|  |
| --- |
|  |

|  |
| --- |
|  |

IPA von Remo Kessler

Zentralisierte Parameterverwaltung für eine Mikroservices-Architektur

Über dieses Dokument

Ablage

|  |  |
| --- | --- |
| Git Repository | https://github.com/kre-cmi/IPA-KRE |

Versionierung

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Datum | Autor | Status | Bemerkung |
| 0.1 | 22.02.2018 | Remo Kessler | In Arbeit | Erstellung des Dokuments inkl. Formatierungen und Ablage im GIt. |
|  |  |  |  |  |

Referenzierte Dokumente

|  |  |  |  |
| --- | --- | --- | --- |
| Dokumentenname | Version | Autor | Datum |
|  |  |  |  |
|  |  |  |  |

Verteiler

|  |  |  |  |
| --- | --- | --- | --- |
| Name / Vorname | Kurzzeichen | Organisation | Rolle |
| Remo Kessler | KRE | CM Informatik AG | Lehrling Informatik |

Konventionen

In diesem Dokument wurden die folgenden Konventionen verwendet:

|  |  |
| --- | --- |
| Was | Beschrieb |
| Gleichstellung | Im Zuge sprachlicher Vereinfachung wird innerhalb des vorliegenden Dokuments jeweils nur eine Form von Personenbezeichnungen (z.B. Projektleiterin, Mitarbeiter etc.) verwendet. Es ist stets auch die andere Form der entsprechenden Personenbezeichnung gemeint und miteingeschlossen. |
| Grün & ??? | Grün hinterlegter Text betrifft Punkte, welche noch besprochen werden müssen |
| Gelb & !!! | Gelb hinterlegter Text betrifft Punkte, welcher bei der Umsetzung speziell beachtet werden müssen. |

Inhaltsverzeichnis

[1 Zweck des Dokuments 6](#_Toc509324057)

[1.1 Thema und Zielsetzung 6](#_Toc509324058)

[1.2 Allgemeines zur Umgebung 6](#_Toc509324059)

[1.3 Endprodukt 6](#_Toc509324060)

[2 Rahmenbedingungen 7](#_Toc509324061)

[2.1 Titel der IPA 7](#_Toc509324062)

[2.2 Auslöser der Aufgabenstellung 7](#_Toc509324063)

[2.3 Grobanforderungen 7](#_Toc509324064)

[2.3.1 Parameterverwaltung 7](#_Toc509324065)

[2.3.1.1 Funktionale Anforderungen 7](#_Toc509324066)

[2.3.1.2 Nicht Funktionale Anforderungen 7](#_Toc509324067)

[2.3.2 «Getting Started» Dokumentation 8](#_Toc509324068)

[2.4 Mittel & Methoden 8](#_Toc509324069)

[2.4.1 Eingesetzte Mittel 8](#_Toc509324070)

[2.4.2 Git Workflow 8](#_Toc509324071)

[2.4.3 Kontroll-Tasks 8](#_Toc509324072)

[2.5 Vorkenntnisse 8](#_Toc509324073)

[2.6 Neue Lerninhalte 8](#_Toc509324074)

[3 Allgemeines 9](#_Toc509324075)

[3.1 Konventionen 9](#_Toc509324076)

[3.1.1 Dokumentation 9](#_Toc509324077)

[3.1.2 Code-Konventionen C# 9](#_Toc509324078)

[3.1.3 Code-Konventionen TypeScript 9](#_Toc509324079)

[3.1.4 Layout-Konventionen 10](#_Toc509324080)

[3.2 Vorgehensmodell 10](#_Toc509324081)

[3.2.1 IPERKA 10](#_Toc509324082)

[4 Umsysteme, Abhängigkeiten, Systemgrenzen und Schnittstellen 11](#_Toc509324083)

[4.1 Umsysteme 11](#_Toc509324084)

[4.2 Abhängigkeiten 11](#_Toc509324085)

[4.3 Systemgrenzen 12](#_Toc509324086)

[4.4 Schnittstellen 13](#_Toc509324087)

[5 Taskerstellung nach den Anforderungen 14](#_Toc509324088)

[5.1 Einzeltasks für die Dokumentation 14](#_Toc509324089)

[5.1.1 Getting Started Dokumentation 14](#_Toc509324090)

