

Werner von Siemens Schule

Frankfurt, den 11.05.2016

Gutleutstraße 333-335

Klasse FI4D

Lehrer: Herr Eick

Marcel Winterer,  
Tom Rosenberger,  
Max Rill,  
Louis Mehring

Projektarbeit:  
TooManyShortscuts

Inhaltsverzeichnis

[1.Vorwort 1](#_Toc450803472)

[2. Pflichtenheft (überarbeitet) 2](#_Toc450803473)

[2.1 Allgemeine Daten zu Arbeitgeber und Arbeitnehmer 2](#_Toc450803474)

[2.1.1 Daten des Arbeitgebers 2](#_Toc450803475)

[2.1.2 Daten des Arbeitnehmers 2](#_Toc450803476)

[2.2 Anforderungen (Lastenheft) 2](#_Toc450803477)

[2.2.1 Zielsetzung 2](#_Toc450803478)

[2.2.2 Weitere Funktionen 2](#_Toc450803479)

[2.3 Umsetzung des Lastenhefts (Pflichtenheft) 3](#_Toc450803480)

[2.3.1 Umsetzung des Auftrags 3](#_Toc450803481)

[2.3.2 Umsetzung der Funktionen (Muss-Kriterien) 3](#_Toc450803482)

[2.3.3 Zusätzliche Funktionen/Kriterien (Soll-Kriterien) 3](#_Toc450803483)

[2.3.4 Produkteinsatz (Anwendungsbereiche & Zielgruppen) 3](#_Toc450803484)

[2.3.5 Benutzeroberfläche 4](#_Toc450803485)

[2.3.6 Darstellung der Meilensteine 5](#_Toc450803486)

[2.3.7 Technische Projektumgebung 5](#_Toc450803487)

[2.3.8 Abgabedaten 5](#_Toc450803488)

[2.3.9 Freigabe/ Genehmigung 5](#_Toc450803489)

[3. Verantwortlichkeiten 6](#_Toc450803490)

[4. Technische Beschreibung des Systems 7](#_Toc450803491)

[4.1 Funktionsweise 7](#_Toc450803492)

[4.2 Einsatzgebiete 7](#_Toc450803493)

[5. Oberfläche (Programmierung) 8](#_Toc450803494)

[5.1 Die Main Form (Hauptfenster) 9](#_Toc450803495)

[5.1.1 Die Steuerelemente 10](#_Toc450803496)

[5.2 Edit Form (Editierungsfenster) 11](#_Toc450803497)

[6. Funktionen der Anwendung (Probleme & Lösungen) 12](#_Toc450803498)

[6.1 Methode RegisterAllShortcuts 13](#_Toc450803499)

[6.2 Methode SpecialIcons 14](#_Toc450803500)

[6.3 Methode DeleteShortcuts 15](#_Toc450803501)

[6.4 Methode UpdateShortcuts 16](#_Toc450803502)

[6.5 Methode lvShortcuts\_DoubleClick / btnNew\_Click 17](#_Toc450803503)

[6.6 Methode Hook\_KeyPressed 18](#_Toc450803504)

[6.7 Methode StartProcess 19](#_Toc450803505)

[6.8 XML-Serialisierung 20](#_Toc450803506)

[6.8.1 Shortcut 20](#_Toc450803507)

[6.8.2 XMLShortcutList 20](#_Toc450803508)

[6.8.3 ListSerializer 20](#_Toc450803509)

[7.Steuerung des Prozesses (MS-Project) 23](#_Toc450803510)

[8. Fazit 24](#_Toc450803511)

# 1.Vorwort

Im Folgenden wird eine Systemdokumentation über die Projektarbeit   
„TooManyShortcuts“ angefertigt. Diese soll besonderen Bezug auf die folgenden Themen nehmen:

* Vergleich mit Pflichtenheft
* Steuerung des Prozesses (MS-Projekt)
* Technische Beschreibung des Systems (Funktionsweise und Einsatzgebiete)
* Oberfläche der Anwendung
* Programmtechnische Funktionen der Anwendung (Probleme & Lösungen)
* Verantwortlichkeiten
* Fazit

Im Themenbereich „Funktionsweise und Einsatzgebiete“ werden die Funktionsart sowie, die Gebiete, in welcher die Anwendung agiert, genannt. Durch diesen Aufbau des Berichtes bekommt der Leser in diesem Themenbereich eine genaue Übersicht für welche Personengruppen der Manager konzipiert wurde.

Weiterhin werden die Oberfläche der Anwendung erklärt sowie die Programmtechnischen Funktionen der Anwendung, welche mit der Oberfläche agieren. Dabei wird auf das Zusammenspiel der Oberflächenkomponenten aber auch der internen Methoden und dem Zusammenhang der Klassen angesprochen.

Der zweite Themenbereich „Verantwortlichkeiten“ klärt über die Verantwortlichkeiten des Projekts auf. Es wird beschrieben wer der Beteiligten sich mit welchem Teilbereich des Projektes beschäftigt hat.

Das letzte Themengebiet „Besondere Probleme und Lösungen“ beschäftigt sich mit den größten Problemen und Lösungen, welche die Programmierung des ShortcutManagers darstellte. Es werden Probleme visuell durch Screenshots dargestellt und verdeutlicht. Auch auf den Denkansatz zur Lösung sowie die problemlösende Programmierung wird genauestens eingegangen.

In der kompletten Dokumentation werden die im Pflichtenheft vorhandenen Meilensteine aufgezeigt sowie hervorgehoben. Zusätzlich werden in allen 5 Themenbereichen die jeweiligen Themen durch Bilder veranschaulichen und untermauert.

# 2. Pflichtenheft (überarbeitet)

## 2.1 Allgemeine Daten zu Arbeitgeber und Arbeitnehmer

### 2.1.1 Daten des Arbeitgebers

Der Auftrag wurde durch Herrn Eick erteilt. Die Projektarbeit wird im Rahmen der Werner von Siemens Schule in Frankfurt durchgeführt. Somit ist auch die ansprechende Adresse: *Gutleutstraße 333-335, 60327 Frankfurt am Main.* Die angegebene Telefonnummer ist somit *„+49 69 21240-100“.*

### 2.1.2 Daten des Arbeitnehmers

Der Auftrag von Herrn Eick wird von Tom Rosenberger, Marcel Winterer, Max Rill sowie Louis Mehring bearbeitet. Diese sind Schüler der Berufsschule und besuchen diese in ihrem 2. Ausbildungsjahr in der Klasse FI4D. Sie sind unter folgenden E-Mail-Adressen zu erreichen.

Louis Mehring: [louis.mehring@web.de](mailto:louis.mehring@web.de)

Marcel Winterer: [marcel-winterer@web.de](mailto:marcel-winterer@web.de)

Max Rill: [max-rill@gmx.de](mailto:max-rill@gmx.de)

Tom Rosenberger: [Tom.Rosenberger@outlook.de](mailto:Tom.Rosenberger@outlook.de)

## 2.2 Anforderungen (Lastenheft)

### 2.2.1 Zielsetzung

Es soll eine Anwendung im Zuge eines Projektes in einer aktuellen Programmiersprache (Java Version 7, Visual Basic, C#, C++) programmiert werden. Diese soll mit einer Datenbank oder einer XML-Datei interagieren.

