

Abschlussprüfung Winter 2022/2023
Fachinformatikerin für Anwendungsentwicklung
 Projektdokumentation



Anwendung zur Erstellung, Bearbeitung und Speicherung von
Konfigurationsdateien

Abgabetermin 23.11.2022

Prüfling:

Lisa Hentschel

Azubinummer: 0000943489

Marburger Str.20

35469 Allendorf (Lumda)



IAD GmbH
Ausbildungsbetrieb (schulisch)
Neue Kasseler Str. 62E
35039 Marburg



MK Versuchsanlagen und Laborbedarf e.K.
Ausbildungsbetrieb (betrieblich)
Stückweg 10
35325 Mücke

Inhaltsverzeichnis

Abbildungsverzeichnis	iv
Tabellenverzeichnis	v
Abkürzungsverzeichnis	vi
1 Beschreibung des Projektes	1
1.1 Betriebsbeschreibung.....	1
1.2 Projektziel	1
1.3 Projektumfeld.....	2
2 Ist-Analyse	2
2.1 Ausgangssituation	2
2.2 Kostenrechnung Ist-Zustand	3
3 Soll-Analyse	3
3.1 Funktionale Anforderungen	3
3.2 Nicht Funktionale Anforderungen.....	3
3.3 Wirtschaftlichkeitsrechnung Soll-Zustand.....	4
4 Projektablauf	4
4.1 Zeitplanung.....	4
4.2 Planung.....	4
4.3 UML Diagramme.....	7
4.4 Planung Corporate Design.....	8
4.5 Implementierung GUI.....	9
4.6 Implementierung Datenbank	9
4.7 Implementierung Geschäftslogik	10
4.8 Testphase	11
4.9 V-Modell	11
4.10 Soll-Ist-Vergleich.....	12
5 Zeitplanung.....	12
5.1 Detaillierte Zeitstruktur	12
5.2 Analyse der Zeitplanung.....	13
6 Quellcodeauszüge	13

6.1	Beispiel Datenbankabfrage	13
6.2	Umwandlung der Daten in Konfigurationsdatei	13
6.3	Style Beispiel GUI.....	14
7	Beschreibung der Anwendung	15
7.1	Ablauf der Anwendung.....	15
7.2	Usability	15
8	Reflexion.....	15
8.1	Lessons Learned	15
8.2	Ausblick.....	15
8.3	Fazit	15
Quellenangabe		vii
Anhang		viii
Kurzanleitung.....		viii
Lastenheft.....		ix
Beispiele Corporate Design		xv

Abbildungsverzeichnis

Abbildung 1	Beispiel Handschuharbeitsplatz	1
Abbildung 2	Datenbankentwurf	5
Abbildung 3	MVVM-Prinzip	6
Abbildung 4	Beispiel unverschlüsselte Konfigurationsdatei	6
Abbildung 5	Aktivitätsdiagramm	7
Abbildung 6	Use Case Diagramm	8
Abbildung 7	GUI der Anwendung	8
Abbildung 8	Beispiel Bindings	9
Abbildung 9	WindowChrome	9
Abbildung 10	Beispiel Property Methode	10
Abbildung 11	Beispiel Command Methode	10
Abbildung 12	Gliederung des V-Modells	11
Abbildung 13	Codebeispiel Datenbankabfrage	13
Abbildung 14	Codebeispiel XML Umwandlung	14
Abbildung 15	Stylebeispiel XAML	14
Abbildung 16	Lastenheft	ix
Abbildung 17	Vergleich Corporate Design	xv

Tabellenverzeichnis

Tabelle 1 Kostenrechnung Ist-Zustand	3
Tabelle 2 Wirtschaftlichkeitsrechnung.....	4
Tabelle 3 Zeitplanung	4
Tabelle 4 Zeitstruktur	12

Abkürzungsverzeichnis

GUI	Graphical User Interface
WPF	Windows Presentation Foundation
MVVM	Model-ViewModel-View Prinzip
XML/XAML	Extensible (Application) Markup Language
GITS	Glove Integrity Test System

1 Beschreibung des Projektes

1.1 Betriebsbeschreibung

MK-Versuchsanlagen ist eine international agierende Firma mit Sitz in Mücke. Hier wurden innerhalb eines Betriebspraktikums praktische Kenntnisse erworben, die die Autorin zum Erarbeiten des Projektes befähigte, ebenso wurde durch die Firma MK-Versuchsanlagen das Projekt beauftragt und dessen Ziele festgelegt. MK-Versuchsanlagen entwickelt, vertreibt und wartet unter anderem Handschuhprüfgeräte für die Pharmaindustrie, sowie die dazugehörige Software. Diese Software nennt sich GITS.



Abbildung 1 Beispiel Handschuharbeitsplatz

Mit den Prüfgeräten, die auf die Öffnung gesetzt werden, wird regelmäßig die Dichtigkeit der Handschuhe, unter bestimmten Parametern, geprüft und dokumentiert. Die Software GITS stellt eine Benutzeroberfläche, sowie einige Tools bereit. Mit diesen Tools kann man beispielsweise einzelne Prüfgeräte direkt ansprechen oder Rezepturen für Testkriterien anpassen. Die Software baut eine Verbindung zur Datenbank des jeweiligen Kunden auf, um dort ihre erfassten Daten abzulegen.

1.2 Projektziel

Beim erstmaligen Starten der Applikation, erstellt diese eine Konfigurationsdatei mit den entsprechenden Verbindungsparametern und legt diese ab. Hierbei handelt es sich um Verbindungsdaten zur entsprechenden Datenbank des Kunden, wie beispielsweise der Name der Datenbank, Passwort und ähnliches. Ohne das Erstellen dieser Datei ist ein Arbeiten mit der GITS Software nicht möglich, da sie sich nicht mit der Kundendatenbank verbinden kann. Um diese Konfigurationsdatei zu erstellen, ist es notwendig, einen Service Dongle am Kundensystem anzustecken. Seit einiger Zeit arbeiten viele der Kunden von MK-Versuchsanlagen allerdings verstärkt mit virtuellen Rechnern und/oder virtuellen Servern. Hier besteht nicht die Möglichkeit, einen USB Dongle am System einzustecken. In diesem Fall, wird ein Laptop der Firma MK-Versuchsanlagen im Kundennetzwerk integriert und die benötigten Parameter werden manuell abgefragt und in einer kundenspezifischen Konfigurationsdatei zusammengestellt. Anschließend wird die Konfigurationsdatei manuell in das Applikationsverzeichnis des Kundensystems kopiert und abgelegt. Bisher wurde für diese Prozedur eine Konsolenanwendung eingesetzt. Da diese sehr umständlich in der Handhabung ist, soll über dieses Projekt eine Anwendung entwickelt werden, die diesen Vorgang vereinfacht, komfortabler macht und die erfassten Konfigurationsdaten dauerhaft verfügbar hält. In der Konsolenanwendung werden die

benötigten Parameter manuell abgefragt. Im Falle einer Falscheingabe muss das Tool neu gestartet werden und alle Eingaben erneut vorgenommen werden. Eine Änderung der Parameter ist ebenfalls nicht möglich, auch hier müssen alle Eingaben erneut getätigt werden. In dieser Anwendung werden die Konfigurationsdateien nicht gespeichert, wodurch ein erneutes Abrufen oder Korrigieren der Parameter unmöglich ist. Um dieses Problem zu lösen, soll eine Anwendung entwickelt werden, welche folgende Anforderungen erfüllt:

- Grafische Oberfläche
- Datenabfrage
- Datenänderung
- Datenergänzung
- Datenlöschung
- Erstellung der Konfigurationsdatei
- Speicherung der Konfigurationsdateien in einer Datenbank
- Abrufen der Konfigurationsdateien anhand der Auftragsnummer

Es soll eine WPF-Anwendung im MK Corporate Design entwickelt werden. Alle Parameter müssen verschlüsselt werden, zudem wird bei Programmstart abgefragt ob ein Service Dongle am System vorhanden ist. Alle Konfigurationsdateien müssen vor Verschlüsselung im XML-Format erstellt werden. Das Schema der XML-Konfigurationsdatei ist durch die GITS Applikation vorgegeben. Für die Verschlüsselung der Daten und für die Abfrage über das Vorhandensein eines Service Dongles kommen, die von MK-Versuchsanlagen entwickelten, Klassen zum Einsatz. Daher sind diese drei Bestandteile vorgegeben und nicht Teil des Projektes.

