**Architekturdokumentation**

MedDevMM

erstellt von

Lehner Roland, Kienböck Daniel, Stöckl Bernhard, Grill Florian, Murrent Mario

*Template Revision: 6.0 DE (Release Candidate)  
19. März 2012*

|  |  |
| --- | --- |
| We acknowledge that this document uses material from the arc 42 architecture  template, <http://www.arc42.de>. Created by Dr. Peter Hruschka & Dr. Gernot Starke. For additional contributors see arc42.de/about/contributors.html |  |

**Änderungsübersicht**

| **Version** | **Datum** | **Bearbeiter** | **Beschreibung** |
| --- | --- | --- | --- |
| 0.1 | 27.10.2014 | Mario Murrent | Basisversion |
| 0.2 | 02.11.2014 | Roland Lehner | Bearbeitung Punkt 8.2. Broker und Pipes und Filters Muster hinzugefügt |
| 0.3 | 12.11.2014 | Daniel Kienböck | Patternentscheidungen und Beschreibung |
| 0.4 | 16.11.2014 | Bernhard Stöckl | Bearbeitung F/R , Command-Processor |
| 0.5 | 17.11.2014 | Mario Murrent | Qualitätsbaum hinzugefügt |
| 0.6 | 18.11.2014 | Mario Murrent | Risiken, Stakeholder, Qualitätsziele hinzugefügt |
| 0.7 | 20.11.2014 | Mario Murrent | Bewertungsszenarien hinzugefügt |
| 0.8 | 21.11.2014 | Daniel Kienböck | Konzepte + Tippfehler ausbessern |
| 0.9 | 05.12.2014 | Daniel Kienböck | Feedback einarbeiten |
| 0.10 | 22.12.2014 | Mario Murrent | Randbedingungen ausgebaut |
| 0.11 | 31.12.2014 | Daniel Kienböck | Domänenmodel |
| 0.12 | 09.01.2015 | Roland Lehner | Konzepte 7 |
| 0.13 | 09.01.2015 | Mario Murrent | Sequenzdiagramm |
| 0.14 | 11.01.2015 | Stöckl Bernhard | Bausteinsicht (Update) |
| 0.15 | 11.01.2015 | Florian Grill | Schnittstellen |
| 0.16 | 11.01.2015 | Daniel Kienböck | Vereinigung zu einem gemeinsamen Stand und Review |
| 0.17 | 12.01.2015 | Bernhard Stöckl | Schriftformat angepasst |
| 0.18 | 15.01.2015 | Florian Grill | Beschreibung externer Schnittstellen |
| 0.19 | 16.01.2015 | Bernhard Stöckl | Bausteinansicht geändert |
| 0.20 | 26.01.2015 | Roland Lehner | Update Dokument |
| 0.20 | 26.01.2015 | Stöckl Bernhard | Ext. Schnittstelle 3 / Format / Bausteinsicht |

**Basisdokumente**

|  |  |
| --- | --- |
| **Dokument** | **Beschreibung** |
|  |  |
|  |  |

Inhaltsverzeichnis

1. Einführung und Ziele 4

1.1 Aufgabenstellung 4

1.2 Qualitätsziele 4

1.3 Stakeholder 4

2. Randbedingungen 5

2.1 Technische Randbedingungen 5

2.2 Organisatorische Randbedingungen 5

2.3 Konventionen 5

3. Kontextabgrenzung 6

3.1 Fachlicher Kontext 6

3.2 Technischer- oder Verteilungskontext 7

3.3 Externe Schnittstellen 8

4. Bausteinsicht 11

4.1 Ebene 1 11

5. Laufzeitsicht 15

6. Verteilungssicht 16

7. Konzepte 17

7.1 Fachliche Strukturen und Modelle 17

7.2 Typische Muster und Strukturen 18

7.3 Persistenz 22

7.4 Benutzungsoberfläche 22

7.5 Transaktionsbehandlung 22

7.6 Sicherheit 22

7.7 Kommunikation und Integration mit anderen IT-Systemen 22

7.8 Logging, Protokollierung, Tracing 22

7.9 Konfigurierbarkeit 22

7.10 Parallelisierung und Threading 23

7.11 Internationalisierung 23

7.12 Migration 23

7.13 Testbarkeit 23

8. Entwurfsentscheidungen 23

8.1 Entscheidung 1 – Bridged Master/Slave Pattern für Quellsysteme 23

8.2 Entscheidung 2 – Pipes and Filters 24

9. Qualitätsszenarien 24

9.1 Qualitätsbaum 25

9.2 Bewertungsszenarien 25

10. Risiken 26

Anmerkung: In der Microsoft-Word-Variante enthält dieses Template Anleitungen und Ausfüllhinweise als „ausgeblendeten Text“. Durch den Befehl „Formate ein-/ausblenden“ können Sie die Anzeige dieser Texte bestimmen.

# Einführung und Ziele

## Aufgabenstellung

Es soll ein System für den medizinischen Bereich, mit dem Fokus auf Krankenhäuser, entwickelt werden, mit dem medizinische Daten aus unterschiedlichen Quellen bzw. Quellsystemen gesammelt und abgerufen werden können. Für die Speicherung der Daten soll ein zentrales Datenmanagement zuständig sein. Das Abrufen und Anzeigen der Daten soll auf unterschiedlichen Devices - mit dem speziellen Fokus auf mobile Devices wie z.B.: Tablet oder Handy - möglich sein. Das System soll sowohl für Ärzte als auch für Patienten nutzbar sein.

## Qualitätsziele

|  |  |  |
| --- | --- | --- |
| Prio | Qualitätsmerkmal | Ziel |
| 1 | Verfügbarkeit | Die Software soll einen hohe Verfügbarkeit bereitstellen |
| 2 | Bedienbarkeit | Usability soll gegen sein. |
| 3 | Unabhängigkeit | Das System soll unabhängig von der darunterliegenden Plattform funktionieren |
| 4 | Erweiterbarkeit | Die Erweiterungen des bestehenden System sollen so einfach wie möglich umgesetzt werden können |

## Stakeholder

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rolle | Beschreibung | Ziel/Intention | Kontakt | Bemerkung |
| Projektmanager |  |  |  |  |
| Auftraggeber |  |  |  |  |
| Software Architekt |  |  |  |  |
| Anwender |  |  |  |  |
| Patienten |  |  |  |  |
| Krankenhausangestellte |  |  |  |  |
| IT Service |  |  |  |  |

# Randbedingungen

Beim Lösungsentwurf sind verschiedene Randbedingungen zu beachten. Dieser Abschnitt stellt sie dar und erklärt auch, wenn nötig, deren Motivation.