### 2.2.2 Weitere Funktionen

Folgende Funktionen soll dieses Programm besitzen und ausführen können:

* Verwalten von Shortcuts (anlegen, löschen, ändern, anzeigen)
* Shortcuts sind über Tastatur oder Kurzschreibweise ausführbar
* Starten des Managers bei Systemstart
* Benachrichtigung bei veralteten Shortcuts

## 2.3 Umsetzung des Lastenhefts (Pflichtenheft)

### 2.3.1 Umsetzung des Auftrags

Als Projekt wurde eine Anwendung, welche aufgrund von Tastenkürzel systemweit Programme ausführt und andere Aktionen automatisiert gewählt. Diese Programme werden auch mit dem Synonyme Shortcutmanager betitelt.

Als Programmiersprache wurde sich für C# aufgrund des Unterrichts in dieser Programmiersprache entschieden. Zusätzlich wird das Programm auf Grundbasis von .Net Framework 3.5 programmiert um eine reibungslose Programmausführung unter den gängigen Windows Betriebssystemen (XP mit Service Pack 3 ,7,8,10) zu gewährleisten, da in allem dieser Betriebssysteme .Net Framework in der Version 3.5 standardmäßig vorhanden ist.

### 2.3.2 Umsetzung der Funktionen (Muss-Kriterien)

Es werden die folgenden Funktionen umgesetzt:

* Verwalten von Shortcuts (anlegen, löschen, ändern, anzeigen)
* Shortcuts sind über Tastatur ausführbar
* Shortcuts sind über Kurzschreibweise in Fenster ausführbar
* Starten des Managers bei Systemstart
* Benachrichtigung bei veralteten Shortcuts

Somit werden alle vom Auftraggeber gewünschten Kriterien umsetzbar sein.

### 2.3.3 Zusätzliche Funktionen/Kriterien (Soll-Kriterien)

* Hinzufügen von Shortcuts über Drag & Drop
* Ausführen des Managers in eigenen Thread
* Doppelte Shortcut-Belegung soll verhindert werden
* Ausführung des SCM auf Linux

### 2.3.4 Produkteinsatz (Anwendungsbereiche & Zielgruppen)

Das Produkt dient zur Verwaltung von selbst definierten Shortcuts. Das Programm soll für Nutzer, welche den alltäglichen Gebrauch mit dem PC automatisieren wollen, konzipiert werden. Es soll durch eine einfache und übersichtliche Oberfläche einen einfachen Umgang mit dem Programm ermöglichen.

### 2.3.5 Benutzeroberfläche

Der ShortcutManager wird in seiner Oberfläche folgende Elemente besitzen:

* Suchen Label
* Suchen Textfeld
* Erstellen Button
* Löschen Button
* Listview zum Anzeigen der Shortcuts
* Editform
* Shorthandform
* ContextMenuStrip zum Auswählen von Autostart und Beenden

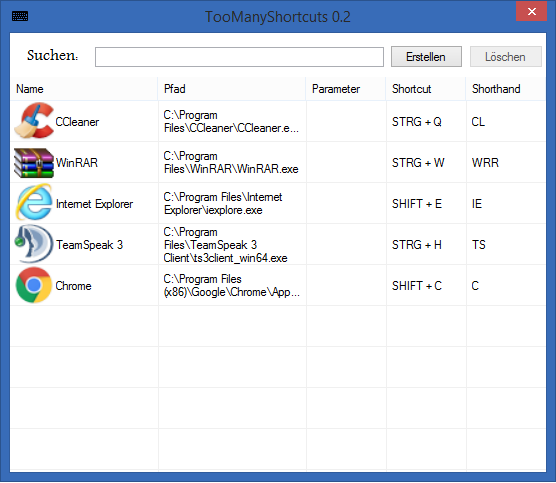


Abbildung : Screenshot Oberfläche

### 2.3.6 Darstellung der Meilensteine

|  |  |
| --- | --- |
| **Funktionscode** | **Funktionalität** |
| F0010 | Shortcuts anzeigen |
| F0020 | Shortcuts anlegen |
| F0030 | Shortcuts ändern |
| F0040 | Shortcuts löschen |
| F0050 | Shortcut direkt über Tastatur ausführen .. |
| F0051 | .. oder Shortcut in Kurzschreibweise ausführen |
| F0060 | SCM mit Systemstart von Windows starten |

Folgende Meilensteine werden für ein erfolgreiches Erreichen des Projektziels benötigt:

Detaillierte Darstellung in dem Anhang.

### 2.3.7 Technische Projektumgebung

Als technische Projektumgebung wurden sich für die Betriebssysteme Windows 7, Windows 8 sowie Windows 10 entschieden. Auf allen diese Umgebungen war das entsprechende .Net Framework in der Version 3.5 enthalten.

### 2.3.8 Abgabedaten

Pflichtenheftabgabe: 13.02.2016

Abgabe des Projektes: 12.05.2016

### 2.3.9 Freigabe/ Genehmigung

In diesem Pflichtenheft wurden die Anforderungen des Lastenhefts sowie deren Umsetzungen in Muss-, Soll- und Wunschkriterien genannt. Weiterhin wurden Meilensteine und die Technische Projektumgebung genannt. Abschließend wurden die Abgabedaten beschrieben.

Auftragnehmer:

Marcel Winterer: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Auftraggeber:

Herrn Eick: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# 3. Verantwortlichkeiten

Im Folgenden werden die Verantwortlichen Ersteller dieses Projekts genannt sowie ihre jeweiligen Einsatzgebiete während des Projekts.

Max Rill programmierte die XML Schnittstelle (inklusive Komponententests) und erstellte das Schema der XML Datei. Er stellte Schnittstellen für Lese und Schreib Vorgänge in der XML Datei zur Verfügung. Weiterhin fertige er den Dokumentationsteil 6.8 an, welcher sich mit den XML Klassen, Methoden und deren Funktionsweise beschäftigte.

Tom Rosenberger erstellte das Projekt auf einer zentralen Verwaltungsplattform (Github) und erstellte dort Vezweigungen sowie den ersten Commit des Programms. Er war an der Projektplanung sowie an Ideen für neue Programm Methoden sowie an der Programmierung der „Icon-Methode“ beteiligt. Er erarbeitete mit Marcel Winterer die Power Point Präsentation zum Projektthema.

Marcel Winter war an der Programmierung einiger Schnittstellen sowie als externer Tester beteiligt. Er erstellte weiterhin das Handout zum Projektthema. Auch überarbeitete er einige Programmstellen und war als ein Motivator der Gruppe zuständig.

Tom Rosenberger sowie Marcel Winterer erstellten weiterhin Unit Tests für einige der vorhandenen Funktionen.

Louis Mehring erstellte sowie programmierte alle Methoden in folgenden Klassen:

* Edit.cs
* Functions.cs
* Keyboardhook.cs
* Main.cs
* ShortcutList.cs
* ShorthandWindow.cs

Sowie alle dazugehörigen Methoden und Eigenschaften. Er programmierte die externe API um und stellte die Kommunikation der Formen untereinander her. Weiterhin überarbeitete er das Lastenheft und fertigte einen Projektplan in Microsoft Project an. Die Dokumentation wurde von ihm bis auf den Punkt 6.8 verfasst. Er testete das Programm ausführlichen im praktischen Gebrauch und stellte Changelogs in Form von Commits in Github für seine Teammitglieder bereit.

Weiterhin wurde im Unterricht von allen vieren über die Oberflächengestaltung sowie die Programlogik diskutiert und beraten.

