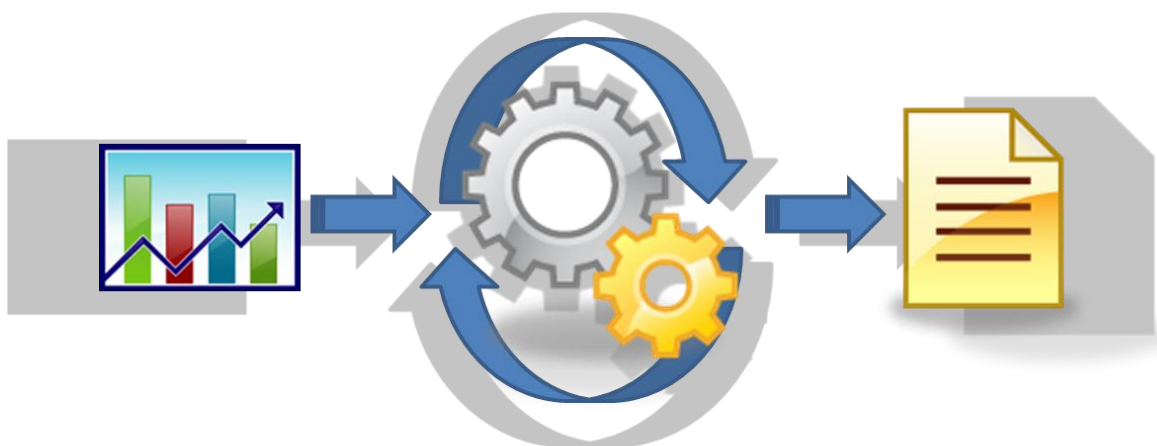


BILDANALYSE SOFTWARE

Betreuer: Matthias Bachmann

Student: Roger Bollmann

Datum: September 2015



Management Summary

Der Grundstein zu dieser Arbeit legte die FINMA. Die FINMA ist die Finanzaufsichtsbehörde und verordnete jedem Finanzunternehmen eine bessere Überwachung von Mitarbeiter. Genauer gesagt, ein Audit muss zu jederzeit feststellen können, wer, wann auf Kundendaten zugegriffen hat. Mit dieser Verordnung wollte die FINMA vermeiden, dass ein weiter Kundendaten CD an andere Behörden weitergeleitet werden kann.

Jedoch was hat das mit diesem Produkt zu tun?

In mehreren Finanzunternehmen setzen sie momentan eine Lösung dazu um. Die einfachste Variante um das zu überprüfen ist die Überwachung des Netzwerkes oder besser gesagt die Überwachung der HTTP Requests oder Responses. Aktuell wird immer mehr auf Webapplikationen gewechselt, was auch in so bleiben wird. Die Analyse von HTTP Requests und Responses oder genauer gesagt des Bodies kann mit Textanalytik Software durchgeführt werden. Jedoch gibt es die Möglichkeit, dass Bilder ebenfalls Kundendaten enthalten. Genau mit dieser Problematik beschäftigt sich diese Arbeit.

Am Schluss sollte ein Produkt vorgestellt werden, bei dem es möglich ist, ein Bild abzufangen und vorbereiten zur Textanalyse. Anhand von durchgeführten Analysen, Evaluierungen und Erstellen eines guten Konzeptes wird aufgezeigt, dass dieses Produkt sich in zwei Teile aufteilen muss, einem Frontend und einem Backend. Der Grund ist, dass das Produkt keinen negativen Einfluss auf die Performance des Webserver haben darf. Da die Umwandlung von Bild in Text sehr Performanceintensiv ist, muss das so durchgeführt werden.

Das Produkt besteht aus drei Hauptkomponenten, dem Sender, dem Empfänger und dem Translator.

Der Sender übernimmt das Abfangen eines Bildes auf dem Webserver. Der Empfänger, wie der Name schon verrät, empfängt das Bild auf dem Bildanalyse System und leitet es zur Umwandlung in Text an den Translator weiter, welcher ebenfalls auf dem Backend sitzt.

Inhaltsverzeichnis

1.	Einleitung.....	1
1.1.	Motivation.....	1
1.2.	Thema.....	1
1.3.	Ausgangslage.....	1
1.4.	Problemstellung	1
1.5.	Ziel der Arbeit.....	1
1.6.	Aufgabenstellung.....	2
1.7.	Erwartete Resultate.....	2
1.8.	Mitwirkende Personen	2
1.9.	Projektplanung	2
1.9.1.	Projektplan	3
1.9.2.	Termine	4
2.	Recherche.....	5
2.1.	Ergebnisse Recherche.....	5
2.1.1.	Abfangen eines Bildes	5
2.1.2.	Übertragungsmethoden	5
2.2.	Ist-Analyse	6
3.	Anforderungsanalyse.....	7
3.1.	Vision	7
3.2.	Stakeholder Analyse	7
3.3.	Kontext- / Systemdiagramm.....	8
3.3.1.	Schnittstellen	9
3.4.	Umweltdiagramm.....	9
3.4.1.	Input	9
3.4.2.	Output	9
3.5.	Rahmenbedingungen	9
3.5.1.	Technische Rahmenbedingungen	10
3.5.2.	Organisatorische Rahmenbedingungen	11
3.6.	Anwendungsfälle	12
3.6.1.	Prozessablauf.....	14
3.7.	Anforderungen	16
3.7.1.	Funktionale Anforderungen	16
3.7.2.	Nicht-funktionale Anforderungen	20
4.	Konzept.....	24
4.1.	Architektur.....	24

4.1.1.	Beschreibung	25
4.1.2.	Begründung	25
4.2.	Sender.....	26
4.2.1.	Logger	26
4.2.2.	Listener	28
4.2.3.	Transfer Handler.....	29
4.3.	Empfänger und Translator.....	31
4.3.1.	Service / Translator	31
4.3.2.	Service	32
4.3.3.	Translator	33
5.	Proof of Concept „PoC“	35
5.1.	Eingesetzte Technologien.....	35
5.1.1.	Programmiersprachen.....	35
5.1.2.	Entwicklerumgebung.....	35
5.1.3.	Versionierung	35
5.1.4.	Infrastruktur	35
5.1.5.	Protokolle	35
5.2.	Sender.....	36
5.2.1.	Logger	36
5.2.2.	Logfile Handler.....	38
5.2.3.	Transfer Handler.....	38
5.3.	Empfänger und Translator.....	39
5.3.1.	Empfänger	39
5.3.2.	Translator	41
5.4.	Nicht funktionale Anforderungen	44
5.4.1.	NFRQ-001 Angemessenheit.....	44
5.4.2.	NFRQ-002 Interoperabilität.....	44
5.4.3.	NFRQ-003 Sicherheit	44
5.4.4.	NFRQ-004 Fehlertoleranz, NFRQ-005 Wiederherstellbarkeit und NFRQ-008 Analysierbarkeit.....	44
5.4.5.	NFRQ-006 Zeitverhalten und NFRQ-007 Verbrauchsverhalten	45
5.4.6.	NFRQ-009 Installierbarkeit	45
5.4.7.	NFRQ-010 Austauschbarkeit	45
6.	Testing	46
6.1.	Unit Test	46
6.1.1.	Sender.....	46

6.1.2.	Empfänger	49
6.1.3.	Translator	50
6.1.4.	Unit Test Abdeckung	51
6.1.5.	Test Resultat	51
6.2.	User Akzeptanz Tests.....	52
6.2.1.	Logger FRQ-001, FRQ-002, FRQ-003	52
6.2.2.	Transfer Handler FRQ-006.....	52
6.2.3.	Translator FRQ-010.....	53
6.2.4.	Translator FRQ-012.....	53
7.	Fazit	54
7.1.	Rückblick.....	54
7.2.	Ausblick.....	55
8.	Verzeichnisse	56
8.1.	Quellenverzeichnis	56
8.2.	Tabellenverzeichnis	57
8.3.	Abbildungsverzeichnis	58
9.	Anhang.....	59
9.1.	Methoden zum Abfangen von Bilder	59
9.1.1.	Module	59
9.1.2.	Advances Logging	61
9.1.3.	HTTP Handler.....	63
9.2.	Übertragungsmethoden.....	63
9.2.1.	WCF (Windows Communication Foundation).....	63
9.2.2.	HTTPTransport.....	66
9.2.3.	TCPTransport	66
9.2.4.	NamePipeTransport	67
9.2.5.	WCF Funktionsübersicht.....	68
9.3.	Textanalyse Software	70
9.3.1.	GOOCR.....	70
9.3.2.	OCR Software Tesseract	70

1. Einleitung

1.1. Motivation

Aktuell arbeite ich in einer Abteilung, welche sich mit dieser Problematik, Überwachung von Mitarbeiter, auseinandersetzt. Bis jetzt hat die Abteilung schon viele Anforderungen umgesetzt. Entweder kamen diese Anforderungen von der FINMA direkt oder vom internen IT Audit. Für das Analysieren von Bildern wurde bis jetzt aber noch keine Lösung gefunden. Dies brachte mich auf die Idee, eine mögliche Lösung zu konzeptionieren und zu programmieren. Die Motivation besteht darin, ein Produkt auf die Beine zu stellen, welche die Hauptanforderung hat, Bilder zur Analyse vorzubereiten. Mit dieser Arbeit will ich beweisen, dass eine Implementierung einer Bildumwandlungssoftware anhand eines „Proof of Concept“, möglich ist.

1.2. Thema

Abfangen von Bildern von einem Webserver und umwandeln in Text zur Analyse.

1.3. Ausgangslage

Die Ausgangslage wird mit den nachfolgenden Kapiteln genauer erläutert.

1.4. Problemstellung

Als Reaktion auf den Diebstahl von zahlreichen Kundendaten, sowie des anschliessenden Verkaufs von illegalen CDs, hat die Eidgenössische Finanzmarktaufsicht (FINMA) alle Schweizer Finanzinstitute aufgefordert, den Zugriff auf Kundendaten verstärkt zu überwachen. Das Unternehmen für das ich arbeite, integriert nun eine Überwachungssoftware, die den HTTP-Traffic nach Kundendaten absucht. Die Software ermöglicht jedoch lediglich eine systematische Analyse von Text. Bilder, die ebenfalls Kundendaten enthalten könnten, werden ignoriert. Grund dafür ist, dass auf dem Markt keine wirklich effizienten Bildanalyse-Softwares angeboten werden.

1.5. Ziel der Arbeit

Das Ziel der Arbeit ist ein Programm zu entwickeln, welches die versendeten Bilder von einem Webserver in Text umwandelt, um eine Analyse durch die Überwachungssoftware zu ermöglichen. Das Programm sollte aus zwei Teilen bestehen, einem Sender und einem Empfänger. Der Sender wird zuständig sein für das Versenden der Bilder, wobei der Empfänger die Bilder empfangen und in Text umwandeln soll.

1.6. Aufgabenstellung

1. Recherche
 - a. Analyse von verschiedenen Methoden zum Abfangen von Bilder auf einem Webserver
 - b. Analyse von verschiedenen Übertragungsmethoden
2. Anforderungen ermitteln und dokumentieren
3. Erstellen eines Konzepts
 - a. Vergleichen von verschiedenen Textanalytik-Applikationen
4. Erstellen eines Proof of Concepts
 - a. Implementierung eines Senders, welcher die Bilder abfängt und an den Empfänger weiterleitet
 - b. Implementierung eines Empfängers, welcher die Bilder empfängt und in Text umwandelt
5. 5.Fazit

1.7. Erwartete Resultate

1. Durchführen einer Recherche
2. Definition einer Anforderungsanalyse
3. Ausarbeitung eines Konzept
4. Durchführen eines Proof of Concept „PoC“
5. Ausführung von Tests
6. Persönliches Fazit

1.8. Mitwirkende Personen

Studierender Roger Bollmann

Betreuungsperson Matthias Bachmann

1.9. Projektplanung

Folgende zwei Punkte zeigen den Projektplan und die vereinbarten Termine.

1.9.1. Projektplan

Die nachfolgende Grafik ist die Projektplanung und zeigt auf, wie das Projekt umgesetzt und wann daran gearbeitet worden ist. Vor dem Design Review wurde der Projektplan nochmals angepasst, weil nicht alles so erfüllt worden ist, wie geplant.

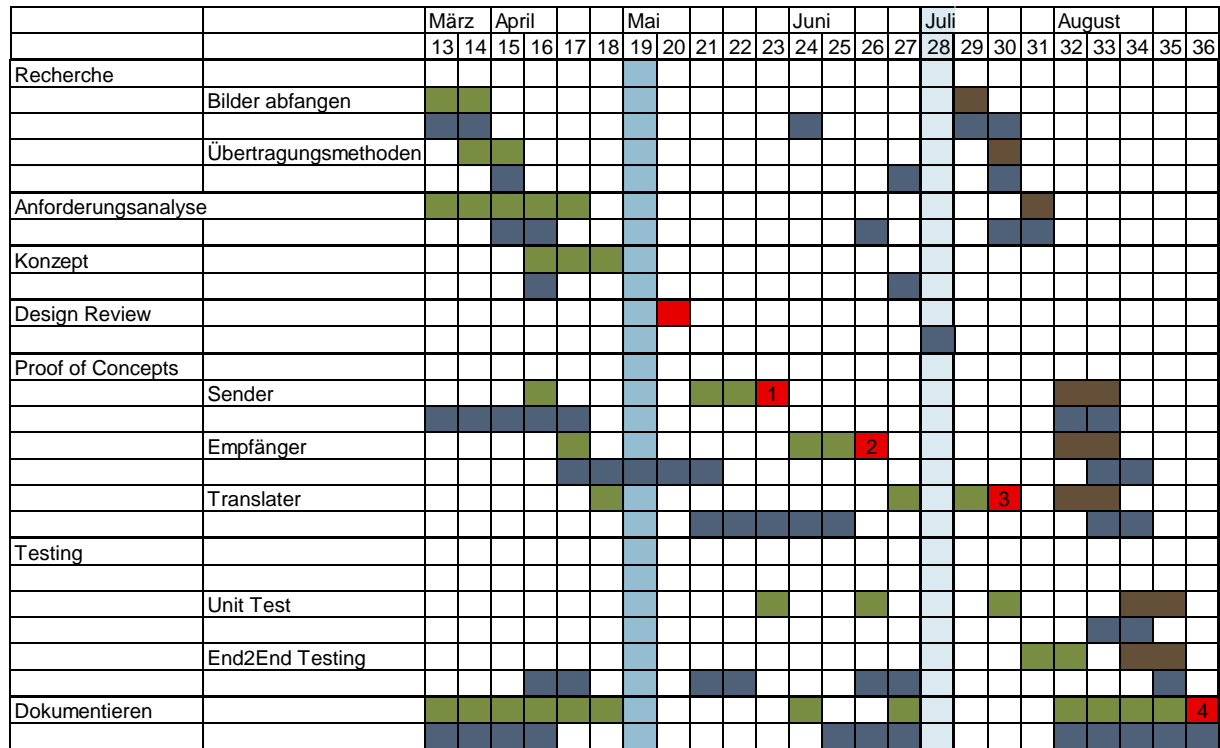


Abbildung 1 Projektplan

Legende	
Ferien	
Milestones	
Review Sender Code	1
Review Empfänger Code	2
Review Translator Code	3
Abgabe	4
Soll	
Ist	
Soll-neu	

Abbildung 2 Legende

In der folgenden Tabelle werden die benötigten Stunden, um das Projekt und die Dokumentation umzusetzen, aufgelistet.

Aufwand	Stunden
Entwicklung	55
Dokumenation	75
Total	130

Tabelle 1 Aufwand

1.9.2. Termine

In der nachfolgenden Tabelle werden die Projekttermine aufgelistet.

Termin	Datum
Kickoff	19.03.2015
Design Review	06.07.2015
Abgabe	05.09.2015
Präsentation	??

Tabelle 2 Termine

2. Recherche

Eine Recherche wurde durchgeführt um die Evaluierung des Produktes zu unterstützen und gewisse Entscheidungen einfach zu machen.

2.1. Ergebnisse Recherche

Genaue Information zu der Recherche befinden sich im Anhang. In diesem Bereich werden die Ergebnisse der Recherche beschrieben. Es wurde eine Recherche über das Abfangen eines Bildes, sowie das Übertragen von Informationen an einen Webserver gemacht.

2.1.1. Abfangen eines Bildes

Die Recherche über das Abfangen eines Bildes von einem Webserver hat ergeben, dass das über die folgenden 3 Möglichkeiten am einfachsten zu implementieren ist.

1. HTTP Modul
2. HTTP Handler
3. Advanced Logging

All diese Möglichkeiten, ihre Eigenschaften und ihre Funktionen sind im Anhang genauer erläutert. Zudem gibt es zu all diesen Funktionen ein Bild abzufangen genügend Information zu finden.

2.1.2. Übertragungsmethoden

Die Recherche über die verschiedenen Übertragungsmethoden hat ergeben, dass die Übertragungsmethoden abhängig sind von der Implementierung des Empfängers. Bezüglich Recherche ist die bevorzugte Methode Informationen zu empfangen für dieses Produkt einen Web Service mit der WCF (Windows Communication Foundation) Plattform zu implementieren. Dabei gibt es drei bevorzugte Übertragungsmethoden sind folgende

1. HTTPTransport
2. TCPTransport
3. NamePipeTransport

Eine Funktionsübersicht befindet sich im Anhang.

2.2. Ist-Analyse

In vielen finanzwirtschaftlichen Unternehmen gibt es zwar ein zentrales Berechtigungssystem, welches dafür zuständig ist, dass Mitarbeiter/innen nur auf diese Applikationen zugreifen können, wo sie selbst auch berechtigt sind. Jedoch gibt es bis jetzt noch keine wirklichen Überprüfung, was genau der Mitarbeiter in der Applikation macht und ob es sinnvoll ist, dass er zum Beispiel auch sensitive Daten anschauen kann, mit welchen diese/r Mitarbeiter/in nicht arbeitet.

Das folgende Bild sollte eine produktivnahe Webapplikationsumgebung beschreiben von einem Unternehmen:

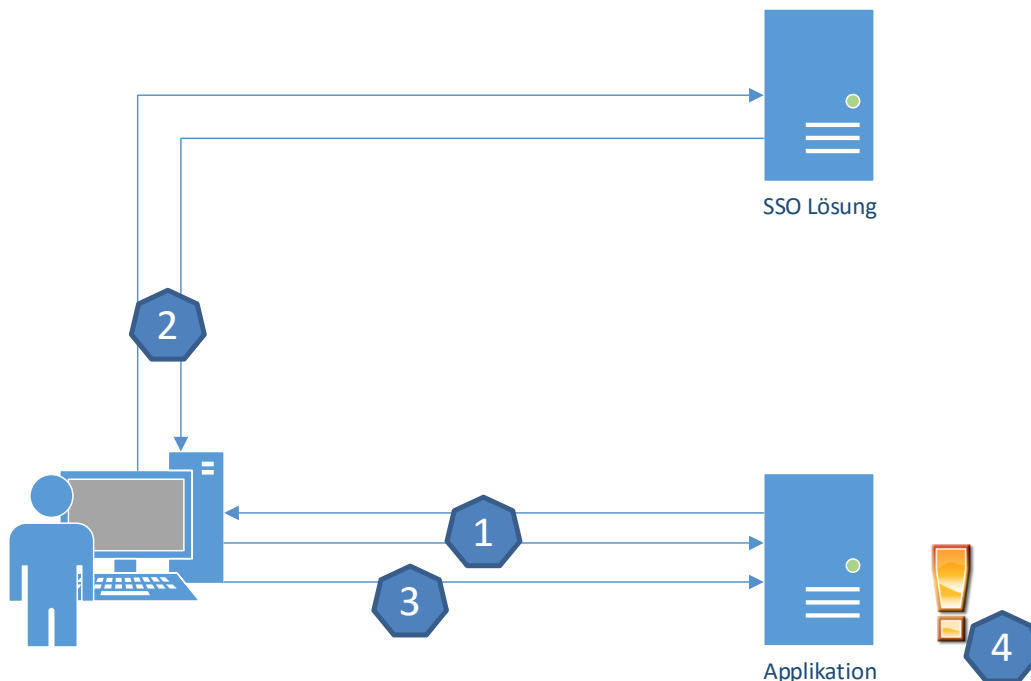


Abbildung 3 Webapplikation Lösung

1. Der Benutzer ruft die Webapplikation in Browser auf
2. Die Applikation verweist den User an ein SSO System, welche die Authentifizierung durchführt.
3. Der Benutzer kann in die Webapplikation einloggen und kann Funktionen ausführen anhand seiner zugewiesenen Berechtigung
4. Abhängig von dem Webserver, welcher installiert ist, wird dann jeder Zugriff auf eine Ressource geloggt.

