventuell des Nachschliffs vorhanden sind, kann die Software die benötigten Materialien berechnet. Es ist wichtig hier den Bestand zwischen den Standorten Nordhausen und Beuren zu unterscheiden. Um das zu erreichen, habe ich zwei Spalten in der Materialtabelle erstellt. Eine für die benötigte Menge in Nordhausen und eine für Beuren. Die Berechnung habe ich in einer eigenen Klasse realisiert.

Im letzten Schritt war gewünscht, dass jede Baugruppenliste bei einer Änderung automatisch archiviert wird. Dafür habe ich die Klassen für das Kopieren und Einfügen wiederverwendet. Allerdings wird ein Archivierungsdatum gesetzt, um die Baugruppenliste von den Aktuellen zu unterscheiden. Sobald die Archivierung abgeschlossen ist, wird die gewünschte Änderung durchgeführt. Bei dem Löschen einer Liste ist das ganze etwas einfacher, hier muss nur das Archivierungsdatum gesetzt werden. In dem treeView werden nur nicht archivierte Listen angezeigt. Diese können dann mit einer Checkbox angezeigt werden. Bei der Ansicht von archivierten Listen ist keine Bearbeitung möglich. Man kann aber die Listen löschen, sofern man die nötigen Rechte hat.

### 2.3.4 Weiterer Ausblick

Nachdem das Programm in der zweiten Version fertiggestellt war, wurde es für den Einsatz in der Firma freigegeben. Dabei gab es noch einige Fehler und Anpassungen, die ich nach und nach behoben habe. Die nächste Ausbaustufe der Software wurde nicht mehr angefangen, da es in der verbleibenden Zeit nicht fertiggestellt werden konnte. Daher habe ich vor allem das vorhandene Programm verbessert und Fehler behoben. In der Zukunft sollen jedoch noch weitere Funktionen hinzugefügt werden. Unter anderem soll die Software mit einer bestehenden Software verbunden sein, welche die Werkzeuge erfasst, die in die Produktion gehen oder zurückkommen. Dazu soll automatisch die Differenz zwischen der Standzeitvorgabe und der tatsächlichen Standzeit berechnet werden. Zusätzlich muss

angegeben werden, auf welcher Maschine das Werkzeug eingesetzt war. Damit kann man dann mögliche Fehlerquellen auf einzelnen Maschinen erkennen. Auch soll die Software mit dem SAP verbunden werden. Hier sollen die Bestandszahlen automatisch aktualisiert werden, sobald ein Werkzeug oder Material eingesetzt wird.

### 2.3.5 Nebenaufgaben

Neben der Entwicklung hatte ich auch andere Aufgaben zum Aushelfen in der Abteilung. So habe ich beispielsweise die Aufgabe bekommen, Computer aufzubauen, die für die Produktion benötigt werden. Nach dem Aufbau musste ich noch die benötigte Software installieren und konfigurieren. Eine weitere Aufgabe war es zusammen mit einigen Kollegen, das eigene Lager aufzuräumen. Hier war vor allem viel altes Büromaterial, welches an andere Abteilungen verteilt werden konnte. An einem Tag wurde ich beauftragt, mit einem Dienstfahrzeug zu einem Kunden in Kölleda zu fahren. Dort musste ich gefertigte Teile abholen. Diese mussten in Nordhausen nocheinmal überprüft werden. Dazu habe ich eine Mercedes V-Class erhalten. Dies war eine sehr interessante Erfahrung, da ich noch kein so großes Fahrzeug gefahren bin. Auch war es das erste Mal, dass ich alleine mit einem Firmenfahrzeug unterwegs war. Dazu musste ich das Fahrtenbuch führen und die Tankquittungen aufbewahren. Insgesamt hat die Dienstreise etwa 4 Stunden gedauert. Des Weiteren habe ich noch Datalogic Scanner konfiguriert, welche aus der Reparatur zurückkamen. Diese mussten mit einer Station gekoppelt werden. Dies habe ich zum ersten Mal gemacht und war eine interessante Erfahrung. Zum Einsatz kam die Software Äladdin"von Datalogic.

## 2.4 Fachspezifische Reflexion über das Praktikum

FPT ist ein mittelständisches Unternehmen in der Automobilbranche, das sich größtenteils mit Produktions- und Verwaltungsaufgaben beschäftigt. Die IT-Abteilung umfasst nur wenige Mitarbeiter. Daher ergibt sich die Chance eine Softwarelösung für fachfremde Personen zu entwickeln. Es hat sich herrausgestellt, dass vor allem die klare Kommunikation unglaublich wichtig ist.

Die Tätigkeit war sehr abwechslungsreich. Man hat viele verschiedene Aufgaben bekommen und konnte genau erleben, wie der Tagesablauf in einer IT-Abteilung ist. Von der Entwicklung einer Software, über das Konfigurieren von Hardware und technische Unterstützung gab es ein breites Feld von Aufgaben.

Wie zu erwarten war, gabe es viele Unterschiede zu reinen Softwareunternehmen oder Forschungsinstituten. So waren die Prioritäten anders gesetzt. Beispielsweise hatte die Neuentwicklung von Software zwar schon einen hohen Stellenwert, aber in der Praxis war es wichtiger, dass die Produktion reibungslos läuft. Somit mussten viele Aufgaben, die nicht direkt mit der Softwareentwicklung zu tun hatten, erledigt werden. Auch die Betreuung von alter Software war ein wichtiger Bestandteil.