[5.2 Einzeltasks für die Realisierung 14](#_Toc509324091)

[5.2.1 Anzeige & Speichern der Parameter 14](#_Toc509324092)

[5.2.2 Implementation des Validierungsmechanismus 15](#_Toc509324093)

[5.2.3 Suchen eines Parameters 15](#_Toc509324094)

[5.2.4 Erstellung der Unit Tests und des Testkonzepts 15](#_Toc509324095)

[5.3 Einzeltaskerstellung der Kontrolltasks 16](#_Toc509324096)

[5.3.1 Durchführen der Tests 16](#_Toc509324097)

[6 Zeitplan 17](#_Toc509324098)

[7 Detailanalyse der erstellten Tasks 18](#_Toc509324099)

[7.1 Umsetzungsreihenfolge 18](#_Toc509324100)

[7.2 Erstellung des Testkonzepts 18](#_Toc509324101)

[7.2.1 Anforderung 18](#_Toc509324102)

[7.2.2 Mögliche Lösungen 18](#_Toc509324103)

[7.2.2.1 Unittests 18](#_Toc509324104)

[7.2.2.2 Integration Tests / e2e Tests 19](#_Toc509324105)

[7.2.2.3 User-Testing 19](#_Toc509324106)

[7.2.3 Umsetzungsbeschreibung 19](#_Toc509324107)

[7.3 Anzeige & Speichern der Parameter 20](#_Toc509324108)

[7.3.1 Anforderung 20](#_Toc509324109)

[7.3.2 Mögliche Lösungen Speichern 20](#_Toc509324110)

[7.3.2.1 Speichern der Parameter als Parametertyp im Json 20](#_Toc509324111)

[7.3.2.2 Speichern der Parameter als generischer Typ im Json 20](#_Toc509324112)

[7.3.3 Umsetzungsbeschreibung Speichern 20](#_Toc509324113)

[7.3.4 Mögliche Lösung Anzeige 21](#_Toc509324114)

[7.3.4.1 Ähnlich wie in Firefox / Waterfox die about:config 21](#_Toc509324115)

[7.3.4.2 Ähnlich wie die Chrome Settings 22](#_Toc509324116)

[7.3.5 Ähnlich wie in Visual Studio 23](#_Toc509324117)

[7.3.6 Umsetzungsbeschreibung Anzeige 24](#_Toc509324118)

[7.4 Implementation des Validierungsmechanismus 25](#_Toc509324119)

[7.4.1 Anforderung 25](#_Toc509324120)

[7.4.2 Mögliche Lösungen 25](#_Toc509324121)

[7.4.2.1 Nur Serverseitig im Service den Parameter selbst testen lassen. 25](#_Toc509324122)

[7.4.2.2 Regular Expression im Parameter 25](#_Toc509324123)

[7.4.3 Umsetzungsbeschreibung 25](#_Toc509324124)

[7.5 Suche eines Parameters 26](#_Toc509324125)

[7.5.1 Anforderung 26](#_Toc509324126)

[7.5.2 Mögliche Lösungen 26](#_Toc509324127)

[7.5.2.1 Volltextsuche über alles 26](#_Toc509324128)

[7.5.2.2 Volltextsuche auf die Parameternamen 26](#_Toc509324129)

[7.5.2.3 Volltextsuche auf Parameternamen und Parameterwert 26](#_Toc509324130)

[7.5.3 Umsetzungsbeschreibung 27](#_Toc509324131)

[7.6 Erstellung der Unittests 27](#_Toc509324132)

[7.6.1 Anforderung 27](#_Toc509324133)

[7.6.2 Umsetzungsbeschreibung 27](#_Toc509324134)

[7.7 Durchführen der Tests 27](#_Toc509324135)

[7.7.1 Anforderung 27](#_Toc509324136)

[7.7.2 Umsetzungsbeschreibung 27](#_Toc509324137)

[7.8 Getting Started Dokumentation 27](#_Toc509324138)

[7.8.1 Anforderung 27](#_Toc509324139)

[7.8.2 Umsetzungsbeschreibung 28](#_Toc509324140)

# Zweck des Dokuments

In diesem Dokument wird der Entwicklungsprozess der IPA «Zentralisierte Parameterverwaltung für eine Microservice-Architektur» beschrieben. Des Weiteren dient diese Dokumentation als Abschlussarbeit von Remo Kessler.