1.3 Projektumfeld

Das Projekt wird innerhalb des Praktikums umgesetzt und somit in der Abteilung Softwareentwicklung der Firma MK-Versuchsanlagen. Der Autorin stehen hierfür folgende Ressourcen zu Verfügung:

- Windows 10
- Microsoft SQL Server
- Microsoft Visual Studio
- C#
- XML/XAML
- Farbvorgaben für das Design

2 Ist-Analyse

2.1 Ausgangssituation

Derzeit ist der Arbeitsaufwand durch die Konsolenanwendung sehr hoch. Zum einen müssen alle Parameter manuell eingegeben werden, bei einem Tippfehler oder einer Korrektur muss die Konsolenanwendung sogar neu gestartet werden. Zum anderen kann man die Daten nicht speichern, was ein wiederholtes Abrufen der Daten unmöglich macht und somit ein erneutes Eingeben der Daten in die Konsolenanwendung erfordert. Das Vorbereiten der Daten vor dem Kundentermin ist dementsprechend nicht möglich, ebenso wenig wie ein Kontrollieren oder

Überwachen der Konfigurationsdateien. So dass ein strukturiertes Abarbeiten und Verwalten der Kundenaufträge unmöglich wird.

2.2 Kostenrechnung Ist-Zustand

Zur Zeit muss ein Mitarbeiter aus der Abteilung jede Konfigurationsdatei manuell eingeben, eventuell mehrfach, bei Falscheingabe kann der einzelne Vorgang in die Länge gezogen werden. Durch das Nicht-Speichern können ist eine Fehleranalyse quasi unmöglich und bei Problemen im Kundenserver werden unter Umständen gleich mehrere Mitarbeiter zeitlich eingebunden. Somit ist das Erstellen der Konfigurationsdateien ein nicht kalkulierbarer und dennoch wiederkehrender Kostenfaktor. Hierzu eine Aufstellung der geschätzten Kosten

Vorgang	Mitarbeiter	Zeit	Lohn	Multiplikationsfaktor	Gesamt
ConfigFile erstellen	1x Fachmitarbeiter	15 min	25,00€	5x pro Ordernummer	125,00€
Geschätzte Kosten pro Ordernummer					125,00€

Tabelle 1 Kostenrechnung Ist-Zustand

Aufgrund der umständlichen Handhabung, braucht ein Fachmitarbeiter etwa 15 Minuten bis die Konsolenanwendung korrekt durchlaufen wird. Es ist davon auszugehen, dass ein Kunde seine Konfigurationsdatei einmal aus Versehen verschiebt, oder aber am System etwas verändert, sodass ein erneutes Erstellen der Konfigurationsdatei nötig ist, deshalb der Multiplikationsfaktor.

3 Soll-Analyse

3.1 Funktionale Anforderungen

- Der Nutzer der Anwendung soll die Möglichkeit haben, Konfigurationsdateien in der Datenbank anhand von Ordnernummern vorzubereiten und in der Datenbank zu speichern, ohne die eigene Konfigurationsdatei zu überspeichern.
- Gleichzeitig soll eine Vorschau der gespeicherten Konfigurationsdatei zu sehen sein.
- Der Nutzer soll darüber hinaus wählen können, ob er die Konfigurationsdatei tatsächlich im korrekten Dateipfad ablegen möchte oder nicht.
- Konfigurationsdateien sollen in der Datenbank und im Dateipfad löscher sein.
- Konfigurationsdateien sollen in der Datenbank und im Dateipfad bearbeitbar sein.

3.2 Nicht Funktionale Anforderungen

- Das Design ist in der Farbgebung und der Gestaltung dem Corporate Design der Firma MK-Versuchsanlagen anzupassen.
- Alle Eingaben müssen in der tatsächlichen Konfigurationsdatei verschlüsselt sein.
- Die Anwendung muss vor dem Start des Programms abfragen, ob ein Service Dongle am System angesteckt ist und bei nicht Vorhandensein das Programm beenden.
- Die Sprache der Anwendung soll zwischen Englisch und Deutsch wählbar sein.
- Die Anwendung muss beweglich und minimierbar sein.
- Ein Infofenster muss zu öffnen und zu schließen sein. Hier soll eine Kurzinformation zur Anwendung erscheinen.

- Das Schema der Konfigurationsdatei muss dem vorgegebenen Schema der Firma MK-Versuchsanlagen entsprechen.
- Siehe Lastenheft im Anhang

3.3 Wirtschaftlichkeitsrechnung Soll-Zustand

Die Wirtschaftlichkeit wird laut untenstehender Tabelle berechnet und somit hat sich das Projekt bereits nach 15 Ordnernummern amortisiert.

Vorgang	Mitarbeiter	Zeit	Lohn	Multiplikationsfaktor	Gesamt
Entwicklungskosten	1x Auszubildender	80h	800,00€	keiner	800,00€
Fachgespräch	1x Fachmitarbeiter	5h	500,00€	keiner	500,00€
Abnahme	1x Fachmitarbeiter	1h	100,00€	keiner	100,00€
ConfigFile erstellen	1x Auszubildender	3min	0,5€	1x pro Ordnernummer	1,00€
ConfigFile abrufen	1x Fachmitarbeiter	3min	5€	5x pro Ordnernummer	25,00€
Entwicklungskosten insgesamt					1400,00€
Kosten pro Konfigurationsdatei und Ordnernummer					26,00€

Tabelle 2 Wirtschaftlichkeitsrechnung

Vorherige Kosten: 15 Ordnernummern x 125€ = 1875€

Neu berechnete Kosten: 1400€ + 15 Ordnernummern x 26€ = 1790€

4 Projektablauf

4.1 Zeitplanung

1. Planung	13h
- Ist-Analyse	3h
- Soll-Analyse	3h
- Grobplanung	3h
- Feinplanung	4h
2. Realisierung	57h
- Erstellung der GUI	8h
- Erstellung der Logik	29h
- Datenbankentwurf	10h
- Testphase	9h
- Abnahme durch MK-Versuchsanlagen	1h
3. Dokumentation	10h
- Erstellung der Dokumentation	10h
Gesamt:	80h

Tabelle 3 Zeitplanung

4.2 Planung

Bei MK-Versuchsanlagen wird nach dem V-Modell entwickelt. Dem auch dieses Projekt unterlag. Die Systemanforderungsanalyse war zu diesem Zeitpunkt bereits abgeschlossen und das Ergebnis wurde bereits im Abschnitt Soll-Analyse erörtert. Auch die Systemarchitektur ist bereits definiert. Eine Anbindung an die Datenbank ist erforderlich. Auch eine Anbindung bzw. die Verfügbarkeit eines vordefinierten Dateipfades wird benötigt. So kommen wir zu dem Punkt eine Anwendung

zu gestalten, die es zum einen möglich macht eine Konfigurationsdatei in der Master Datenbank der Firma MK-Versuchsanlagen zu hinterlegen. Es muss möglich sein, diese anhand von Ordnernummern für die einzelnen Kunden im Vorhinein vorzubereiten, wiederabrufbar zu machen und archivieren zu können. Außerdem müssen beim Kunden die in der Datenbank hinterlegten Konfigurationsdateien über die Anwendung ausgelesen und im korrekten Dateipfad abgelegt werden. Hierfür ist eine Anbindung an die Master Datenbank notwendig. Es wird also eine eigene Tabelle in der Master Datenbank benötigt, die alle Parameter der Konfigurationsdatei, sowie eine Verknüpfung zur Ordertabelle enthält. Als Systementwurf hier eine Abbildung der erforderlichen Tabelle und ihre Anbindung an die Master Datenbank.