# 4. Technische Beschreibung des Systems

## 4.1 Funktionsweise

Ein ShortcutManager erlaubt dem Benutzer der Anwendung schnell und effektiv Programme zu öffnen, Webseiten auszuführen sowie Texte einzufügen. Dazu wird lediglich eine vom Benutzer festgelegte/r Tastenkombination / Tastenkürzel benötigt. Diese können unabhängig von dem aktuell geöffneten Programm angewendet werden und sind somit systemübergreifend. Damit können User komplexe Programmaufrufe wie die Windows Kommandozeile, mit spezifischen Parametern (mit anderem Konto ausführen) sehr schnell und effektiv aufrufen. Jedoch wird ein ShortcutManager mit vielen Shortcuts oft unübersichtlich und man vergisst leicht einen Shortcut. Hier schaffen Shorthands Abhilfe, da sie aus Zeichenkombinationen bestehen die leichter zu merken sind, da diese Abkürzungen für das Programm darstellen können. Somit ist der Wiedererkennbarkeitswert von Beispiel dem Shortcut „Internet Explorer“ durch den Shortcut „STRG+I“ weniger gegeben wie durch die Zeichenfolge „IE“.

## 4.2 Einsatzgebiete

Oft verwenden IT-Administratoren sowie sogenannte Power-User den Großteil ihrer Zeit die Tastatur für Eingaben und Steuerung in und von Programmen. Sie greifen eher auf Tasten wie "Tab" zurück anstatt in das nächste Textfeld zu klicken. Weiterhin benutzen sie Windows-Tasten Kombinationen wie "Windows-Taste + D" zum Minimieren aller geöffneten Programme anstatt diese von Hand zu schließen. Somit bedienen sich diese User oft Hilfsprogrammen, welche Ihnen das Leben erleichtern. Ein Shortcut Manager kann eines dieser Programme sein. Im Internet finden sich zahlreiche Programme für diese Funktion. Ein alter, minimalistischer jedoch ressourcenschonender ShortcutManager zum Release von Windows Vista war "QOpen". Er gibt dem User die Möglichkeit mit wenigen Tastendrücken Programme mit definierten Parametern zu starten, verschiedene Tastenkürzel (sogenannte Shorthands) anzulegen und sogar eigene Icons für die Programme anzulegen. Jedoch funktionierte dieser unter Windows 7 und darauffolgenden Windows Versionen nicht mehr und wurde nicht mehr weiterentwickelt. Viele andere ShortcutManager sind sehr überladen an Funktionen und oft nicht portabel. Deswegen hat sich es angeboten einen ShortcutManager zu entwickeln der eine einfache aber effiziente Arbeitsmethodik besitzt und zudem auch noch portabel ist.

# 5. Oberfläche (Programmierung)

Bei der Programmierung der Oberfläche kam es widererwartend zu einigen Problemen, deren Lösungen des Öfteren einige Zeit beansprucht haben. Zwar stellt C# unter Visual Studio viele Komponenten zur Darstellung von Sachverhalten zur Verfügung jedoch müssen diese bei größeren Projekten explizit benannt, in richtige Beziehung" zueinander gesetzt sowie strukturiert werden. Da sich das Projekt auf Windows Forms basiert, konnten viele Möglichkeiten zum Gestalten der Anwendung, welche durch WPF (Windows Presentation Foundation) gegeben wären, nicht genutzt werden. Dies vermindert jedoch keineswegs die Benutzbarkeit des Programms, da dieses eine simplen, performanten und übersichtlichen Shortcutmanager darstellen soll. Dieser benötigt keine großen Animationen, welche Ressourcen verbrauchen. Auch benötigen einige WPF Funktionen das Microsoft Net Framework 4.0. Dieses konnte nicht eingesetzt werden, da wir in unserem Lastenheft die Net Framework Version 3.5 als Basis für unser Programm deklariert haben.

In den folgenden Problembeschreibungen wird grundlegend aber nicht explizit auf Syntax und genaue Funktion von Schlüsselworten, sowie Standardfunktionen von C# eingegangen, da diese zwar dem Leser einen Einblick in die jeweiligen Funktionalitäten sowie Problemstellungen und Lösungen geben sollen aber eine komplette Analyse des Programmcodes den Rahmen dieser Projektarbeit sprengen würde.

Da auch während des Schreibens der Dokumentation immer wieder Fehler behoben und Performance verbessert wurde, kann der finale Quellcode Abweichungen zu den hier gezeigten Screenshots besitzen.

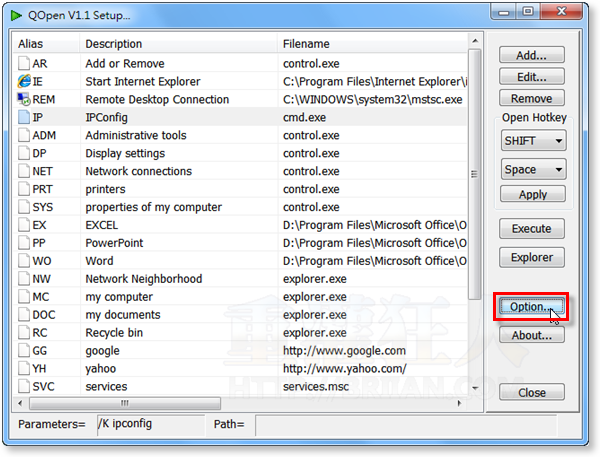


Abbildung 2: Screenshot QOpen

## 5.1 Die Main Form (Hauptfenster)

Um das spätere Anordnen der einzelnen Komponenten zu erleichtern, wurde das eigentliche Fenster mithilfe mehrerer sogenannter Panels in zwei kleinere Bereiche unterteilt. Dies hat auch den Vorteil, dass sich die sich Komponenten bei einer Änderung der Fenstergröße nicht anderen Stellen neu positionieren, sondern sich immer zusammen mit den anderen Komponenten ihres Panels bewegen.

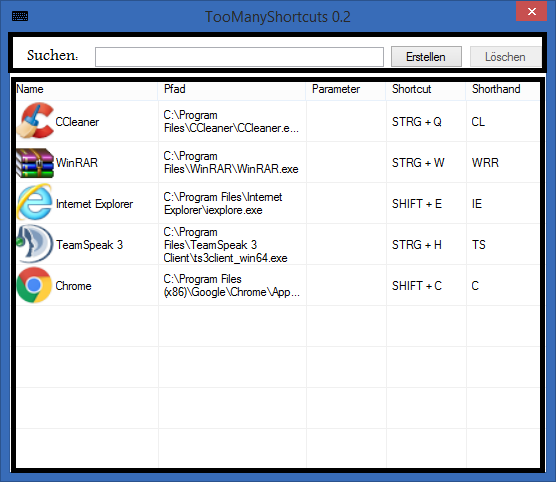
Die Panels wurden auf dem Fenster wie folgt positioniert:

Abbildung : Screenshot Panelunterteilung

Dies sieht anfangs etwas komplex aus, ist aber recht simpel und schnell erklärt. Das Fenster an sich ist in zwei Panels unterteilt: das obere Panel, in dem Sucheingaben getätigt werden, und das untere, in welchem sich die auszugebende Liste mit den Shortcuts befindet. In dem unteren Panel wird beim Programmstart eine Form zur Darstellung der Shortcuts geladen. Links neben den Shortcuts werden Icon der jeweiligen Programme sowie Custom Icons falls der Pfad z.B. auf eine „.mp4“ Datei verweist. Dies wäre nicht mit einem ListBox-Element möglich, da dieses keine Darstellung von Icon unterstützt. Weiterhin sind die Columns dynamisch festgelegt und bieten somit bestmögliche Flexibilität.