## Thema und Zielsetzung

Diese Projektarbeit befasst sich mit dem Thema von Microservices und wie man diese zentral verwalten kann. Dies unter Berücksichtigung der Microservice-Architektur.

Dies ist nun das Ausgangsproblem für diese Arbeit. Als Endprodukt soll eine funktionierende, zentrale Parameterverwaltung für das oben genannte Problem entstehen. Dies natürlich unter der Berücksichtigung von der Microservice-Architektur.

## Allgemeines zur Umgebung

Das Projekt wird in einem eigenen Repository unabhängig vom restlichen Projekt umgesetzt. Um das ganze möglichst zu vereinfachen sind sämtliche Funktionen, die nichts mit der IPA zu tun haben, nicht in diesem Repository enthalten. Im Anschluss des Projekts wird die Parameterverwaltung in die Entwicklungsumgebung und anschliessend in die Live-Umgebung eingebaut.

## Endprodukt

Nach Abschluss der Projektarbeit liegen folgende Produkte vor:

* Vollständige Dokumentation des Lösungsweges
* Arbeitsjournale
* Programm als Visaul Studio respektive Webstorms Projekte

# Rahmenbedingungen

## Titel der IPA

Zentralisierte Parameterverwaltung für eine Mikroservices-Architektur

## Auslöser der Aufgabenstellung

Im Projekt Viaduc, welches von der CM Informatik AG derzeit umgesetzt wird, geht es derzeit um die Erstellung einer Software, welche zur Aufgabe hat einen klassischen Lesesaal online verfügbar zu machen. Durch das die hohen und komplexen Anforderungen an den Backendbereich, entschloss man sich dieses mit einer Microservicearchitektur um zu setzen. Dies hat zur Folge, dass die Parametrierung dieser Services nicht zentral gespeichert werden kann, da sonst die Eigenständigkeit der Services nicht mehr gewährleistet wird und die Microservice-Architektur dann keine richtige Microservice-Architektur mehr wäre.

## Grobanforderungen

### Parameterverwaltung

Dies sind die Kriterien aus der Aufgabenstellung für die Parameterverwaltung.

#### Funktionale Anforderungen

* Der Administrator kann alle Parameter aller Systemdienste an einem Ort pflegen
* Das Auffinden des gewünschten Parameters wird vom System unterstützt
* Default-Werte sind optisch als solche identifizierbar
* Validierungsfehler werden optisch hervorgehoben
* Ein Hilfetext kann für jeden Parameter hinterlegt werden
* Die Validierung erlaubt die Kontrolle bei der Erfassung
* Die Validierung kann auch separat aufgerufen werden
* Der Validierungsmechanismus soll versch. Formate unterstützen
* Die PV erkennt selbstständig die parametrierbaren Dienste und ihre Werte

#### Nicht Funktionale Anforderungen

* Die Parameter werden im Service gespeichert, nicht zentral(!)
* Die Services funktionieren auch, wenn die Parameterverwaltung (PV) nicht läuft
* Die PV funktioniert rechnerübergreifend, ohne File-Sharing
* Es werden sprechende Namen für Klassen / Methoden verwendets
* Eine Beispielimplementierung im Projekt Viaduc ist vorhanden und einsehbar
* Der Programmierer braucht sich nicht um GUI Aspekte zu kümmern
* Ist es nachvollziehbar, warum gerade diese Lösung gewählt wurde? Was waren die Kriterien?
* Werden mögliche Fehler mit den entsprechenden Mitteln erkannt und behandelt? Mögliche Fehler: fehlerhafte Parameter, fehlende Parameter

### «Getting Started» Dokumentation

Dies sind die Kriterien aus der Aufgabenstellung an die «Getting Started» Dokumentation. Da dies eine Reine Dokumentation ist, macht es keinen Sinn in die Unterteilung Funktional und nicht Funktionale Anforderungen.