Das Ausrufezeichen beim Punkt 4, sollte aufzeigend sein für was dann der User genau macht? Welche sensitiven Daten der User besichtigt? Wie viele Daten der User besichtigt? Solche und noch mehr Informationen sollen gemäss FINMA nachvollziehbar sein. Wie die Umsetzung durchgeführt wird, ist jedem Unternehmen überlassen. Momentan gibt es viele Unternehmen welche noch keine Lösung dazu haben (Stand 2015/07).

3. Anforderungsanalyse

In den nachfolgenden Kapiteln werden die Anforderungen definiert und detailliert beschrieben.

3.1. Vision

Die Realisierbarkeit der folgenden Vision soll im Zuge dieser Arbeit evaluiert und umgesetzt werden:

„Es soll eine Lösung zur Analysierung von Bildmaterialien im Intranet zur Verfügung gestellt werden. Dies dient zur Überwachung von Mitarbeiter, welche Bilder von einer Webseite hoch oder herunterladen.“

Wie bereits in den Zielen dieser Arbeit beschrieben, soll sich die Lösung hauptsächlich auf die in der Einleitung beschriebenen Problemstellung beziehen, jedoch zukünftig erweitert werden.

3.2. Stakeholder Analyse

Eine Stakeholder Analyse wird durchgeführt um herauszufinden ob dieses Produkt potentielle Käufer haben könnte und wer sonst noch alles Interesse haben könnte an dem Produkt.

In der nachfolgenden Tabelle werden die potentiellen Stakeholder aufgelistet, welche mögliche Interesse am Ausgang und Entwicklung dieses Projekts haben könnten.

Stakeholder	Beschreibung
Finanzwirtschafts Unternehmen: Abteilung IT-Security	Alle Unternehmen im Bereich Finanzwirtschaft könnten ein potenzieller Abnehmer dieses Produktes sein, da alle diese Anforderung der FINMA umgesetzt werden müssen.
IT-Security Unternehmen	IT-Security Unternehmen könnten eventuell Interesse haben, eine solche Lösung zusätzlich an ihre Kunden anbieten zu können.
ZHAW	Für die ZHAW ist es massgeblich, dass die Semesterarbeit gemäss den organisatorischen Vorgaben und dem Reglement durchgeführt wird. Zudem soll für die ZHAW ersichtlich sein, dass der Student die Arbeit gemässe den erlernten wissenschaftlichen Ansätzen und Methoden gelöst hat
Student: Roger Bollmann	Der Student selber hat ein grosses Interesse daran, mit dieser Arbeit den Anforderungen und Erwartungen der ZHAW gerecht zu werden und ein gutes Resultat zu erzielen. Falls dieses Produkt von einigen Unternehmen eingesetzt würde, könnte zudem noch Geld verdient werden.

Tabelle 3 Stakeholder Analyse

3.3. Kontext- / Systemdiagramm

Das folgende Kontext-/Systemdiagramm dient der Modellierung einer Produktumgebung und es dient dazu das Produkt von seiner Umwelt abzugrenzen und zu definieren.

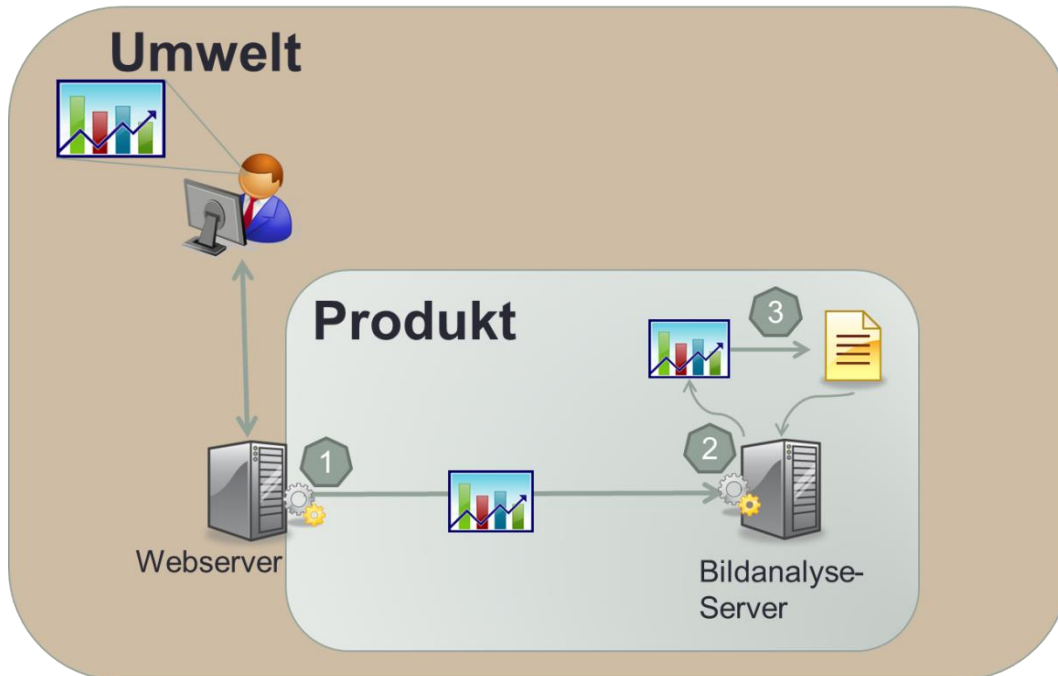


Abbildung 4 Systemdiagramm

Legende:

1. Sender
2. Empfänger
3. Translator

Das Produkt begrenzt sich auf Sender, Empfänger und Translator. Der Sender ist eine Komponente, welche auf dem Webserver installiert werden muss, damit die Daten online abgefangen werden, welche an den User versendet werden. Es werden nur Daten mit dem Mime-Type Image, also Bilder, weiterverarbeitet. Der Empfänger wird auf dem sogenannten Backend installiert und wird zuständig sein um die Daten und Bildinformationen zu empfangen. Der Translator wird das Bild in Text umwandeln und für mögliche Weiterverwendung bereitstellen.

3.3.1. Schnittstellen

In diesem Abschnitt wird die neue Schnittstelle, welche durch das Produkt entsteht, definiert. Die einzig zusätzliche Schnittstelle ist das Senden des Bildes über HTTP/S an den Webserver.

INT-001	Webserver → Bildanalyse System
Beschreibung	Übermittlung von Bilder an den Webservice des Bildanalyse System.
Periodizität	Bei jedem angezeigten Bild auf der Webseite.
Protokoll	HTTP oder HTTPS
Komponente	Bytestream
Zustand	Das Produkt liefert ein Konfigurationsfile mit, welche die Schnittstellt beschreiben soll.

Tabelle 4 Schnittstellen-Analyse

3.4. Umweltdiagramm

Beim Umweltdiagramm werden der Input und der Output beschrieben.



Abbildung 5 Input-Output Diagram

3.4.1. Input

Der Input ist ein Request an den Webservice, welcher von einem User ausgefügt wird. Anhand des Request stellt das Produkt ein Logeintrag, welcher dann weiter prozessiert wird.

3.4.2. Output

Der Output ist ein Textfile, welches den Text eines aufgerufenen Bildes auf der Webseite beinhaltet. Das heisst, dass ein Bild in ein Textfile umgewandelt und abgespeichert wird.

3.5. Rahmenbedingungen

In diesem Kapitel werden die ersten groben Anforderungen an das Produkt gemäss des Systemdiagramms (Abb. 4) als Rahmenbedingung in technischer und organisatorischer Form definiert.

3.5.1. Technische Rahmenbedingungen

3.5.1.1. Allgemein

Es muss eine Webapplikation zur Verfügung gestellt werden, welche Bilder anzeigt, damit das Produkt integriert werden kann. Zudem muss ein Server zur Verfügung gestellt werden, auf dem das Backend (Bildanalyse System) installiert und zur Anwendung freigegeben werden kann.

3.5.1.2. Technologie

Damit ein PoC (Proof of Concept) durchgeführt werden kann, müssen die zur Verfügung gestellten System nur eine technologische Anforderung erfüllen, es muss .NET Framework 4.5 installiert sein. Das Produkt konzentriert sich momentan zur Umsetzung auf Windows Server 2008.

3.5.1.3. Erweiterbarkeit

Da die Anforderung von FINMA ziemlich strickt definiert sind, lässt sich das Produkt nicht gross erweitern. Es sollte jedoch unterschiedliche Kommunikationswege zur Verfügung stellen, welche in der Umsetzung miteinbezogen werden.

3.5.1.4. Wartbarkeit

Das Produkt sollte über eine gute Fehler- und Benachrichtigung verfügen. Zudem sollten auftretende Fehler so gut wie möglich ohne zusätzlichen Programmieraufwand beheben werden können.

3.5.1.5. Sicherheit

Die Sicherheit der Daten muss zu jeder Zeit gewährleistet werden. Es darf kein Datenverlust während der Verarbeitung geben. Der Zugriff von unbefugten Benutzern muss von jeweiligen Serververantwortlichen umgesetzt werden.

3.5.1.6. Stabilität

Das System muss den Betrieb stabil aufrechterhalten.

3.5.1.7. Performance

Die Performance des Produktes wird Anhand der Umsetzung überprüft und sollte auf jeden Fall keinen Einfluss auf die Webapplikation haben. Da es sich jedoch nicht um ein Onlineanalyse Tool handelt, muss das Backend (Bildanalyse System) nicht hoch-performant laufen.

3.5.1.8. Mehrsprachlichkeit

Wie in der Erweiterbarkeit erwähnt sollte das Produkt nur für Bild in Text Übersetzung eingesetzt werden. Jedoch ist es wichtig, dass die Übersetzung in Text weiter analysiert werden kann. Die verwendete Übersetzungssprache spielt aber zu diesem Zeitpunkt keine Rolle, weil vor allem Kundendaten analysiert werden müssen, welche in den meisten Sprachen gleich geschrieben werden. Es wird jedoch auf die Sprache Englisch gesetzt, weil es die meist verbreitete Sprache ist.

3.5.2. Organisatorische Rahmenbedingungen

Da es sich hier um einen PoC handelt, lassen sich die organisatorischen Rahmenbedingungen noch nicht definitiv definieren.

3.5.2.1. Zeitlicher Rahmen

Momentan gibt es keinen zeitlichen Rahmen für die erste Inbetriebnahme des Produktes.

3.5.2.2. Menschliche Ressourcen

Für die Integration des Produktes fallen natürlich Aufwände an, jedoch müssen diese so tief und einfach wie möglich gehalten werden.

3.5.2.3. Budget

Das Budget ist stark abhängig von der potentiellen Firma, welche das Produkt integrieren möchte.

3.6. Anwendungsfälle

Basierend auf dem Kontext und Systemdiagramm (Kapitel 3.3) werden die nachfolgenden relevanten Anwendungsfälle der Applikation mit Hilfe von Use-Case Diagrammen abgeleitet. Die daraus resultierenden Anforderungen werden im Kapitel Anforderungen genauer erläutert.

UC-001	Abfangen von Bilder	
Beschreibung	Das Produkt muss in der Lage sein, Bilder, welche an User verschickt werden, abzufangen und den Zugriff zu loggen	
Diagramm	<p style="text-align: center;">Bildanalyse System</p> <pre> graph TD Actor((User)) --> UC((User schaut Webseite an: Setzt Requests an Webserver ab)) UC --> P[Webserverschickt Inhalt zurück] P --> D{Bild?} D -- Nein --> F[Fertig] D -- Ja --> L[Zugriff loggen] </pre>	
Version	1.0	
Vorbedingung	Der Benutzer greift über einem Webbrowser auf die Webapplikation zu	
Daraus resultierende Anforderungen	FRQ-001, FRQ-002, FRQ-003	
Standard-Ablauf	Benutzer	System
	Öffnen des Webbrowser	
	Eingabe der URL zu einer Webapplikation	
		Schickt angeforderten Inhalt zurück
		Webb es ein Bild ist, schreibt das System ein Logfile, welche folgende Informationen beinhalten soll: 1. Zeitpunkt 2. Bild 3. User 4. Server/URL 5. Applikationsnamen
Autor	Roger Bollmann	

Tabelle 5 UC-001 Abfangen von Bilder

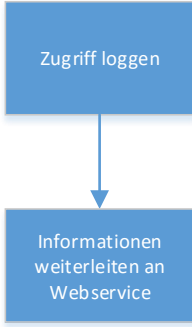
UC-002	Verschicken von Bildern	
Beschreibung	Anhand des generierten Logeintrag von UC-001 wird das Bild an das Backend weitergeleitet für weiter Analyse	
Diagramm	<p style="text-align: center;">Bildanalyse System</p>  <pre> graph TD A[Zugriff loggen] --> B[Informationen weiterleiten an Webservice] </pre>	
Version	1.0	
Vorbedingung	Ein Logeintrag ist erstellt worden von einem verschicken Bild	
Daraus resultierende Anforderungen	FRQ-004, FRQ-005, FRQ-006, FRQ-007, FRQ-011	
Standard-Ablauf	System (Webapplikation)	System (Bildanalyse)
	Listener auf Logfile	
	Sobald ein neuer Eintrag gemacht wird, Bild auf der Festplatte suchen und an Bildanalyse System weiterleiten	
		Empfangen des Bildes
		Abspeichern des Bildes auf Festplatte
Autor	Roger Bollmann	

Tabelle 6 UC-002 Verschicken von Bildern

UC-003	Transferieren von Bild in Text.			
Beschreibung	Anhand des erhalten Bildes von UC-002 soll nun das Bild in Text umgewandelt werden.			
Diagramm	<div><p>Bildanalyse System</p><pre>graph TD; A[Webservice transferiert Bild in Text] --> B([Text wird abgespeichert]);</pre><p>Das Diagramm zeigt den Prozess der Bildanalyse. Ein blauer rechteckiger Kasten oben mit der Aufschrift 'Webservice transferiert Bild in Text' ist mit einem blauen Pfeil verbunden mit einem blauen ovalen Kasten unten mit der Aufschrift 'Text wird abgespeichert'.</p></div>			
Version	1.0			
Vorbedingung	Neues Bild ist auf der Festplatte abgespeichert			
Daraus resultierende Anforderungen	FRQ-008, FRQ-009, FRQ-010, FRQ-012			
Standard-Ablauf	<table><tr><td>System (Bildanalyse)</td></tr><tr><td>Abgespeichertes Bild an OCR weiterleiten</td></tr><tr><td>ORC Software transferiert das Bild in Text.</td></tr></table>	System (Bildanalyse)	Abgespeichertes Bild an OCR weiterleiten	ORC Software transferiert das Bild in Text.
System (Bildanalyse)				
Abgespeichertes Bild an OCR weiterleiten				
ORC Software transferiert das Bild in Text.				
Autor	Roger Bollmann			

Tabelle 7 UC-003 Transferieren von Bild in Text

3.6.1. Prozessablauf

Ein User sieht sich auf einer Webseite einige Bilder an, welche potentiell Kundendaten enthalten können. Der Sender bekommt das mit und schickt das Bild, zur Überprüfung ob es Kundendaten enthält, an den Empfänger weiter. Der Empfänger wandelt das Bild in Text um und kann den Text zur Analyse weiterleiten.

1. Benutzer greift auf eine Webapplikation zu.
2. Der Webserver schickt den angeforderten Inhalt zurück.
3. Falls ein Bild verschickt wird, schreibt der Webserver das in das Logfile.
4. Der Sender hat ein Listener auf dem Logfile. Sobald ein neuer Eintrag hinzugefügt wird, liest er daraus die nötigen Informationen.
5. Der Sender kopiert das Bild in einen neuen Ordner, um es danach an den Empfänger weiterzuleiten.
6. Der Sender sendet das Bild über die ausgewählte Übertragungsmethode an den Empfänger.
7. Der Empfänger bekommt das Bild und legt es an einen bestimmten Ort ab.
8. Der Translator nimmt das Bild auf und wandelt es in Text um und legt der Text in einem bestimmten Ordner ab.

Diagramm:

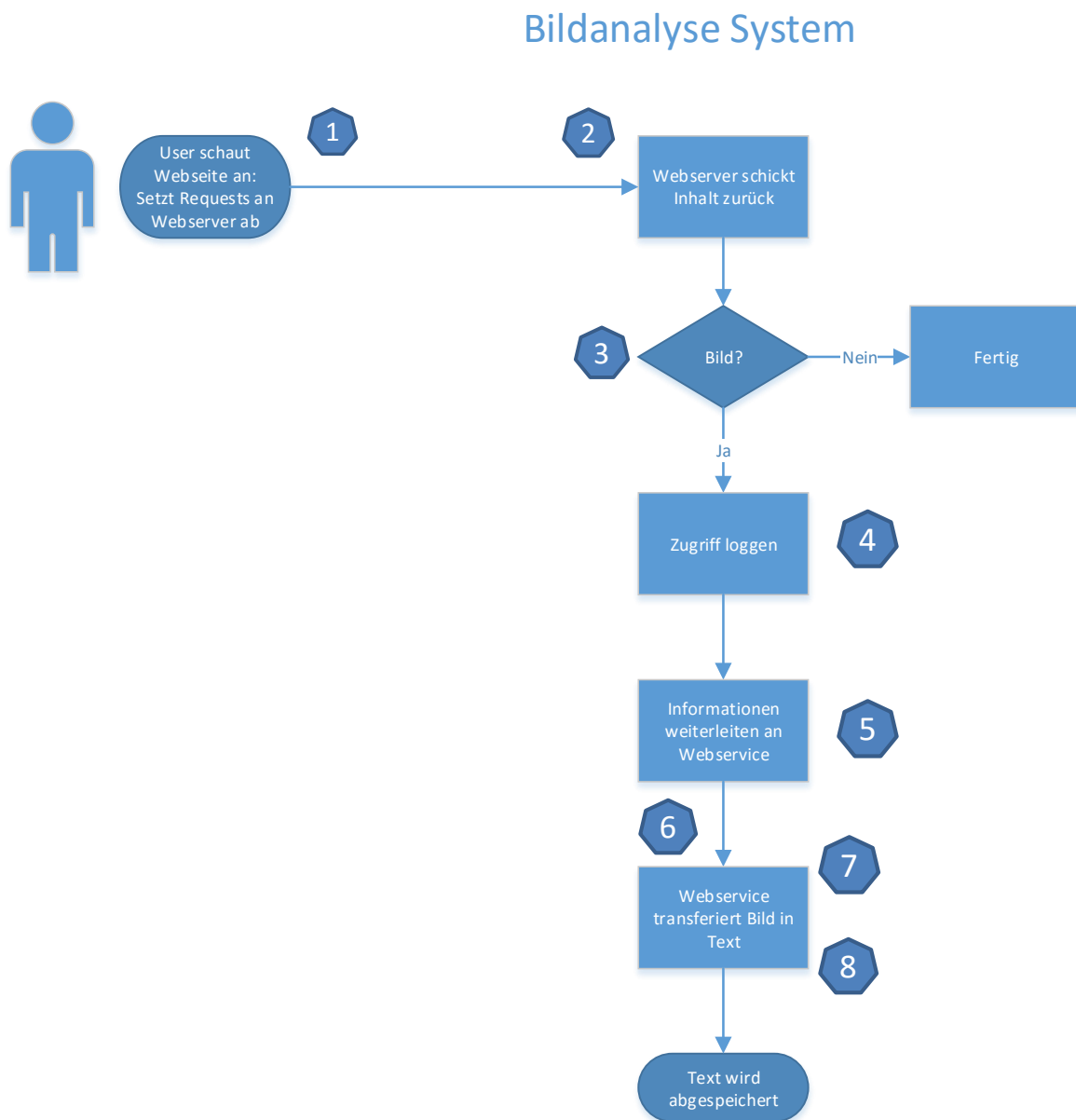


Abbildung 6 Bildanalyse System Prozess

3.7. Anforderungen

Ausgehend von der bisherigen Analyse zu den Bedingungen und dem Umfeld wurden im vorherigen Kapitel die Anwendungsfälle definiert. Aus diesen gewonnen Erkenntnissen werden nun die nachfolgenden funktionale und nicht funktionale Anforderungen definiert.