### 5.1.1 Die Steuerelemente

Neben den zwei Panels befinden sich auf dem Fenster:

* Ein Label
* Eine Textbox
* Zwei Buttons
* Eine Listview

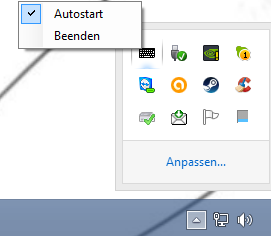
Die **Textbox** wird als Suchmaske für die Shortcuts verwendet. Durch sie ist es möglich sich je nach Wunsch des Anwenders nach Name, Pfad und Shorthand zu suchen. Hierbei wurde sich gegen eine Durchsuchung der Shortcuts sowie Parameter entschieden, da diese keinen Nährwert für eine Suche darstellen. Es wurde sich weiterhin gegen einen „Suchenbutton“ entschieden, da dies optisch veraltet ist und sich eine Laufzeitnahe Suche nach Shortcuts besser darstellt.

Die **Buttons** besitzen die relevantesten Funktionen in der Oberfläche, da sich durch sie Shortcuts anlegen sowie löschen lassen. Das Anlegen eines Shortcuts kann nur über den Button „Anlegen“ erfolgen. Löschen kann man Shortcuts jedoch auch mit der „Entf“ (Entfernen) Taste auf der Tastatur. Der Löschen Button ist anwählbar wenn kein Element in der Listview selektiert wurde.

Die Listview zeigt die Shortcuts in einer übersichtlichen weise an und ist in die Teilbereiche (Columns) Name, Pfad, Parameter, Shortcut und Shorthand unterteilt. Hier werden die Informationen zum jeweiligen Shortcut angezeigt.

Das **Label** gibt einzig und allein Informationen zu dem Steuerelement der Suche und hat sonst keine weitere Funktion.





Weiterhin besitzt die Form noch zwei Steuerelemente, welche jedoch in der Ausführung nicht direkt im Fenster sichtbar sind. Eines dieser Elemente ist das ContextMenuStrip, welches uns die Anzeige von Rechtsklickoptionen ermöglicht. Somit können wir das Programm im Systemtray beenden oder den Autostart aktivieren sowie deaktivieren.

Abbildung : Screenshot Systemtray

Das NotifyIcon hat die Funktion ein Icon in dem Systemtray darzustellen wenn unser Programm im Hintergrund läuft. Es ist direkt mit dem ContextMenuStrip verbunden.

## 5.2 Edit Form (Editierungsfenster)

In dem Editierungsfester werden Shortcuts angelegt und geändert. Alle Steuerelmente befinden sich in einer Groupbox. Alle Textboxen werden automatisch befüllt, wenn man in der Hauptform ein Doppelklick auf ein selektiertes Item ausführt. Somit ist eine optimale und schnelle Abänderung von Shortcuts kein Problem. Die Textbox für den Pfad und den Namen wird automatisch befüllt, wenn ein Drag Drop von einer Datei, einem Ordner oder einer Webseite auf das Textfeld des Pfades erfolgt. Der Button „Speichern“ ist standartmäßig deaktiviert, wenn nicht folgende Bedingungen gegeben sind:

* Alle Textboxen außer der Parametertextbox dürfen nicht leer sein
* Der Shortcut muss einmalig sein und darf nicht schon einmal vergeben sein
* Selbiges gilt für den Shorthand, welcher zusätzlich nur 1-3 Zeichen lang sein darf
* Zusätzlich dürfen keine Windowsspezifischen Shortcuts wie „STRG+C“ etc. verwendet werden.

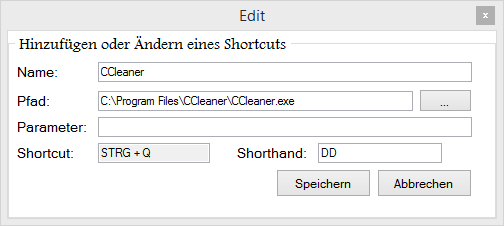


Abbildung : Screenshot Editform

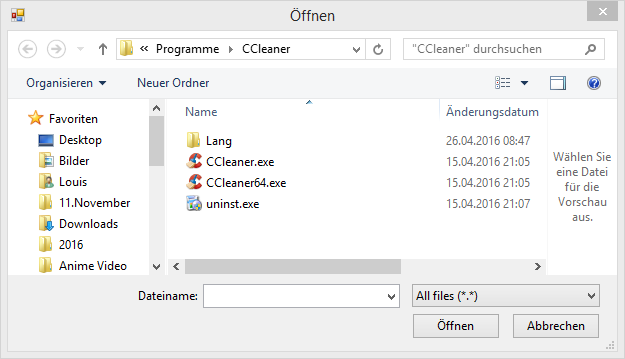
Weiterhin existiert noch der Button „…“. Dieser öffnet einen OpenFileDialog der dem User die Möglichkeit bietet eine beliebige Datei auszuwählen und somit auch den Pfad ohne Drag & Drop anzugeben.

Abbildung : OpenFileDialog

Weiterhin sind alle TextBoxen außer die ShortcutTextbox frei beschreibbar. Somit sind auch händische Eingaben möglich.

# 6. Funktionen der Anwendung (Probleme & Lösungen)

Bei dem Programmcode wurde sich sehr an die Struktur sowie Vorgehensweise des Unterrichts orientiert. Es wurden klare Abgrenzungen zwischen Klassen definiert. Weiterhin wurde eine klare Logik und Programmierstruktur vorgegeben. Weiterhin konnten auch viele Themen des Unterrichts (statischen Methoden, Parameterübergabe, Konvertierung etc.) wieder aufgegriffen werden. Durch diese klaren Parallelen fällt es einem Schüler der FI4D sehr einfach die jeweiligen Programmierschritte und die Gedankengänge des Programmierers nachzuvollziehen. Jedoch wurde der Programmcode auch so konzipiert, dass er auch für einen Außenstehenden leicht verständlich und strukturiert ist. Die Oberfläche sowie die Funktionsalgorithmen besitzen eine eigene Klasse. Somit ist eine Abspaltung in Struktur sowie Programmcode gegeben.

Somit wurden in der Klasse „ShortcutList.cs“ sowie „Functions.cs“ folgende Methoden für die Shortcut-Erstellung sowie andere wichtige Funktionen angelegt:

* „ShortcutList.cs“
  + RegisterAllShortcuts
  + SpecialIcons
  + DeleteShortcuts
  + UpdateShortcuts
  + lvShortcuts\_DoubleClick / btnNew\_Click
  + txtSearch\_Changed
* „Functions.cs“
  + Hook\_KeyPressed
  + StartProcess
  + RegisterHotKey

Die Methoden wurden so aufgebaut, dass sie komplett unabhängig voneinander agieren können. Somit können die Methoden aus dem C# Quellcode einfach per „Copy-Paste“ in andere Klassen oder Codes implementiert und genutzt werden. Im weiteren Verlauf werden des Arbeitsprozessberichtes wird auf jede dieser unabhängigen Klassen eingegangen und deren Aufbau, Aufgabe sowie deren Bedeutung für das gesamte Rechenkonzept erläutert. Auch werden die genauen „Problemstellen“ im Programmcode aufgezeigt und die entsprechende Lösung wird erklärt und veranschaulicht.