* Bietet einen konzeptionellen Überblick
* Schritt für Schritt Anleitung für Einbindung
* Mind. 2 Beispiele für die Validierung
* Mind. 2 Screenshots der resultierenden Darstellung im GUI

## Mittel & Methoden

### Eingesetzte Mittel

* Visual Studio für die Backendprogrammierung in C#
* Webstorms für die Frontendprogrammierung in TypeScript
* MS Office für die Erstellung der Dokumentation
* Office at Work Vorlagen, um das CI/CD der Dokumentation ein zu halten
* Draw.io für das Zeichnen der Diagramme.
* Konventionen der Programmierung im Backend und Frontend wie bei der CM Informatik AG üblich

### Git Workflow

Das gesamte Projekt wird mit Git respektive Github versioniert und abgelegt. Da es im Projekt nur einen «Contributer» gibt, macht es wenig Sinn immer einen eigenen Branch pro Task zu erstellen, da man keinen Nutzen daraus gewinnt und Zeit verliert durchs Mergen etc. Wegen der Sicherheit der Daten wird dennoch jeder Commit auch gleich gepusht. Zudem wird es jeden Tag eine Version geben.

### Kontroll-Tasks

* Nach den IPERKA Phasen Planen, Entscheiden und Realisieren wird das Dokument zum gegenlesen gegeben. Am anschliessenden Tag, werden jeweils 45 Minuten eingerechnet, um zu besprechen und allfällige Fehler zu beheben.
* Jeweils am Ende einer Woche, wird die Anforderungsliste mit dem Ist Stand verglichen.

## Vorkenntnisse

* C# Programmierung während der Lehre bei der Greenshare AG / CM Informatik
* TypeScript Programmierung / LESS / Angular seit einem halben Jahr bei der CM Informatik AG im Projekt Viaduc
* HTML und CSS Kenntnisse aus der Berufsschule und dem Viaduc Projekt

## Neue Lerninhalte

* Microservices-Architektur
* RabbitMQ / MassTransit

# Allgemeines

## Konventionen

### Dokumentation

|  |  |
| --- | --- |
| Bereich | Vorgabe |
| Seitenumbruch | Alle Überschriften erster Stufe beginnen auf einer neuen Seite. Keine Überschrift steht ganz unten auf einer Seite. |
| CI/CD | Von OAW zu entnehmen |

### Code-Konventionen C#

|  |  |
| --- | --- |
| Bereich | Vorgaben |
| Namensgebung Methoden/Variablen | Methoden- und Variablennamen soll Prägnant und in Englisch sein zudem immer in Camel-Case Fachbegriffe in Deutsch |
| Private / Public | Private klein, Public gross |
| Methodenlänge | Methoden sollen nicht zu lang sein. |
| Service | Ein Service besteht aus dem Host, dem Manager und dem Contract.  Im Host befindet sich nur der Service selbst, im Manager die Logik und im Contract die gemeinsame Logik von Empfänger und Sender. |
| Projektnamensgebung | CMI.(was es ist: Host | Manager | Service | Web).ServiceName |

### Code-Konventionen TypeScript

|  |  |
| --- | --- |
| Bereich | Vorgaben |
| Namensgebung Methoden/Variablen | Methoden- und Variablennamen soll Prägnant und in Englisch sein zudem immer in Lower-Camel-Case Fachbegriffe in Deutsch |
| Private / Public | Private mit Underscore, Public nur Lower-Camel-Case |
| Methodenlänge | Methoden sollen nicht zu lang sein. |
| Component | Pro Component ist ein eigenes Verzeichnis an zu legen  KomponentenName.component.(ts | html | less)  Services sind im Services Verzeichnis ab zu legen und via Dependency Injection mit Angular an zu steuern. |

### Layout-Konventionen

|  |  |
| --- | --- |
| Bereich | Vorgabe |
| CI / CD | CI / CD des Bundes ist ein zu halten. Sagt dieses nichts aus ist auf Bootstrap zurück zu greifen. Ist da ebenfalls nichts definiert, so muss ein Vorschlag gemacht werden. |
| Barrierefreiheit | Es muss alles mit der Tastatur bedienbar sein. |

## Vorgehensmodell

Das Projekt wird mit IPERKA umgesetzt. Die CM Informatik AG setzt in der Entwicklung auf Scrum. Dies ist jedoch bewusst anders gewählt, da die CM Informatik AG mit einem 1 Monatssprint arbeitet. Dies ist nicht Sinnvoll für ein Projekt von 3 Wochen Dauer. Zudem kommen die Vorteile der Agilen Entwicklung in einem in sich selbst Abgeschlossenen Projekt nicht zum Tragen. Da die IPA eine Projektarbeit unter idealen Bedingungen darstellt, muss auch nicht auf allfällige Anforderungsänderungen eingegangen werden.