Eine Anforderung verfügt über eine eindeutige Kennung bestehend aus zwei Typen, FRQ (Funktionale Anforderungen) und NFRQ (Nicht-funktionale Anforderungen).

Die Anforderungen werden gemäss IEEE 830-1998 gemäss ihrer Notwendigkeit (Degree of necessity) in folgende Klassen eingeteilt:

Notwendigkeit	Beschreibung
Essential	Impliziert dass die Software nicht akzeptabel ist bis die komplette Anforderung geliefert und umgesetzt ist
Conditional	Impliziert dass diese Anforderungen die Software verbessern, jedoch nicht unbedingt notwendig sind, damit die Software funktioniert
Optional	Impliziert eine Klasse von Funktionen welche eventuell umgesetzt werden. Das gibt den Auftraggeber die Möglichkeit, etwas vorzuschlagen, welche über die vorhandenen Anforderungen herausgeht.

Tabelle 8 Notwendigkeit

Zudem werden die Anforderungen noch gemäss Kritikalität klassifiziert:

Kritikalität	Beschreibung
Hoch	Im Falle eines Fehlers können Daten verloren gehen.
Mittel	Im Falle eines Fehlers können falsche Daten als Input oder Output entstehen.
Niedrig	Im Falle eines Fehlers können Daten langsam oder später übertragen werden, was zu Stau oder nicht-performanten System führen kann.

Tabelle 9 Kritikalität

3.7.1. Funktionale Anforderungen

Die funktionalen Anforderungen definieren die Funktionalität des Produktes. Die nachfolgenden Tabellen beschreiben die funktionalen Anforderungen im Detail, abgeleitet von den Anforderungsfällen:

FRQ-001	Bilder erkennen
Beschreibung	Jedes Bild das von einem Webserver verschickt wird, muss erkannt werden
Version	1.0
Notwendigkeit	Essential
Kritikalität	Hoch
Abnahmekriterien	UAT 6.2.1 Logger
Autor	Roger Bollmann

Tabelle 10 Funktionale Anforderung FRQ-001

FRQ-002	Logeintrag
Beschreibung	Jedes Bild das von einem Webserver verschickt wird, muss in ein Logfile geschrieben werden.
Abnahmekriterien	Das Logfile muss folgende Informationen beinhalten: <ol style="list-style-type: none"> 1. Zeit 2. Physikalische Pfad des Bildes 3. Mime Type 4. Status Code 5. User (optional) 6. Server (optional) 7. Applikationsnamen (optional) Die Informationen werden durch ein Pipe „ “ abgetrennt.
Version	1.0
Notwenigkeit	Essential
Kritikalität	Hoch
Abnahmekriterien	UAT 6.2.1 Logger
Autor	Roger Bollmann

Tabelle 11 Funktionale Anforderung FRQ-002

FRQ-003	Logfile konfigurierbar
Beschreibung	Der Pfad des Logfiles muss konfigurierbar sein.
Abnahmekriterien	Der Pfad muss in einem Konfigurationsfile des Webserver ersichtlich sein.
Version	1.0
Notwenigkeit	Optional
Kritikalität	Hoch-Mittel
Abnahmekriterien	UAT 6.2.1 Logger
Autor	Roger Bollmann

Tabelle 12 Funktionale Anforderung FRQ-003

FRQ-004	Logeintrag muss gelesen werden
Beschreibung	Jeder Eintrag in dem Logfiles muss gelesen werden und für weiter Verwendung weitergeleitet werden
Abnahmekriterien	Sobald sich das Logfile verändert muss das vom Produkt war genommen und der neuste Eintrag gelesen werden.
Version	1.0
Notwenigkeit	Essential
Kritikalität	Hoch
Abnahmekriterien	Unit Test 6.1.1.1 Loglistener
Autor	Roger Bollmann

Tabelle 13 Funktionale Anforderung FRQ-004

FRQ-005	Bild inkl. Bildinformationen verschicken
Beschreibung	Der Sender muss das Bild inklusive vorher definierten Bildinformationen verschicken können
Abnahmekriterien	Der Sender muss folgende Information verschicken: <ol style="list-style-type: none"> 1. Bild 2. Bildinformationen (Pfad Mime-Type Status)

	Code Usernamen Applikation Servername) 3. Bildnamen
Version	1.0
Notwenigkeit	Essential
Kritikalität	Hoch
Abnahmekriterien	Unit Test 6.1.1.1 Loglistener
Autor	Roger Bollmann

Tabelle 14 Funktionale Anforderung FRQ-005

FRQ-006	Endpunktadresse muss konfigurierbar sein
Beschreibung	Der Sender muss von einem Konfigurationsfile die Endpunktadresse herauslesen können.
Abnahmekriterien	Die Endpunktadresse des Empfängers muss in einem Konfigurationsfile eingetragen oder verändert werden.
Version	1.0
Notwenigkeit	Conditional
Kritikalität	Mittel
Abnahmekriterien	UAT 6.2.2 Transfer Handler FRQ-006
Autor	Roger Bollmann

Tabelle 15 Funktionale Anforderung FRQ-006

FRQ-007	Bild inkl. Bildinformationen empfangen
Beschreibung	Bild inkl. Bildinformationen müssen empfangen werden können und für weiter Verarbeitung vorbereitet werden
Abnahmekriterien	Der Empfänger muss folgende Informationen erhalten <ol style="list-style-type: none"> 1. Bild 2. Bildinformationen (Pfad Mime-Type Status Code Usernamen Applikation Servername) 3. Bildnamen
Version	1.0
Notwenigkeit	Essential
Kritikalität	Hoch
Abnahmekriterien	Unit Test 6.1.2.1 Webservice
Autor	Roger Bollmann

Tabelle 16 Funktionale Anforderung FRQ-007

FRQ-008	Bild in Text umwandeln
Beschreibung	Sobald der Empfänger das Bild erhält, muss das Bild in Text umgewandelt werden.
Abnahmekriterien	Empfangenes Bild muss in Text umgewandelt werden
Version	1.0
Notwenigkeit	Essential
Kritikalität	Hoch
Abnahmekriterien	Unit Test 6.1.3.1 Translate
Autor	Roger Bollmann

Tabelle 17 Funktionale Anforderung FRQ-008

FRQ-009	Text abspeichern
Beschreibung	Der Translator muss der Output des umgewandelte Bild abspeichern
Abnahmekriterien	Der Text muss lokal auf dem Bildanalyse Server abgespeichert werden.
Version	1.0
Notwenigkeit	Essential
Kritikalität	Hoch
Abnahmekriterien	Unit Test 6.1.3.1 Translate
Autor	Roger Bollmann

Tabelle 18 Funktionale Anforderung FRQ-009

FRQ-010	Input und Output für Tranlator konfigurierbar
Beschreibung	Der Translator muss von einem Konfigurationsfile der Input und Output Pfad herauslesen können.
Abnahmekriterien	Input und Output Pfad muss konfigurierbar sein
Version	1.0
Notwenigkeit	Optional
Kritikalität	Niedrig
Abnahmekriterien	UAT 6.2.3 Translator FRQ-010
Autor	Roger Bollmann

Tabelle 19 Funktionale Anforderung FRQ-010

FRQ-011	Bildinformationen im Output File
Beschreibung	Es müssen sich alle Bildinformationen im Output File vom Translator befinden.
Abnahmekriterien	<p>Folgende Informationen müssen zusätzliche im Output File ersichtlich sein:</p> <ol style="list-style-type: none"> 1. Pfad 2. Mime-Type 3. Status Code 4. Usernamen 5. Applikation 6. Servername <p>Die Informationen sollen durch einen Pipe „ “ getrennt werden.</p>
Version	1.0
Notwenigkeit	Conditional
Kritikalität	Mittel
Abnahmekriterien	Unit Test 6.1.2.1 Webservice
Autor	Roger Bollmann

Tabelle 20 Funktionale Anforderung FRQ-011

FRQ-012	Translator muss alle Ausführungen loggen
Beschreibung	Der Translator soll aus Audit zwecken alle übersetzten Bilder loggen.
Abnahmekriterien	<p>Folgende Informationen müssen im Logfile vorhanden sein:</p> <ol style="list-style-type: none"> 1. Bild 2. Startzeit 3. Endzeit 4. Ausführungszeit <p>Die Informationen sollen durch einen Pipe „ “ getrennt werden.</p>
Version	1.0

Notwendigkeit	Optional
Kritikalität	Niedrig
Abnahmekriterien	UAT 6.2.4 Translator FRQ-012
Autor	Roger Bollmann

Tabelle 21 Funktionale Anforderung FRQ-012

3.7.2. Nicht-funktionale Anforderungen

Nicht-funktionale Anforderungen oder auch Qualitätsmerkmale legen fest, welche Eigenschaften eine Software grundsätzlich vorweisen muss. Die internationale Qualitätsnorm ISO/IEC 9126 beschreibt ein Qualitätsmodell bestehend aus 6 Hauptqualitätsmerkmalen welche auf alle Arten von Software anwendbar sind. Die für dieses Produkt relevanten Anforderungen werden gemäss nachfolgenden Tabellen definiert.

3.7.2.1. Funktionalität

NFRQ-001	Angemessenheit
Beschreibung	Die Software muss geeignete Funktionen zur Verfügung stellen für die spezifizierten Anforderungen der Bildanalyse.
Abnahmekriterien	Alle funktionalen Anforderungen müssen umgesetzt worden sein.
Version	1.0
Notwendigkeit	Essential
Kritikalität	Hoch
Autor	Roger Bollmann

Tabelle 22 Nicht funktionale Anforderung NFRQ-001

NFRQ-002	Interoperabilität
Beschreibung	Die Software muss mit dem vorgegebenen System zusammenwirken.
Abnahmekriterien	Das Produkt muss auf Windows 2008 Server mit IIS 7.0 und .NET 4.5 laufen
Version	1.0
Notwendigkeit	Essential
Kritikalität	Hoch
Autor	Roger Bollmann

Tabelle 23 Nicht funktionale Anforderung NFRQ-002

NFRQ-003	Sicherheit
Beschreibung	Die Software muss in der Lage sein, ungewollten Zugriff auf Daten zu verhindern.
Version	1.0
Notwendigkeit	Essential
Kritikalität	Hoch
Autor	Roger Bollmann

Tabelle 24 Nicht funktionale Anforderung NFRQ-003

3.7.2.2. Zuverlässigkeit

NFRQ-004	Fehlertoleranz
Beschreibung	Die Software muss in der Lage sein, bei einem Fehlerverhalten ihre Schnittstelle zum Bildanalyse System zu bewahren
Abnahmekriterien	Das Produkt muss in der Lage sein, bei einem Fehlerfall die Schnittstelle zum Endprodukt wieder herstellen zu können.
Version	1.0
Notwendigkeit	Essential
Kritikalität	Hoch
Autor	Roger Bollmann

Tabelle 25 Nicht funktionale Anforderung NFRQ-004

NFRQ-005	Wiederherstellbarkeit
Beschreibung	Die Software muss in der Lage sein, das spezifizierte Leistungsniveau wiederherzustellen und die direkt betroffenen Daten wieder zugewinnen.
Abnahmekriterien	Keine Daten dürfen verloren gehen.
Version	1.0
Notwendigkeit	Essential
Kritikalität	Hoch
Autor	Roger Bollmann

Tabelle 26 Nicht funktionale Anforderung NFRQ-005

3.7.2.3. Effizienz

Für die Abnahmekriterien der Effizienz muss zuerst eine Baseline festgelegt werden.

NFRQ-006	Zeitverhalten
Beschreibung	Die Antwort- und Verarbeitungszeit sowie der Durchsatz bei der Funktionsausführung muss in einer angemessen Zeit ausgeführt sein.
Version	1.0
Notwendigkeit	Conditional
Kritikalität	Mittel
Autor	Roger Bollmann

Tabelle 27 Nicht funktionale Anforderung NFRQ-008

NFRQ-007	Verbrauchsverhalten
Beschreibung	Ressourcenverbrauch, wie CPU oder Festplattenzugriff muss in Rahmen gehalten werden. Die Software darf vor allem keinen negativen Einfluss auf die Webapplikation haben.
Version	1.0
Notwendigkeit	Essential
Kritikalität	Hoch
Autor	Roger Bollmann

Tabelle 28 Nicht funktionale Anforderung NFRQ-009

3.7.2.4. Wartbarkeit

NFRQ-008	Analysierbarkeit
Beschreibung	Die Software soll in der Lage sein, bei Problemen alle notwendigen Informationen zur Analyse des Problems bereitzustellen.
Abnahmekriterien	Bei einem möglichen Fehlerfall muss das Produkt in der Lage sein, die Fehlermeldung in ein Logfile zu schreiben.
Version	1.0
Notwendigkeit	Conditional
Kritikalität	Mittel
Autor	Roger Bollmann

Tabelle 29 Nicht funktionale Anforderung NFRQ-010

3.7.2.5. Übertragbarkeit

NFRQ-009	Installierbarkeit
Beschreibung	Die Software muss einfach zu integrieren und installieren sein auf den spezifizieren System
Abnahmekriterien	Das Produkt muss mit wenig manuellen Aufwand installierbar sein.
Version	1.0
Notwendigkeit	Conditional
Kritikalität	Mittel
Autor	Roger Bollmann

Tabelle 30 Nicht funktionale Anforderung NFRQ-012

NFRQ-010	Austauschbarkeit
Beschreibung	Die Software muss in der Lage sein, spezifische Komponente austauschbar zu machen.
Abnahmekriterien	Wichtige Komponenten müssen mit wenig Programmieraufwand austauschbar sein.
Version	1.0
Notwendigkeit	Conditional
Kritikalität	Mittel
Autor	Roger Bollmann

Tabelle 31 Nicht funktionale Anforderung NFRQ-013

4. Konzept

In diesem Kapitel werden Konzepte ausgearbeitet, welche die definierten Anforderungen erfüllen. Es wird eine Architektur dargestellt inklusive einzelne Teilaufgaben evaluiert und grafisch dargestellt.

Dazu wird zuerst eine Übersichtsgrafik, anhand der zusammengetragenen Informationen, detailliert aufgezeigt. Einzelne Teilaufgaben des Produktes werden danach evaluiert, um die bestmögliche Lösung umzusetzen. Die Lösungen werden anhand ausgewählten Szenarien überprüft, inwiefern die einzelnen Lösungen den Anforderungen gerecht werden.

4.1. Architektur

Die Architektur stellt einen Überblick über das Produkt grafisch dar. Es beinhaltet sowohl Schnittstellen wie auch einzelne Teilaufgaben, welche das Produkt mit sich bringt. Einzelne Teilaufgaben sind im Diagramm mit Nummern gekennzeichnet. Die Legende dazu befindet sich ebenfalls oben rechts im Bild.

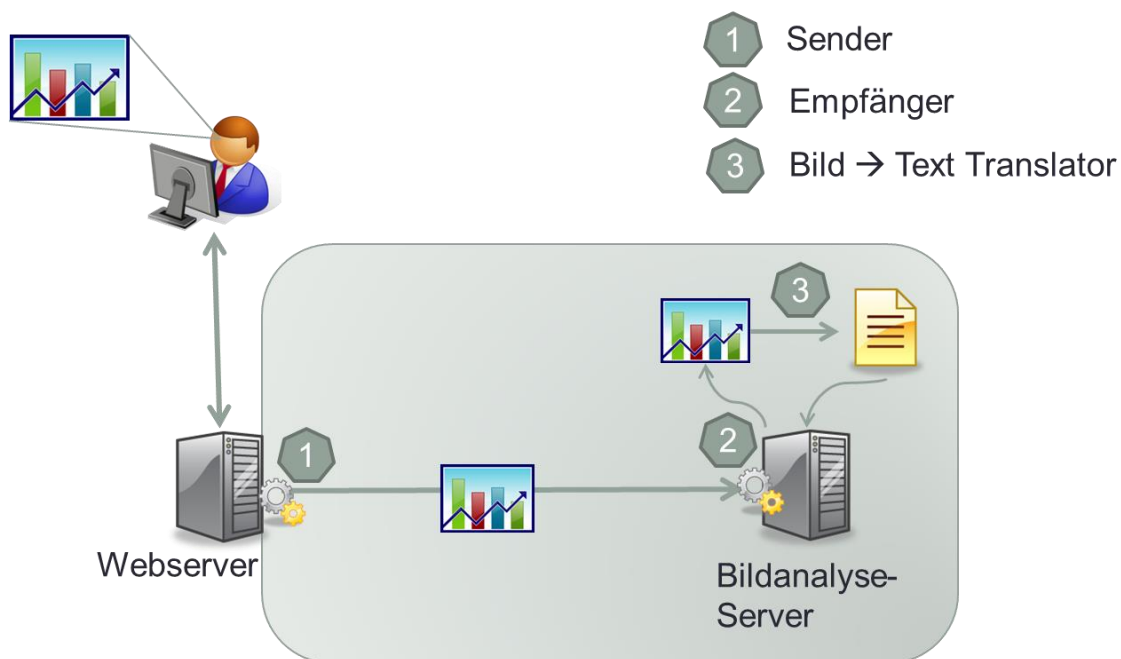


Abbildung 7 Architektur

4.1.1. Beschreibung

Sobald ein Benutzer ein Bild von einer Webseite erhält, gibt es ein Mechanismus, welcher alle versendeten Bilder aufzeichnet. Diese Funktion wird im Produkt Sender (1) genannt. Dieser zeichnet nicht nur auf, sondern ist auch für das Übertragen des Bildes danach an den Bildanalyse Server zuständig. Die nächste Funktion des Produktes ist für das Empfangen der Bilder verantwortlich, der Empfänger (2). Er legt das Bild lokal ab und übergibt es danach an die nächste und letzte Funktion, dem Translator(3). Er übersetzt das Bild in Text und speichert es danach zur Weiterverarbeitung lokal auf dem Filesystem. Die Weiterverarbeitung ist nicht teil des Produktes.

4.1.2. Begründung

Die Architektur wurde in zwei Bereichen aufgeteilt, einem Sender und einem Bildanalyse System. Im Wesentlichen wird die Architektur von den nicht-funktionalen Anforderungen definiert. Dabei ist es wichtig, dass das Produkt keine negative Auswirkung auf die Performance des Webserverns haben darf. Aktuell werden die meisten Server aufgesetzt, dass möglichst wenig Performance vergeudet wird, was das Abspalten des Bildanalyse Servers nur noch unterstützt.

4.2. Sender

Wie bereits erwähnt hat der Sender die Aufgabe die versendeten Bilder zu loggen und anschliessend das Bild zu verschicken. Folgende Grafiken sollen einen Überblick über die einzelnen Teilaufgaben vom Sender darstellen. Sie sind aufgeteilt in 3 Unterkategorien, Logger, Listener (Überwacher) und Transfer Handler (Übermittler).

4.2.1. Logger

Wie bereits erwähnt wird der Logger alle Bilder welche versendet werden in einem Logfile festhalten. Die folgende grafische Übersicht erläutert die Funktionalität des Loggers.