Schlussendlich können Leser sowie Auftraggeber (Herr Eick), welche eventuell einen eigenen „ShortcutManager“ programmieren wollen sich über gewissen Problematiken und Lösungsansätze vergewissern und den Aufwand jeder Methode betrachten.

## 6.1 Methode RegisterAllShortcuts

Diese Methode erlaubt dem Benutzer, alle Shortcuts auf einmal zu registrieren. Dies wird unter anderem beim Programmstart benötigt jedoch auch bei anderen Programmzwischenschritten. Es wird kein Übergabe Parameter benötigt, da die Methode auf die öffentliche Variable XMLList zugreift und diese mit einer Foreach Schleife Schritt für Schritt durchläuft. Es werden somit Schritt für Schritt die „RegisterHotKey“ Methode der Klasse „Functions“ aufgerufen auf welche später eingegangen wird.

Problematiken beim Registrieren des Shortcuts war, dass eine sehr lange Zeit das XML-Schema nicht festgelegt wurde. Somit musste Übergangsweise mit einem DataTable gearbeitet werden sowie einem String Array. Somit wurde viel Arbeit in Methoden investiert um eine vergleichbare Testumgebung zu schaffen, jedoch mussten diese später komplett abgeändert werden. Der Lerneffekt war hier jedoch sehr groß, da mit komplett neuen Steuerelementen und ihren Methoden gearbeitet wurde. Somit wurde auch das Thema Databinding in den Vordergrund geschoben, was sich in Bezug auf den Lernerfolg im Thema Datenbanken sehr positiv auswirkte.

Weitere Problematiken stellte die Formatierung des XML-Schemas „Shortcut.Keycombo“ dar, da dieser erst fehlerhaft war.

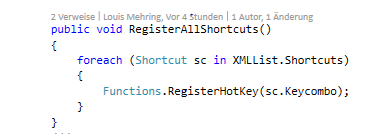


Abbildung 7: Methode (RegisterallShortcuts)

Auf die Frage wieso eine so kleiner Codeteil nicht einfach in die größere Methode „UpdateShortcuts“ eingebunden wird muss man sagen, dass es Szenarien gibt in denen alle Shortcuts zwar neu registriert werden müssen jedoch die Liste nicht neu geschrieben werden muss. Dies kann der Fall sein, wenn z.B. die Form Edit aufgerufen und mit dem Abbrechen Button wieder geschlossen wird. Die Form Edit löscht alle Shortcuts um Meldungen richtig anzeigen zu können. Somit müssen diese auch bei einem Abbrechen des Programms neu registriert werden aber nicht neu in die Listview geschrieben werden. Dazu dient also diese Funktion.

## 6.2 Methode SpecialIcons

In dieser Methode werden Speziale Icons für Programme festgelegt, welche z.B. standardmäßig keine Icons besitzen (Webseiten). Die Abfrage bezieht sich auf viele Ebenen und wird mit folgender Struktur durchgeführt:

Es werden der jeweilige Shortcut, welchem die Special Icons zugeteilt werden sollen sowie das Listview Item in welchem sie angezeigt werden sollen, benötigt.

Es wird zuerst abgefragt ob das Programm / der Ordner existiert. Ist dies der Fall wird überprüft ob es sich wirklich um eine Datei handelt. Trifft dies zu wird die ImgList mit Icons befüllt. Diese Icons sind die Anzeige Icons der Dateien, welche auch von Windows dargestellt werden. Dort wird die maximale Auflösung abgerufen und zurückgegeben. Dann wird überprüft ob es sich bei der Datei um eine „mp4“ Datei, eine „mp3“ oder ein Bild (jpg,png) handelt. Ist dies der Fall wird das nötige Special Icon eingefügt und überschreibt in der Listview das programmspezifische Icon von Windows.

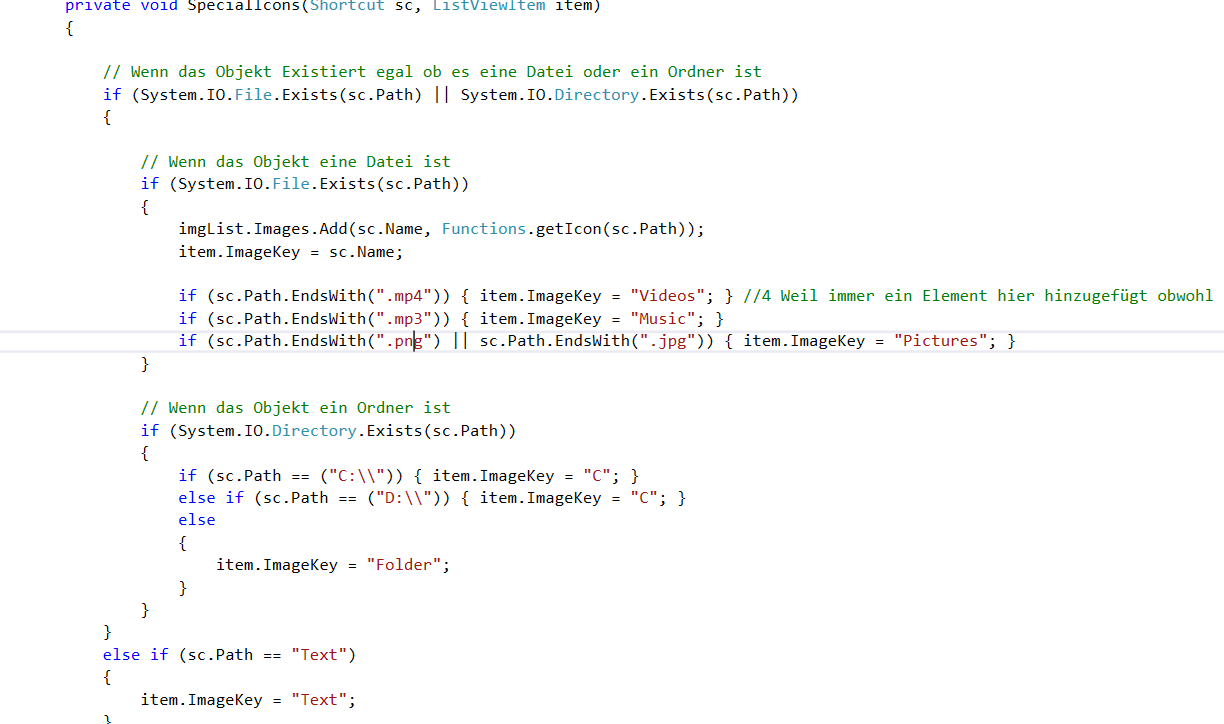
So sieht auch der Verlauf bei der Ordnerüberprüfung aus ist hier der Pfad „C:\“ oder „D:\“ gegeben werden spezielle Icons verwendet.

Abbildung : Methode (SpecialIcon)

Falls keine der Abfragen zutrifft wird davon ausgegangen, dass es weder eine Webseite, noch eine Datei noch ein Ordner ist. Somit wird ein Error Icon ausgegeben, welches dem Nutzer symbolisiert, dass der Shortcut womöglich Fehler auslöst oder ohne Funktion ist.