### IPERKA

Die sechs Phasen von IPERKA gehen in einander über. Welcher Task zu welchem schritt gehört ist im Zeitplan ersichtlich

# Umsysteme, Abhängigkeiten, Systemgrenzen und Schnittstellen

Das ganze System ist anhand der Vereinfachten Bedingungen während der IPA aufgezeigt, da das Livesystem zu gross und komplex ist für dieses Projekt. Die Services, die im Livesystem vorhanden sind, verhalten sich gleich, wie jene, die sich im IPA-System befinden.

## Umsysteme

Für das IPA-Projekt gibt es keine relevanten Umsysteme, da man sich innerhalb des Projektes Viaduc befindet. Die Kommunikation mit den Umsystemen findet im Viaduc über einzelne Microservices stat. Da diese aber in der IPA nicht enthalten sind, kommuniziert das System mit keinem Umsystem.

## Abhängigkeiten

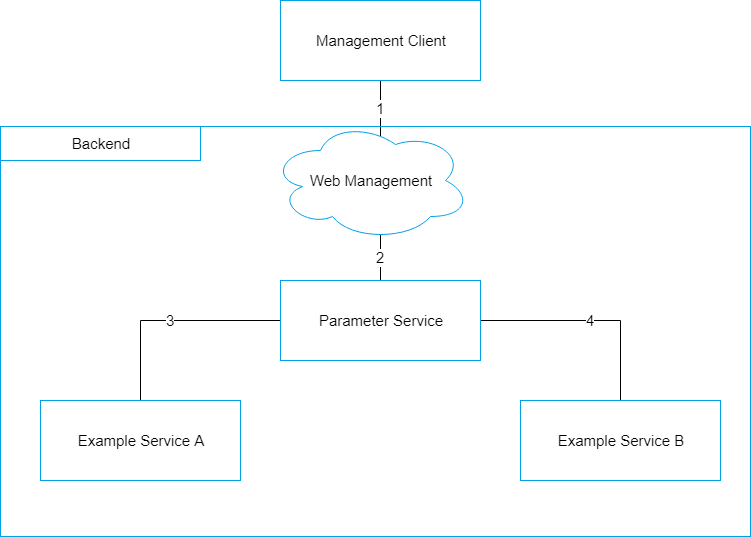
Aufgrund der Microservices-Architektur ist das ganze Programm so unabhängig wie es geht. Jeder Service ist in sich selbst gekapselt. Die einzelnen Services sind nur über die jeweilige Contract DLL miteinander verbunden.

Im Fall dieser Arbeit ist dies die CMI.Contract.Parameter, welche als einzige eine Abhängigkeit hat, eine zentrale Rolle. In dieser DLL werden sämtliche Kommunikationsinterfaces für den Parameter Service definiert.

## Systemgrenzen

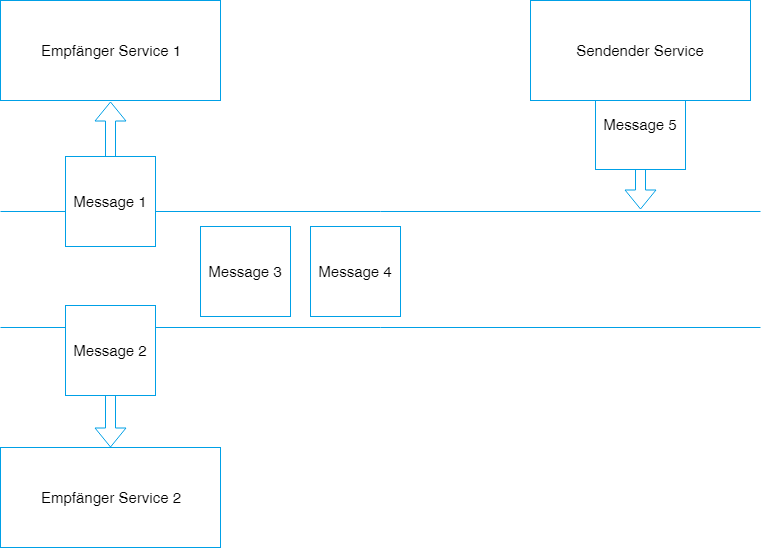
Im Projekt Viaduc haben wir verschiedenste Services. Diese sind im Rahmen dieser Projektarbeit durch die «Example Services» A und B repräsentiert. Der Web Managementteil dient nur zum Informationseingang Sämtliche Logik ist im Parameter Service zu schreiben.