## Technische Randbedingungen

|  |  |
| --- | --- |
| Randbedingung | Erläuterung |
| Moderate Hardwareausstattung | Die Lösung muss auf marktüblichen Geräten, wie Tablets, Notebooks, PCs betrieben werden können. |
| Implementierung der Clients in HTML5 / JavaScript und native Implementierung | Die Implementierung soll auf HTML5 / JavaScript basieren um eine Plattformunabhängigkeit zu gewährleisten. Plattform spezifische Lösungen werden in nativen Implementierungen bereitgestellt. |
| Implementierung des Servers | Die Implementierung wird in Java umgesetzt. |

## Organisatorische Randbedingungen

|  |  |
| --- | --- |
| Randbedingung | Erläuterung |
| Team |  |
| Zeitplan |  |
| Entwicklungswerkzeuge | Der Entwurf erfolgt mit Stift und Papier, ergänzend Visio / Pencil oder Enterprise Architect. Die gewonnenen Erkenntnisse werden in diesem Arc42 Template persistiert. |
| Konfigurations- und Versionsverwaltung | Subversion |
| Testwerkzeuge |  |
| Vorgehensmodell |  |
| Projektbudget | Für die Entwicklung wurde ein Festpreis vereinbart |

## Konventionen

|  |  |
| --- | --- |
| Konvention | Erläuterung |
| Architekturdokumentation | Gliederung nach dem deutschen arc42-Template |
| Coderichtlinien für JavaScript | siehe CodingGuidelines JS 1.5.docx |
| Modelle | UML wird zur Erstellung von Diagrammen und zur einfachen Beschreibung verwendet |
| Sprache | Innerhalb dieser (deutschen) arc42-Architekturdokumentation werden Dinge (Komponenten, Schnittstellen) in Diagrammen und Texten deutsch benannt |

# Kontextabgrenzung

## Fachlicher Kontext

Namensgebung aus Sicht des Endbenutzers

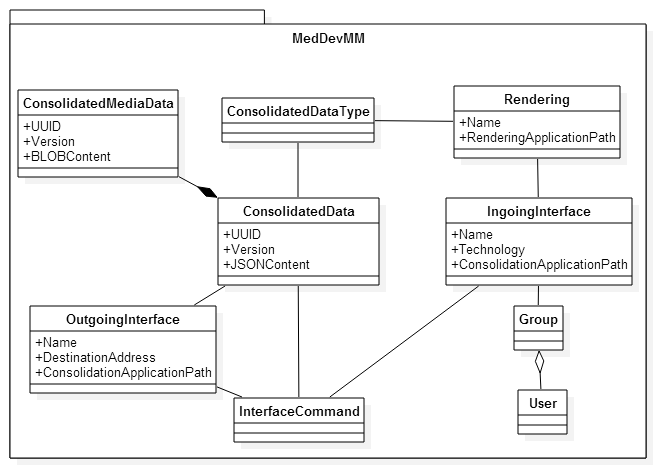


Abbildung 1 - Domänenmodell

* IngoingInterface repräsentiert die Daten eines Endgerät-Interfaces
* Daten zu Authentication-System beim IngoingInterface (Group/User)
* IngoingInterface erzeugt ein nachvollziehbares InterfaceCommand zur Abarbeitung
* IngoingInterface repräsentiert eine Ausgabe (Rendering) einer Datenart
* InterfaceCommand wird von IngoingInterface generiert und am OutgoingInterface im Zusammenspiel mit den zupassenden Daten in einer gewissen Version ausgeführt
* Das OutgoingInterface repräsentiert ein internes System
* Daten kommen vom IngoingInterface und werden dort in das konsolidierte Format übertragen
* Zusätzlich zu den konsolidierten Daten können noch weitere Infos gespeichert werden

Abkürzungen: UUID = Unique Universal Identifier

BLOB = Binary Large Object

## Technischer- oder Verteilungskontext

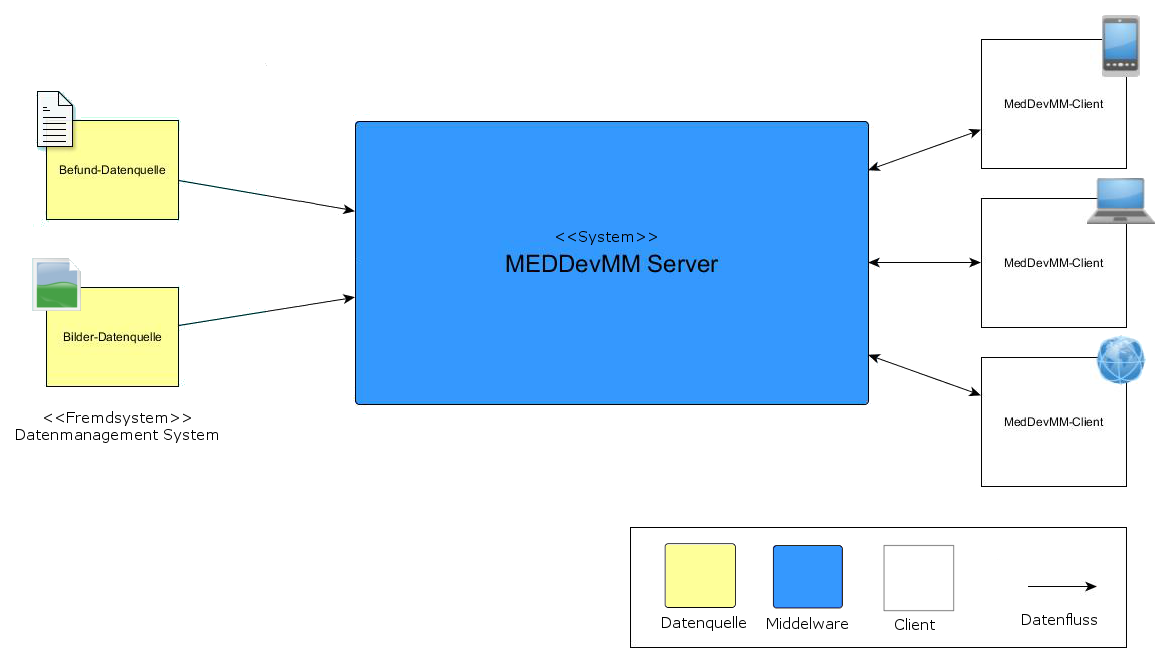


Abbildung 2: Technische Interaktion zwischen MedDevMM und den Komponenten

**MEDDevMM-Client**

Die Anbindung an das MEDDevMM System ist über verschiedene Client Systeme möglich. Unter anderem werden Clients für Microsoft PCs, diverse Smartphones und Tablets zur Verfügung gestellt. Weiters wird es auch möglich sein über eine Weboberfläche auf das System zuzugreifen.