## 6.3 Methode DeleteShortcuts

Diese Methode ist wie der Name vermuten lässt für das Löschen der Shortcuts zuständig. Sie muss ausgeführt werden wenn die Entfernen Taste oder der Löschen Button gedrückt wird. Hier wurde sich einer LinQ Funktion bedient, welche von einer List Funktion Namens „Find“ abgeleitet ist.

Diese ermöglicht es auf extrem performante weise und ohne Schleife eine Liste zu durchsuchen. Somit wird in einem Try-Catch Block, welcher eine Exception wirft, wenn kein Element in der Listview selektiert wurde, nach dem Shortcut gesucht der gerade in der Listview selektiert wurde. Dieser wird dann durch den Remove Befehl der Liste entfernt. Weiterhin wird auch dieser Eintrag in der Listview gelöscht. Der Listserializer schreibt dann lediglich die veränderte ShortcutListe in die XML-Datei und unregistriert dann alle Shortcuts um sie erneut zu registrieren und die Listview komplett zu erneuen.



Abbildung 9: Methode (DeleteShortcut)

Auch in dieser Methode musste auf viele Fehler geachtet werden, da man daran denken muss, dass der Shortcut zwar aus der XML-Datei sowie aus der Listview Liste gelöscht ist jedoch noch registriert ist. Da sich einzeln Shortcuts nicht unregistrieren lassen müssen alle Shortcuts gecleart und neu eingelesen werden. Es wurde versucht in der „Keyboardhook“ diesem unperformanten Szenario vorzubeugen jedoch ist diese Klasse an eine externe API gebunden. Somit gestaltete sich eine umschreiben des Codes zunehmend schwer und wurde auf spätere Fixes verschoben.

## 6.4 Methode UpdateShortcuts

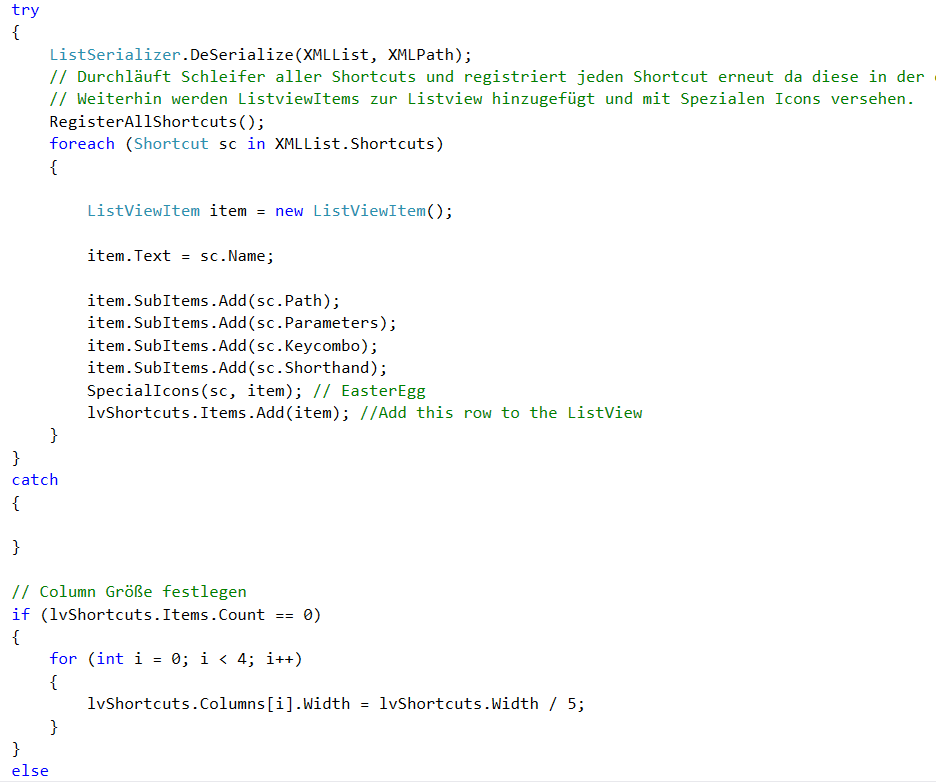
Diese Methode muss aufgerufen werden falls Shortcuts geändert, gelöscht, erstellt werden. Sie bedient sich der „RegisterAllShorts“ Methode sowie der „SpecialIcons“ Methode, welche oben erläutert wurden. Zuerst werden einige Abfragen zur Editform gestellt. Danach wird wie im unteren Bild zu sehen die „try catch Schleife“ ausgeführt. Diese wird vom ListSerializer benötigt um eine Exception zu werfen falls die XML-Datei nicht vorhanden ist. Danach wird für jeden Shortcut ein ListViewElemt erstellt, welches die nötigen werden durch die Shortcuteigenschaften zugewiesen bekommt. Weiterhin kommt unten die dynamische Column „Berechnung“ zum Einsatz. Diese überprüft die Größe des Gesamtfensters und teilt diese durch fünf um jeden der Columns den gleichen Platz zuzuweisen.

Abbildung : Methode UpdateShortcuts

Problematiken traten durch die Listview, welche sich trotz der Eigenschaft Dock.Fill nicht auf die richtige Größe skalieren lies. Dies wurde jedoch mit einer Abfrage, welche die Breite der Listview explizit auf die Größe der Main Form setzte behoben.

## 6.5 Methode lvShortcuts\_DoubleClick / btnNew\_Click

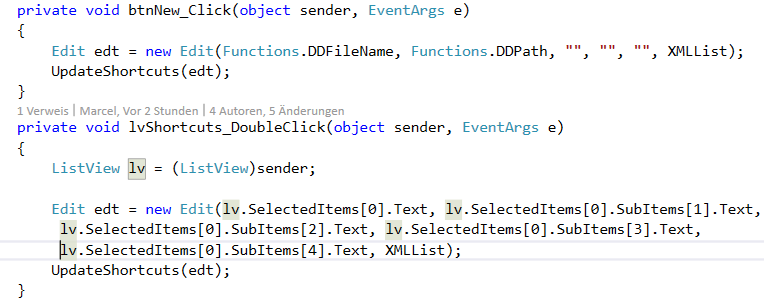
Diese beiden Methoden rufen die EditForm auf und befüllen sie mit verschiedenen Werten. Falls der Button „Erstellen“ gedrückt wird, öffnet sich die Editform mit den Parametern „Functions.DDFileName“, welcher leer ist falls kein Objekt per „Drag & Drop“ in die Form gezogen wurde. Falls jedoch ein Doppelklick auf ein selektiertes ListViewItem ausgeführt wird, werden dessen Werte an die Edit Form übergeben.