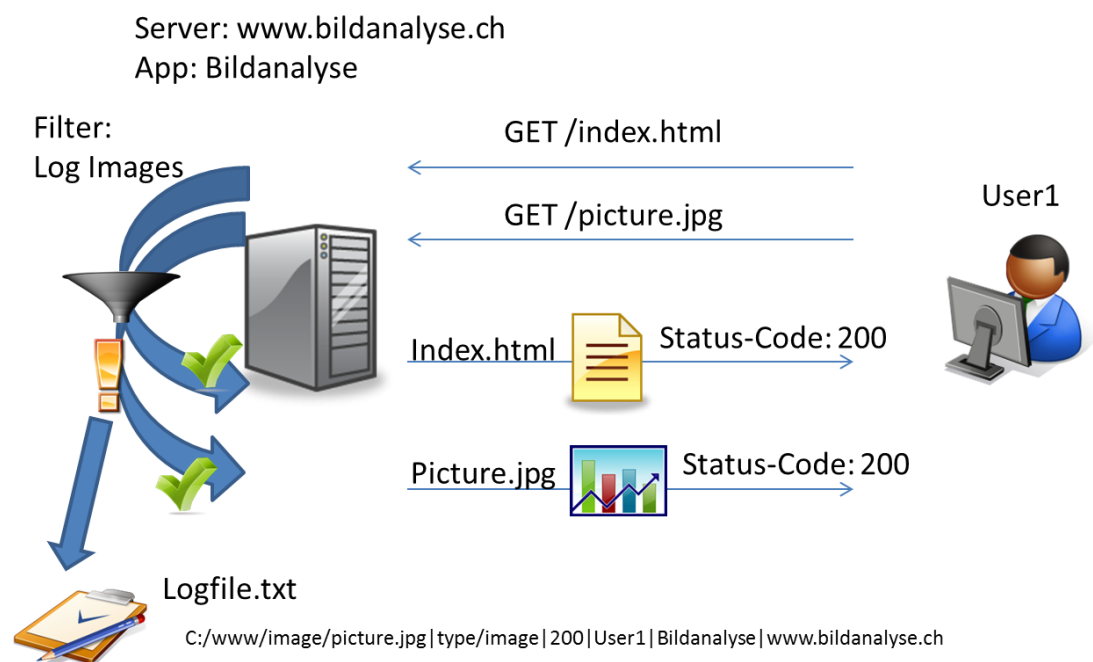


Abbildung 8 Logger

Sobald ein Bild verschickt wird, wird der Logger das feststellen und die alle wichtigen Informationen in einem Logfile festhalten.

4.2.1.1. Evaluierung Logger

Es gibt verschiedene Wege ein Bild, welches verschickt worden ist, in einem Logfile festzuhalten. 3 verschiedene Arten werden in diesem Kapitel festgehalten und evaluiert, anhand nachfolgenden definierten Kriterien.

4.2.1.1.1. Kriterien

In diesem Bereich der Arbeit werden die Kriterien definiert, welche zur Evaluation benötigt werden.

Name	Beschreibung	Quelle
Integration	Einfachheit der Integration in das vorhandene System	NFRQ-009
Erweiterbarkeit	Wie erweiterbar ist die Funktion?	NFRQ-010
Fehlerhandling	Wie gut reagiert es in Falle eines Fehlers	NFRQ-005
Erfüllung Anforderung	Erfüllt es alle Anforderung, welche definiert worden sind	NFRQ-001
Wartbarkeit	Wie aufwändig ist die Wartbarkeit der Funktion	NFRQ-008

Tabelle 32 Logger Kriterien

4.2.1.1.2. Typen

In diesem Bereich werden die unterschiedlichen Typen zur Umsetzung des Loggers aufgezeigt und beschrieben. Genauere Erläuterungen zu den Typen befinden sich im Anhang.

ID	Namen	Beschreibung
1	Advanced Logging	Advanced Logging ist eine von Microsoft zur Verfügung gestelltes Add-On für einen Webserver, welche für erweitertes Logging benötigt und eingesetzt wird.
2	Http Module	Ein Http Modul ist ein Webmodul, welches in die IIS Pipeline geladen werden kann. Es muss jedoch selber definiert und programmiert werden. Es wird vorallem für Logging verwendet.
3	Http Handler	Ein Http Handler wird ebenfalls in die IIS Pipeline geladen. Es wird an spezifische Requests gebunden, welche zu einer Ressource zeigt, um danach den dazugehörigen Response zu manipulieren.

Tabelle 33 Logger Typen

4.2.1.1.3. Bewertungstabelle

Die Bewertungstabelle wird Aufschluss geben über die bestmögliche Lösung. Dabei wurde die Gewichtung der einzelnen Kriterien selber gewählt. Erreichte Punktzahl wird mit 0-10 definiert. Wobei 0-3 (nicht genügend), 3-5 (genügend), 5-7 (gut) und 7-10 (sehr gut) bezeichnet werden.

Das Ergebnis, der erreichten Punkte pro Kriterium wird folgendermassen berechnet:

Gewichtung * erreichte Punktzahl = Total

Kriterium	Gewichtung	Type 1		Type 2		Type 3	
		Punkte	Total	Punkte	Total	Punkte	Total
Integration	40	8	320	7	280	6	240
Erweiterbarkeit	30	7	210	10	300	10	300
Fehlerhandling	10	5	50	10	100	9	90
Wartbarkeit	20	6	120	2	40	1	20
Total	100		700		720		650

Tabelle 34 Logger Bewertungstabelle

4.2.1.1.4. Fazit

Type 2, eine Http Module, ist aus meiner Sicht die beste Möglichkeit, um die gewünschten Anforderungen zu erfüllen. Es gibt zwar etwas Aufwand um diesen zu programmieren, jedoch wird es dadurch sehr flexibel und erweiterbar.

4.2.2. Listener

Der Listener (Überwacher) ist dafür zuständig, den vorher erstellten Logeintrag zu erkennen und alle nötigen Informationen zu Übermittlung weiterzuleiten an den Transfer Handler. Die folgende grafische Darstellung erläutert die Funktionalität des Listeners.

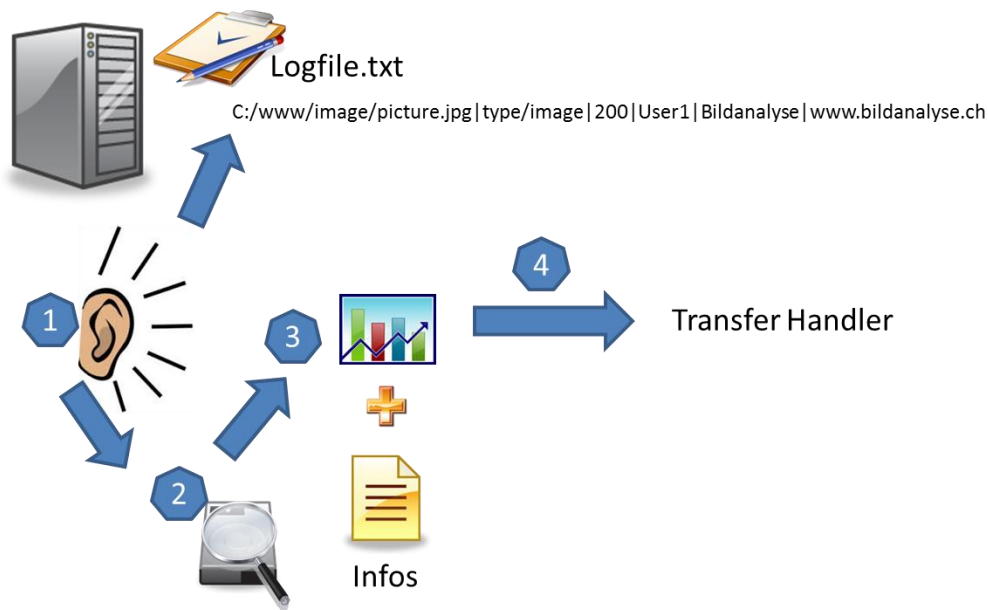


Abbildung 9 Listener

4.2.2.1. Beschreibung

Der Listener überwacht ein gewisses Logfile, welches von dem Logger erstellt wird. Sobald ein neuer Eintrag generiert wird, sucht es das Bild auf der lokalen Festplatte, nimmt es auf und übergibt es samt den Informationen aus dem Logfile an den Transfer Handler.

4.2.2.2. Begründung

Diese Funktion muss nicht evaluiert werden, weil es dabei nicht viele unterschiedliche Möglichkeiten gibt ein Logfile zu überwachen. Die Funktion wird bei der Umsetzung mit C# implementiert.

4.2.3. Transfer Handler

Der Transfer Handler (Übermittler) ist dafür zuständig die erhaltenen Informationen vom Listener an den Webservice zu übertragen. In der folgenden grafischen Darstellung wird die Funktion des Transfer Handler übersichtlich dargestellt:

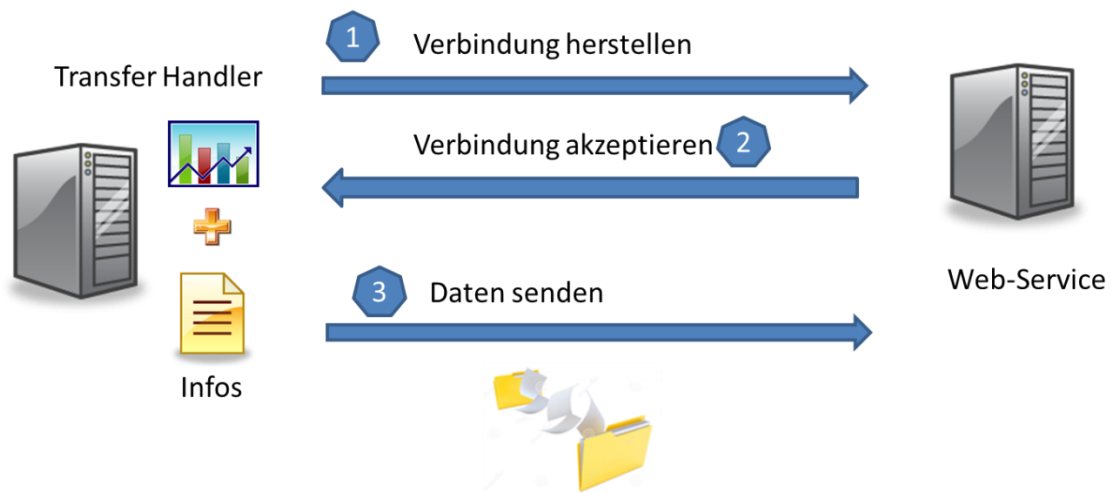


Abbildung 10 Sender Transfer Handler

4.2.3.1. Beschreibung

Sobald der Transfer Handler die definierten Informationen erhält, stellt er eine Verbindung zum Webservice her und übermittelt das Bild und die dazugehörigen Bildinformationen.

4.2.3.2. Begründung

Der Transfer der Daten ist abhängig von der Implementation des Webserver, darum wird eine Evaluation der Übertragung beim Webservice durchgeführt.

4.3. Empfänger und Translator

In diesem Bereich der Arbeit werden Empfänger und Translator des Produktes detailliert beschrieben und Lösungen evaluiert. Diese zwei Komponenten werden in einem Kapitel beschrieben, weil beide Teil des Bildanalyse Systems sind. Die Evaluation wird dann in den jeweiligen Unterkapiteln dokumentiert.

4.3.1. Service / Translator

Der Webservice ist zuständig für das Empfangen der Bilder inklusiv Bildinformationen. Der Translator hingegen ist für die Übersetzung des Bildes in Text verantwortlich. Die folgende grafische Darstellung wird die Funktionen des Webservice und des Translators genau erläutern.

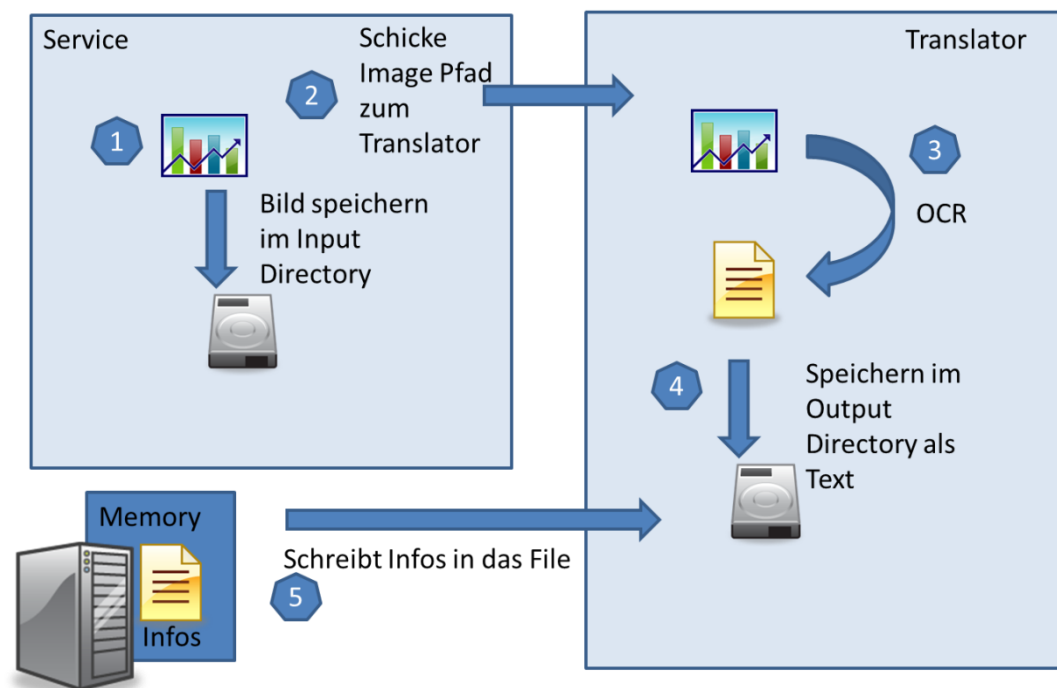


Abbildung 11 Service / Translator

4.3.1.1. Beschreibung

Sobald der Webservice ein Bild inklusive Bildinformationen erhält, speichert er das Bild an einem vordefinierten Pfad ab und hält die weiteren Bildinformationen im Memory. Der Pfad des Bildes wird dann an den Translator weitergegeben, wobei er das Bild aufnimmt, übersetzt in Text und der Text dann in dem Output Pfad abspeichert. Der Webservice wird danach die notwendigen Bildinformationen anhängen.

4.3.2. Service

Der Service ist ein Webservice welche mit WCF (Windows Foundation Communication) Plattform umgesetzt wird. Es gibt unterschiedliche Wege um Daten an einen Webservice zu übertragen. Diese verschiedenen Übertragungsmethoden werden nun evaluiert, um die bestmögliche Lösung herauszufinden.

4.3.2.1. Kriterien

In diesem Bereich der Arbeit werden die Kriterien definiert, welche zur Evaluation benötigt werden.

Name	Beschreibung	Quelle
Integration	Einfachheit der Integration in das vorhandene System	NFRQ-009
Erweiterbarkeit	Wie erweiterbar ist die Funktion?	NFRQ-010
Fehleranalyse	Wie gut kann eine Fehleranalyse durchgeführt werden.	NFRQ-005
Wartbarkeit	Wie aufwändig ist die Wartbarkeit der Funktion	NFRQ-008
Sicherheit	Ist bei der Übertragung die Sicherheit gewährleistet	NFRQ-003

Tabelle 35 Service Kriterien

4.3.2.2. Typen

In diesem Bereich werden die 3 unterschiedlichen Typen zur Umsetzung des Webservices aufgezeigt und beschrieben. Genauere Erläuterungen zu den Typen befinden sich im Anhang.

ID	Namen	Beschreibung
1	HTTP Transport	HTTP Transport basiert auf einer Übertragung über HTTP, welches als primäre Übertragungsmethode genutzt wird. Dieser Transport verwendet URIs der Art: „http://hostname/path“
2	TCP Transport	TCP Transport werdet einen TCP Socket zur Übertragung der Daten an den Webservice. Dieser Transport verwendet URIs der Art: „net.tcp://hostname/path“
3	NamePipe Transport	NamePipe Transport ist auch unter FIFO (First in, first out) bekannt. Das heisst, die Daten welche zuerst kommen, werden zuerst bearbeitet. Dieser Transport verwendet URIs der Art „net.pipe://hostname/path“.

Tabelle 36 Service Typen

4.3.2.3. Bewertungstabelle

Die Bewertungstabelle wird Aufschluss geben über die bestmögliche Lösung. Dabei wurde die Gewichtung der einzelnen Kriterien selber gewählt. Erreichte Punktzahl wird mit 0-10 definiert. Wobei 0-3 (nicht genügend), 3-5 (genügend), 5-7 (gut) und 7-10 (sehr gut) bezeichnet werden.

Das Ergebnis, der erreichten Punkte pro Kriterium, wird folgendermassen berechnet:

Gewichtung * erreichte Punktzahl = Total

Kriterium	Gewichtung	Type 1		Type 2		Type 3	
		Punkte	Total	Punkte	Total	Punkte	Total
Integration	30	8	240	6	180	6	180
Erweiterbarkeit	10	10	100	10	100	10	100
Fehleranalyse	20	8	160	6	120	6	120
Wartbarkeit	10	6	60	6	60	6	60
Sicherheit	30	10	300	10	300	10	300
Total			860		760		760

Tabelle 37 Service Bewertungstabelle

4.3.2.4. Fazit

Type 1, HTTP Transport, ist aus meiner Sicht die am besten zu verwendende Lösung. Die Kommunikation über das Protokoll http ist heutzutage fast Standard und bietet ebenfalls sehr gute Verschlüsselung, sprich Sicherheit, an. Zudem ist das Protokoll meistens freigeschalten bei den üblichen Firewalls und muss somit keinen weiteren Integrationsaufwand betrieben werden. HTTP ist unabhängig von der Plattform und bietet somit einen Service auch an Clients an, welche nicht WCF unterstützen.

4.3.3. Translator

Der Translator ist zuständig für die Übersetzung des Bildes in Text. Sobald er das Bild vom Webservice erhält, wird das Bild durch die OCR Software durchgeschickt und der Output in einem definierten Pfad abgespeichert. Um die bestmögliche OCR Software zu nutzen, werden einzelne OCR Software nun beschrieben und evaluiert anhand nachfolgender Kriterien.

4.3.3.1. Kriterien

In diesem Bereich der Arbeit werden die Kriterien definiert, welche zur Evaluation benötigt werden.

Name	Beschreibung	Quelle
Integration	Einfachheit der Integration in das vorhandene System	NFRQ-009
Unterstützte Bildtypen	Wie viele Bildtypen werden unterstützt	Roger Bollmann
Support	Gibt es Support zu diesem Produkt?	Roger Bollmann
Genauigkeit	Wie gross ist die durchschnittliche Texterkennung?	Roger Bollmann
Erfahrung	Seit wann wird diese OCR Software eingesetzt?	Roger Bollmann

Tabelle 38 Translator Kriterien

4.3.3.2. Typen

In diesem Bereich werden auf die 2 meist verwiesenen ORC Software zur Umsetzung des Translators aufgezeigt und beschrieben, wobei nur ORC Tools, welche auf Befehlsebene aufgerufen werden können und gratis sind, analysiert werden. Genauere Erläuterungen zu den Typen befinden sich im Anhang.

ID	Namen	Beschreibung
1	Tesseract	Tesseract ist eine ORC Software welche von Google erfunden worden ist.
2	GOCR	GOCR ist ebenfalls eine ORC Software und wurde entwickelt von Jürg Schulenburg im Jahre 2000.

Tabelle 39 Translator Typen

4.3.3.3. Bewertungstabelle

Die Bewertungstabelle wird Aufschluss geben über die bestmögliche Lösung. Dabei wurde die Gewichtung der einzelnen Kriterien selber gewählt. Erreichte Punktzahl wird mit 0-10 definiert. Wobei 0-3 (nicht genügend), 3-5 (genügend), 5-7 (gut) und 7-10 (sehr gut) bezeichnet werden.