**MEDDevMM-Server**

Der MEDDevMM-Server fungiert als zentrale Middleware. Er nimmt Anfragen von den verschiedenen Clients entgegen und bereitet die Daten für die spezifischen Client vor. Um die Anfragen der Clients beantworten zu können ist das System mit Datenmanagement-Systemen verbunden welches die Informationen zu den Anfragen liefert.

**Datenmanagement System (Fremdsystem)**

Ein Datenmanagement System ist die Informationsquelle des MEDDevMM-Servers in Bezug auf die Anfragen der Clients. Es stellt Schnittstellen nach außen bereit um eine problemlose Anbindung zu ermöglichen.

Inhalt

Festlegung der Kanäle zwischen Ihrem System, den Nachbarsystemen und der Umwelt;

Zusätzlich eine Mappingtabelle, welcher logische Input (aus 3.1) über welchen Kanal ein- oder ausgegeben wird.

Motivation

Verstehen, über welche Medien Informationen mit Nachbarsystemen bzw. der Umwelt ausgetauscht werden.

Form

z.B.: UML Deploymentdiagramm mit den Kanälen zu Nachbarsystemen, begleitet von einer Mappingtabelle Kanal x Input/Output.

## Externe Schnittstellen

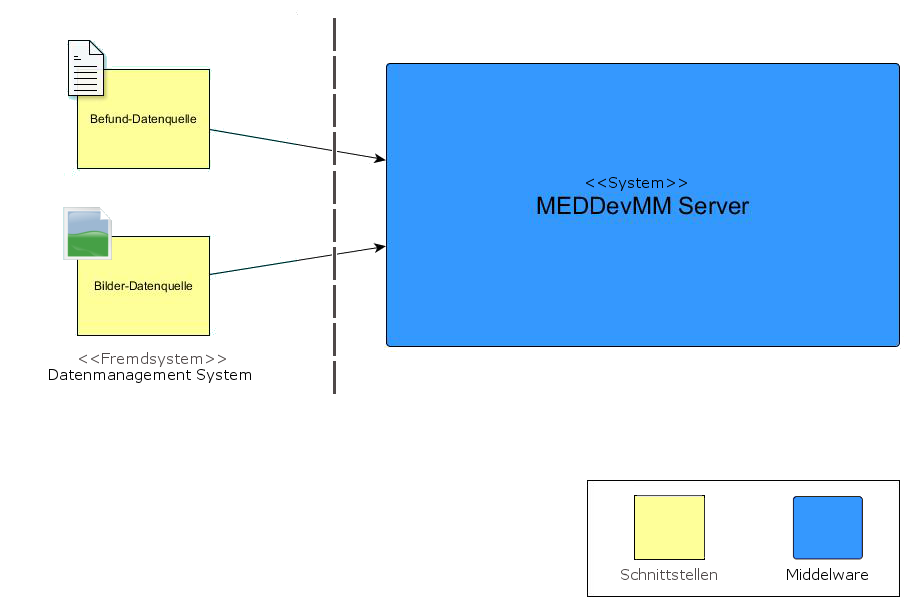


Abbildung 3: Schnittstellen des Datenmanagement System

Die Datenmanagement Systeme bieten primär folgende zwei Schnittstellen an.

1. Schnittstelle zu den Befunden

* Liefert Befunde in Textformat (Textfiles)

1. Schnittstelle zu medizinischen Bilddaten

* Liefert digitale Bilder im DICOM Format (Röntgenbilder, Magnetresonanzbilder…)

Diese Schnittstellen werden in diesem System nur von der Middleware, nämlich dem MEDDevMM Server, aufgerufen.

Contents

Specification of the communication channels linking your system to neighboring systems and the environment.

### Externe Schnittstelle 1

#### Identifikation der Schnittstelle

|  |  |
| --- | --- |
| Name / Bezeichnung der Schnittstelle | <Befund-Datenquelle> |
| Version | 1.0 |
| Änderungen gegenüber Vorversion | - |
| Wer hat geändert und warum? | - |
| Verantwortlicher Ansprechpartner / Rolle | - |

#### Fachlicher Kontext der Schnittstelle

Die Schnittstelle verbindet die Befund-Datenquelle mit MedDevMM Server.

#### Fachliche Abläufe

Der MEDDEV Server bekommt von einem Client eine Anfrage über einen Befund. Ist der Client berechtigt wird die Anfrage über die Befund-Datenquelle abgewickelt. Für die Anbindung nutzt der Server eine definierte Schnittstelle. Liefert die Befund-Datenquelle ein Ergebnis wird dieses für die Darstellung auf dem spezifischen Client vorbereitet und zurückgesendet.

#### Fachliche Bedeutung der Daten

Das Abrufen von Patientendaten stellt ein wesentliches Kernelement vom MEDDevMM System dar, deshalb ist diese Schnittstelle von besonderer Bedeutung in Bezug auf die Funktionalität des Systems.

Um die Daten korrekt anzuzeigen werden diese von der MedDevMM Server Middleware entsprechend aufbereitet.

#### Anforderungen an die Schnittstelle

##### Sicherheitsanforderungen

Um die Daten dem Client zur Verfügung stellen zu können muss das System prüfen ob der entsprechende Client berechtigt ist die Daten einzusehen. Dafür benötigt das System entsprechende Login Informationen des Clients.

Ohne entsprechende Authentifizierung dürfen keine Daten übermittelt werden.

##### Mengengerüste

**Laufzeit**: Soll gering gehalten werden, das bedeutet die Abfrage der Daten darf nur wenig Zeit in Anspruch nehmen um ein Benutzerfreundliches Verhalten zu gewährleisten.

* Mehr als 150ms darf eine Anfrage nicht überschreiten

**Verfügbarkeit:** Da die Schnittstelle die Kernfunktionalität des Systems bereitstellt soll sie eine hohe Verfügbarkeit aufweisen.

* Eine Verfügbarkeit von 99,7% soll gewährleistet sein

#### Syntax: Methoden/Funktionen

GetDiagnoses(long patientId) 🡪 Lieftet alle Befunde des Patienten

GetDiagnosis(long patientId, long diagnosisId) 🡪 Liefert einen konkreten Befund

#### **Ablauf der Schnittstelle**

##### Technische Infrastruktur

Es wird eine SQL-Verbindung über eine TCP/IP Verbindung mit definiertem Port und IP Adresse verwendet.

##### Fehler- und Ausnahmebehandlung

Falls ein Angeforderter Befund nicht zur Verfügung steht wird dem Client eine entsprechende Fehlermeldung zugesandt.

Ist die Datenquelle nicht erreichbar wird der Client über diesen Zustand benachrichtigt. Weiters sendet die Middleware in diesem Fall automatisch eine E-Mail und ein SMS an den Systemverantwortlichen.

##### Einschränkungen und Voraussetzungen

 Berechtigungen: die Daten dürfen nur an Berichtigte Personen weitergegeben werden.