Abbildung : Methode lvShortcuts & btnNew Events

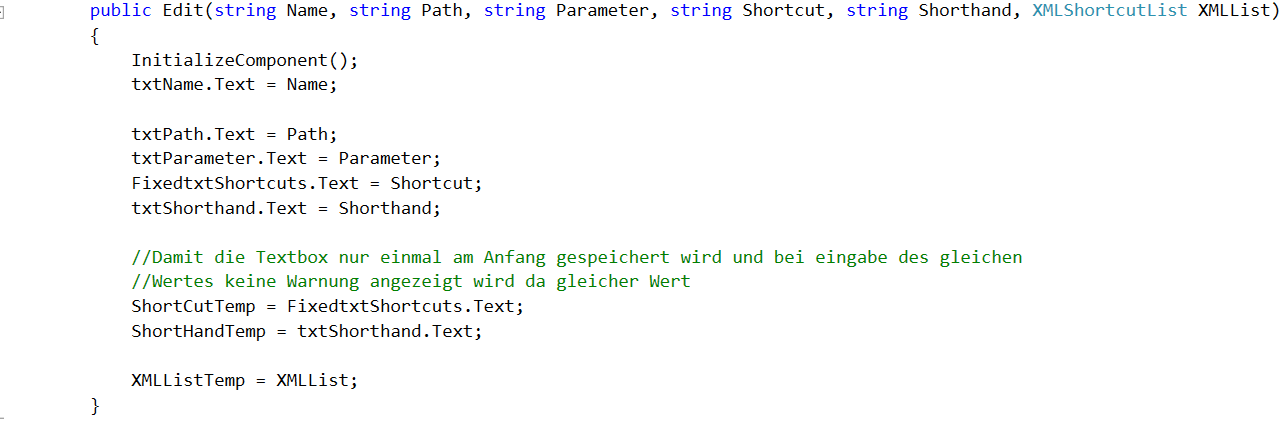
Auch bei diesen Funktionen kam es häufig zu Fehlern da erst eine statische DataTable anstatt einer nicht statischen XMLList genutzt wurde. Die Übergabe der beiden Methoden musste mit einem zweiten Konstruktor in der Editform gelöst werden.

Abbildung : Konstruktor Edit

Dieser Konstruktor weist einer temporären XML List die originale XMLList zu um diese dann Klassenweit zu verwenden. Am Ende wird die TempXMLList zur aufrufenden Form zurückgegeben und überschreibt dort die originale XMLList.

## 6.6 Methode Hook\_KeyPressed

Abbildung : Methode hook\_KeyPressed

Diese Methode hat einige Schwierigkeiten bereitet, da sie jede Menge Fehleingaben einfangen musste. Zuerst muss ein Application.DoEvents() ausgeführt werden um nicht den Shortcut Beispielsweise doppelt zu starten. Danach muss kontrolliert welchen Modifier der mitgegebene Parameter „e“ besitzt. Weiterhin soll bei der Tastenkobination „STRG+ SPACE (Leertaste)“ das ShorthandWindow geöffnet werden. Dies befand sich nie im Vordergrund selbst mit einem TopMost Befehl. Einerseits dürfte das Main Window nicht minimiert sein und zweitens musste überprüft werden ob ShorthandWindow schon gestartet worden ist. Wenn dies der Fall war dann musste der Focus wieder auf das Fenster gelegt werden. Somit wurden durch einen „Hide()“ Befehl anstatt eines Minimized in der Main sowie Focus und BringMeToFront in dem Shorthandwindow das gewünschte Ergebnis erreicht.

## 6.7 Methode StartProcess

Abbildung : Methode StartProcess

StartProcess verursachte bei der automatisierten Eingabe von Text, welche über ein SendKeys-Event geregelt war dauernt den Aufruf aller möglichen Shortcuts. Dies geschah, da während der Eingabe noch die STRG Taste gedrückt war und somit der Start eines Shortcuts durch das SendKeys-Event ausgelöst wurde. So hat Beispielsweise der Text „asd“ die Shortcuts STRG + A, STRG + S und STRG + D ausgeführt.

Dieses Problem wurde gelöst indem das SendKeys-Event durch das Clipboard-Event, welches den jeweiligen Text in die Zwischenablage schreibt und dann einfach den Tastendruck STRG + V (Einfügen), ersetzt wurde. Weiterhin wurde sich nach längerer Zeit für den Process als „Starter“ des Shortcuts entschieden, da er den riesigen Vorteil hat über ProcessInfo den Parameter als einzelnes Argument anzugeben. Weiterhin besitzt er viele Einstellung wie z.B. die WindowsState beim ausführen des Fensters etc.

Der „Try-Catch“ Block wirft eine entsprechende Meldung aus wenn der Parameter ungültig ist oder gar das Programm nicht vorhanden ist. Falls Process.Start mit einem Link ausgeführt wird, dann startet der Process den entsprechenden Standardbrowser und verweist auf die entsprechende Seite. Somit ist auch hier keine weitere Programmierung notwendig.

## 6.8 XML-Serialisierung

Um die Shortcuts als XML-Dokument serialisieren(schreiben) und deserialisieren(lesen) zu können werden drei Klassen benötigt: Die eigentliche Shortcut-Klasse, deren Attribute dem vorgegebenen Format im XML-Schema entsprechen. Die XMLShortcutList-Klasse, die hauptsächlich dazu dient alle Shortcuts auf einmal zu serialisieren und außerdem noch Informationen über das XML-Wurzelelement und die Referenz auf das Schema beinhaltet. Die ListSerializer-Klassse, die Shortcuts liest, schreibt und gegen das Schema validiert.

### 6.8.1 Shortcut

Erzeugt ein Shortcut-Objekt mit den dazugehörigen Eigenschaften "Name", "Shorthand"(Kurzschreibweise zur Nutzung mit strg+space), "Keycombo"(Tastenkombination), "Path"(Pfad, der aufgerufen werden soll).

Darüber hinaus gibt es noch eine optionale Eigenschaft für zusätzliche Parameter mit denen der Shortcut ausgeführt werden soll, die nur serialisiert wird, wenn sie einen Wert hat.

### 6.8.2 XMLShortcutList

Hat zwei Eigenschaften: ein Listenobjekt, in das alle Shortcuts geladen werden und eine Variable, die den Pfad für das XML-Schema festlegt und mit zusätzlichen Serialisierungsanweisungen versehen ist um beim Erstellen der XML-Datei einen Verweis auf das Schema einzutragen.

### 6.8.3 L**istSerializer**

Diese Klasse regelt den gesamten Vorgang des Lesens aus XML und Schreibens in XML.

Sie hat drei Methoden:

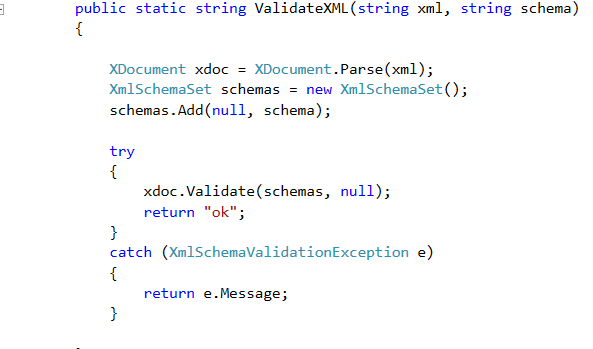
ValidateXML nimmt als Argumente einen string, der XML enthält und einen Dateipfad für ein XML-Schema an. Die Methode prüft, ob der Inhalt der string-Variablen ein nach dem gegebenen Schema gültiges XML-Dokument ist und gibt entweder den string "ok" zurück oder im Falle eines Validierungsfehlers die entsprechende Fehlermeldung.

Abbildung : ValidateXML

SerializeXML nimmt als Argumente das zu serialisierende XMLShortcutList-Objekt sowie den Dateipfad, an dem das fertige Dokument gespeichert werden soll, an. Die Methode nutzt die in C# vorhandene "XMLSerializer"-Klasse und einen StringWriter um die Shortcuts einen nach dem anderen als XML in eine Stringvariable zu schreiben, die der ValidateXML-Methode zusammen mit dem Wert der in der gegebenen Liste hinterlegten Pfadangabe für das dazugehörige Schema übergeben wird. Nach erfolgreicher Validierung wird der XML-string als Datei geschrieben.