|  |  |
| --- | --- |
| Bezeichnung | Was es ist |
| Management Client | Der Web Client vom Projekt |
| Web Management | Der Webendpunkt, wo sich die APIs befinden |
| Parameter Service | Der Service, welcher während der IPA |
| Example Service A & B | Diese zwei Services stellen zwei beliebige Viaduc Microservices dar |
| 1 | API Call und Response |
| 2, 3, 4 | RabbitMQ / MassTransit Kommunikation |



## Schnittstellen

Die Kommunikation entsteht durch RabbitMQ und MassTransit. Mit diesen beiden Frameworks ist es möglich verschiedene Kommunikations-Queues zu erstellen. Diese werden dann nach dem First in First out Prinzip abgearbeitet. Das gesamte Backend kommuniziert über solche RabbitMQ Queues. Die Kommunikation ist wie folgt gezeigt sich vor zu stellen.



Dieses Diagramm zeigt die Event-Kommunikation von RabbitMQ und MassTransit auf. Der Sender Service gibt Messages in die Queue (der Balken in der Mitte) und von der Queue aus gelangen diese dann an den entsprechenden Empfänger Service und werden so abgearbeitet. Einmal erfolgreich abgearbeitet, werden die Messages aus der Queue entfernt.

# Taskerstellung nach den Anforderungen

## Einzeltasks für die Dokumentation

### Getting Started Dokumentation

|  |
| --- |
| Erfüllte Ziele |
| Bietet einen konzeptionellen Überblick |
| Schritt für Schritt Anleitung für Einbindung |
| Mind. 2 Beispiele für die Validierung |
| Mind. 2 Screenshots der resultierenden Darstellung im GUI |

|  |
| --- |
| Anforderung |
| Die Getting Started Dokumentation soll es nach der IPA erleichtern den anderen Projekt-Entwicklern selbst einen Parameter zu erstellen für die Parameterverwaltung |

**Geschätzter Aufwand**: 2h

## Einzeltasks für die Realisierung

### Anzeige & Speichern der Parameter

|  |
| --- |
| Erfüllte Ziele |
| Ein Hilfetext kann für jeden Parameter hinterlegt werden |
| Default-Werte sind optisch als solche identifizierbar |
| Werden mögliche Fehler mit den entsprechenden Mitteln erkannt und behandelt? Mögliche Fehler: fehlerhafte Parameter, fehlende Parameter |
| Die Parameter werden im Service gespeichert, nicht zentral |
| Die PV erkennt selbstständig die parametrierbaren Dienste und ihre Werte |
| Der Administrator kann alle Parameter aller Systemdienste an einem Ort pflegen |

|  |
| --- |
| Anforderung |
| Die Parameterverwaltung muss die Parameter Speichern können. Dies muss zentral im Management Client geschehen. Zu diesem Zweck muss die Parameterverwaltung alle Parameter Abfragen und Validieren können.  Eine funktionierende Validierung ist zur Umsetzung dieses Tasks erforderlich. |

**Geschätzter Aufwand**: 12h

### Implementation des Validierungsmechanismus

|  |
| --- |
| Erfüllte Ziele |
| Die Validierung kann auch separat aufgerufen werden |
| Die Validierung erlaubt die Kontrolle bei der Erfassung |
| Validierungsfehler werden optisch hervorgehoben |
| Der Validierungsmechanismus soll versch. Formate unterstützen |
| Werden mögliche Fehler mit den entsprechenden Mitteln erkannt und behandelt? Mögliche Fehler: fehlerhafte Parameter, fehlende Parameter |

|  |
| --- |
| Anforderung |
| Die Parameterverwaltung muss die Richtigkeit der Daten überprüfen können. Diese soll bei Bedarf auch separat aufgerufen werden, sodass z.B. beim neuerfassen eines Service gleich hervorgehoben werden kann, was noch zu korrigieren ist und wo die Default-Werte genügen. |