##### Betrieb der Schnittstelle

Das Hosting der Schnittstelle wird vom Betreiber der Befund-Datenquelle übernommen und abgewickelt.

### Externe Schnittstelle 2

#### Identifikation der Schnittstelle

|  |  |
| --- | --- |
| Name / Bezeichnung der Schnittstelle | <Bilder-Datenquelle> |
| Version | 1.0 |
| Änderungen gegenüber Vorversion | - |
| Wer hat geändert und warum? | - |
| Verantwortlicher Ansprechpartner / Rolle | - |

#### Fachlicher Kontext der Schnittstelle

#### Die Schnittstelle verbindet die Bilder-Datenquelle mit MedDevMM Server.

#### Fachliche Abläufe

Der MEDDEV Server bekommt von einem Client eine Anfrage über einen Bild (Röntgen-Bild, MRT etc). Ist der Client berechtigt wird die Anfrage über die Bilder-Datenquelle abgewickelt. Für die Anbindung nutzt der Server eine definierte Schnittstelle. Liefert die Bilder-Datenquelle ein Ergebnis wird dieses für die Darstellung auf dem spezifischen Client vorbereitet und zurückgesendet.

#### Fachliche Bedeutung der Daten

Das Abrufen von Bilddaten(Röntgen-Bild, MRT etc.) stellt ein wesentliches Kernelement vom MEDDevMM System dar, deshalb ist diese Schnittstelle von besonderer Bedeutung in Bezug auf die Funktionalität des Systems.

Um die Bilder korrekt anzuzeigen werden diese von der MedDevMM Server Middleware entsprechend aufbereitet.

#### Anforderungen an die Schnittstelle

##### Sicherheitsanforderungen

Um die Daten dem Client zur Verfügung stellen zu können muss das System prüfen ob der entsprechende Client berechtigt ist die Daten einzusehen. Dafür benötigt das System entsprechende Login Informationen des Clients.

Ohne entsprechende Authentifizierung dürfen keine Bilder übermittelt werden.

##### Mengengerüste

**Laufzeit:** Soll gering gehalten werden, das bedeutet die Abfrage der Daten darf nur wenig Zeit in Anspruch nehmen um ein Benutzerfreundliches Verhalten zu gewährleisten.

* Mehr als 500ms darf eine Anfrage nicht überschreiten

Verfügbarkeit: Da die Schnittstelle die Kernfunktionalität des Systems bereitstellt soll sie eine hohe Verfügbarkeit aufweisen.

* Eine Verfügbarkeit von 99,7% soll gewährleistet sein

#### Syntax: Methoden/Funktionen

GetImage(long patientId, long diagnosisId) 🡪 Lieftet alle Befunde des Patienten

#### **Ablauf der Schnittstelle**

##### Technische Infrastruktur

Es wird eine SQL-Verbindung über eine TCP/IP Verbindung mit definiertem Port und IP Adresse verwendet.

##### Fehler- und Ausnahmebehandlung

Falls ein Angefordertes Bild nicht zur Verfügung steht wird dem Client eine entsprechende Fehlermeldung zugesandt.

Ist die Datenquelle nicht erreichbar wird der Client über diesen Zustand benachrichtigt. Weiters sendet die Middleware in diesem Fall automatisch eine E-Mail und ein SMS an den Systemverantwortlichen.

##### Einschränkungen und Voraussetzungen

 Berechtigungen: die Daten dürfen nur an Berichtigte Personen weitergegeben werden.

##### Betrieb der Schnittstelle

Das Hosting der Schnittstelle wird vom Betreiber der Befund-Datenquelle übernommen und abgewickelt.

### Externe Schnittstelle 3

#### Identifikation der Schnittstelle

|  |  |
| --- | --- |
| Name / Bezeichnung der Schnittstelle | <REST Serviceinterface> |
| Version | 1.0 |
| Änderungen gegenüber Vorversion | - |
| Wer hat geändert und warum? | - |
| Verantwortlicher Ansprechpartner / Rolle | - |

#### Fachlicher Kontext der Schnittstelle

Diese Schnittstelle bietet die Serviceanbindung der Client-Devices an. Anfragen und Response werden über die REST-Schnittstelle abgewickelt.

#### Fachliche Abläufe

Die Clients senden eine Anfrage im JSON-Format an den REST-Service an welche Daten dieser benötigt. Über den LDAP-Authentifizierung-Server werden die Berechtigungen für die Anfrage überprüft. Mittels JSON-Respons werden die angeforderten Daten zurück an den Client übermittelt.

#### Fachliche Bedeutung der Daten

Die Daten werden wie schon beschrieben im JSON-Format übertragen. Die Wahl wurde getroffen da JSON ein leicht verständliches und gut strukturiertes Format aufweist und für die Übertragungen mittels REST am besten geeignet ist.

#### Anforderungen an die Schnittstelle

Jeder Service muss über einen RESTful konformen Uniform Resource Identifier (URI) angesprochen werden.

z.B.: <http://172.70.34.10:8080/RIS/Findings/Patient/2034239/>

für medizinische Befunde oder:

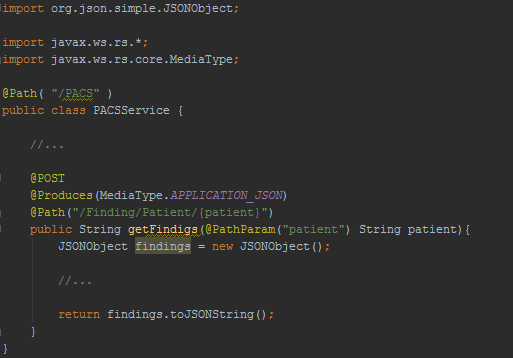
z.B.: <http://172.70.34.10:8080/PACS/PIC/Patient/2034239/>

für medizinische Bilder des Patienten „2034239“.

Für jedes Quellsystem muss ein eigener Service implementiert werden. (RIS,PACS,…)

#### Syntax: Methoden/Funktionen

Beispielklasse des Service für Befunde des PACS-Systems:



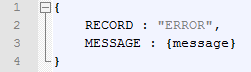
#### **Ablauf der Schnittstelle**

##### Technische Infrastruktur

Verwendet wird die Java API für RESTful Services (JAX-RS) die in eine Webcontainer von Jersey in einer Tomcat-Webserver-Instance gehostet wird. Jene Komponente ist im Bereich DeviceCommunication enthalten (siehe 4.2.1) und als RESTServiceProvider im RequestResponsHandler dargestellt.

