Inhaltsübersicht

**1 Projektdefinition**

* 1. Projektumfeld (Ist)
  2. Projektauftrag (Soll)
  3. Projektziele
  4. Projektabgrenzung

**2 Projektplanung**

2.1 Zeit- und Ablaufplanung

2.2 Ressourcenplanung

2.3 Kostenplanung

**3 Projektdurchführung**

* 1. Analysephase
  2. Designphase
     1. Architektur
     2. Benutzeroberfläche/View
     3. Geschäftslogik/Model
     4. Verknüpfung/Presenter

1. **Projektdefinition**
   1. Projektumfeld

…

Sowohl neue als auch bestehende Geschäftskontakte werden häufig in Form von Visitenkarten übergeben. Diese werden von dem betreffenden Mitarbeiter selbstständig verwaltet. Oft werden diese in ein digitales Adressbuch, z.B. das einer E-Mail-Anwendung, manuell übertragen oder bei jedem Bedarf erneut herausgesucht.

* 1. Projektauftrag

Es soll eine Desktopanwendung entwickelt werden, welche digitalisierte Daten von Visitenkarten erkennt, zuordnet und in das Microsoft Outlook Adressbuch exportiert. Eine Aktualisierungsmöglichkeit für vorhandene Kontakte soll vorhanden sein. Die Software soll in der Lage sein, mit gängigen Scannern zu arbeiten. Korrekturmöglichkeiten, zur Bearbeitung eventuell falsch erkannter Schriftzeichen, sollen gegeben sein. Die Sprache der Benutzeroberfläche soll in deutscher Sprache gehalten sein. Der Fokus der Erkennung und Zuordnung der Visitenkartenfelder liegt auf der Bundesrepublik Deutschland. Eingriffe durch den Benutzer sind auf ein Minimum zu beschränken.

* 1. Projektziele

Die Software soll dem Mitarbeiter und somit der Firma Zeit bei der Aufnahme und Pflege von Geschäftskontakten ersparen. Weiterhin sinkt die Hürde, Kontakte zu digitalisieren durch einfachere und schnellere Handhabung durch die Anwendung im Vergleich zu einer manuellen Eingabe.

* 1. Projektabgrenzung

Der Fokus des Projektes liegt auf der Funktionalität der Anwendung, weswegen auf aufwendige Designarbeiten verzichtet wird. Weiterhin wird auf eine zweite Speicherung der Daten z.B. in einer Datenbank verzichtet.

1. **Projektplanung**
   1. Zeit-und Ablaufplanung
   2. Ressourcenplanung

Da es sich bei der Anwendung um eine mit Microsoft Outlook kommunizierende Software mit grafischer Oberfläche handeln soll, steht die Zielplattform mit Microsoft Windows bereits fest. Die Wahl der Programmiersprache fiel auf C# und ging aus einer Nutzwertanalyse (siehe Nutzwertanalyse im Anhang) hervor. Weitere Ressourcen befinden sich im Anhang.

* 1. Kostenplanung

1. **Projektdurchführung**
   1. Analysephase

In einem Gespräch mit dem Auftraggeber werden der Ist-Zustand sowie die Anforderungen an die Anwendung dargelegt. Aus den Gesprächsnotizen wird ein Lastenheft erstellt. Auf Grundlage des Lastenheftes wird ein Pflichtenheft erstellt, welches spezifische Informationen über die Funktionen der Software, sowie deren Anwendung und Implementierung enthält. Nach Fertigstellung wird das Pflichtenheft dem Auftraggeber vorgelegt.

* 1. Designphase
     1. Architektur

Das Projekt wird auf Basis des MVP-Passive-View Designpatterns realisiert. Neben der logischen Trennung der Komponenten Ansicht und Programmlogik und dadurch besseren Übersicht, kann bei späteren Tests auf die der Ansichten verzichtet werden. Daraus resultiert unmittelbar, dass die grafischen Elemente des Programms ausgetauscht werden können. Außerdem bleibt die Anwendung mit steigendem Quellcode übersichtlich und verständlich, auch wenn das wohl auf die meisten UI-Patterns zutrifft. Außerdem kann der Model-Code in anderen Anwendungen wiederverwendet werden. So ist z.B. eine spätere Portierung in eine WPF-Anwendung möglich, um eine Verbesserung der GUI zu erreichen.

* + 1. Verknüpfung/Presenter

Der Presenter dient der Kommunikation zwischen View und Model. Er reagiert auf Ereignisse und holt oder setzt daraufhin Daten aus bzw. in den jeweiligen Schichten. Der jeweilige Presenter implementiert ein oder mehrere Interfaces, welche die Methoden der jeweiligen Ereignisse des Models vorgeben. Für die View-Seite reichten lediglich Referenzen zu den in den Interfaces beschriebenen Form-Objekten.

* + 1. Benutzeroberfläche/View

Die Benutzeroberfläche besteht aus vier Fenstern. Der Benutzer kann zwischen zwei Arten von Schnittstellen wählen, mit dem der Scanner arbeitet. Einer allgemeinen Schnittstelle, die mit einer Ordnerüberwachung hinterlegt ist, womit Scanner mit proprietären Treibern betrieben werden können und einer WIA1-Schnittstelle (Windows Image Acquisition). Im weiteren Dialog, kann der Benutzer die zugeordneten Ergebnisse überprüfen, abändern und ergänzen. Im Folgenden, optionalen Fenster werden eventuelle Übereinstimmungen zu bereits vorhandenen Datensätzen angezeigt. Der Benutzer hat hier die Möglichkeit den Datensatz mit den eingescannten Daten zu aktualisieren, den Vorgang abzubrechen oder den Datensatz anzulegen.

Alle „Views“ sind von Model und Presenter unabhängig und damit austauschbar. Sie implementieren jeweils ein Interface, mit den Eigenschaftsmethoden, auf die später zugegriffen werden soll. Dieses Interface dient dem Presenter später zur Steuerung des Views.

* + 1. Programmlogik/Model

In der Model-Schicht ist die gesamte Programmlogik hinterlegt. Alle Model-Klassen implementieren ein eigenes generisches Interface vom Typ „Presenter“, über welches dieser sich injiziert. Entsprechende Ereignisse können so die im Presenter hinterlegte Funktion aufrufen.

Alle Klassen und Interfaces sind in dem Klassendiagramm unter Anhang xx aufgeführt.

* 1. Realisierungsphase
     1. Erstellung des Grundgerüsts des MVP-Patterns

Für Presenter- und Modelschicht werden jeweils die beiden Grundinterfaces, sowie die beiden klassenspezifischen Interfaces mit deren Implementierungen erstellt. Für die View reicht ein klassenspezifisches Interface und eine Form. Alle auf der Form bereitgestellten Controls oder Controleigenschaften müssen mit ihren Eigenschaftsmethoden sowohl im Interface als auch in der Formklasse nach der Implementierung des Interfaces aufgeführt werden. Das Grundinterface des Presenters deklariert ein ObjektÜber eine Referenz auf dieses Interface im Presenter, hat dieser Zugriff auf diese Methoden. Danach werden

Der Presenter implementiert weiterhin sein spezifisches Interface und implementiert