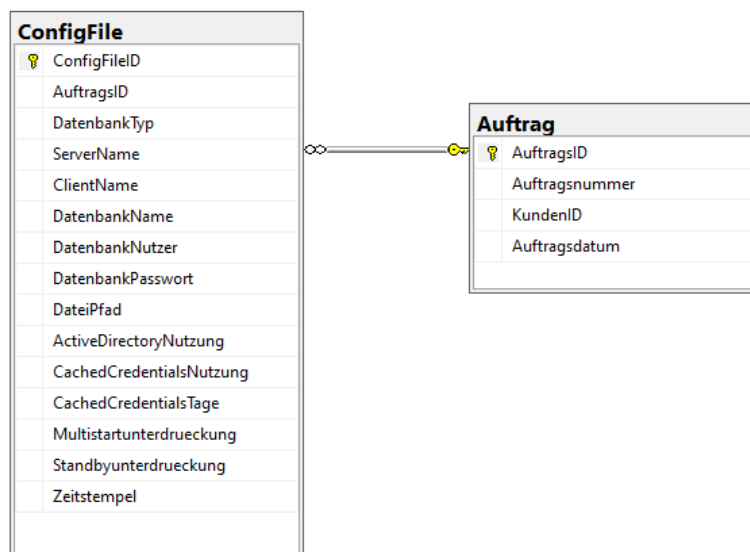


Abbildung 2 Datenbankentwurf

Auch die Softwarearchitektur ist vordefiniert, denn in der Firma MK-Versuchsanlagen wird ausschließlich mit dem MVVM Prinzip gearbeitet, d.h. alle Windows Presentation Foundation (WPF) Anwendungen werden in drei Ebenen gegliedert. Und zwar das View, das ViewModel und das Model. In dem View wird mit Hilfe von XAML die graphische Oberfläche (GUI) gestaltet. Es werden Bindings gesetzt, die entweder Ereignisse (Events) oder Eigenschaften (Properties) aus dem Code ansprechbar machen. Das ViewModel kann man als Kommunikationsebene verstehen. Hier werden alle Bindings aus dem View aufgegriffen und ansprechbar für den eigentlichen Code gemacht. Zudem wird die im Code festgelegte Logik über das ViewModel an das View weitergeben. Das Ganze funktioniert nur mit Hilfe zweier Komponenten, zum einen der Klasse Relay Commands, zum anderen dem EventHandler INotifyPropertyChanged. Die Klasse Relay Commands wird als eigenständige Klasse im Model hinterlegt. Sie steuert die Commands, also den Zugriff auf die Events. Der EventHandler INotifyPropertyChanged wird in das ViewModel eingebunden. Hierüber werden die Veränderungen der Properties gesteuert. Somit steht das Grundschema dieses Projektes bereits im Vorfeld fest.



Abbildung 3 MVVM-Prinzip

Um nun in die Feinplanung zu kommen ist es zunächst wichtig zu verstehen welche Daten in der Konfigurationsdatei benötigt werden und wie die bisherige Konsolenanwendung arbeitet. Wo liegen die Verbesserungsansätze? Das Schema, welches in der Konfigurationsdatei hinterlegt ist, muss zwingend so aussehen, wie hier dargestellt, sonst ist sie nicht verwendbar/lesbar.

```
<root>
  <CONFIG_DBTYPE value="MSSQL" />
  <CONFIG_DBSERVERNAME value="localhost\\sql2017express" />
  <CONFIG_UNIQUECLIENTNAME value="" />
  <CONFIG_DBNAME value="DBgits" />
  <CONFIG_DBUSERNAME value="gitsAppUser" />
  <CONFIG_DBP value="55BmOT5w8C7oVdc3RysRgw==" />
  <CONFIG_DBPATHFILE value="" />
  <CONFIG_USE_ACTIVE_DIRECTORY value="False" />
  <CONFIG_USE_CACHED_CREDENTIALS value="False" />
  <CONFIG_CACHED_CREDENTIALS_DAYS value="7" />
  <CONFIG_PREVENT_MULTISTART value="True" />
  <CONFIG_DISABLE_STANDBY value="True" />
  <LAST_SAVE_TIMESTAMP value="Mo. 19.09.2022 10:30:03" />
</root>
```

Abbildung 4 Beispiel unverschlüsselte Konfigurationsdatei

Die Konsolenanwendung fragt alle Parameter der abgebildeten Konfigurationsdatei einzeln ab. Wenn man sich vertippt, muss die Anwendung neu gestartet werden. Nur im Fall, dass die Anwendung einmal alle Parameter abfragen konnte und alle Parameter korrekt eingegeben wurden, wird einmalig die Konfigurationsdatei erzeugt und im korrekten Dateipfad abgelegt. Einzig und allein der Zeitstempel hilft ein wenig bei der Überwachung, wie aktuell eine Konfigurationsdatei bearbeitet wurde. Denn sie werden in keiner anderen Art und Weise archiviert. Ein Vorbereiten im Vorhinein, bevor man zum Kunden fährt, ist unmöglich.

Die neu zu entwickelnde Anwendung sollte gleichzeitig dazu in der Lage sein, die Daten in die Datenbank zu speichern und/oder daraus abzurufen. Anhand dieser Daten muss es möglich sein, eine Konfigurationsdatei zu erstellen. Die Konfigurationsdatei sollte in einer Vorschau unverschlüsselt angezeigt werden. Bei Bedarf ist sie im korrekten Dateipfad verschlüsselt abzulegen. Es werden also Textboxen für die Eingabe zur Speicherung und Bearbeitung der Daten für die Datenbank benötigt. Wenn Konfigurationsdateien aus der Datenbank abgerufen werden, soll der Inhalt in die passenden Textboxen geladen werden. Darüber hinaus wird eine ComboBox

zur Auswahl der Ordnernummern aus der Datenbank benötigt. In einem Label soll eine Vorschau der unverschlüsselten Konfigurationsdatei gezeigt werden. Zudem braucht man ein Button zum Laden der Konfigurationsdateien, einen weiteren zum Speichern und einen zum Löschen. Für die bessere Übersicht wird ein Button implementiert, der alle Felder zu leert. Um ein Arbeiten mit der Anwendung überhaupt möglich zu machen, ist eine CheckBox für die Auswahl, ob nun auch im Dateipfad tatsächlich eine verschlüsselte Konfigurationsdatei abgelegt werden soll, essentiell. Sonst überspeichert man mit jedem Eintrag in die Datenbank die eigene Konfigurationsdatei im eigenen System und ein Speichern in der Datenbank im Vorhinein ist nicht möglich.