Das Ergebnis, der erreichten Punkte pro Kriterium, wird folgendermassen berechnet:

Gewichtung * erreichte Punktzahl = Total

Kriterium	Gewichtung	Type 1		Type 2	
		Punkte	Total	Punkte	Total
Integration	30	8	240	9	30
Unterstützte Bildtypen	20	7	140	1	20
Support	10	6	60	1	10
Genauigkeit	30	7	210	5	150
Erfahrung	10	8	80	6	60
Total			730		270

Tabelle 40 Translator Bewertungstabelle

4.3.3.4. Fazit

Type 1, Tesseract, ist von HP entwickelt und von Google weitergetrieben worden. Die Integration ist ziemlich einfach, weil es als File mitgeliefert und per Befehlszeile ausgeführt werden kann. Tesseract hat mittlerweile sogar ein Package für Visual Studio 2013, welches einfach hinzugefügt werden kann. Der Support ist einigermaßen okay, jedoch nicht von Google selber, sondern weil es so viel Leute gibt, die das Produkt einsetzen. Tesseract unterstützt von Hause auf bereits mehrere Sprachen, was jedoch für dieses Produkt nicht weiter von Bedeutung ist, weil es mehrheitlich Informationen abfangen möchte, die in allen Sprachen gleich geschrieben werden. Leider wurden keine weiteren brauchbare Befehlszeilen OCR Tools gefunden, was diesen Vergleich nicht besonders aussagekräftig macht. Jedoch wird im Internet mehrheitlich auf Tesseract verwiesen. Wie genau Tesseract Bilder in Text umgewandelt werden, wird im Anhang etwas genauer erläutert.

5. Proof of Concept „PoC“

In diesem Bereich der Arbeit wird einen Proof of Concept “PoC” durchgeführt und dokumentiert. Das heisst, dass das vorherige definierte Konzept umgesetzt wird und somit bestätigt wird, dass es funktioniert. In den folgenden Kapiteln wird die Umsetzung der einzelnen Teilkomponenten des Produktes genauer erläutert. Zuerst werden jedoch die verwendeten Technologien und welche Infrastruktur dabei benutzt worden sind beschrieben.

5.1. Eingesetzte Technologien

5.1.1. Programmiersprachen

Das Produkt ist mit C# (C-sharp) programmiert worden. C# ist eine Programmiersprache welche von Microsoft entwickelt worden ist und zählt zu den objektorientierten Sprachen. Erstmals wurde es im Jahre 2002 veröffentlicht und eingesetzt.

5.1.2. Entwicklerumgebung

Visual Studio 2013 wurde als Entwicklerumgebung eingesetzt. Der Vorteil ist, dass es bereits sehr viele vordefinierte Templates hat und ebenfalls sehr gut dokumentiert ist Seitens Microsoft.

5.1.3. Versionierung

Als Versionierungs-Tool wird GitHub benutzt. Somit ist sichergestellt, dass es eine Versionierung des Produktes gibt und zusätzlich als Datensicherung benutzt wird.

5.1.4. Infrastruktur

Zur Überprüfung ob das Produkt auch funktionsfähig ist, wird ein virtueller Windows 2008 Server mit VMWare aufgesetzt. Auf dem Server ist .NET 4.5 und IIS 7.0 installiert.

5.1.5. Protokolle

Die Daten werden über das HTTP Protokoll auf einem definierten Port übertragen.

5.2. Sender

In den nachfolgenden Kapiteln wird die Umsetzung von den Teilaufgaben für den Sender, anhand der definierten Anforderungen dokumentiert.

5.2.1. Logger

In diesem Unterkapitel wird die Umsetzung des Loggers detailliert aufgezeigt.

5.2.1.1. FRQ-001 Bilder erkennen

Um ein versendet es Bild zu erkennen, wurde ein HTTP Module in C# geschrieben. Dafür muss die Klasse vom Interface IHttpModule ableiten und alle Funktionen integrieren:

```
public class HelloWorldModule : IHttpModule
```

Funktion Init

```
public void Init(HttpApplication application) { }
```

Die Funktion Init() wird zum Starten dieses Modul benutzt. Hier wird das Modul an einen Request Event registriert. Im Anhang sind die einzelnen Request Events genauer erläutert.

Bei diesem Modul ist es vorallem wichtig, dass so viele Daten wie möglich geloggt werden. Das heisst umso später mein Modul vom Webserver aufgerufen wird, umso mehr Informationen hat das Modul zur Verfügung. Informationen wie http Status oder mime-type befinden sich im Response Header.

Registrierung auf Request Event

```
application.EndRequest += (new EventHandler(this.Application_EndRequest));
```

Sobald der EndRequest Event aufgeführt wird, wird die Funktion Application_EndRequest aufgerufen.

Zugriff auf Header Informationen

```
private void Application_EndRequest(Object source, EventArgs e)
{
    HttpApplication application = (HttpApplication)source;
    HttpContext context = application.Context;
    string contentType = context.Response.ContentType;
}
```

Das ganze http Objekt (Object source) wird an die Funktion weitergegeben. Die Header Informationen befinden sich im Context des Objektes und sind so zugriffbar.

Funktion Dispose

```
public void Dispose() { }
```

Die Funktion Dispose wird aufgerufen, sobald der Event fertig ist. Bei dieser Funktion werden alle benötigten Ressourcen automatisch wieder freigegeben.

5.2.1.2. FRQ-002 Logeintrag

Der Logeintrag wird ebenfalls vom Modul geschrieben. Dabei werden folgende Informationen geloggt:

- Datum
- Physikalischer Pfad zum Bild
- Content Type
- Status Code
- User Namen
- Server Namen
- Applikation Namen

Diese Daten werden mit Hilfe der StreamWriter Klasse in das Logfile „logfile“ angehängt.

```
using (StreamWriter stream = File.AppendText(logfile))
{
    stream.WriteLine("{0:HH:mm:ss}|{1}|{2}|{3}|{4}|{5}|{6}",
        DateTime.Now, context.Request.PhysicalPath, contentType,
        statusCode, userName, serverName, applicationName);
}
```

5.2.1.3. FRQ-003 Logfile konfigurierbar

Da das Modul in den Webserver hineingehängt wird, muss die Konfiguration des Logfiles in der Konfiguration des Webserver „web.config“ unter appSettings hinzugefügt werden.

Konfiguration im web.config

```
<appSettings>
.....
<add key="logFilePath" value="C:\inetpub\logs\LogFiles\test_end.txt"/>
</appSettings>
```

Danach kann anhand folgendem Befehl darauf zugegriffen werden:

```
logfile = ConfigurationSettings.AppSettings["logFilePath"];
```

5.2.2. Logfile Handler

5.2.2.1. FRQ-004 Logeintrag muss gelesen werden

Das vorher vom Logger erstellte Logfile muss nun gelesen werden. Dafür wird eine Klasse `FileSystemWatcher` verwendet. Dabei wird eine Methode an ein System Event registriert, welche bei dem System Event aufgerufen wird:

```
watcher.Changed += new FileSystemEventHandler(OnChanged);
```

Sobald das File geändert wird, sprich etwas hinzugefügt wird, wird die Funktion `OnChanged` aufgerufen. Zudem ist es möglich, dass nur einzelnen Events registriert werden. Mit folgender Einstellung, kann diese Registrierung noch detaillierter konfiguriert werden:

```
watcher.NotifyFilter = NotifyFilters.Size | NotifyFilters.FileName;
```

Hier ist zu beachten, dass einzelne Events wie zum Beispiel Zugriff auf das File, vom Betriebssystem zweimal als Event ausgeführt werden, was dazu führen kann, dass die Methode zweimal aufgerufen wird. Um das zu umgehen, wird das Produkt nur auf Grösse (Size) und auf Filenamen (FileName) konfiguriert.

Die `OnChanged` Methode liest dann den neu hinzugefügten Logeintrag und überreicht diese Informationen dann an den Transfer Handler, welcher für den Transfer der Daten zuständig ist:

```
TransferImageHandler trans =  
    new TransferImageHandler(path, imageInformation, imageName);  
trans.SendImage();
```

5.2.3. Transfer Handler

5.2.3.1. FRQ-005 Bild inkl. Bildinformationen verschicken

Der Transfer Handler schickt die Informationen über http an den Webservice. Die benötigten Webservice Informationen können entweder direkt im Source Code oder anhand einer Konfiguration definiert werden. In diesem Produkt ist die Konfigurations Variante bevorzugt worden, welches in dem `app.config` zu finden ist:

```
<client>  
  <endpoint address="http://localhost:1234" binding="basicHttpBinding"  
    bindingConfiguration="BasicHttpBinding_IService1"  
    contract="ServiceReference1.IService1"  
    name="BasicHttpBinding_IService1" />  
</client>
```

Bei der Endpoint Konfiguration muss die ABC Regel eingehalten werden. Adresse, Binding und Contract müssen in der Konfiguration vorhanden sein, um Daten an einen Webservice zu schicken.

Die erhaltenen Informationen werden dann anhand dieser Konfiguration verschickt.

```
public void SendImage()
{
    try
    {
        ServiceReference1.Service1Client client = new
        ServiceReference1.Service1Client("BasicHttpBinding_IService1");
        byte[] fileByte = File.ReadAllBytes(imagePath);
        client.UploadImage(fileName, imageInformation, fileByte);
    }
    catch (Exception ex)
    {
        using (StreamWriter writer = File.AppendText(logFilePathTransHand))
        {
            writer.WriteLine("{0}|{1}|{2}|{3}", imagePath, imageInformation, file
            Name, ex.Message);
        }
    }
}
```

UploadImage ist eine Funktion welche von dem Webservice zur Verfügung gestellt wird.

5.2.3.2. FRQ-006 Endpunktadresse muss konfigurierbar sein

Wie bereits im vorherigen Kapitel 5.2.3.1 gezeigt, kann die Endpunkt Adresse konfiguriert werden in dem Konfigurationsfile „app.config“.

5.3. Empfänger und Translator

Die Umsetzung des Empfängers und Translators werden in diesem Kapitel zusammengefasst und detailliert beschrieben.

5.3.1. Empfänger

Der Empfänger ist ein Webservice welches anhand des WCF Framework umgesetzt worden ist.

5.3.1.1. FRQ-007 Bild inkl. Bildinformationen empfangen

Damit Daten empfangen werden können, muss der Webservice sich auch an die ABC Regel halten.

Die Adresse und das Binding werden in der Konfigurationsdatei definiert:

Adresse

```
<host>
  <baseAddresses>
    <add baseAddress = "http://localhost:1234" />
  </baseAddresses>
</host>
```

Binding

```
<bindings>
  <basicHttpBinding>
    <binding name="basicHttpEndpointBinding" closeTimeout="01:01:00"
      openTimeout="01:01:00" receiveTimeout="01:10:00" sendTimeout="01:01:00"
      allowCookies="false" bypassProxyOnLocal="false"
      hostNameComparisonMode="StrongWildcard"
      maxBufferSize="2147483646" maxBufferPoolSize="2147483646"
      maxReceivedMessageSize="2147483646"
      messageEncoding="Mtom" textEncoding="utf-8"
      transferMode="StreamedRequest"
      useDefaultWebProxy="true">
      <readerQuotas maxDepth="2147483646" maxStringContentLength="2147483646"
        maxArrayLength="2147483646"
        maxBytesPerRead="2147483646" maxNameTableCharCount="2147483646" />
      <security mode="None">
        <transport clientCredentialType="None" proxyCredentialType="None"
          realm="" />
        <message clientCredentialType="UserName" algorithmSuite="Default"
          />
      </security>
    </binding>
  </basicHttpBinding>
</bindings>
```

Im Binding werden alle benötigten Informationen zum Austausch der Daten definiert. Für den PoC wurde Sicherheit nicht als höchste Priorität angesehen, weil die Implementation stark von der Implementation der Sicherheit in dem Unternehmen abhängt.

Der **Contract** (Vertrag) wird in einem Service Interface definiert, welches dann von eigentlichen Service abgeleitet wird:

```
public interface IService1
{
    [OperationContract]
    void UploadImage(string fileName, string fileInfo, byte[] data);
}
```

In diesem Bereich werden diese Methoden definiert, welche der Webservice öffentlich zur Verfügung stellt. Die Methode wird mit dem Tag [OperationContract] beschrieben.

Die empfangenen Informationen werden danach aufbereitet und an den Translator weitergeleitet.

```
Writer = new BinaryWriter(File.OpenWrite(filePath));

// Writer raw data
Writer.Write(data);
Writer.Flush();
Writer.Close();

//start translation to text
Translator trans = new Translator(fileName);
```

Da die Daten binär daherkommen müssen Sie mit Verwendung von BinaryWriter auf die Harddisk geschrieben „Flush()“ werden.

5.3.1.2. FRQ-011 Bildinformationen im Output File.

Sobald die Daten vom Translator übersetzt worden sind. Erhält der Service den Pfad zum Output File zurück, wobei dann die übrigen Bildinformationen angehängt werden können. Dies geschieht mit der Verwendung von StreamWriter:

```
string fileOutPath = trans.transLate() + ".txt";

using (StreamWriter writer = File.AppendText(fileOutPath))
{
    writer.WriteLine(Environment.NewLine+fileInfo);
}
```

5.3.2. Translator

Sobald der Translator das Bild vom Empfänger erhält wird das Bild in Text umgewandelt. In den nachfolgenden Kapiteln wird die Umsetzung detaillierter beschrieben.

5.3.2.1. FRQ-008 Bild in Text umwandeln

Der Translator erhält der Pfad zum Bild, welches sich nun lokal auf dem Server befindet. Danach wird das Bild mit Tesseract umgewandelt:

```
public string transLate()
{
    string outputName = Path.GetFileNameWithoutExtension(imageName)+".txt";
    string pathToImage = readInput + "\\\" + imageName;
    string pathToOutput = readOutput + "\\\" + outputName;

    string language = "eng";

    string command = tesseract+" " + pathToImage + " " + pathToOutput + " -l " +
        language;
    Process pProcess = new Process();

    try
    {
        pProcess.StartInfo.FileName = tesseract;
        pProcess.StartInfo.Arguments = pathToImage + " " + pathToOutput + " -l "
            + language;
        pProcess.StartInfo.UseShellExecute = false;
        pProcess.StartInfo.CreateNoWindow = true;
        DateTime startTime = DateTime.Now;
        pProcess.Start();
        pProcess.WaitForExit();
        DateTime endTime = DateTime.Now;
        TimeSpan duration = endTime.Subtract(startTime);
        pProcess.Close();
        File.Delete(pathToImage);
        using (StreamWriter writer = File.AppendText(logFile))
        {
            writer.WriteLine("{0}|{1}|{2}|{3}", pathToImage,
                startTime.ToString(), endTime.ToString(), duration.Duration());
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    finally
    {
        pProcess.Close();
    }

    return pathToOutput;
}
```

Im Try Block wird das Bild dann von Tesseract in Text umgewandelt und lokal abgespeichert. Tesseract wird auf der Befehlsebene ausgeführt, weil erstens, es sehr einfach austauschbar ist und zweitens, Argument zum Ausführen mitgegeben werden können.

5.3.2.2. FRQ-009 Text abspeichern

Tesseract speichert der Ouptut automatisch mit folgender Konfiguration lokal ab:

Outputfile definition:

```
string outputName = Path.GetFileNameWithoutExtension(imageName)+".txt";
```

Parameter zum Ausführen von Tesseract:

```
pProcess.StartInfo.Arguments = pathToImage + " " + pathToOutput + " -l " + language;
```

Der Parameter pathToOutput ist der Output Pfad zum erstellten Textfile.

5.3.2.3. FRQ-010 Input und Output für Tranlator konfigurierbar

Input und Output Pfad werden in der Konfigurationsdatei app.config definiert:

```
<add key="output" value="C:\Users\Roger\Pictures\Output" />  
<add key="input" value="C:\Users\Roger\Pictures\Input"/>
```

Und können folgendermassen gelesen werden:

```
this.readInput = ConfigurationSettings.AppSettings["input"];  
this.readOutput = ConfigurationSettings.AppSettings["output"];
```

5.3.2.4. FRQ-012 Translator muss alle Ausführungen loggen

Bei jeder Ausführung von Tesseract wird ein Logfileeintrag erstellt:

```
using (StreamWriter writer = File.AppendText(logFile))  
{  
    writer.WriteLine("{0}|{1}|{2}|{3}", pathToImage, startTime.ToString(),  
        endTime.ToString(), duration.Duration());  
}
```

Es werden folgende Information ins Logfile geschrieben:

- Pfad zum Bild
- Start Zeit
- End Zeit
- Durchführungszeit

Zudem kann der Logfilepfad ebenfalls im App.config unter appSettings definiert werden.

```
<add key="logfile" value="C:\Users\Roger\Pictures\Output\translator.txt"/>
```


Und folgendermassen gelesen werden

```
this.logFile = ConfigurationSettings.AppSettings["logfile"];
```

5.4. Nicht funktionale Anforderungen

Die Umsetzung von den nicht-funktionalen Anforderungen wird in den folgenden Unterkapiteln genauer erläutert. Jedoch werden nur diese aufgegriffen, welche auch Sinn machen.

5.4.1. NFRQ-001 Angemessenheit

Wie in den vorherigen Kapiteln ersichtlich, wurde jede Anforderung umgesetzt und ist somit erfüllt.

5.4.2. NFRQ-002 Interoperabilität

Das Produkt unterstützt in diesem Zustand nur Windows Plattformen, so wie es definiert worden ist.

5.4.3. NFRQ-003 Sicherheit

Grundsätzlich muss das System, auf dem das Produkt (Sender, Empfänger und Translator) installiert werden, die vorgegebene Sicherheit erfüllen. Einzig die Sicherheit beim Transport der Daten vom Sender zum Empfänger muss gewährleistet werden. Weil jedoch die Implementierung der Sicherheit mit Zertifikaten abhängig von dem Unternehmen ist, wurde keine sichere Übertragung gewählt, sondern ganz normale http Übertragung.

5.4.4. NFRQ-004 Fehlertoleranz, NFRQ-005 Wiederherstellbarkeit und NFRQ-008 Analysierbarkeit

Bei einem Fehlerverhalten vom Empfänger werden alle nicht-versendeten Bilder inklusiv Bildinformationen in ein separates Logfile geschrieben, welches beim Neustart vom Sender aufgegriffen wird.

Beim Senden der Daten werden Fehler abgefangen und in ein Logfile geschrieben, wie im Kapitel 5.2.3.1 ersichtlich ist. Beim Starten des Senders wird dann dieses Logfile aufgegriffen und versucht das Bild nochmals zu verschicken:

```
string logFileTransHandler = ConfigurationSettings.AppSettings["logFileTransHandler"];  
  
var failureLine = "";  
  
//read all lines in from the logfile which was created by TransferHandler  
in case of Errors.  
if (File.Exists(logFileTransHandler))  
{  
  
    var lines = File.ReadAllLines(logFileTransHandler);
```

```
File.Delete(logFileTransHandler);  
foreach (var item in lines)  
{  
    string[] info = item.Split(new Char[] { '|' });  
    string imagePath = info[0];  
    string imageInformation = info[1];  
    string fileName = info[2];  
  
    TransferImageHandler tim = new TransferImageHandler(imagePath,  
        imageInformation, fileName);  
    tim.SendImage();  
}  
}
```

Das Logfile ist auch hier konfigurierbar in dem App.conf mit dem key „logFileTransHandler“. Falls es beim Neustart immer noch Probleme mit dem Empfänger gibt, wird das Logfile komplett eingelesen und danach gleich gelöscht. Somit wird sichergestellt, dass bei einem erneuten Fehler, den Eintrag nicht doppelt vorhanden ist. Das Erstellen eines separaten Logfiles hilft vor allem auch bei Analysearbeiten in Problemfällen.