##### Fehler- und Ausnahmebehandlung

Fehlermeldungen werden ebenfalls in einer JSON-Struktur an die Clients zurückgesendet, die wie Folgt aussehen:



# Bausteinsicht

MedDevMM besteht aus unterschiedlichen Modulen die wiederum in weitere kleiner Module untergliedert werden. Die erste Ebene zeigt eine Übersicht über das Gesamtsystem wobei diese als Subsysteme bezeichnet werden und mit ihren Schnittstellen dargestellt werden. Das Zusammenspiel zur Laufzeit ist in Abschnitt „Laufzeitsicht“ näher erklärt.

## Ebene 1

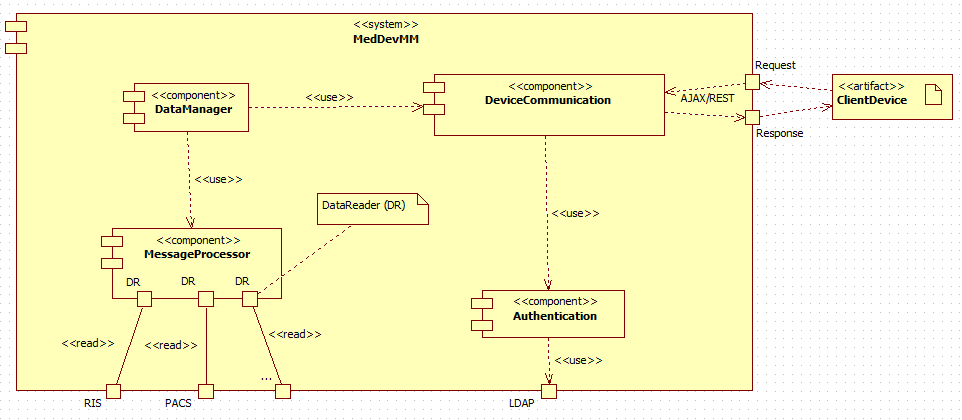
An dieser Stelle beschreiben Sie die White-Box-Sicht der Ebene 1 gemäß dem Whitebox-Template. Die Struktur ist im folgenden bereits vorgegeben.

Das Überblicksbild zeigt das Innenleben des Gesamtsystems in Form der Bausteine 1 - n, sowie deren Zusammenhänge und Abhängigkeiten.

Sinnvoll sind hier auch Beschreibungen der wichtige Begründungen, die zu dieser Struktur führen, insbesondere die Beschreibung der Abhängigkeiten (Beziehungen) zwischen den Bausteinen dieser Ebene.

Evtl. verweisen Sie auch auf verworfene Alternativen (mit der Begründung, warum es verworfen wurde

Die folgende Abbildung zeigt die Hauptbausteine unseres Systems und deren Abhängigkeiten. Das System MedDevMM wird in vier weiter Komponenten unterteilt. An den Systemrändern werden die Interaktionspunkte zu anderen Systeme dargestellt.



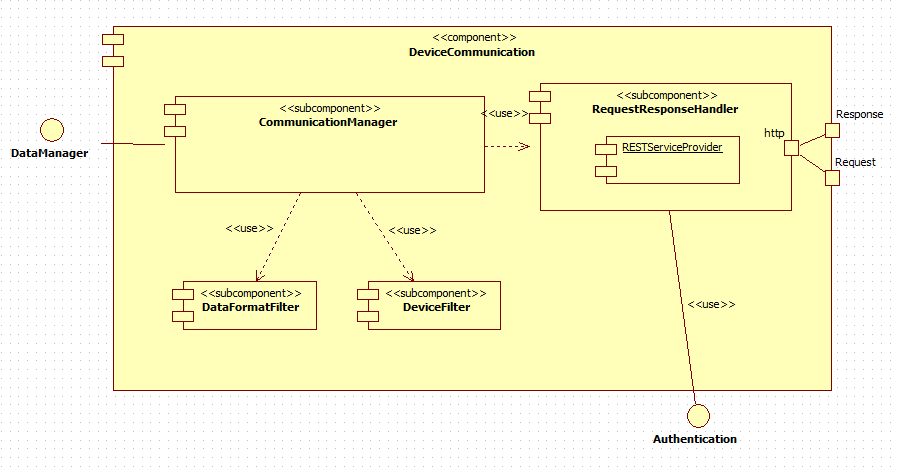
|  |  |
| --- | --- |
| Paket | Kurzbeschreibung |
| DeviceCommunication | Nimmt die Daten der Devices entgegen.  Wendet Filter auf Daten an (Deviceformate, Imageformate)  Speichert die Anfragedaten der Devices im DataManager. |
| Authentication | Verwalten von Benutzerdaten und ihren Rechten und authentifiziert diese auf Bedarf. |
| DataManager | Zuordnung der Bilddaten an die Devices; Validierung; Freigabe |
| MessageProcessor | Hat Zugriff auf die Quellsysteme (PACS, RIS,…).  Bereitet Bilddaten auf und stellt die Daten für den DatenManager zur Verfügung. |

Es folgen Erläuterungen zu Strukturen und Abhängigkeiten der Ebene 1.

## Ebene 2

### DeviceCommunication

1. Struktur gemäß Black-Box- Template:
2. Zweck / Verantwortlichkeit:
3. Schnittstelle(n):
4. Erfüllte Anforderungen:
5. Variabilität:
6. Leistungsmerkmale:
7. Ablageort / Datei:
8. Sonstige Verwaltungsinformation:
9. Offene Punkte:



Struktur gemäß Black-Box- Template:

1. Zweck / Verantwortlichkeit: Nimmt über RequestResponseHandler Daten von den unterschiedlichen Devices entgegen und gibt die gefilterte Daten an den DataManager weiter.
2. Schnittstelle(n): Eingang: RESTService in RequestResponseHandler (JSON-Data), Anbindung an Authenticator, DataManager
3. Erfüllte Anforderungen: Empfang von Anfragen über medizinische Daten. Filtern auf unterschiedliche Formate (Datentyp, Datenformat, Datengrösse). Filtern auf unterschiedliche Devices. Leicht erweiterbar.
4. Variabilität: Filtermöglichkeiten konfigurierbar.
5. Leistungsmerkmale: Kommunikation mit Devices. Anfragebearbeitung der Devices
6. Ablageort / Datei: Namespace Communication in der Source-Codeverwaltung in MedDev.Core.Communication

### Authentication

Struktur gemäß Black-Box- Template:

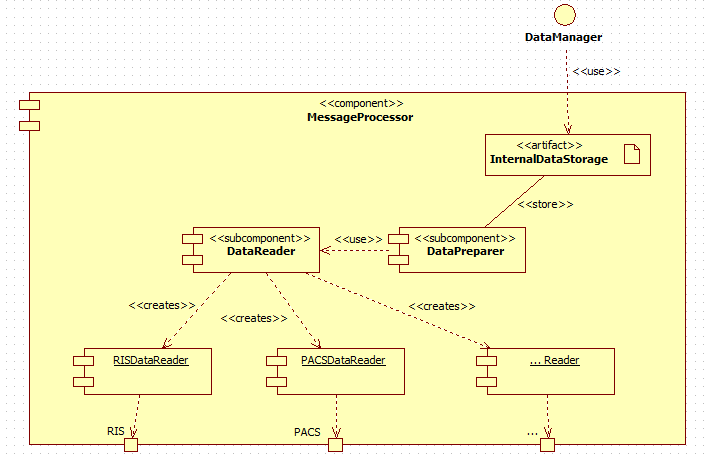
1. Zweck / Verantwortlichkeit: User Authentifizierung
2. Schnittstelle(n): Erhält Daten von DeviceCommunication (Input). Erhält Authentifizierungstoken von LDAP und liefert diesen an den DeviceCommunication (Output) zurück.
3. Erfüllte Anforderungen: Feststellung der Userberrechtigung und Authentifizierung, um Zugriff auf die Quellsysteme zu haben.
4. Variabilität: IP-Adresse des LDAP Servers
5. Leistungsmerkmale: Wrappt Zugriff auf LDAP in einem Objekt-Orientierten Ansatz.
6. Ablageort / Datei: Namespace Authentication in der Source-Codeverwaltung in MedDev.Core.Authentication

### DataManager

Struktur gemäß Black-Box- Template:

1. Zweck / Verantwortlichkeit: Weitergabe von Filterdaten an DataProcessing. Daten an die Device Communication.
2. Schnittstelle(n): DeviceCommunication, MessageProcessor
3. Erfüllte Anforderungen: Freigabeprozess, Vermittler zwischen DataProcessing und Communication
4. Variabilität: keine
5. Ablageort / Datei: Namespace Workflow in der Source-Codeverwaltung in MedDev.Core.Management

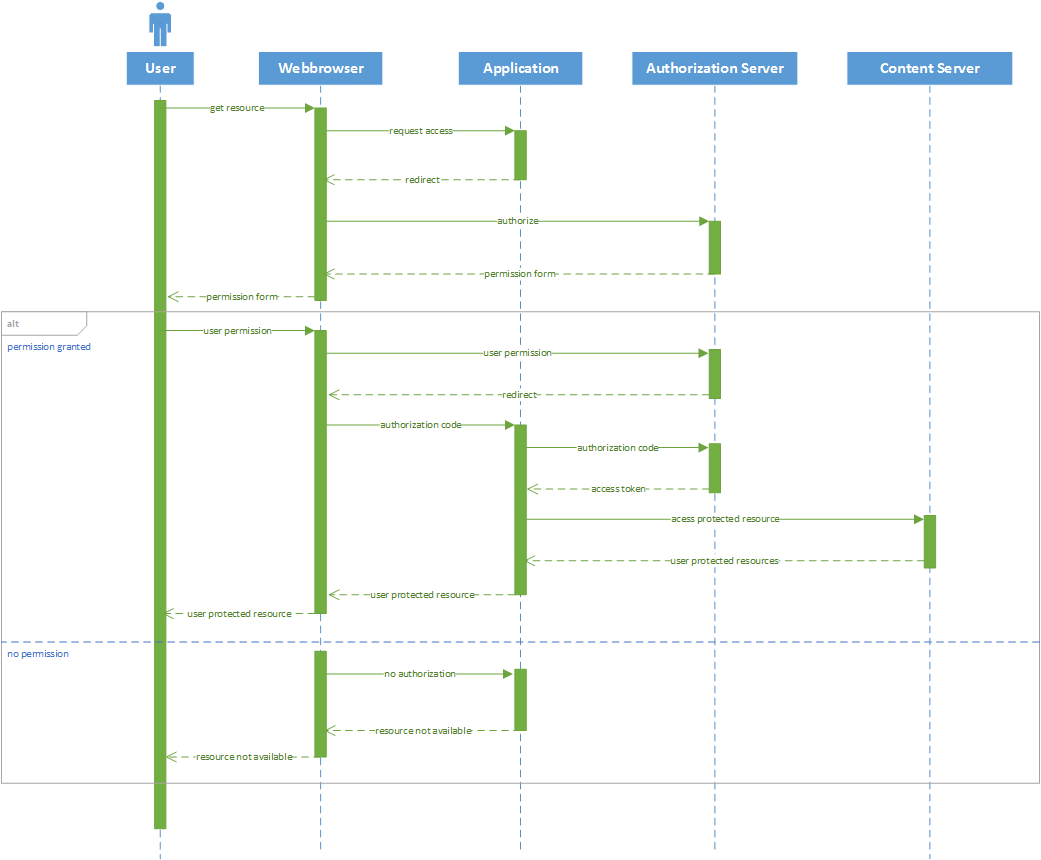
### MessageProcessor



Struktur gemäß Black-Box- Template:

1. Zweck / Verantwortlichkeit: Nimmt über Außenschnittstellen an den Systemen RIS und PACS medizinische Daten entgegen.
2. Schnittstelle(n): Eingang: Befunddaten (Textfile), Bilddaten (DICOM Files). Ausgang: DataPreparer bereitet die Daten auf ein einheitliches Format auf (JSON). Speicherung in internen Liste. Schnittstellen selbst sind mit dem Bridge-Pattern implementiert.
3. Erfüllte Anforderungen: Datenbeschaffung von Quellsystemen. Aufbereitung der medizin. Daten.
4. Variabilität: Erweiterbar auf weitere Quellsysteme.
5. Leistungsmerkmale: Bereitstellen von medizinischen Daten (Befunde, Bilder, etc …).
6. Ablageort / Datei: Namespace FragebogenManger in der Source-Codeverwaltung in MedDev.Core.Processing

# Laufzeitsicht



# Verteilungssicht

Da die Datenquellen als stabile Dienste im Netzwerk zu sehen sind werden diese hier als ein einzelner Server dargestellt (auf die interne Verteilung, wie bspw. der Clusterung wird nicht eingegangen, da für MedDevMM transparent).





MedDevMM



# Konzepte

Dieser Abschnitt beschreibt allgemeine Strukturen und Aspekte, die systemweit gelten. Darüber hinaus stellt er verschiedene technische Lösungskonzepte vor.

## Fachliche Strukturen und Modelle

**Middleware:**

* **Datenquellen-Schnittstellen (gelb)**

Datenquelle Schnittstelle

Kern

Endanwender Schnittstelle

* **Anfragen von Endbenutzern (grün)**
* **Kern (blau)**

Middleware

Client 1

Client 2

Client 3

Client 4

Client 5

Client 6

…

Client X

Datenquelle1

Datenquelle2

Datenquelle3

Datenquelle4

Datenquelle5

Datenquelle6

…

DatenquelleX

Cmd-Pattern

Broker

Master/Slave  
(bridged)

Pipes / Filters

Anfrage immer gleich:

* Pipes and Filters ändert die Anfrage sodass die Anfrage für das Endgerät optimiert werden kann (Mobile / Tablet / Browser)

## Typische Muster und Strukturen

Funktionale Anforderungen beschreiben die Möglichkeiten und nicht-funktonale Anforderungen die Qualität der Software. Infolge werden die Anforderungen mit den zugehörigen Mustern genauer spezifiziert.