4.3 UML Diagramme

Aktivitätsdiagramm eines möglichen Durchlaufs der Anwendung

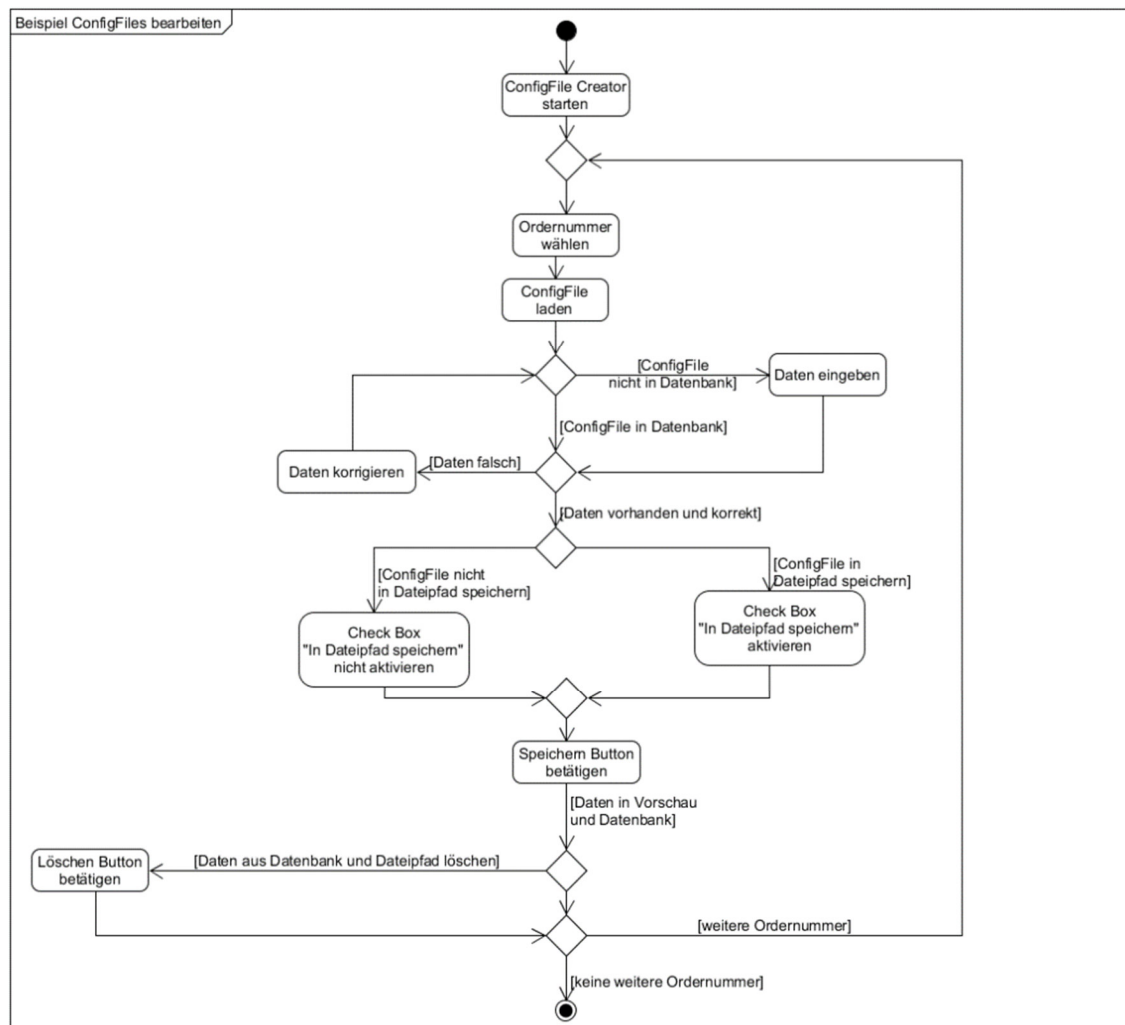


Abbildung 5 Aktivitätsdiagramm

Use-Case Diagramm eines möglichen Durchlaufs der Anwendung

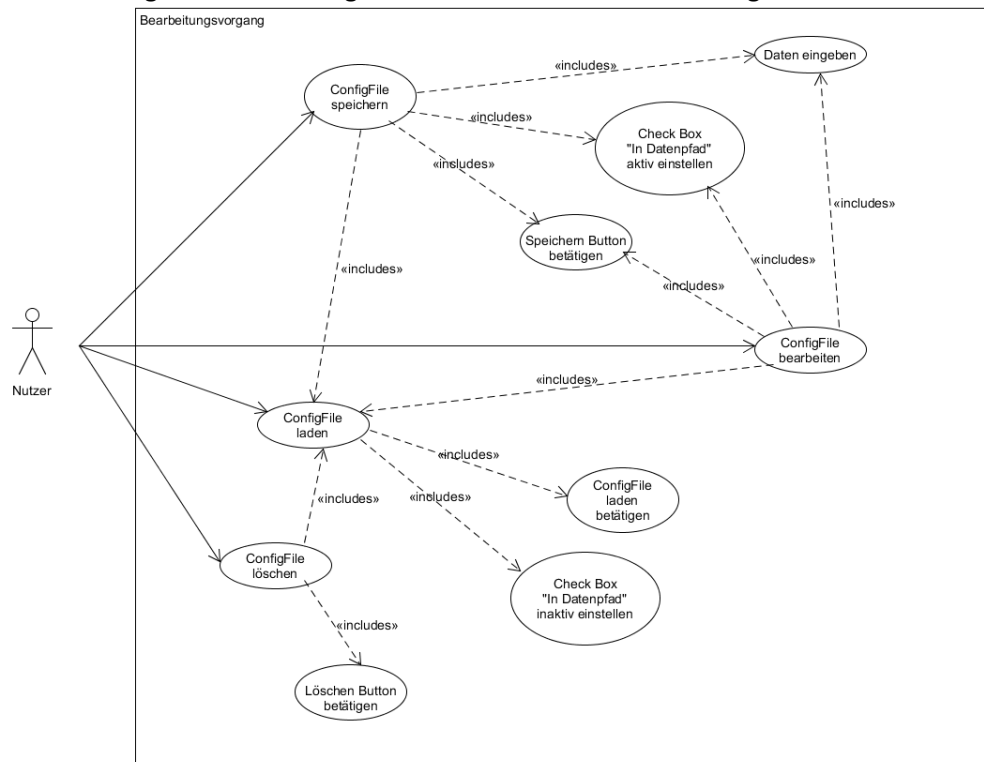


Abbildung 6 Use Case Diagramm

4.4 Planung Corporate Design

In der Firma MK-Versuchsanlagen gibt es bereits eine Sammlung von Tools, die alle einzelne Verbindungsstellen/Aufgaben mit und um die GITS Software darstellen bzw. erfüllen. Diese sind zusammengefasst im Toolcenter. Über Icons kommt man in die einzelnen Tools und ab hier sind Abmessungen und Farbgebung identisch. So sind beispielsweise die Maße des Logos, Position des Logos, Maße des seitlichen Menüs, sowie des unteren Menüs, zu Verfügung stehende Farben, der fehlende Rand, die Sprachwahlbuttons, der Info-Button und dessen Funktion, der Minimieren-Button und der Schließen-Button von anderen Tools abzuleiten gewesen. Siehe Vergleichstools im Anhang.



Abbildung 7 GUI der Anwendung

4.5 Implementierung GUI

Bevor mit der Implementierung begonnen werden kann, ist es notwendig das zuvor beschriebene Grundgerüst des MVVM Prinzips zu erstellen. Hierzu werden einzelne Ordner für das View, das ViewModel und das Model angelegt und mit den entsprechenden Klassen bzw. EventHandlern befüllt. Bei der Implementierung der GUI liegt das Hauptaugenmerk auf der richtigen Verknüpfung der Events und Properties durch Bindings.

```
<Button x:Name="Löschen" Content="{Binding Source={StaticResource ViewModel}, Path=Loeschen_Button}"
        HorizontalAlignment="Left" Height="32" Margin="129,634,0,0" VerticalAlignment="Top"
        Width="130" Background="white" FontSize="14" FontWeight="Bold"
        Command="{Binding DateiLoeschen, Source={StaticResource ViewModel}}" Grid.Column="2"/>
```

Abbildung 8 Beispiel Bindings

Es gilt ebenso die Ansprüche, die aus dem Corporate Design resultieren, umzusetzen. Um beispielsweise zu ermöglichen, dass die Anwendung keinen Rand hat, aber trotzdem beweglich bleibt, ist es notwendig für das „Window“ einen eigenen Style festzulegen. Darüber hinaus muss eine Größe für den Window Chrome Bereich festgelegt werden, sprich für den Bereich, der außerhalb des Bereichs liegt, der für den Nutzer erreichbar ist.

```
</Window.Resources>
<Window.Style>
    <Style TargetType="Window">
        <Setter Property="ResizeMode" Value="CanMinimize" />
        <Setter Property="WindowStyle" Value="ToolWindow" />
        <Setter Property="BorderBrush" Value="#FFFE64140" />
        <Setter Property="BorderThickness" Value="5" />
    </Style>
</Window.Style>
<WindowChrome.WindowChrome>
    <WindowChrome CaptionHeight="80"/>
</WindowChrome.WindowChrome>
```

Abbildung 9 WindowChrome

Da nun aber bestimmte Elemente der GUI innerhalb dieses nicht erreichbaren Bereichs des Nutzers liegen, müssen diese die Eigenschaft „IsHitTestVisibleInChrome“ mit „True“ zugewiesen bekommen, damit sie weiterhin für den Nutzer erreichbar bleiben. Das sind hier die Sprachwahlbuttons, sowie die Buttons für „Minimieren“, „Schließen“ und das Info Fenster. Eigene Styles werden ebenso für alle Elemente benötigt, die farblich oder durch die Form an das Corporate Design angepasst wurden. Eine Besonderheit stellt hier die ComboBox dar, denn bei dieser ist es nicht möglich, einzelne Style Elemente (Properties) direkt anzusprechen und anzupassen, denn das würde bedeuten, dass die Funktionalität der Combo Box verloren ginge. Hier ist es also nötig die gesamte Style Vorlage innerhalb der Programmierung anzupassen.