5.4.5. NFRQ-006 Zeitverhalten und NFRQ-007 Verbrauchsverhalten

Beim Modul wurde so wenig Logik wie möglich hinzugefügt, so dass er möglichst keinen Einfluss auf den Webserver hat. Beim Übertragen der Daten ist es einfach Abhängig von der Bandbreite, der Grösse des Bildes und wie viele Bilder aufs Mal verschickt werden. Auf der Entwicklungsumgebung hat der Translator (Bild zu Text) eine durchschnittliche Ausführungszeit von ~8 Sekunden, was nicht besonders optimal ist.

5.4.6. NFRQ-009 Installierbarkeit

Da es mit Visual Studio erstellt worden ist, wird nach jedem Built ein exe File erstellt, welches auf der definierten unterstützten Plattform installieren werden kann.

5.4.7. NFRQ-010 Austauschbarkeit

Die Austauschbarkeit ist ein ziemlicher Vorteil eines Webservices. Der Sender kann grundsätzlich durch ein anderes Programm ausgetauscht werden. Es muss nur möglich sein, die REST API zu verwenden. Der Translator kann mit geringem Programmieraufwand ausgetauscht werden. Einzige Bedingung ist, dass es ein Befehlszeilen „Commandline“ Tool ist.

6. Testing

In dem nachfolgenden Kapitel wird das umgesetzte Produkt getestet. Einerseits wird das mit Unit Test (Automatisiertem Testen) gemacht, andererseits wird ein User Akzeptanz Test das Produkt auf Herz und Nieren testen.

6.1. Unit Test

In diesem Bereich werden die automatisierten Tests, sogenannte Unit Tests durchgeführt. Dabei werden die einzelnen Klassen so gut wie möglich unabhängig zu einander getestet. Bei einigen Tests ist das jedoch nicht möglich, weil gewisse Abhängigkeiten zu anderen Klassen vorhanden sind.

6.1.1. Sender

Beim Sender wurde der Fokus des Testes auf den LogListener und auf den TransferImageHandler gesetzt. In den folgenden Unterkapiteln werden die jeweiligen Funktionen inklusiv Methoden getestet.

6.1.1.1. LogListener

Zuerst wird die Klasse LogListener unabhängig erstellt und getestet. Damit Visual Studio die Klasse als Testklasse ansieht muss die Option [TestClass] oberhalb der Klasse stehen, sowie auch die Methode muss mit der Option [TestMethod] definiert werden:

```
[TestClass]
public class UnitTestBildanalyse1
{
    [TestMethod]
    public void TestLogListener()
    {
        //starte loglistener
        string logFilePath = @"C:\Users\Roger\Pictures\Bildanalyse\test.txt";
        string directoryPath = @"C:\Users\Roger\Pictures\Bildanalyse";
        string logFileName = "test.txt";
        LogListener logListener = new LogListener(logFilePath);

        //check logdirectory
        Assert.AreEqual(directoryPath, logListener.logDirectory);

        //check logfilepat
        Assert.AreEqual(logFilePath, logListener.logPath);

        //check logfileName
        Assert.AreEqual(logFileName, logListener.logFileName);
    }
}
```

Als erstes wird ein neuer LogListener erstellt und die benötigten Informationen der Klasse übergeben. Danach wird mit der Assert.AreEqual() Methode zwei Eingaben miteinander verglichen. Dabei werden alle möglichen Properties dieser Klasse überprüft, ob sie auch genau das zurückgeben, wie es erwartet wird.

Anschliessend wurde der ganze Ablauf „End-to-End“ von dieser Klasse aus getestet, ob das Endresultat, ein Textfile, erstellt wird. Dazu wurde ein Logeintrag der Logger Komponente simuliert:

```
using (StreamWriter fileWriter = File.AppendText(logFilePath))
{
    string user = "Hans";
    string server = "svtest";
    string code = "200";
    string mime = "image/jpg";
    string app = "TestApp";
    string path = @"C:\Users\Roger\Documents\Visual Studio
2013\Projects\ConsoleApplication3\Log\IMG_20140723_113303.jpg";
    fileWriter.WriteLine("{0:HH:mm:ss}|{1}|{2}|{3}|{4}|{5}|{6}",
        DateTime.Now, path, mime, code, user, app, server);
}

object source = new Object();
WatcherChangeTypes test = new WatcherChangeTypes();
FileSystemEventArgs e = new FileSystemEventArgs(test, directoryPath,
logFileName);
logListener.OnChanged(source, e);

//check if file is sendet and translated to text
Assert.AreEqual(File.Exists(@"C:\Users\Roger\Pictures\Output\svtestIMG_20140723_
113303.txt"), true);
```

Mit StreamWriter wird einen möglichen Logfileeintrag zur Simulation erstellt und anschliessend die OnChanged() Methode aufgerufen, damit der LogListener der Eintrag im Logfile einliest (FRQ-004). Nachdem der Logeintrag geschrieben wurde, werden diese Informationen weitergeleitet, bis schlussendlich der Text vom Bild abgespeichert wird (FRQ-005). Der abgespeicherte Text wird danach mit Assert.AreEqual überprüft. Dazu wird die File Methode Exists (existiert) aufgerufen. Der Rückgabewert wird entweder true (wahr) oder false (falsch) zurückgeben, ob das File existiert. Danach muss nur noch mit true (wahr) verglichen werden.

Visual Studio hat einen Test Explorer, welcher beim Ausführen der Tests behilflich ist. Dieser sieht folgendermassen aus:

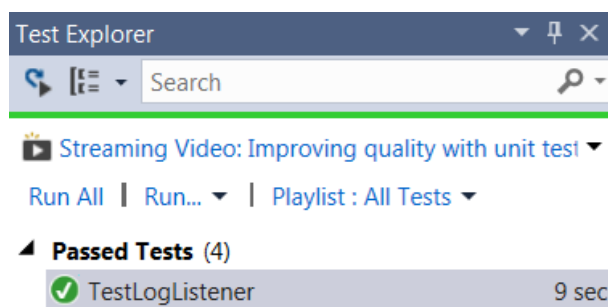


Abbildung 12 Test Explorer

Dabei wird im Falle eines positiven Testfalls ein grünes Häkchen angehängt oder bei einem negativen, direkt das Problem beschrieben.

6.1.1.2. TransferImageHandler

TransferImageHandler ist die Klasse, welche das Bild an den Webservice weiterleitet. Um den Transfer zu testen musste dabei der Service als Referenz hinzugefügt und entsprechend konfiguriert werden. Die Methode TestTransferImageHandler wurde folgendermassen umgesetzt:

```
[TestMethod]
public void TestTransferImageHandler()
{
    string user = "Hans";
    string serverName = "svtest";
    string statusCode = "200";
    string mime = "image/jpg";
    string imageInformation = "User: " + user + ", Server: " + serverName +
        ", StatusCode: " + statusCode + ", Mime-Type: " + mime;
    string path = @"C:\Users\Roger\Documents\Visual Studio
        2013\Projects\ConsoleApplication3\Log\IMG_20140723_113303.jpg";

    //string imagePath = @"C:\Users\Roger\Documents\Visual Studio
        2013\Projects\ConsoleApplication3\Log\IMG_20140723_113303.jpg";
    string imageName = serverName + Path.GetFileName(path);

    TransferImageHandler trans = new TransferImageHandler(path,
        imageInformation, imageName);
    trans.SendImage();

    //check path
    Assert.AreEqual(path, trans.imagePath);

    //check imageInformation
    Assert.AreEqual(imageInformation, trans.imageInformation);

    //check imageName
    Assert.AreEqual(imageName, trans.fileName);

    //check if file is sended and translated to text
    Assert.AreEqual(File.Exists(@"C:\Users\Roger\Pictures\Output\svtestIMG_20
        140723_113303.txt"), true);
}
```

Ebenfalls wurde in diesem Beispiel die Klasse zuerst unabhängig getestet. Das heisst, er wurde ein Instanz erstellt und die Properties danach überprüft mit der Assert.AreEqual() Methode. Zudem wurde auch ab dem TransferHandler der restliche Ablauf getestet indem ein Test Bild vom TransferHandler Instanz an den Webservice verschickt und danach mit File.Exists() der Output überprüft wird.

6.1.2. Empfänger

Beim Empfänger wird der Webservice automatisiert getestet. Die nachfolgenden Unterkapitel beschreiben die ausgeführten Tests in Detail.

6.1.2.1. Webservice

Beim Testen von einem Webservice ist es jeweils wichtig, dass der Service während des Tests am Laufen ist. Die Testmethode sieht folgendermassen aus:

```
[TestMethod]
public void TestTransferImageToService()
{

    //path|mime|code|user|app|servername
    string user = "Hans";
    string serverName = "svtest";
    string statusCode = "200";
    string mime = "image/jpg";

    string imagePath = @"C:\Users\Roger\Documents\Visual Studio
2013\Projects\ConsoleApplication3\Log\IMG_20140723_113303.jpg";

    string imageInformation = "User: " + user + ", Server: " + serverName + ",
    StatusCode: " + statusCode + ", Mime-Type: " + mime;

    string fileName = "IMG_20140723_113303.jpg";

    //create a Service client and send the image to Webservice
    Service1Client client = new Service1Client("BasicHttpBinding_IService1");
    Byte[] fileByte = File.ReadAllBytes(imagePath);
    client.UploadImage(fileName, imageInformation, fileByte);

    Assert.AreEqual(File.Exists(@"C:\Users\Roger\Pictures\Output\IMG_20140723
_113303.txt"), true);

    string lastLine = "";

    var lines = File.ReadLines(
        @"C:\Users\Roger\Pictures\Output\IMG_20140723_113303.txt");
    string lastLine = lines.Last();

    Assert.AreEqual(imageInformation, lastLine);
}
```

Der Webservice kann getestet werden indem zuerst einen ServiceClient erstellt wird. Anhand vom Service Client können auf die publizierten Methoden des Services zugegriffen werden. In diesem Beispiel wurde ein Test Bild mit allen wichtigen zusätzlichen Bildinformationen an den Service geschickt und das Resultat danach überprüft (FRQ-007). Zum Schluss werden die Bildinformationen, welche mitgeschickt worden sind, mit der letzten Line des Output Files überprüft (FRQ-011).

Hiermit wurde der Service Client und die Methode UploadImage getestet.

6.1.3. Translator

Beim Translator wird das Umwandeln von Bild in Text überprüft. Diese Komponente kann unabhängig getestet werden, da diese Komponente nur vom Webservice aufgerufen wird.

6.1.3.1. Translate

Die Methode Translate hat die Aufgabe das Bild in den Text umzuwandeln. Dies wird mit der OCR Software Tesseract durchgeführt. Hierbei wird jedoch nur getestet, ob die Methode auch das gewünschte Resultat als Output hat.

```
[TestMethod]
public void TestTranslator()
{
    string fileName="IMG_20140723_113303.jpg";
    string source = @"C:\Users\Roger\Documents\Visual Studio
2013\Projects\ConsoleApplication3\Log";
    string dest = @"C:\Users\Roger\Pictures\Input";

    string sourceFile = Path.Combine(source, fileName);
    string destFile = Path.Combine(dest, fileName);

    File.Copy(sourceFile, destFile, true);
    Translator trans = new Translator(fileName);
    string pathOutputFile = trans.translate();

    string expectedOutputPath =
"C:\\Users\\Roger\\Pictures\\Output\\IMG_20140723_113303";

    Assert.AreEqual(expectedOutputPath, pathOutputFile);
}
```

Der Rückgabewert der Methode translate ist der Outputpfad des Textfiles, welches umgewandelt worden ist vom Bild (FRQ-008 und FQR-009). Da der Outputpfad bekannt ist, werden diese zwei Werte mit AreEqual() verglichen. Falls sie gleich sind, ist der Testfall positiv.

6.1.4. Unit Test Abdeckung

In der folgenden Tabelle wird die Abdeckung der einzelnen Methoden aufgezeigt.

Komponente	Funktion	Methode	Keine Abdeckung	Abdeckung
Sender	LogListener	LogListener	0.0%	100%
Sender	LogListener	OnChanged	14.29%	85.71%
Sender	TransferImageHandler	SendImage	25%	75%
Sender	TransferImageHanlder	TransferImageHandler	0.0%	100%
Empfänger	Webservice	ServiceClient	0.0%	100%
Empfänger	Webservice	UploadImage	33.33%	66.67%
Translator	Translator	Translate	0.0%	100%
Total			10.38%	89.62%

Tabelle 41 Test Abdeckung

6.1.5. Test Resultat

Im der nachfolgender Grafik sieht man die Test Resultat von den Unit Tests und ihre Ausführungszeit.

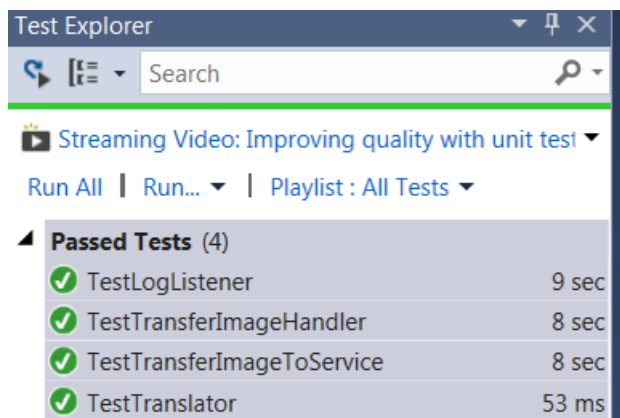


Abbildung 13 Test Resultat

6.2. User Akzeptanz Tests

Mit dem User Akzeptanz Test werden diese Tests durchgeführt, welche nicht unbedingt mit dem Unit Test abgefangen werden können.

6.2.1. Logger FRQ-001, FRQ-002, FRQ-003

Mit dem nachfolgenden Testfall werden die drei funktionalen Anforderungen FRQ-001, FRQ-002 und FRQ-003 überprüft.

Schritt ID	Beschreibung	Aktuell	Resultat
1	Öffne die Website Bildanalyse	Seite öffnet sich.	Passed
2	Gehe auf die Seite „Image“	Bilder werden geladen	Passed
3	Schaue in das web.config wo das Logfile ist. Unter Appsettings mit der ID logFilePath	C:\inetpub\logs\LogFiles\test_end.txt	Passed
4	Kontrolliere ob das Logfile vorhanden ist auf dem System	Logfile ist vorhanden	Passed
5	Öffne das Logfile und überprüfe ob es einen neuen Eintrag mit folgender Struktur erstellt worden ist: datetime path mime code user app servername	Die erwähnt Struktur wurde erzeugt	Passed

Tabelle 42 UAT FQR-001, FQR-002, FRQ-003

Das IIS Modul erkennt dass ein Bild an einen User weitergeleitet worden ist (FRQ-001). Anschliessend wurde ein Logeintrag generiert mit der definierten Struktur (FRQ-002). In der Konfigurationsdatei kann das Logfile konfiguriert werden mit dem Schlüsselwort logFilePath (FRQ-003).

6.2.2. Transfer Handler FRQ-006

Mit dem nachfolgenden Testfall wird die funktionale Anforderung FRQ-006 überprüft.

Schritt ID	Beschreibung	Aktuell	Resultat
1	Öffne das App.config vom Sender	App.config öffnet sich	Passed
2	Überprüfe ob die Enpunkadresse konfigurierbar ist unter configuration → system.serviceModel → client → endpoint address	Die Endpunktadresse ist konfigurierbar	Passed

Tabelle 43 UAT FQR-006

Die Endpunktadresse des Webservices kann über das App.config vom Sender konfiguriert werden.

6.2.3. Translator FRQ-010

Mit dem nachfolgenden Testfall wird die funktionale Anforderung FRQ-010 überprüft.

Schritt ID	Beschreibung	Aktuell	Resultat
1	Öffne das App.config vom Webservice	App.config öffnet sich	Passed
2	Überprüfe ob das Input Verzeichnis konfigurierbar ist: configuration → appSettings → key = „input“	Das Input Verzeichnis kann in der Konfigurationsdatei vom Webserver konfiguriert werden	Passed
3	Überprüfe ob das Output Verzeichnis konfigurierbar ist: configuration → appSettings → key = „output“	Das Output Verzeichnis kann in der Konfigurationsdatei vom Webserver konfiguriert werden	Passed

Tabelle 44 UAT FRQ-010

Die funktionale Anforderung, dass das Input und das Output Verzeichnis vom Translator konfigurierbar muss sein, ist somit erfüllt und der Test erfolgreich bestanden.

6.2.4. Translator FRQ-012

Mit dem nachfolgenden Testfall wird die funktionale Anforderung FRQ-012 überprüft.

In diesem Testfall muss vorher ein Bild übersetzt worden sein, bevor der Test durchgeführt werden kann. Da der Logfileeintrag dynamisch anhand der Zeit erstellt wird, ist es wesentlich einfacher diesen Fall manuell zu überprüfen.

Schritt ID	Beschreibung	Aktuell	Resultat
1	Öffne das App.config vom Webservice	App.config öffnet sich	Passed
2	Schaue wo das Logfile des Translators erstellt wird. Unter appSettings → key = logFile	Aktuell ist es eingestellt auf folgendes Logfile: C:\Users\Roger\Pictures\Output\translator.txt	Passed
3	Überprüfe in diesem Logfile, ob einen Eintrag mit folgender Struktur erstellt worden ist: Bild Startzeit Endzeit Ausführungszeit	C:\Users\Roger\Pictures\Input\IMG_20140723_113303.jpg 15.08.2015 11:25:05 15.08.2015 11:25:14 00:00:09.4225390	Passed

Tabelle 45 UAT FRQ-012

Der Logeintrag vom Translator wurde somit erfolgreich getestet.

7. Fazit

Die nachfolgenden Kapiteln wird ein Rückblick sowie ein Ausblick über die Arbeit beschrieben.

7.1. Rückblick

Durch die Anforderung von der FINMA an die Finanzunternehmen, die Überwachung von Mitarbeiter zu verschärfen, entstand der Grundgedanke dieser Arbeit. Viele Unternehmen haben sich danach an die Umsetzung herangewagt und überlegt wie der Zugriff geloggt werden kann. Die einfachste Methode ist die Analyse des Web-Verkehrs, also Analyse basierend auf http Requests und Responses. Die Analyse von Text, sprich dem Body, kann mit Textanalyse Programme umgesetzt werden, jedoch ist da noch das Problem eines Bildes, welches nicht so einfach analysiert werden kann.

Durch die Ist-Analyse wurde aufgezeigt wie momentan eine mögliche Webapplikation umgesetzt ist. Dadurch wird aufgezeigt, wo genau investiert werden muss, um ein Bild abzufangen. Der erste Teil des Produktes wird in diese Kette implementiert, damit die notwendigen Informationen abgefangen werden können.

Aufgrund der Recherche und der Evaluierung der potentiellen Lösung, wie Bilder transferiert und in Text umgewandelt werden können, wurde ein Konzept ausgearbeitet. Bei der erstellen der IT-Architektur wurde klar, dass diese Produkt aufgespaltet werden muss. Einerseits gibt es einen Sender, welche das Bild abfängt und an das Bildanalyse System weiterleitet. Die Hauptaufgabe findet dann auf dem Bildanalyse System statt, das Umwandeln eines Bildes in Text. Um die finale Architektur zu erstellen, wurden Evaluierungen verschiedener Programmen oder Methoden vorgenommen, um die bestmögliche Lösungen umzusetzen.

Die Umsetzung des PoC's erfolgte anhand der funktionalen sowie auch nicht-funktionalen Anforderungen. Mit Hilfe der Code Beispiele wurde aufgezeigt, dass die Umsetzung aller Anforderungen durchgeführt worden sind.

Damit sichergestellt werden kann, dass das Produkt auch wie gewünscht funktioniert wurden Unit Tests und User Akzeptanz Tests spezifisch auf die Anforderungen durchgeführt, um das Produkt auf Herz und Nieren zu überprüfen.