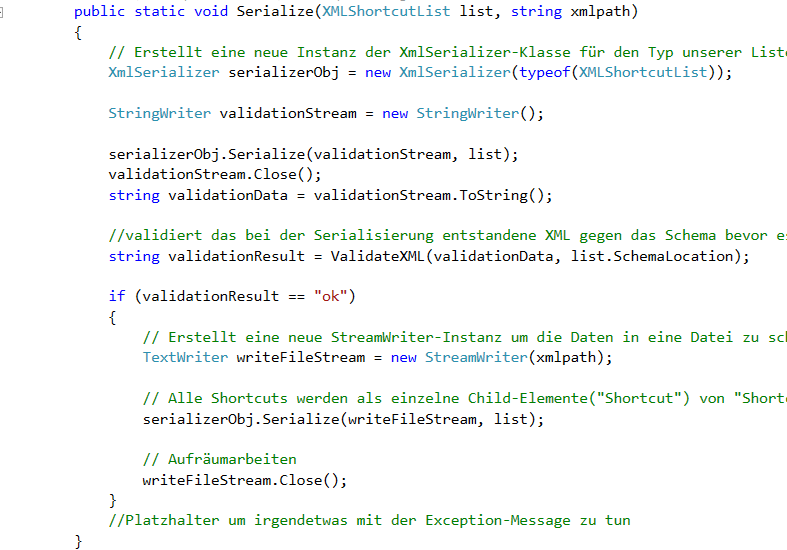
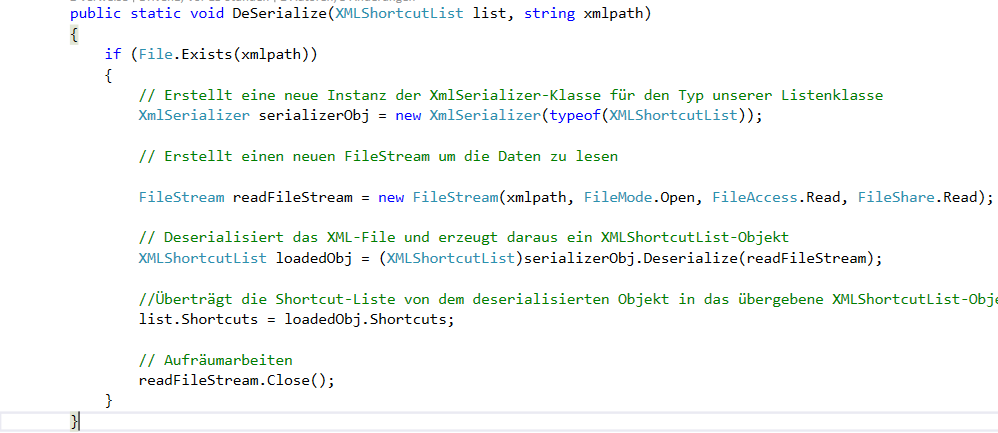


Abbildung 16: XML Serialize

DeSerializeXML nimmt als Argumente ein XMLShortcutList-Objekt, in das die Shortcuts geladen werden sollen, und den Dateipfad für das zu lesende XML-Dokument an. Die Methode liest über die C#-eigene "FileStream"-Klasse das gegebene Dokument aus und übergibt es dann dem XMLSerializer, der aus dem eingelesenen XML Shortcut-Objekte erzeugt und in die gegebene Liste füllt. 

XML-Schema "SCM\_Schema.xsd"

In dem Schema wird festgelegt, wie die Shortcuts in XML aufgebaut sein müssen.

Jeder Shortcut wird als eigenes xs:element gespeichert, diese Elemente sind komplexe Typen (im Gegensatz zu einfachen Datentypen wie xs:string und xs:int), gekennzeichnet durch "xs:complexType", und beinhalten mehrere Unterelemente, die in einer xs:Sequence (Abfolge) beschrieben werden. Die einzelnen Elemente spiegeln die Eigenschaften der Shortcut-Klasse wieder und sind entweder Pflichtfelder (müssen genau 1-mal auftreten) oder optionale Felder (tritt 0- oder 1-mal auf). Mit "xs:unique" werden noch mehrere Regeln festgelegt, welche Felder im Kontext der Shortcutliste einzigartige Werte haben müssen. In diesem Fall handelt es sich dabei um die Felder "Shorthand", "KeyCombo" und "Name". Ein gültiges XML-Dokument darf keine zwei Einträge haben bei denen der Inhalt eines dieser Felder genau übereinstimmt. Die einzelnen Shortcut-Elemente sind Teil der Sequenz des übergeordneten komplexen Elternelements "Shortcuts", dass die Listen-Eigenschaft der XMLShortcutList-Klasse abbildet.

# C:\Users\dengler\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Gantdiagram.jpgC:\Users\dengler\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Programplan.jpg7.Steuerung des Prozesses (MS-Project)

Abbildung : MS-Project Tagesplan

Abbildung 17: GanttDiagram

# 8. Fazit

Die Programmierung des ShortcutManager hat uns die Programmierung mit C# im fortgeschrittenen Bereich nähergebracht. Es wurden viele Fragestellungen zu Befehlen sowie Syntaxstellungen aufgeworfen, welche oft nur mit systematischen Herangehensweisen gelöst werden konnten. Es wurde oft Online Hilfeforen zur Rate gezogen, da diese eine ausführliche Hilfe zu vielen Befehlen lieferten. Vor allem die externe API musste oft in Frage gestellt werden und einige Beiträge zu ihr eingehend studiert werden.

Im Thema Funktionsweise und Einsatzgebiete musste man sich bewusstwerden, welche Zielgruppen unserer ShortcutManager genau anspricht und wozu er genutzt werden kann. Weiterhin musste eine passende Programmiersprache sowie ein passendes und allgemein verwendbare .Net Framework Version gefunden werden.

Im Thema der Oberflächenprogrammierung stellte eine besondere Herausforderung die Programmierung mehrere Formen im Zusammenspiel miteinander dar, da diese sich individuell ergänzen und untereinander kommunizieren mussten. Dies programmtechnisch umzusetzen fiel Louis Mehring trotz seiner fortgeschrittenen C#-Kenntnisse nicht leicht. Weiterhin sorgten einige Focus Optionen der Panels und Formen für viel Aufwand, da für die Kommunikation zwischen Main und Shorthand Form eine eigene Überprüfungsmethode konzipiert werden musste.

Auch stellten uns Registryeinträge aufgrund fehlender Berechtigungen an einigen Rechnern der Schule vor Testprobleme. Man musste nicht ob da Programm sich einen Autostart des Shortcutmanagers veranlasste, wenn nicht die nötigen Berechtigungen gegeben waren. Dies musste mit vielen Tests behoben werden. Auch die externe API warf oft Fehler aus, welche schwer identifizierbar oder unverständlich waren.

Das Projekt erforderte auch jede Menge Teamarbeit, da die XML-Datei mit der Oberfläche sowie der dahinterstehenden Logik abgestimmt sein musste, um reibungslos zu funktionieren.

Letzten Endes musste auch ein ordentlicher und sauberer Arbeitsprozessbericht angefertigt werden, welche den Ansprüchen einer Berufsschulklasse der Werner von Siemens Schule genügt.