4.6 Implementierung Datenbank

Auf Basis der bereits erläuterten Datenbankstruktur wurde die vorhandene Master Datenbank der Firma MK-Versuchsanlagen um die Tabelle „ConfigFiles“ erweitert. Die „ConfigFileID“ wurde in dieser Tabelle als Primary-Key festgelegt. Darüber hinaus wird eine Verknüpfung mit der bereits bestehenden Tabelle „Auftrag“ benötigt. Das wird über den Primary-Key dieser Tabelle, nämlich die „AuftragsID“ ermöglicht, der als Foreign-Key der Tabelle „ConfigFiles“ festgelegt wurde. Um aus der Anwendung auf die Datenbank zugreifen zu können, wurde eine SQL Authentifizierung eingerichtet.

4.7 Implementierung Geschäftslogik

Zu dem bereits bestehenden Grundgerüst aus dem MVVM Prinzip und der bereits implementierten GUI wird nun die Kommunikationsebene, also das ViewModel befüllt. Hier ist es notwendig allen festgelegten Bindings aus der GUI Variablen bzw. für die Events Commands zuzuordnen. Für jedes Binding, welches einer Property zugeordnet ist, wird eine `RaisePropertyChanged` Methode erstellt. Diese ermöglicht das Zugreifen und Verändern der Property aus dem Model, über das ViewModel, in der GUI.

```
/// <summary>
/// Die öffentliche Variable für die Bindung des Textes auf dem Speichern Button
/// </summary>
1 Verweis
public string Speichern_Button
{
    get
    {
        return _Speichern_Button;
    }
    set
    {
        _Speichern_Button = value;
        RaisePropertyChanged("Speichern_Button");
    }
}
```

Abbildung 10 Beispiel Property Methode

Das Gleiche Prinzip gilt für die Events, nur sind es hier Commands, die ausgeführt werden sollen. Es wird also eine Methode festgelegt, die beim Eintreffen des Events ablaufen soll.

```
/// <summary>
/// Das öffentliche Event für den Click auf "Info"
/// </summary>
0 Verweise
public ICommand Info
{
    get
    {
        if (_Info == null)
        {
            _Info = new MRelayCommand(param => Info_Los());
        }
        return _Info;
    }
}

1 Verweis
public void Info_Los()
{
    _Label_InfoBox = MProjekt_HeLi.Info_Los(Label_InfoBox);
    RaisePropertyChanged("Label_InfoBox");
    _Button_InfoBox = MProjekt_HeLi.Button_da(Button_InfoBox);
    RaisePropertyChanged("Button_InfoBox");
}
```

Abbildung 11 Beispiel Command Methode

Um nun die eigentliche Geschäftslogik zu implementieren, werden zwei weitere Klassen dem Model hinzugefügt. Zum einen eine Klasse die alle Methoden beinhaltet, mit denen auf die Daten aus der Datenbank zugegriffen wird, alle Methoden mit denen eine Umformatierung der Daten ermöglicht wird, alle Methoden mit denen ein Lesen bzw. Schreiben von Datensätzen in oder aus einem Dateipfad ermöglicht wird. Diese Klasse nennt sich „MDatenbankzugriff“. Eine weitere Klasse „MProjekt_HeLi“ beinhaltet sämtliche Methoden die den eigentlichen Programmablauf

In der Klasse „MProjekt_HeLi“ werden alle Methoden zu Verfügung gestellt, die über die Bindings der Commands an die Events in der GUI gebunden sind. Neben den nicht für die Funktion des Programms relevanten Methoden zur Befüllung der ComboBox, den Methoden zur Sprachwahl, den Methoden für die Funktionalität der Info Box, der Methode zum Minimieren und der Methode zum Schließen der Anwendung, eben auch die relevanten Methoden, die für die Funktion der Anwendung ausschlaggebend sind. Dies umfasst je eine Methode für die Funktionalität der Buttons „Daten speichern“, „Daten löschen“, „ConfigFile laden“ und „Felder leeren“.

Während der gesamten Entwicklung wurden immer wieder einzelne Teilbereiche ausführlich getestet und ihre Funktion geprüft. Es wurden dynamische Unittests durchgeführt, nach Fertigstellung des Projektes erfolgte ein abschließende Abnahmetest aller Anforderungen durch MK-Versuchsanlagen. Mit Abnahme des Projektes ist die Autorin nicht mehr integriert in den letzten drei Punkte des V-Modells. Der Integrationstest, die Systemintegration, sowie die Nutzung des Programms führte MK-Versuchsanlagen durch.

```
graph TD; SA[Systemanforderungsanalyse] --> SA_A[System-Architektur]; SA_A --> SE[System-Entwurf]; SE --> SWA[Software-Architektur]; SWA --> SEW[Software-Entwurf]; SEW --> UT[Unit-Tests]; UT --> IT[Integrations-Tests]; IT --> SI[System-Integration]; SI --> AN[Abnahme und Nutzung]; SA --> SI; SA_A --> IT; SE --> SI; SWA --> IT; SEW --> IT;
```

Wie bereits erwähnt, wird bei MK-Versuchsanlagen ausschließlich nach dem V-Modell entwickelt. Dies bringt verschiedene Vorteile mit sich.

- Einzelne Projektphasen sind gut zu überwachen und voneinander abzutrennen
- Ergebnisse einzelner Phasen sind bindend und auch im Nachhinein gut nachvollziehbar
- Durch fest integrierte Tests, wird Qualität des Endprodukts erhöht
- Einzelne Phasen sind durch die Grundstruktur und die Philosophie in der Softwareentwicklung der Firma MK-Versuchsanlagen bereits festgelegt.

Für das vorliegende Projekt gilt:

Systemanforderungsanalyse —————> siehe Soll-Analyse

Systemarchitektur —————> Anbindung Datenbank und Dateipfad

Systementwurf —————> siehe Datenbankentwurf

Softwareentwurf —————> siehe Implementierung

Unit-Test —————> dynamisch, während der Entwurfsphase durch die Autorin

Die erfolgte Abnahme durch MK-Versuchsanlagen, Integrationstest und Systemintegration erfolgte ebenfalls durch MK-Versuchsanlagen. Dies war bereits im Vorhinein so festgelegt und nicht Teil des Projektes.

4.10 Soll-Ist-Vergleich

Es wurden alle Funktionalitäten und Anforderungen gemäß Lastenheft erfüllt und alle funktionalen und nicht-funktionalen Problemstellungen umgesetzt. Die durch das Projekt umgesetzte Anwendung wird bereits durch MK-Versuchsanlagen genutzt.

5 Zeitplanung

5.1 Detaillierte Zeitstruktur

4. Planung	13h
- Ist-Analyse	3h
- Soll-Analyse	3h
- Grobplanung	3h
- Feinplanung	4h
5. Realisierung	57h
- Erstellung der GUI	8h
- Erstellung der Logik	29h
- Datenbankentwurf	10h
- Testphase	9h
- Abnahme durch MK-Versuchsanlagen	1h
6. Dokumentation	10h
- Erstellung der Dokumentation	10h
Gesamt:	80h

Tabelle 4 Zeitstruktur

5.2 Analyse der Zeitplanung

Der zu Beginn des Projektes festgelegte Zeitplan konnte eingehalten werden. An zwei Punkten konnte die Autorin Zeit einsparen, bei der Testphase und dem Datenbankentwurf. Diesen Zeitpuffer konnte die Autorin für die Ausarbeitung der Dokumentation gut nutzen.