Das Praktikum hat einiges an Erfahrung geboten, welche in dem Studium nur theoretisch vermittelt werden kann. Allerdings gab es einen Unterschied zwischen dem wissenschaftlichen Arbeiten an der Universität und dem praktischen Arbeiten. Beispielsweise gab es keine Anfertigung von Pflichten- oder Lastenheften, da alle „Auftraggeber“ Mitarbeiter von FPT waren. Auch gab es keine Anforderungsanalyse, da die Anforderungen direkt von den Mitarbeitern kamen.

# III Fazit und Bewertung

Insgesamt war das Praktikum eine sehr interessante und lehrreiche Erfahrung. Es hat mir gezeigt, wie das Tagesgeschäft eines Informatikers aussehen kann. Die Tätigkeit war sehr abwechslungsreich und hat mir gezeigt, dass auch nach dem Studium noch viele neue Methoden und Technologien zu lernen sind. Auch konnte ich erstmals Erfahrungen sammeln, wie es ist in der Praxis zu arbeiten. Natürlich wird sich die Praxis auch von anderen Wirtschaftssektoren oder Forschungseinrichtungen unterscheiden. Dennoch konnte ich erfahren, wie sich die Zusammenarbeit mit Kollegen gestaltet. Persönlich habe ich diese Zusammenarbeit als sehr angenehm und vor allem hilfreich empfunden. Die Kollegen waren stets bereit zu helfen und haben mir viele Tipps und Tricks gezeigt. Außerdem haben sie mir gelehrt, dass es besser ist zu viele Fragen zu stellen, als dass das Projekt scheitert. Trotzdem haben sie mir auch die Möglichkeit gegeben, selbstständig zu arbeiten und eigene Nachforschungen anzustellen. Hier konnte ich einige Probleme nach eigener Recherche lösen. Dazu konnte ich unter Anderem eigenes Wissen aus dem Studium anwenden. Zum Beispiel habe ich die Datenbankanfragen und die Datenbankstruktur selbst erstellt. Es hat sehr geholfen, dass in der Vorlesung „Datenbanken“ diese Themen schon behandelt wurden. Auch die Programmierung in C# war mir nicht fremd, da ich in der Vorlesung „Algorithmen und Datenstrukturen“ schon mit Java gearbeitet habe. Die Umstellung von Java zu C# war nicht sehr schwer, da die Syntax sehr ähnlich ist. Die generelle Abläufe der Softwareentwicklung waren mir auch nicht fremd, da ich in der Vorlesung „Softwaretechnik“ schon mit den Phasen der Softwareentwicklung vertraut gemacht wurde. In der Praxis wurden die Phasen nicht streng eingehalten, wie es in der Theorie gelehrt wird. So gab es beispielsweise keine Anforderungsanalyse oder Pflichtenhefte. Die Anforderungen kamen direkt von den Mitarbeitern. Auch gab es keine klare Trennung der Phasen. Beispielsweise wurde die Implementierung und das Testen parallel durchgeführt. Teilweise war ich mit der Fehlerbehebung von dem ersten Teilprogramm beschäftigt, während ich schon mit der Neuentwicklung des zweiten Teils angefangen hatte.

Im Praktikum konnte ich vor allem die Kommunikation mit Kunden und Kollegen lernen. Solche Erfahrungen können im Studium nicht vermittelt werden. Auch die Tatsache, dass die Software für fachfremde Personen entwickelt wird, muss bei der Entwicklung berücksichtigt werden. Dazu konnte ich innerbetriebliche Abläufe kennenlernen und wie die Software in diese Abläufe integriert wird. Im Studium wird vor allem Wert auf die Theorie und wissenschaftliche Arbeit gelegt. In der Praxis legt man mehr Wert auf die Funktionalität und die Benutzerfreundlichkeit. Auch die Wartung und Pflege von alter Software ist ein wichtiger Bestandteil. Trotzdem finde ich es essentiell, dass die Theorie nicht vernachlässigt wird. Die Theorie ist die Grundlage für die Praxis. Meiner Meinung nach sollte man die Grundlagen in der Theorie lernen und verstehen. Damit kann man in der Praxis ein weiteres Verständnis entwickeln und viele verschiedene Methoden und Technologien schnell anwenden.

# IV Abbildungen

# V Anhang

- Kopie des Praktikumszeugnisses und ggf. Kopie von Arbeitsproben
- Bestätigung der Teilnahme am Praktikum mit Stempel und Unterschrift der Institution.

# VI Eidesstattliche Versicherung

Ich versichere eidesstattlich durch eigenhändige Unterschrift, dass ich die Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen sind, habe ich als solche kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht und ist in gleicher oder ähnlicher Weise noch nicht als Studienleistung zur Anerkennung oder Bewertung vorgelegt worden. Ich weiß, dass bei Abgabe einer falschen Versicherung die Prüfung als nicht bestanden zu gelten hat.

Rostock

| (Abgabedatum) | (Vollständige Unterschrift) |
|---|---|