### Nicht funktionale Muster

### Broker

Der Broker ist dafür verantwortlich die Anfrage an die bestehende Schnittstelle weiterzuleiten (mapping von Request auf Data-Source). Der Broker nimmt die bereits angepasste Anfrage (siehe pipes and filters) entgegen und fragt die Datenquellen an. Sollten neue Datenquellen hinzugefügt werden, muss der Broker um die Datenquellen erweitert werden.

### Master-Slave

Master

+ result

+launch()  
*+makeSlave() : Slave*

*Slave*

+ result

*+ run()*

ConcreteSlave

+ run()

ConcreteMaster

+ makeSlave() : Slave

Für die Anfrage der Daten aus verschiedenen Schnittstellen wir prinzipiell im Schnittstellen – Modul ein Objekt als Schnittstellen-Master (Singleton) instanziiert und in Folge vom Master pro Anfrage ein (paralleler) Slave erstellt (siehe Bridge-Pattern zu Implementierungsdetails des Slave). Das Pattern dient der Erhöhung der Verfügbarkeit.

### Command-Prozessor

Clients können die Änderungen (Command) an Daten in den Ressourcen wieder rückgängig machen. Sobald ein Client seine Änderungen durchführt wird das Command in einer Liste gespeichert und andere Clients können über die Undo-Funktion die Änderungen bzw. das Command rückgängig machen.



### Funktionale Muster

### Pipes and Filters

Das Pipes and Filters Architekturmuster, kommt bei der Anfrage der Clients zum Einsatz. Dieser Prozess wird in der Middleware durchgeführt. Es werden somit dem Endgerät die optimierten Daten zur Verfügung gestellt, die es benötigt. Zum Beispiel soll ein mobiles Endgerät keine großen Bilderdaten erhalten, um den Datenkonsum klein zu halten.

Bsp.: = Pipe

Endgerättypenbestimmung

Auth-Token-Validierung

Bildfilter

Pipes and Filters dient der leichten Erweiterbarkeit.

### Bridge

Abstraction

- impl : Implementor

+ function()

*Implementor*

+ implementation()

ConcreteImplementor

+ implementation()

RefinedAbstraction

+ function()

Da viele verschiedene Schnittstellen (und dessen libraries) unter einen Hut gebracht werden müssen, wurde das Bridge-Pattern gewählt, um hier ein gemeinsames (triviales) Interface für jede Schnittstelle verwenden zu können. Jede neue Schnittstelle zu einem Fremdsystem kann somit über die „Abstraction“ (siehe Bild; interface zur Verwendung) aufgerufen werden. Die RefinedAbstraction (siehe Bild; Code, der den Zugriff der Abstraktion auf die konkrete Library mapped) muss beim Integrationsprozess ausdefiniert werden.

**Anwendungsfall Schnittstellen**

* Medizinische Befunde mit Bildern werden beim Holen der Daten konsolidiert und werden in einem standardisierten JSON-Format ausgegeben bzw. Bilder im Portable Network Graphics (png)-Format mit einem eindeutig vergebenen Namen hinzugefügt (UUID) mit Referenz im JSON-File.
* Mit demselben JSON/PNG Format werden DICOM Files gelesen und genauso konsolidiert gespeichert.

Über das Pattern ist es möglich dem implementierenden Team nicht ein Interface sondern nur ein Requirement (also abstrakte Anforderung bspw. von einem reinen, nicht technischen Projekt-Manager) vorzugeben (bzw. Software zuzukaufen, wobei man dem Hersteller meist nur schwer eine Schnittstelle vorgeben kann). Das Requirement wird dann implementiert, getestet und über die RefinedAbstraction in das Projekt von einem - unter Umständen anderen - Team integriert.

## Benutzungsoberfläche

Grundsätzlich wird die Benutzeroberfläche in einem Webinterface dargestellt. Wird das System von einem mobilen Endgerät angesteuert, kann eine mobile Website, oder eine native Implementierung der Benutzeroberfläche zu Grunde liegen. Die mannigfaltigen Benutzeroberflächen stärken das Konzept der Bedienbarkeit; das Setzen auf Web-Technologien das Konzept der Unabhängigkeit.

## Transaktionsbehandlung

In Geschäftstransaktionen wird als Transaktionsmodell VCRP (Visibility; Consistency; Recovery; Permanence) statt dem normalen ACID Modell aus den Standard OLTP – DB-Systemen gewählt. Recovery wird hierbei über das Command Pattern gewährleistet.

## Sicherheit

Eine Anfrage auf einen Datensatz wird mit Hilfe eines Zugriffstokens gewährleistet. Zu Grunde liegt das Pipes and Filters Pattern, wo Sicherheitsfilter implementiert werden, die diesen Token überprüfen.

## Kommunikation und Integration mit anderen IT-Systemen

Wird ein neues System angeschlossen, muss dieses ein Interface implementieren, über welches die Systeme miteinander kommunizieren können. Hierfür wird ein Bridge Pattern verwendet, welches alle Verbindungen unter einen Hut bringen kann, um so eine einfache Erweiterbarkeit zu gewährleisten.

## Logging, Protokollierung, Tracing

Kommt es zu Problemen im System wird dies in einem Logfile protokolliert. In diesem Logfile kann der Fehler mit der jeweiligen Komponente in Verbindung gebracht werden. Google Analytics kommt zum Einsatz, um genau nachvollziehen zu können bei welchem User Interface der Fehler aufgetreten ist. Falls es Datenschutzbedenken geben sollte, kann eine eigene Lösung für Traceability auf einem eigenen Server gehostet werden. Hierfür müssten Produkte wie zum Beispiel Webtrekk zugekauft werden.

## Konfigurierbarkeit

Der Broker ist dafür verantwortlich, dass die Datenquellen die richtigen Anfragen bekommen. Hier kann für jede Datenquelle eine Konfiguration durchgeführt werden.

## Parallelisierung und Threading

Parallelisierung und Threading wird im Master/Slave Pattern behandelt. Werden zwei geleichzeitige Anfragen auf ein Service gesendet, so werden auch zwei Slaves erstellt, die diese Anfrage abarbeiten.