6 Quellcodeauszüge

6.1 Beispiel Datenbankabfrage

Mit dem abgebildeten Beispiel wird eine komplette Zeile aus der Tabelle ConfigFiles geholt, die Anhand einer vorher bereitgestellten AuftragsID ausgewählt wird. Die Daten befinden sich anschließend in einem DataSet, was einem ItemArray entspricht. Um die Daten in eine weiter verarbeitbare Form zu bringen, werden die Daten in das ConfigFile_Array gelesen, welches durch das gesamte Programm immer wieder, entweder mit leeren Strings oder mit Werten aus der ConfigFiles Datenbank gefüllt wird. Jede TextBox oder CheckBox auf der GUI ist einer bestimmten Position im ConfigFile_Array zugeordnet.

```
/// <summary>
/// Holt Config File Daten aus der Datenbank
/// </summary>
/// <returns>ConfigFile_Array</returns>
2 Verweise
public ArrayList HoleDaten_ConfigFile()
{
    CMD.CommandText = MProjekt_HeLi.HoleConfigFile;
    CMD.Connection = con;
    if (!ConnectionOpen)
    {
        OpenZugriff();
    }
    DA = new SqlDataAdapter(CMD);
    DA.Fill(DataSet_SQL);
    CloseZugriff();
    foreach (DataRow Zeilen in DataSet_SQL.Tables[0].Rows)
    {
        for (int i = 0; Zeilen.ItemArray.Length > i; i++)
        {
            string Wert = Convert.ToString(Zeilen.ItemArray[i]);
            ConfigFile_Array.Add(Wert);
        }
    }
    DataSet_SQL.Tables.Clear();
    return ConfigFile_Array;
}
```

Abbildung 13 Codebeispiel Datenbankabfrage

6.2 Umwandlung der Daten in Konfigurationsdatei

In dieser Methode wird mit Hilfe eines Streamwriter Objektes und der zuvor erstellten XML Datei mit den Spaltennamen und Werten der Konfigurationsdatei, eine Text-Datei erstellt, die optisch der lesbaren Version der Konfigurationsdatei für die GITS Software, entspricht. Zunächst wird die Zeile „<root>“ separat geschrieben, da diese in der Datenbank und dementsprechend in der XML Datei nicht vorkommt. Im Anschluss gehen wir in eine erste Zähl-Schleife mit der Anzahl der Spalten in der Konfigurationsdatei, um so viele Zeilen in der Konfigurationsdatei zu erzeugen wie benötigt werden. In der folgenden foreach-Schleife wird dann pro Wert in der Konfigurationsdatei die korrekte Zeile erstellt. Zum Schluss wird nochmals separat „</root>“ ausgegeben.

```
/// <summary>
/// Das Dataset mit der XML Datei wird über zwei verschachtelte Schleifen in eine Textdatei umgewandelt
/// </summary>
1 Verweis
public static void TXT_Bereitstellen()
{
    StreamWriter Schreiben = new StreamWriter(MDatenbankzugriff.Dateiname_Pfad_TXT);
    DataTable TBL = XMLReturn.Tables[0];
    string Columnname;
    string Zeile = "<root>";
    Schreiben.WriteLine(Zeile);
    int Zaehler = 0;
    string xmlZeile;
    for (int i = 0; i < TBL.Columns.Count; i++)
    {
        foreach (DataRow DR in TBL.Rows)
        {
            Columnname = TBL.Columns[i].ColumnName;
            Zeile = DR[Zaehler].ToString();
            xmlZeile = "<" + Columnname + " value=" + "\"" + Zeile + "\"" + " />\n";
            Schreiben.WriteLine(xmlZeile);
            Zaehler++;
        }
    }
    Zeile = "</root>";
    Schreiben.WriteLine(Zeile);
    Schreiben.Close();
}
```

Abbildung 14 Codebeispiel XML Umwandlung

6.3 Style Beispiel GUI

Unten zu sehen ist ein Ausschnitt aus der XML Datei, die für die Gestaltung der GUI benötigt wird. Hier wurde ein Button angelegt, der die Info Box wieder schließen soll, wenn man sie geöffnet hat. Dieser Button liegt über der gesamten Anwendung, aber hinter der geöffneten Info Box. Er ist durchscheinend, d.h. man erkennt ihn nicht als Button, sondern als grauen Schatten über der Anwendung. Man sieht zum einen das Binding für die Property Visibility, denn er soll nur sichtbar sein, wenn die Info Box geöffnet ist, darüber hinaus das Binding für das Command, um das Click Event auslösen zu können. Im angehängten Button.Style wird die Hintergrundfarbe angesprochen. Es wird aber keine Hintergrundfarbe festgelegt. Mit dem Style.trigger wird die Property des MouseOver Effekts angesprochen. D.h. mit diesem Code Ausschnitt wird der MouseOver Effekt für diesen einzelnen Button entfernt.

```
<Button x:Name="Button_Stop" HorizontalAlignment="Left" Margin="-5,-25,-10,-15" VerticalAlignment="Center"
Height="699" Width="1004" Grid.ColumnSpan="3" Opacity="0.5"
Visibility="{Binding Source={StaticResource ViewModel}, Path=Button_InfoBox}"
Command="{Binding Source={StaticResource ViewModel}, Path=Stoppen_InfoBox}">
<Button.Style>
<Style TargetType="{x:Type Button}">
<Setter Property="Template">
<Setter.Value>
<ControlTemplate TargetType="{x:Type Button}">
<Border Background="{TemplateBinding Background}">
<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>
</Border>
</ControlTemplate>
</Setter.Value>
</Setter>
<Style.Triggers>
<Trigger Property="IsMouseOver" Value="True">
</Trigger>
</Style.Triggers>
</Style>
</Button.Style>
</Button>
```

Abbildung 15 Stylebeispiel XAML

7 Beschreibung der Anwendung

7.1 Ablauf der Anwendung

Nach dem Öffnen der Anwendung, gibt es verschiedene Anwendungsmöglichkeiten. Ganz gleich welche Anwendungsmöglichkeit ausgewählt wird, es ist auf jeden Fall erforderlich, eine Ordernummer auszuwählen und zu laden. Hier gibt es bereits verschiedene Anwendungsmöglichkeiten. Ist für die Ordernummer bereits eine Konfigurationsdatei angelegt und die CheckBox „In Dateipfad speichern“ ist aktiv, wird die Konfigurationsdatei direkt beim Laden in den korrekten Dateipfad abgelegt und verschlüsselt. Ist die CheckBox nicht aktiv, aber die Konfigurationsdatei Daten bereits in der Datenbank hinterlegt, wird sie lediglich in der Vorschau angezeigt. Nun kann man einzelne Werte bearbeiten und speichern. Auch hier gilt, hat man die CheckBox aktiviert wird die Konfigurationsdatei bereits im korrekten Dateipfad gespeichert und verschlüsselt. Ist keine Konfigurationsdatei in der Datenbank hinterlegt, ist dies über ein Füllen der Text- und Checkboxes möglich. Kurzanleitung siehe Anhang.

7.2 Usability

Die Anwendung ist leicht verständlich und über die Benennung der Buttons und die Funktionen selbsterklärend, jeder Nutzer kann sich ohne weitere Hilfestellung mit der Anwendung am Bearbeiten der Konfigurationsdateien beteiligen.

8 Reflexion

8.1 Lessons Learned

Im Zuge des Projektes war es der Autorin möglich, tiefere Einblicke in das MVVM-Prinzip bei der Programmierung von WPF Anwendungen zu bekommen. Auch in Sachen Planung und Durchführung konnten erweiterte Erfahrungen gesammelt werden. Datenbanken und deren Anbindung, sowie die Einbindung von Dateipfaden, also das Einlesen und Ablegen von Dateien machten ebenfalls einen großen Teil des Projektes aus.

8.2 Ausblick

Da die Anwendung bereits im MK-Versuchsanlagen System integriert ist, war das Projekt erfolgreich. Es ist ein in sich geschlossenes System und bietet kaum Erweiterungsspielraum. Aber natürlich ist es denkbar das weitere Tools ins Tool Center der Firma MK-Versuchsanlagen Einzug halten.

8.3 Fazit

Das Projekt hat sich für alle Beteiligten bezahlt gemacht, die Autorin hat gute Einblicke bekommen und viel gelernt. Die Firma MK-Versuchsanlagen hat ein komfortables, gut nutzbares Tool erhalten. Auch die Kunden der Firma MK-Versuchsanlagen bekommen eine vereinfachte, schnellere Serviceleistung.

Quellenangabe

- <https://akrick.wordpress.com/2013/05/18/mvvm/> (hier wurde Abbildung 3 genutzt, am 15.10.2022, 12:30Uhr)
- <https://de.wikipedia.org/wiki/V-Modell> (hier wurde Abbildung 12 genutzt, am 15.10.2022, 14:00Uhr)

Anhang

Kurzanleitung

Mit dem Tool „Config File Creator“ haben Sie vier Anwendungsmöglichkeiten:

- Laden
- Speichern
- Bearbeiten
- Löschen

von Konfigurationsdateien. Nach dem Starten der Anwendung ist es in jedem Anwendungsfall notwendig, eine Ordernummer über das Drop Down Menü der Auftragsnummer auszuwählen und zu laden. Dies geschieht über den Button „ConfigFile laden“.