Abschliessend zeigt sich, dass sich das Aufzeigen der Ist-Situation und die methodischen Ansätze des Requirements-Engineering, der Lösungs-Selektion und –Evaluation bewährt haben, da dadurch eine fundamentierte Empfehlung über eine mögliche Lösung der Problemstellung dieser Arbeit umgesetzt werden kann.

7.2. Ausblick

Aufgrund das dieses Produkt nur ein PoC ist, gibt es einige Verbesserungen bis zu einer Enterprise-Ready Software. Vor allem in Bezug auf Performance und Stabilität, müsste das Produkt noch verbessert werden. Mit diesem Produkt ist jedoch eine gute Basis gelegt. Leider gibt es momentan nicht viele Befehlszeilen Programme um Bilder in Text umzuwandeln.

Performance

Die Ausführungszeit von Tesseract muss verbessert werden. Bei 8-9 Sekunden pro Bilder kann dies nicht an Unternehmen übergeben werden, welche vielleicht 1000 Bilder pro Minute haben. Darum muss ein skalierbares Produkt noch ausgearbeitet werden.

Web-Service

Die Übertragung von Information an einen Webserver muss effizienter durchgeführt werden, um ein Enterprise Produkt zu bekommen. Dabei ist es wichtig, dass für die Zukunft eine Lösung gefunden wird, bei dem eine Warteschlange implementiert wird, um den Translator nicht zu überlasten.

HTTP Modul

Das HTTP Modul ist etwas instabil bei mehreren Bilder auf der gleichen Webseite. Dabei kann es vorkommen, dass einige Bilder nicht an den User übertragen werden. Dies müsste noch verbessert werden.

Sicherheit

Die Übertragung der Informationen müssen sicher übertrag werden an das Bildanalyse System. Dies wurde mit dieser Arbeit nicht umgesetzt, weil es sich dabei um einen PoC handelt und die Übertragungssicherheit auch abhängig von Unternehmen selber ist.

Zukunftsgedanke

Eine zukünftige Lösung könnte ein Cluster System sein, welcher aus Arbeiter Systeme und Manager System besteht. Der Manager könnte die Warteschlange implementieren und die Tasks an die Arbeiter Systeme verteilen um dadurch bessere Performance und Stabilität zu erreichen.

8. Verzeichnisse

8.1. Quellenverzeichnis

MSDN – HTTP Applikation. (25.04.2015). [https://msdn.microsoft.com/de-de/library/vstudio/system.web.httpapplication_events\(v=vs.100\).aspx](https://msdn.microsoft.com/de-de/library/vstudio/system.web.httpapplication_events(v=vs.100).aspx) abgerufen

IIS – Framework. (16.05.2015) <http://www.iis.net/learn/develop/runtime-extensibility/developing-iis-modules-and-handlers-with-the-net-framework> abgerufen

IIS – Kette. (30.05.2015) http://i2.iis.net/media/7179629/aspnet-integration-with-iis-243-fig2.jpg?cdn_id=2015-04-08-001 abgerufen

GOOCR – OCR. (06.06.2015) <http://jocr.sourceforge.net/index.html> abgerufen

OCR – Tesseract. (28.04.2015) <http://tesseract-ocr.googlecode.com/svn/trunk/doc/tesseractidcar2007.pdf> abgerufen

OCR – Tesseract Trainings Daten. (28.04.2015) <https://code.google.com/p/tesseract-ocr/wiki/TrainingTesseract3> abgerufen

Typographie – Fachwörter. (01.08.2015) <https://en.wikipedia.org/wiki/Typeface> abgerufen

Typographie – Kerning. (01.08.2015) <https://en.wikipedia.org/wiki/Kerning> abgerufen

MSDN – HTTP Handler & HTTP Module. (11.07.2015) <https://msdn.microsoft.com/de-de/library/Bb398986%28v=vs.90%29.aspx> abgerufen

MSDN – HTTP Handler. (11.07.2015) https://msdn.microsoft.com/de-de/library/system.web.ihttphandler_members%28v=vs.90%29.aspx abgerufen

MSDN – Übertragungsmethoden. (25.07.2015) <https://msdn.microsoft.com/en-us/library/ms733769%28v=vs.110%29.aspx> abgerufen

IEEE 830-1998 – Software Spezifikation. (25.07.2015)
<http://www.google.ch/url?sa=t&rct=j&q=&esrc=s&frm=1&source=web&cd=2&ved=0CCMQFjABahUKewxzPnMr-TGAhXI6HIKHQivAjY&url=http%3A%2F%2Fwww.math.uaa.alaska.edu%2F~afkjm%2Fcs401%2FIEEE830.pdf&ei=Ch2qVbGfEuXRywOI3oqwAw&usq=AFQjCNHznuy4ji6mL-rEUUUm4QFS-moI2Q> abgerufen

ISO 9126 – Qualitätsmerkmale (25.07.2015) https://de.wikipedia.org/wiki/ISO/IEC_9126 abgerufen

8.2. Tabellenverzeichnis

Tabelle 1 Aufwand.....	3
Tabelle 2 Termine.....	4
Tabelle 3 Stakeholder Analyse	7
Tabelle 4 Schnittstellen-Analyse	9
Tabelle 5 UC-001 Abfangen von Bilder.....	12
Tabelle 6 UC-002 Verschicken von Bildern.....	13
Tabelle 7 UC-003 Transferieren von Bild in Text.....	14
Tabelle 8 Notwendigkeit	16
Tabelle 9 Kritikalität.....	16
Tabelle 10 Funktionale Anforderung FRQ-001.....	16
Tabelle 11 Funktionale Anforderung FRQ-002.....	17
Tabelle 12 Funktionale Anforderung FRQ-003.....	17
Tabelle 13 Funktionale Anforderung FRQ-004.....	17
Tabelle 14 Funktionale Anforderung FRQ-005.....	18
Tabelle 15 Funktionale Anforderung FRQ-006.....	18
Tabelle 16 Funktionale Anforderung FRQ-007.....	18
Tabelle 17 Funktionale Anforderung FRQ-008.....	18
Tabelle 18 Funktionale Anforderung FRQ-009.....	19
Tabelle 19 Funktionale Anforderung FRQ-010.....	19
Tabelle 20 Funktionale Anforderung FRQ-011.....	19
Tabelle 21 Funktionale Anforderung FRQ-012.....	20
Tabelle 22 Nicht funktionale Anforderung NFRQ-001	20
Tabelle 23 Nicht funktionale Anforderung NFRQ-002	20
Tabelle 24 Nicht funktionale Anforderung NFRQ-003	20
Tabelle 25 Nicht funktionale Anforderung NFRQ-004	21
Tabelle 26 Nicht funktionale Anforderung NFRQ-005	21
Tabelle 27 Nicht funktionale Anforderung NFRQ-008	22
Tabelle 28 Nicht funktionale Anforderung NFRQ-009	22
Tabelle 29 Nicht funktionale Anforderung NFRQ-010	22
Tabelle 30 Nicht funktionale Anforderung NFRQ-012	22
Tabelle 31 Nicht funktionale Anforderung NFRQ-013	23
Tabelle 32 Logger Kriterien	27
Tabelle 33 Logger Typen.....	27
Tabelle 34 Logger Bewertungstabelle	28
Tabelle 35 Service Kriterien.....	32
Tabelle 36 Service Typen.....	32
Tabelle 37 Service Bewertungstabelle	33
Tabelle 38 Translator Kriterien.....	33
Tabelle 39 Translator Typen	34
Tabelle 40 Translator Bewertungstabelle	34
Tabelle 41 Test Abdeckung.....	51
Tabelle 42 UAT FQR-001, FQR-002, FRQ-003.....	52
Tabelle 43 UAT FQR-006.....	52
Tabelle 44 UAT FRQ-010.....	53
Tabelle 45 UAT FRQ-012.....	53

Tabelle 49 Eventregistrierung	61
Tabelle 50 WCF Transport	69

8.3. Abbildungsverzeichnis

Abbildung 1 Projektplan	3
Abbildung 2 Legende	3
Abbildung 3 Webapplikation Lösung	6
Abbildung 4 Systemdiagramm	8
Abbildung 5 Input-Output Diagram	9
Abbildung 6 Bildanalyse System Prozess	15
Abbildung 7 Architektur	24
Abbildung 8 Logger	26
Abbildung 9 Listener	29
Abbildung 10 Sender Transfer Handler	30
Abbildung 11 Service / Translator	31
Abbildung 12 Test Explorer	47
Abbildung 13 Test Resultat	51
Abbildung 14 IIS Pipeline	59
Abbildung 15 Log Definition	62
Abbildung 16 Webservice Architektur	64
Abbildung 17 Tesseract Linenfinder	70
Abbildung 18 Tesseract Baseline	71
Abbildung 19 Tesseract Charaktererkennung	71
Abbildung 20 Tesseract Kerning	71
Abbildung 21 Tesseract Überlappung 1	72
Abbildung 22 Tesseract Überlappung 2	72

9. Anhang

9.1. Methoden zum Abfangen von Bilder

9.1.1. Module

Es gibt zwei Arten von Manipulieren eines IIS 7, Module und Handler. Ein Handler wird vorallem eingesetzt um Requests zu behandeln und den Response zu manipulieren. Ein Module wird erstellt, wenn der Request prozessiert werden muss. Also das heisst, vorallem wenn der Inhalt analysiert werden muss, wie zum Beispiel für das Logging und Monitoring. Der Sender ist nichts anderes als ein Logger der Webapplikation..

Seit IIS 7 werden die Module direkt in der IIS Pipeline integriert. Dies ermöglicht es, ein Modul zu platzieren, indem es an einen Event registriert wird. Folgendermassen sieht die Architektur von einem IIS aus:

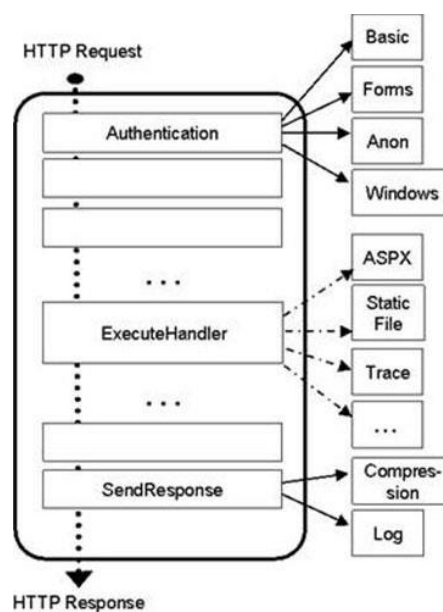


Abbildung 14 IIS Pipeline

Quelle: http://i2.iis.net/media/7179629/aspnet-integration-with-iis-243-fig2.jpg?cdn_id=2015-04-08-001

Wo das Modul in der IIS Kette aufgerufen wird, ist abhängig von der Registrierung an einem Event. An folgende Event kann ein Modul registriert werden:

Name	Beschreibung
AcquireRequestState	Tritt ein, wenn ASP.NET den aktuellen Zustand (z. B. den Sitzungszustand) erhält, der der aktuellen Anforderung zugeordnet ist.
AuthenticateRequest	Tritt ein, wenn die Identität des Benutzers von einem Sicherheitsmodul eingerichtet wurde.
AuthorizeRequest	Tritt ein, wenn die Benutzerautorisierung von einem Sicherheitsmodul überprüft wurde.
BeginRequest	Tritt als erstes Ereignis in der HTTP-Pipelinekette der Ausführung ein, wenn ASP.NET auf eine Anforderung antwortet.
Disposed	Tritt ein, wenn die Anwendung verworfen wird.
EndRequest	Tritt als letztes Ereignis in der HTTP-Pipelinekette der Ausführung ein, wenn ASP.NET auf eine Anforderung antwortet.
Error	Tritt beim Auslösen einer nicht behandelten Ausnahme ein.
LogRequest	Tritt auf, bevor ASP.NET eine Protokollierung für die aktuelle Anforderung ausführt.
MapRequestHandler	Infrastruktur. Tritt auf, wenn der Handler ausgewählt wird, um auf die Anforderung zu reagieren.
PostAcquireRequestState	Tritt ein, wenn der Anforderungszustand (z. B. der Sitzungszustand) abgerufen wurde, der der aktuellen Anforderung zugeordnet ist.
PostAuthenticateRequest	Tritt ein, wenn die Identität des Benutzers von einem Sicherheitsmodul eingerichtet wurde.
PostAuthorizeRequest	Tritt ein, wenn der Benutzer für die aktuelle Anforderung autorisiert wurde.
PostLogRequest	Tritt auf, wenn die Verarbeitung aller Ereignishandler für das LogRequest-Ereignis von ASP.NET abgeschlossen wurde.
PostMapRequestHandler	Tritt ein, wenn ASP.NET dem entsprechenden Ereignishandler die aktuelle Anforderung zugeordnet hat.
PostReleaseRequestState	Tritt ein, wenn ASP.NET das Ausführen aller Ereignishandler der Anforderung abgeschlossen hat und die Zustandsdaten der Anforderung gespeichert wurden.
PostRequestHandlerExecute	Tritt ein, wenn der ASP.NET-Ereignishandler (z. B. eine Seite oder ein XML-Webdienst) die Ausführung beendet.
PostResolveRequestCache	Tritt ein, wenn ASP.NET die Ausführung des aktuellen Ereignishandlers umgeht und ermöglicht, dass ein Cachemodul eine Anforderung aus dem Zwischenspeicher behandelt.
PostUpdateRequestCache	Tritt ein, wenn ASP.NET die Aktualisierung von Cachemodulen und das Speichern von Antworten abschließt, mit denen nachfolgende Anforderungen aus dem Cache behandelt werden.
PreRequestHandlerExecute	Tritt unmittelbar vor dem Moment ein, bevor ASP.NET einen Ereignishandler (z. B. eine Seite oder einen XML-Webdienst) ausführt.
PreSendRequestContent	Tritt ein, kurz bevor ASP.NET Inhalt an den Client sendet.
PreSendRequestHeaders	Tritt ein, kurz bevor ASP.NET HTTP-Header an den Client sendet.
ReleaseRequestState	Tritt ein, nachdem ASP.NET die Ausführung aller Ereignishandler der Anforderung abgeschlossen hat. Dieses Ereignis veranlasst die Zustandsmodule, die aktuellen Zustandsdaten zu speichern.

ResolveRequestCache	Tritt ein, wenn ASP.NET ein Autorisierungsereignis abschließt, damit die Cachemodule Anforderungen aus dem Cache behandeln können, wobei sie die Ausführung des Ereignishandlers (z. B. einer Seite oder eines XML-Webdiensts) umgehen.
UpdateRequestCache	Tritt ein, wenn ASP.NET die Ausführung eines Ereignishandlers abschließt, damit Cachemodule Antworten speichern können, die für das Behandeln nachfolgender Anforderungen aus dem Cache verwendet werden.

Tabelle 46 Eventregistrierung

Quelle: [https://msdn.microsoft.com/de-de/library/vstudio/system.web.httpapplication_events\(v=vs.100\).aspx](https://msdn.microsoft.com/de-de/library/vstudio/system.web.httpapplication_events(v=vs.100).aspx)

Beim Sender macht es am meisten Sinn sich an den Event EndRequest zu registrieren, weil viele Informationen vom Request und vom Response Header verwendet werden.

Das Modul muss das Interface IHttpModule integrieren, welches zwei Funktionen hat:

1. Dispose

Dispose wird ausgeführt sobald das Modul gestoppt wird. Alle verwendeten Ressourcen werden dann freigelassen.

2. Init

Init wird aufgerufen sobald das Modul gestartet wird. Der erste Aufruf ist die Event Registration.

9.1.2. Advances Logging

Advances Logging ist ein IIS Features, welches von Microsoft zur Verfügung gestellt wird. Die Einstellungen vom Logging können dadurch verändert werden. Diese Methode ist die am einfachsten umzusetzende Methode für IIS, um eine gewisse Struktur in ein Logfile zu bringen. Es kann eine neue Log Definition hinzugefügt werden über den IIS Manager.

Log Definition

Base file name:
TestLog

☒ Enabled
☐ Publish real-time events
☒ Write to disk

Log File Rollover

☒ Schedule
Daily
☐ Start new log file when configuration changes
☐ Maximum duration (in seconds)
86400
☐ Maximum file size (in kilobytes)
1024

Selected Fields

ID	Header Name	Required	Default Value
Date-UTC	date	No	
Time-UTC	time	No	
CPU-Utilization	CPU-Utilization	No	
RequestsPerSecond	RequestsPerSecond	No	
Author		No	
URI-Stem	cs-uri-stem	No	
Server-IP	s-ip	No	
Status	sc-status	No	
Substatus	sc-substatus	No	
Bytes Sent	sc-bytes	No	

Edit...
Move First
Move Up
Move Down
Move Last

Select Fields... Remove

Filter
Edit Filter...
Filter details:

Abbildung 15 Log Definition

Quelle: http://i1.iis.net/media/7178232/advanced-logging-for-iis---custom-logging-579-LogDefinition-Move.jpg?cdn_id=2015-07-21-001

In diesem Bereich können die Felder definiert werden, welche zum Loggen des Requests notwendig sind. Anschliessend wird der Logfilepfad noch angegeben und schon werden die Request in das Logfile geschrieben.

9.1.3. HTTP Handler

HTTP Handler werden vor allem zum Manipulieren von Requests benötigt. Sobald eine Anfrage an den Webserver kommt, kann der Handler eingreifen und den Response bearbeiten. Zum Beispiel, wenn ein Benutzer ein Bild herunterladen möchte in einem bestimmten Format, kann der Handler den Request entgegen nehmen und das Format des Bildes anpassen.

Der HTTP Handler muss das Interface `IHandler` implementieren. Dazu gehören zwei Funktionen, `ProcessRequest` und `IsReusable`.

`ProcessRequest` ermöglicht die Verarbeitung von Webanfragen durch einen benutzerdefinierten Handler. `IsReusable` ruft einen Wert ab, welcher angibt ob eine weitere Anforderung diese Instanz verwenden kann.

9.2. Übertragungsmethoden

Im folgenden Bereich werden die Übertragungs- und Empfangsmethoden beschrieben.

9.2.1. WCF (Windows Communication Foundation)

WCF ist eine dienstorientierte Kommunikationsplattform für Windows Systeme. Durch diese Plattform werden die Kommunikationstechnologien DCOM, MSMQ und Web-Service einheitlich zusammengefasst. Es ermöglicht es einfach Daten über das Netzwerk auszutauschen, zu manipulieren oder zu prozessieren.

DCOM (Distributed Component Object Model) ist ein objektorientiertes RPC-System (Remote Procedure Call). Es ermöglicht es eine Funktion von einem Server remote von einem Client aus aufzurufen über das Netzwerk.

MSMQ (Message Queuing) ist ein Protokoll von Microsoft, welches Nachrichten-Warteschlangen zur Verfügung stellt. MSMQ garantiert eine definitive Übertragung indem es die Nachrichten in Warteschlangen ablegt, bis der Service die Message entgegen nimmt. Dies auch wenn der Service kurze Zeit offline ist.

9.2.1.1. Web-Service

Ein Web-Service ist ein Programm, welches Funktionen in einem Netzwerk bereitstellt. Es wird über einen Uniform Resource Identifier (URI) aufgerufen. Dadurch ist es im Netzwerk eindeutig identifizierbar. Die Schnittstellenbeschreibung, also wie der Service integriert wird, wird durch WSDL (Web Service Description Language) definiert. Die Kommunikation läuft mehrheitlich über Internetprotokolle wie http und anderen XML-basierten Protokolle.

Webservice basieren auf serviceorientierten Architekturen (SOA) und vereinen somit verteilte und objektorientierte Programmierstandards.

Die WCF abstrahiert das Konzept des Endpunktes durch die Trennung von Address, Binding und Contract (ABC-Prinzip).

Die Adresse (Address) ist ein URI, der die eindeutige Identifikation im Netzwerk des Services beschreibt.