## Internationalisierung

Bei der Internationalisierung wird auf die Spracheinstellung des jeweiligen Endgerätes zurückgegriffen. Die Sprachfiles befinden sich bei den Webanwendungen (Mobil/Desktop) direkt auf dem Server. Bei nativen Anwendungen für mobile Endgeräte muss für jede Plattform ein eigenes Sprachfile erstellt werden.

## Migration

Eine Migration ist nicht notwendig, da die Middleware zu existierenden Services geschaffen wird. Die Datenquellen bleiben unverändert.

## Testbarkeit

* Unit-Test
* Integrationstest
* Äquivalenzklassentest pro Dokumententyp

# Entwurfsentscheidungen

In diesem Kapitel wird vor allem auf die Entwurfsentscheidungen im Bereich der gewählten Muster (Patterns)

* Command -Pattern
* Master-Slave -Pattern
* Broker -Pattern
* Pipes / Filters -Pattern
* Bridge -Pattern

unter Berücksichtigung der Qualitätsziele:

* Verfügbarkeit
* Bedienbarkeit
* Unabhängigkeit
* Erweiterbarkeit

eingegangen.

## Entscheidung 1 – Bridged Master/Slave Pattern für Quellsysteme

Das Schnittstellenmodul zu den Quellsystemen ist zentraler Baustein für diese Middleware-Applikation. Hierbei sollte beachtet werden, dass so ein zentrales Modul so performant wie möglich gestaltet werden soll.

Das Pattern Master/Slave dient der Verfügbarkeit / Hoch-Verfügbarkeit. Anstatt sequenzielle Abfragen abzuarbeiten werden die Abfragen pro Quelle parallel pro Datenquelle (=1 Slave pro Datenquelle) ausgeführt (bridged).

Annahme dabei ist, dass die Anfragen gut verteilt auf die Datenquellen kommen, um wirklich einen Performance-Gewinn durch das Pattern zu erreichen.

Alternativ wäre es möglich einen Load-Balancer zu verwenden, jedoch kann bei dedizierten Verbindungen jede Verbindung zum Quellsystem effektiver genutzt werden (keine Zeitverluste durch connection-establishing).

## Entscheidung 2 – Pipes and Filters

Bedienbarkeit, Unabhängigkeit und Erweiterbarkeit sind Argumente, die alle in eine Richtung schlagen: keiner weiß was morgen ist. Es geht um die Möglichkeit schnell und flexibel auf verschiedenste, sinnvolle Endgeräte eine akkurate und gut bedienbare Lösung zu schaffen. So gilt aktuell noch Android Wear (bspw. Uhren mit Android-Betriebssystem, die mit dem Handy gekoppelt Nachrichten anzeigen können und über Audio-Interface einfache Dienste wie Mail und SMS schreiben übernehmen können) als absolutes Entwickler-Hacker-Spielzeug. Es ist jedoch durchaus möglich, dass sich diese Technologie in den nächsten 3-5 Jahren im Endkundensegment etabliert und durchgesetzt haben wird. Es ist in der Lösung somit angestrebt, dass auch für solche „esoterischen“ Geräte als auch noch nicht berücksichtigbare Softwarelösungen berücksichtigt werden und die Lösung leicht erweiterbar ist, um diesen kommenden Herausforderungen stand zu halten.

Annahme hierbei ist, dass das Internet als solches mit dem Konzept von Webseiten und einem Browser sich zwar weiter-entwickeln wird, jedoch nicht komplett als Technologie wie wir sie jetzt kennen abgelöst wird. Anzubindende Endgeräte haben entweder eine Kommunikationsschnittstelle, die Daten in einem definierten Format zu erhalten haben (Bspw. REST-Service mit XML-Daten für eine i-Phone App) oder einen Browser installiert haben, auf den eine Webseite optimiert auf das Endgerät (Bspw. Webseite am Surface-Tablet sieht anders aus als auf dem Linux-Desktop-System) angezeigt wird.

Pipes and Filters bieten hierbei die gewünschten Möglichkeiten. Beispielsweise können verschiedene Informationen je nach Authentifizierung durch einen Filter entfernt werden oder aus verschiedenen Renderings (Ausgabeformaten) optimiert für das Endgerät gewählt werden. Bilder können für kleine Geräte heruntergerechnet und im Filterelement (als Mini-Baustein) ge-cache-t werden, um auch hier Performance-Gewinne zu erzielen. Komplett neue Anforderungen können durch das Hinzufügen eines neuen Filters bewältigt werden.

Alternativ könnte man verschiedene Endpunkte angeben, die verschieden abgearbeitet werden und die jede Funktionalität (bspw. Authentifizierungsabhängige Sichten) ausprogrammieren, jedoch ist hierbei eine - auf den ersten Blick - komplex wirkende Lösung zu entwickeln, wie man gemeinsame Teile wiederverwenden kann. Außerdem ist diese Art der Anbindung für den Anwender nicht transparent.

# Qualitätsszenarien

## Qualitätsbaum



## Bewertungsszenarien

|  |  |
| --- | --- |
| Nr | Szenario |
| 1 | Ein Interessierter mit Grundkenntnissen in UML möchte einen Einstieg in die Architektur von MedDevMM finden.  Lösungsstrategie und Entwurf erschließen sich innerhalt einer Stunde. |
| 2 | Ein Architekt, der arc42 anwenden möchte, sucht zu einem beliebigen Kapitel des Templates einen konkreten Beispielinhalt und findet ihn unverzüglich in der Dokumentation |
| 3 | Ein erfahrener Webentwickler sucht die Implementierung eines im Entwurf beschriebenen Bausteins.  Er findet sie ohne Umwege oder fremde Hilfe im Quelltext |
| 4 | Ein System Engineer integriert ein neues Gerät in das bestehende System. Er kann dies ohne Probleme und zusätzlichen Aufwand tun |
| 5 | Das System wird auf Grund geänderter Anforderungen umgebaut. Durch die sauber definierten Schnittstellen des Systems und die gewählte Architektur ist es möglich Änderungen ohne Herunterfahren des Systems zuzulassen. |
| 6 | Der Anwender gibt seine Logindaten ein und drückt auf Login. Die Benutzerdaten werden vom validiert und liefert entweder eine Fehlermeldung zurück oder leitet den Anwender zur vorgesehenen Seite weiter. |

# Risiken

Die folgenden Risiken wurden zu Beginn des Projekts identifiziert. Diese beeinflussten die Planung der Iterationen maßgeblich.

|  |  |  |
| --- | --- | --- |
| Prio | Risiko | Erläuterung |
| 1 |  | Die Schnittstellen ändern sich ungewollt gegen Ende des Projekts |
| 2 |  | Das System ist zum definierten Zeitpunkt nicht ausreichend getestet |
| 3 |  | Die verwendeten Frameworks bzw. Libraries weisen nicht die gewollte Stabilität auf |
| 4 |  | Nichterreichen der verlangten Leistungen |