Laden mit Speicherung im korrekten Dateipfad

Um die Konfigurationsdatei direkt beim Laden im korrekten Dateipfad abzulegen, aktivieren Sie die CheckBox „Im Dateipfad speichern“. Ist die geladene Ordernummer bereits mit einer Konfigurationsdatei in der Datenbank hinterlegt, wird diese abgelegt und verschlüsselt. Im Vorschauenfenster erscheint eine unverschlüsselte Version der abgelegten Datei.

Laden ohne Speicherung im korrekten Dateipfad

Möchten Sie die Konfigurationsdatei lediglich laden, um sie zu prüfen und/oder zu bearbeiten, ist es zwingend erforderlich die CheckBox „Im Dateipfad speichern“ zu deaktivieren, denn sonst überspeichern Sie ihre eigene in Ihrem System hinterlegte Konfigurationsdatei. Auch hier erscheint eine Vorschau der Konfigurationsdatei. Abgelegt und verschlüsselt wird sie aber nicht.

Bearbeiten/Speichern

Sind zu der geladenen Konfigurationsdatei keine Daten in der Datenbank hinterlegt, ist es möglich über die Eingabe von Daten in die beschrifteten Text- und CheckBoxen, eine Konfigurationsdatei anzulegen. Auch hier gilt, ist die CheckBox „Im Dateipfad speichern“ aktiviert, wird die Konfigurationsdatei im korrekten Dateipfad abgelegt und verschlüsselt. Ist die CheckBox nicht aktiviert, werden die Daten lediglich in der Datenbank gespeichert. Sind falsche Daten hinterlegt, können hier auf die gleiche Art und Weise Daten bearbeitet werden. Hat man nach Bearbeitung der Daten den Button „ „Daten speichern“ betätigt, erscheint die bearbeitete unverschlüsselte Konfigurationsdatei im Vorschauenfenster.

Löschen

Geladene Konfigurationsdateien können über den Button „Löschen“ aus der Datenbank und dem Dateipfad entfernt werden. Hier gilt, alle Daten werden gelöscht, die aus der Datenbank und, falls vorhanden, aus dem Dateipfad.


Felder leeren

Um zwischenzeitlich den Arbeitsplatz zu „reinigen“, ist es möglich alle Felder zu leeren. Dies dient lediglich der besseren Übersicht, hierbei werden keine Daten verändert.

Lastenheft

Das Lastenheft wurde durch die Firma MK-Versuchsanlagen erstellt und der Autorin übergeben.

Abbildung 16 Lastenheft


NUR FÜR INTERNE VERWENDUNG		
 DOKUMENT URS- GITSCon	URS – USER REQUIREMENT SPECIFICATION	SEITE 2 VON 7
	GITS Connection Config	VERSION 1.0 DATUM 18.07.2022
VERSIONSHISTORIE		
VERSION	ÄNDERUNGEN	
1.0	• Dokument erstellt	• HK 18.07.2022
<small>Bitte beachten: Ein neues Dokument startet mit Version 1.0 und es müssen nur Änderungen zur aktuellen Version festgehalten werden!</small>		

ENTWURF

© Lisa Hentschel

ix

NUR FÜR INTERNE VERWENDUNG

	URS – USER REQUIREMENT SPECIFICATION	SEITE 3 VON 7
	GITS Connection Config	VERSION 1.0
		DATUM 18.07.2022
DOKUMENT URS- GITSCon		

INHALT

1. Zweck dieses Dokumentes.....	4
2. Projektumfeld	4
3. Projektziel.....	5
4. Funktionsspezifikation	5
5. Abschließende Bemerkungen.....	7


NUR FÜR INTERNE VERWENDUNG

NUR FÜR INTERNE VERWENDUNG

MK_Technisches Dokument_Intern_OE_1.1.docx

NUR FÜR INTERNE VERWENDUNG

NUR FÜR INTERNE VERWENDUNG

	URS – USER REQUIREMENT SPECIFICATION	SEITE 4 VON 7
	GITS Connection Config	VERSION 1.0
		DATUM 18.07.2022
DOKUMENT URS- GITCon		

1. ZWECK DIESES DOKUMENTES

Die User Requirement Specification stellt die Anforderungen an ein neu zu entwickelndes Tool zusammen. Die Applikation GITS Connection Conf soll als interne Anwendung bei Fa. MK Versuchsanlagen und Laborbedarf e. K. eingesetzt werden. Das Dokument dient als Grundlage für das zu erstellende Pflichtenheft.

2. PROJEKTUMFELD

Die Applikation GITS bietet die Vollautomatische Erstellung der benötigten Datenbank beim erstmaligen Start der Anwendung. Im Dialog für die Datenbankverbindung sind die Informationen zum Server, der Instanz des Benutzers und dessen Kennwort einzugeben. Damit dieser Dialog genutzt werden kann ist es notwendig, einen MK Service Dongle am System auf dem die Software gestartet wird, anzustecken.

Seit ca. 2 Jahren werden diese Systeme immer häufiger auf virtuellen Rechnern installiert. Gerade Kunden die eine Enterprise-Lizenz erworben haben, möchten das System voll in ihre Infrastruktur integrieren, wobei diese in der Regel aus mehreren virtualisierten Servern besteht. Diese Systeme bieten im Normalfall nicht die Möglichkeit, einen USB Dongle am System anzustecken. Die Informationen, die beim Erststart vom Endbenutzer abgefragt werden und in das Program-Config-File geschrieben werden, müssen in diesen Fällen durch Fa. MK manuell erstellt werden um dann in das Applikationsverzeichnis kopiert zu werden.

Fa. MK hat für diesen Zweck ein Tool in Form einer Konsolenanwendung im Einsatz. Dieses Tool ist jedoch sehr unflexibel was die Abfrage der notwendigen Daten betrifft. Bei einer Konsolenanwendung werden die Daten sequenziell vom Benutzer abgefragt, wurde in einem Schritt eine falsche Angabe getätigt, muss das Tool neu gestartet werden und alle Eingaben erneut vorgenommen werden.

Weiterhin muss auch für kleine Änderungen immer die Gesamtheit der Informationen neu erfasst werden da diese Konfigurationsdateien nicht gespeichert werden.


NUR FÜR INTERNE VERWENDUNG

NUR FÜR INTERNE VERWENDUNG

MK_Technisches Dokument_Interne_Doc_1.1.docx

NUR FÜR INTERNE VERWENDUNG

NUR FÜR INTERNE VERWENDUNG

	URS – USER REQUIREMENT SPECIFICATION		SEITE 5 VON 7
	GITS Connection Config		VERSION 1.0
	DOKUMENT URS- GITSCon		DATUM 18.07.2022

3. PROJEKTZIEL

Es soll eine Anwendung entwickelt werden, die die Funktionalitäten der Konsolenanwendung in einer grafischen Oberfläche zur Verfügung stellt. Alle Eingaben können jederzeit verändert werden und per Knopfdruck wird aus den Informationen die entsprechende Konfigurationsdatei erstellt. Die Parameter werden zeitgleich im zentralen System gespeichert.

Zusätzlich soll die Konfiguration Kundenspezifisch, anhand einer Auftragsnummer im zentralen System gespeichert werden. Somit ist auch bei kleineren Änderungen in der Konfiguration nicht die erneute Eingabe aller Daten erforderlich, die Konfiguration kann aus dem zentralen System erneut eingelesen und abgeändert werden.