Die Anbindung (Binding) beschreibt die Art der Kommunikation, darunter fallen die Kodierung, Sicherheit und das verwendete Übertragungsprotokoll.

Der Vertrag (Contract) definiert die verfügbaren Methoden eines Dienstes.

Die Architektur sieht folgendermassen aus:

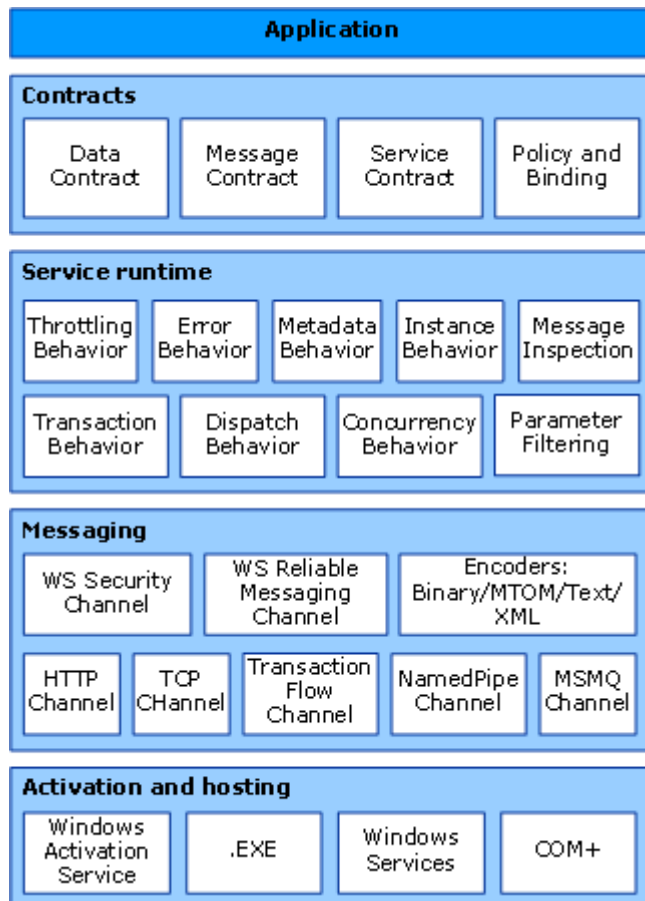


Abbildung 16 Webservice Architektur

Es beschreibt die Hauptelemente von WFC.

9.2.1.2. Verträge

Der Datenvertrag (Data Contract) beschreibt alle Parameter, welcher der Dienst erstellen und verarbeiten kann. Die Parameter werden in XSD-Dokumente (XML Schemadefinitionssprache) definiert. Dadurch kann jedes XML-fähiges System die Dokumente verarbeiten.

Der Nachrichtenvertrag (Message Contract) definiert anhand von SOAP-Protokollen bestimmte Nachrichtenteile und ermöglicht eine detaillierte Steuerung der Teile einer Nachricht.

Der Dienstvertrag (Service Contract) beschreibt die genauen Methodensignaturen eines Dienstes und wird als Schnittstelle in einer der unterstützenden Programmiersprache verteilt (C# oder Visual Basic).

Richtlinien und Bindungen legen die Bedingungen für eine Kommunikation fest. Bindungen liegt beispielsweise fest, dass mindestens ein verwendeter Transport (z.B. http) und eine Kodierung angegeben werden muss. Richtlinien schliessen vor allem Sicherheitsanforderungen ein.

9.2.1.3. Service Runtime

Service Runtime umfasst alle Verhaltensweise, die während der Ausführung des Dienstes auftreten.

- Throttling: Drosslung des Taktes eines Prozessors, bei bevorstehender Überhitzung
- Fehlerverhalten: Definiert das Fehlerverhalten bei einem internen Dienstfehler. Z.B. welche Informationen an den Client weitergereicht werden.
- Metadatenverhalten: Definiert wie und wo Metadaten öffentlich verfügbar gemacht werden.
- Instanzverhalten: Definiert wie viele Instanzen des Dienstes ausgeführt werden können.
- Transaktionsverhalten: Definiert einen Rollback von durchgeführten Aktionen bei einem Fehler.
- Verteilungsverhalten: Steuerung der Verarbeitung von Nachrichten durch die WCF-Infrastruktur

Die Erweiterbarkeit ermöglicht eine Anpassung der Laufzeitprozesse. Z.B. können mit der Nachrichtinspektion einzelne Teile einer Nachricht überprüft werden oder mit der Parameterfilterung Nachrichtenheaders anhand eines Filters durchsuchen.

9.2.1.4. Messaging

Es gibt zwei verschiedene Arten von Kanälen: die Transport- und Protokollkanäle.

Transportkanäle lesen und schreiben Nachrichten aus dem Netzwerk. Bei einigen Transporten wird ein Umwandler verwendet, um Nachrichten in und aus Bytestreamdarstellung zu konvertieren. HTTP, Pipes, TCP und MSMQ sind Beispiele für Transporte. Beispiele für Codierung sind XML und optimierte Binärdateien.

Protokollkanäle implementieren Nachrichtenverarbeitungsprotokolle, damit das Lesen und Schreiben von zusätzlichen Informationen in Headers einer Nachricht möglich ist. Zu diesen Protokollen gehörten beispielsweise WS-Security und WS-Reliability (Stellt Nachrichtenübermittlung sicher).

9.2.1.5. Hosting und Aktivierung

Ein Dienst ist ein Programm und muss wie andere Programme in einer ausführbaren Datei ausgeführt werden.

Dienste werden jedoch auch gehostet oder in einer ausführbaren Datei von einem externen Agent verwaltet ausgeführt. Entweder kann er manuell als .EXE oder per Windows Dienst ausgeführt werden.

9.2.2. HTTPTransport

HTTPTransport wird auf der MSDN Homepage von Microsoft sehr gut erläutert:

HTTP ist ein Anforderung-/Antwortprotokoll für den Austausch zwischen Clients und Servern. Diese Transportart wird am häufigsten von Webbrowserclients verwendet, die mit einem Webserver kommunizieren. Der Client sendet eine Anforderung an einen Server, der nach Anforderungsnachrichten von Clients lauscht. Wenn der Server eine Anforderung empfängt, gibt er eine Antwort zurück, die den Status der Anforderung enthält. Wenn dieser Vorgang erfolgreich ist, werden optionale Daten wie eine Webseite, eine Fehlermeldung oder andere Informationen zurückgegeben.

Das HTTP-Protokoll basiert nicht auf einer Verbindung. Nachdem die Antwort gesendet wurde, wird kein entsprechender Zustand beibehalten. Die Anwendung muss jeden erforderlichen Zustand beibehalten, um Transaktionen mit mehreren Seiten verarbeiten zu können.

Unter WCF wird die HTTP-Transportbindung aus Gründen der Interoperabilität mit älteren Nicht-WCF-Systemen optimiert. Wenn alle kommunizierenden Parteien WCF verwenden, sind auf TCP oder Named Pipes basierende Bindungen schneller.¹

9.2.3. TCPTransport

TCPTransport wird auf der MSDN Homepage von Microsoft sehr gut erläutert:

TCP ist ein verbindungsbasierter, datenstromorientierter Zustellungsdienst mit End-to-End-Fehlererkennung und -behebung. Verbindungsbasiert bedeutet, dass vor dem Datenaustausch eine Kommunikationssitzung zwischen Hosts eingerichtet wird.

TCP ermöglicht die zuverlässige Datenzustellung und Benutzerfreundlichkeit. TCP benachrichtigt den Absender der Paketzustellung, stellt sicher, dass Pakete auch in der Sendereihenfolge zugestellt werden, überträgt verloren gegangene Pakete neu und sorgt dafür, dass Datenpakete nicht dupliziert werden. Beachten Sie, dass diese zuverlässige Zustellung zwischen zwei TCP/IP-Knoten gilt. Dies entspricht nicht WS-ReliableMessaging, das für Endpunkte unabhängig davon gilt, wie viele Zwischenknoten diese enthalten.

Der WCF-TCP-Transport ist für das Szenario optimiert, bei dem beide Enden der Kommunikation WCF verwenden. Diese Bindung ist die schnellste WCF-Bindung für Szenarien, bei denen die Kommunikation zwischen verschiedenen Computern erfolgt. Die Vorgänge des Nachrichtenaustauschs verwenden das BinaryMessageEncodingBindingElement für die optimierte Nachrichtenübertragung. TCP ermöglicht die Duplexkommunikation und kann daher verwendet werden, um Duplexverträge zu implementieren, auch wenn der Client sich hinter NAT (Network Address Translation) befindet.²

¹ Quelle: <https://msdn.microsoft.com/de-de/library/ms733769%28v=vs.110%29.aspx>

² Quelle: <https://msdn.microsoft.com/de-de/library/ms733769%28v=vs.110%29.aspx>

9.2.4.NamePipeTransport

NamePipeTransport wird auf der MSDN Homepage von Microsoft sehr gut erläutert:

Eine benannte Pipe ist ein Objekt im Windows-Betriebssystemkernel, zum Beispiel ein Abschnitt eines gemeinsam genutzten Speichers, den Prozesse für die Kommunikation verwenden können. Eine benannte Pipe hat einen Namen und kann für die unidirektionale Kommunikation oder Duplexkommunikation zwischen Prozessen auf einem einzelnen Computer verwendet werden.

Wenn die Kommunikation zwischen verschiedenen WCF-Anwendungen auf einem einzelnen Computer erforderlich ist und Sie jegliche Kommunikation von einem anderen Computer verhindern möchten, können Sie den Transport mittels benannter Pipes verwenden. Eine weitere Einschränkung besteht darin, dass Prozesse, die über Windows-Remotedesktop ausgeführt werden, ggf. auf die Windows-Remotedesktopsitzung beschränkt sind, wenn sie nicht über erweiterte Berechtigungen verfügen.³

³ Quelle: <https://msdn.microsoft.com/de-de/library/ms733769%28v=vs.110%29.aspx>

9.2.5. WCF Funktionsübersicht

In der nachfolgenden Tabelle wird aufgezeigt, welcher WCF Transport bei welchen Situation eingesetzt werden können.

Attribut	Beschreibung	Häufig verwendete Transporte
Diagnose	Die Diagnose ermöglicht es Ihnen, Probleme mit der Transportkonnektivität automatisch zu erkennen. Alle Transporte unterstützen die Fähigkeit, Fehlerinformationen zurückzusenden, die die Konnektivität beschreiben. WCF enthält jedoch keine Diagnosetools zum Untersuchen von Netzwerkproblemen.	Keine
Hosting	Alle WCF-Endpunkte müssen innerhalb einer Anwendung gehostet werden. IIS 6.0 und ältere Versionen unterstützen nur Hostanwendungen, die den HTTP-Transport verwenden. Windows Vista unterstützt das Hosten von allen WCF-Transporten, auch von TCP und Named Pipes. Weitere Informationen finden Sie unter Hosten in Internetinformationsdiensten und Hosten in WAS (Windows Process Activation Service).	HTTP
Inspektion	Die Inspektion ist die Fähigkeit, während der Übertragung Informationen aus Nachrichten zu extrahieren und zu verarbeiten. Das HTTP-Protokoll trennt Routing- und Steuerungsinformationen von den Daten, um das Erstellen von Tools zu vereinfachen, die Nachrichten untersuchen und analysieren. Transporte, die leicht zu überprüfen sind, erfordern ggf. auch weniger Verarbeitungsleistung in Netzwerkeinrichtungen. Die verwendete Sicherheitsebene wirkt sich darauf aus, ob Nachrichten überprüft werden können.	HTTP
Wartezeit	Die Wartezeit ist die Mindestmenge an Zeit, die erforderlich ist, um einen Austausch von Nachrichten durchzuführen. Alle Netzwerkvorgänge weisen je nach gewähltem Transport mehr oder weniger Wartezeit (Latenz) auf. Das Verwenden der Duplexkommunikation oder unidirektionalen Kommunikation mit einem Transport, der das systemeigene Nachrichtenaustauschmuster Anforderung/Antwort verwendet, zum Beispiel HTTP, kann zu einer längeren Wartezeit führen, da die Korrelation von Nachrichten erforderlich ist. Erwägen Sie in dieser Situation, einen Transport zu verwenden, der als systemeigenes Nachrichtenaustauschmuster Duplex verwendet, zum Beispiel TCP.	TCP, Named Pipe
Reichweite	Die Reichweite eines Transports gibt an, in welchem Umfang der Transport eine Verbindung zu anderen Systemen herstellen kann. Der Transport mittels benannter Pipes besitzt nur eine geringe Reichweite. Er kann nur eine Verbindung zu Diensten herstellen, die auf dem gleichen Computer ausgeführt werden. Die Transportarten TCP und HTTP verfügen jeweils über eine ausgezeichnete Reichweite und können auch einige	HTTP, TCP

	NAT- und Firewallkonfigurationen durchdringen. Weitere Informationen finden Sie unter Arbeiten mit NATs und Firewalls.	
Sicherheit	<p>Die Sicherheit ist die Fähigkeit, während der Übertragung Nachrichten zu schützen, indem die Vertraulichkeit, Integrität oder Authentifizierung sichergestellt wird. Die Vertraulichkeit schützt eine Nachricht davor, untersucht zu werden, die Integrität schützt eine Nachricht davor, geändert zu werden, und die Authentifizierung liefert zuverlässige Informationen zum Absender oder Empfänger der Nachricht.</p> <p>WCF unterstützt die Übertragungssicherheit auf der Nachrichtenebene und auf der Transportebene. Die Nachrichtensicherheit wird mit einem Transport verknüpft, wenn der Transport einen gepufferten Übertragungsmodus unterstützt. Die Unterstützung für die Transportsicherheit ändert sich in Abhängigkeit vom gewählten Transport. Die Transportarten HTTP, TCP und benannte Pipe verfügen in Bezug auf die Unterstützung der Transportsicherheit über eine angemessene Parität.</p>	Alle
Durchsatz	Der Durchsatz misst die Datenmenge, die in einem bestimmten Zeitraum übertragen und verarbeitet werden kann. Wie die Wartezeit auch, kann der gewählte Transport sich auf den Durchsatz für Dienstvorgänge auswirken. Die Maximierung des Durchsatzes für einen Transport erfordert sowohl die Minimierung des Mehraufwands für die Übertragung von Inhalten als auch die Reduzierung des Wartezeitraums für den Abschluss des Nachrichtenaustauschs. Sowohl der TCP-Transport als auch der Transport mittels benannter Pipes fügen dem Nachrichtentext wenig Mehraufwand hinzu und unterstützen eine systemeigene Duplexform, die die Wartezeit auf Antworten für Nachrichten reduziert.	TCP, benannte Pipe
Tools	Die Tools umfassen die Unterstützung für Drittanbieteranwendungen für ein Protokoll, mit dem die Entwicklung, die Diagnose, das Hosten und andere Aktivitäten durchgeführt werden können. Entwicklungstools und Softwareanwendungen, die in Verbindung mit dem HTTP-Protokoll verwendet werden können, stellen eine besonders hohe Investition dar.	HTTP

Tabelle 47 WCF Transport

Quelle: <https://msdn.microsoft.com/de-de/library/ms733769%28v=vs.110%29.aspx>

9.3. Textanalyse Software

9.3.1. GOCR

GOCR wurde von Jörg Schulenburg erstmals 2000 publiziert. Es wurde basierend auf der GNU Public Lizenz entwickelt und ist somit Open Source Software. Die Unterstützung lässt jedoch zu wünschen übrig. Support findet man nur schwierig im Internet. 2010 wurde es erstmal für Windows zur Verfügung gestellt. Mehr Informationen ist unter folgender Adresse publiziert:

<http://jocr.sourceforge.net/index.html>

9.3.2. OCR Software Tesseract

Tesseract wurde als HP Research Prototype im Jahre 1984-1994 entwickelt. Es begann als eine Doktorarbeit und wurde bis 2005 von HP weiterentwickelt. Danach wurde es erstmals als Open Source Software 2005 publiziert und wird seit dahin von Ray Smith bei Google weiterentwickelt.

Tesseract ist die erste Software die sowohl Schwarz-auf-Weiss als auch Weiss-auf-Schwarz erkennen kann.

Architektur

Ein Bild wird anhand eines Linienfinder Algorithmus in Linien unterteilt. Danach werden diese Linien einzeln durch einen Wortfinde Algorithmus geschickt, welcher eine 2-Pass-Prozedur ausführt.

2-Pass-Prozedure:

1. Versuchen ein Wort anhand von dem Abstand zu finden. Das Wort wird dann anhand einer mitgelieferten Datenbank mit Wörtern in dieser Sprache verglichen.
2. Falls einzelne Wörter nicht richtig gefunden werden, werden diese nochmals analysiert

In den nachfolgenden Kapiteln wird der Algorithmus etwas genauer erklärt.

9.3.2.1. Linienfinder Algorithmus

Dieser Algorithmus wurde so entworfen, dass auch abgeschrägte Linien gefunden und analysiert werden können. Ohne diese Funktion würde die Qualität der Bilder erheblich verschlechtert werden. Die Hauptteile von dem Prozess sind Blob Filterung und Linien Erstellung.



Abbildung 17 Tesseract Linienfinder

Anschliessend wird ein Baseline gezogen, welche in der nachfolgender Grafik dargestellt wird.



Abbildung 18 Tesseract Baseline

9.3.2.2. Fixed Pitch Detection and Chopping

Pitch Detection ist das Erkennen vom Zeichenabstand, wobei „Chopping“ die Linie in einzelne Charakter unterteilt. Danach wird nach fixen Abständen gesucht. Dadurch kann die Linie in Wörter unterteilt werden. Sobald das abgeschlossen ist, werden die Wörter in einzelne Buchstaben unterteilt.

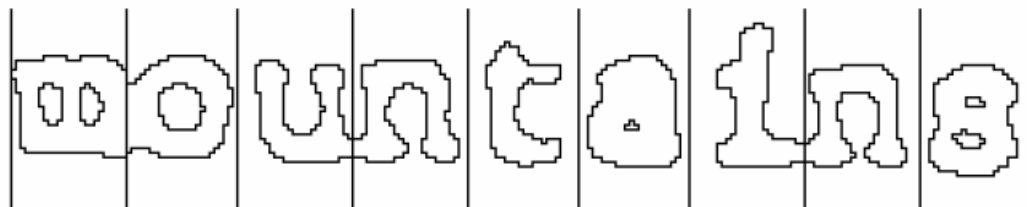


Abbildung 19 Tesseract Charaktererkennung

9.3.2.3. Problem

Dabei treffen aber immer wieder bekannte Problem, wie Kerning „Überlappen“ auf.

9.3.2.4. Kerning Problem

Das Kerning Problem ist die Abstandserkennung von Buchstaben eines Wortes, welche überlappen. In der nachfolgenden zwei Grafiken wird das Problem beschrieben:



Abbildung 20 Tesseract Kerning

Quelle: <https://en.wikipedia.org/wiki/Kerning>

fear of financial collapse,

Abbildung 21 Tesseract Überlappung 1

Zwischen „of“ und „financial“ besteht kein Abstand, welcher zu diesem Zeitpunkt als Fuzzy „unscharf“ bezeichnet und am Schluss nochmals mit einer zweiten Methode analysiert.

2.Methode:

Es wird nicht mehr der Abstand von der einen Box zur anderen Box angeschaut, sondern es wird eine neue Box gemacht und zwar nur von der Baseline zur Medianline und danach dort der Abstand gemessen:

ir of finan

Abbildung 22 Tesseract Überlappung 2

Somit können diese Buchstaben separiert werden.

9.3.2.5. Wort recognision

Die analysierten Wörter werden danach anhand „Liguistic Analyis“ analysiert. Das heisst sie werden gegen Wörter in Wörterbücher, gegen numerischen Wörter, gegen Grossbuchstabenwörtern und gegen Keimbuchstabenwörtern verglichen. Das Wort mit der kleinsten Abweichung gegenüber allen, wird dann als Wort verwendet.