4. FUNKTIONSSPEZIFIKATION

Position/Bereich	Funktionalität	Muss	Bemerkung
1. System			
1.1 Client	Standard-PC-MK mit Win10	Ja	
2. Datenspeicher		Ja	Integration in MDT
2.1	Zentraler Server MS-SQL 2016	Ja	
2.2	Datenbankzugriff über Datenbankbenutzer	Ja	
2.3	Backup automatisch über Serverroutine (IT MK)	Ja	
2.4	Restore über IT Abteilung MK	Ja	
2.5	Daten aus der Anwendung müssen mit Zeitstempel gespeichert werden	Ja	
3. UserInterface			
3.1	WPF Oberfläche MK Corporate Design	Ja	
4. Funktionalitäten			
4.1 Programmstart	Beim Programmstart muss die	Ja	


NUR FÜR INTERNE VERWENDUNG

NUR FÜR INTERNE VERWENDUNG

MK_Technisches Dokument_Intern_OIG_1.1.docx

NUR FÜR INTERNE VERWENDUNG

NUR FÜR INTERNE VERWENDUNG

	URS – USER REQUIREMENT SPECIFICATION		SEITE 6 VON 7
	GITS Connection Config		VERSION 1.0
			DATUM 18.07.2022
DOKUMENT URS- GITSCon			

	Präsenz eines MK Service Dongle abgefragt werden		
4.2 Auftragsbezug	Für die Eingabe der Konfigurationsdaten ist zwingend die Auswahl einer Auftragsnummer erforderlich	Ja	
	Auch das erneute Einlesen der Daten erfolgt durch Auswahl einer Auftragsnummer	Ja	Einlesen der auf dem zentralen System gespeicherten Daten siehe 4.5
4.3 Verschlüsselung der Konfigurationsdaten	Alle eingegebenen Daten müssen in der zu erstellenden Datei verschlüsselt werden	Ja	
4.4 Dateiformat	Die Konfigurationsdateien müssen vor der Verschlüsselung im XML Format erstellt werden.		
4.5 Ablage der Konfigurationsdateien im zentralen System	Alle Konfigurationsparameter werden im zentralen System gespeichert	Ja	


NUR FÜR INTERNE VERWENDUNG

NUR FÜR INTERNE VERWENDUNG

MK_Technisches Dokument_Intern_OE_1.1.docx

NUR FÜR INTERNE VERWENDUNG

NUR FÜR INTERNE VERWENDUNG

	URS – USER REQUIREMENT SPECIFICATION	SEITE 7 VON 7
	GITS Connection Config	VERSION 1.0
		DATUM 18.07.2022
DOKUMENT URS- GITSCon		

5. ABSCHLIEßENDE BEMERKUNGEN

Das Projekt wird im Rahmen der Abschlussprüfung zur Fachinformatikerin für Anwendungsentwicklung von Lisa Hentschel durchgeführt.

Das Schema für die XML-Konfigurationsdatei ist durch die GITS Applikation vorgegeben, der Entwurf ist also nicht Bestandteil des Projektes. Für die Verschlüsselung wird die von MK entwickelte Klasse verwendet, diese ist nicht Bestandteil des Projektes. Für die Abfrage eines präsenten Service-Dongles wird die von MK entwickelte Klasse verwendet, sie ist ebenfalls nicht Bestandteil des Projektes.

NUR FÜR INTERNE VERWENDUNG

NUR FÜR INTERNE VERWENDUNG

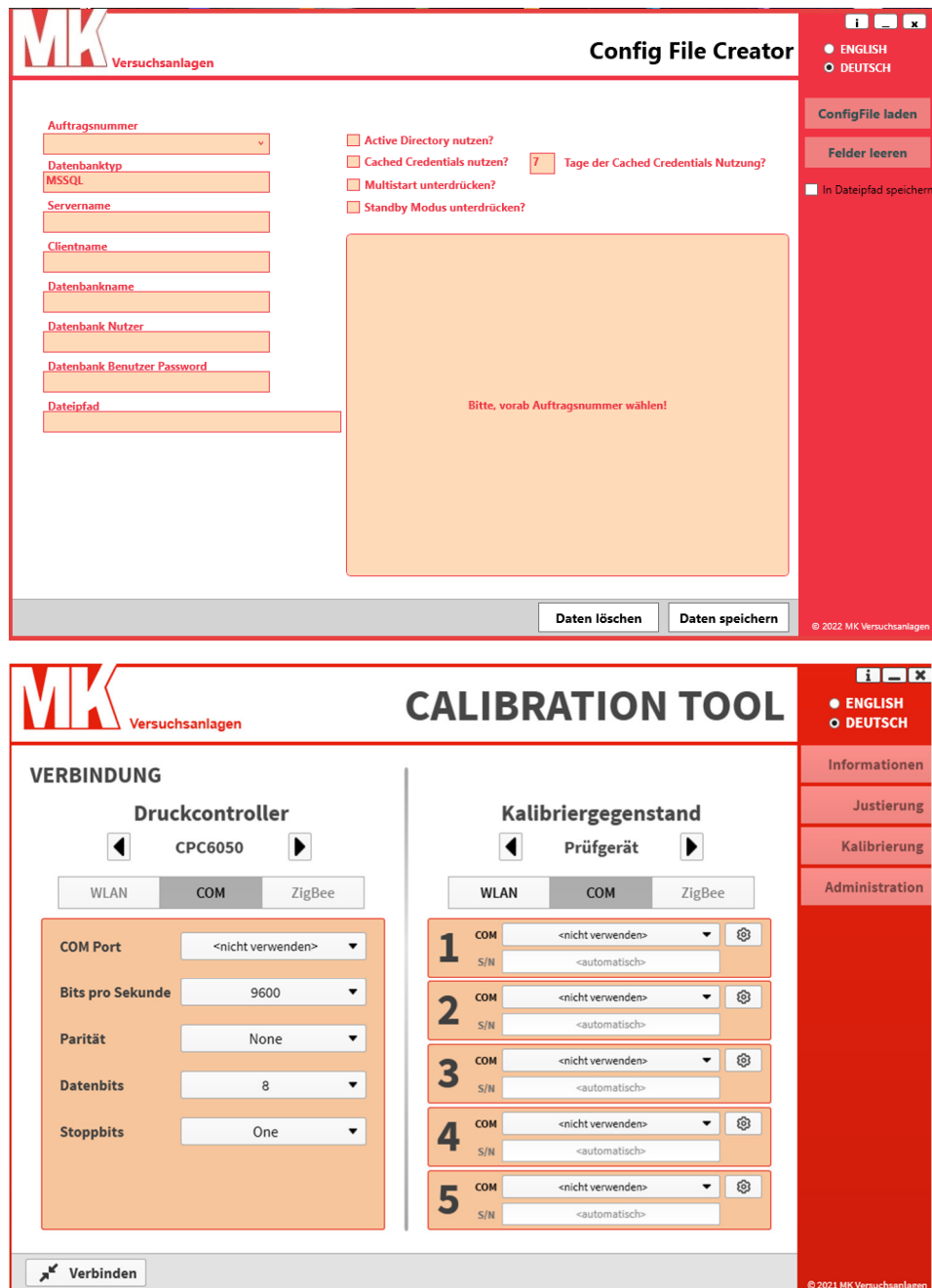
MK_Technisches_Dokument_Interne_Doc_1.1.docx


NUR FÜR INTERNE VERWENDUNG

Beispiele Corporate Design

Das erste Beispiel zeigt die GUI der von der Autorin erstellten Anwendung. Farbnummern wurden vorgegeben, Maße und Anordnungen hat die Autorin aus den bereits bestehenden Tools heraus gearbeitet, ohne deren Code einsehen zu können. Die folgenden drei Beispiele sind bereits bestehende Tools.

Abbildung 17 Vergleich Corporate Design



 **Versuchsanlagen**

Engineering Tool **4**

● ENGLISH
○ DEUTSCH

Informationen
Autokonfig
Allgemein
SAM
Pumpen
Test
BMS
Kommunikation
Reporting


Verbindung

WLANCOMHTTPBMS

IP-Adresse192.168.0.
Port10023
Prüfg. Scanner

☒ WLAN Parameter auslesen

Verbinden oder Konfig. ladenWebinterface

 **Versuchsanlagen**

SERVICE TOOL

● ENGLISH
○ DEUTSCH

Auftragsdaten
Bericht
Administration

VERBINDUNG

Wartungsgegenstand
◀ Prüfgerät ▶

WLANInfrarotZigBee

IP-Adresse192.168.0.
Port10023
Prüfg. Scanner

Verbinden

© 2022 MK Versuchsanlagen