**Geschätzter Aufwand**: 8h

### Suchen eines Parameters

|  |
| --- |
| Erfüllte Ziele |
| Das Auffinden des gewünschten Parameters wird vom System unterstützt |

|  |
| --- |
| Anforderung |
| Das System muss um Benutzerfreundlich zu bleiben dem Benutzer dabei behilflich sein, einen gewünschten Parameter schnell zu finden, da zurzeit noch ungewiss ist, wie viele Parameter wirklich in der Parameterverwaltung vorkommen werden. |

**Geschätzter Aufwand**: 8h

### Erstellung der Unit Tests und des Testkonzepts

|  |
| --- |
| Anforderung |
| Für Funktionen, welche daten Verarbeiten oder Validieren, muss ein mindestens ein Unit-Test geschrieben werden. |

**Gesamter Geschätzter Aufwand:** 4h

## Einzeltaskerstellung der Kontrolltasks

### Durchführen der Tests

|  |
| --- |
| Erfüllte Ziele |
| Die Services funktionieren auch, wenn die Parameterverwaltung (PV) nicht läuft |
| Die PV funktioniert rechnerübergreifend, ohne File-Sharing |
| Der Programmierer braucht sich nicht um GUI Aspekte zu kümmern |
| Eine Beispielimplementierung im Projekt Viaduc ist vorhanden und einsehbar |
| Es werden sprechende Namen für Klassen / Methoden verwendets |
| Ist es nachvollziehbar, warum gerade diese Lösung gewählt wurde? Was waren die Kriterien? |

|  |
| --- |
| Anforderung |
| Überprüfung ob alle oben genannten Ziele erfüllt wurden.  Zudem muss folgendes getestet werden.  **Frontend**   * der Barrierefreiheit (alles mit der Tastatur bedienbar?) * Ist das CI/CD vom Bund, oder falls dies nichts sagt, von Bootstrap eingehalten?   **Backend**  Auf dem Server müssen die Tests durchlaufen, Sobald diese geschrieben sind. |

**Geschätzter Aufwand pro Durchgang**: 1h

Dieser Task muss 1 Mal nach jedem Realisiertem Realisierung-Punkt Umgesetzt werden.

**Gesamter Geschätzter Aufwand:** 4h

# Zeitplan

# Detailanalyse der erstellten Tasks

## Umsetzungsreihenfolge

Ich brauche das Testkonzept, um die Tests, welche nicht durch Unittests abgedeckt werden können, zu testen. Ich brauche einen Parameter, um die Validierung schreiben zu können, und die suche baut ebenfalls auf der Validierung auf. Die Erstellung der Unittests kann erst mit der Fertigstellung des restlichen Source Codes abgeschlossen werden. Und zu Letzt die Getting Started Doku, da diese Screenshots benötigt und die konkrete Umsetzung, bevor sie geschrieben werden kann. Dies ergibt folgende Reihenfolge.

|  |  |
| --- | --- |
| Nr. | Task |
| 1 | Erstellung des Testkonzepts |
| 2 | Anzeige & Speichern der Parameter |
| 3 | Implementation des Validierungsmechanismus |
| 4 | Suche eines Parameters |
| 5 | Erstellung der Unittests |
| 6 | Durchführen der Tests |
| 7 | Getting Started |

## Erstellung des Testkonzepts

### Anforderung

Das Testkonzept muss alle Test-Cases beinhalten. Es soll möglichst einfach zu testen sein. Zudem darf der Zeitaufwand beim Testen und beim Test schreiben nicht zu gross sein.

### Mögliche Lösungen

#### Unittests

|  |  |
| --- | --- |
| Kriterium | Punktzahl |
| Einfach aus zu führen | 25 / 25 |
| Einfach zu schreiben / definieren | 15 / 25 |
| Zeitaufwand des Tests | 25 / 25 |
| Zeitaufwand des Schreibens der Test | 15 / 25 |
| **Total** | 80 / 100 |

#### Integration Tests / e2e Tests

|  |  |
| --- | --- |
| Kriterium | Punktzahl |
| Einfach aus zu führen | 15 / 25 |
| Einfach zu schreiben / definieren | 10 / 25 |
| Zeitaufwand des Tests | 25 / 25 |
| Zeitaufwand des Schreibens der Test | 0 / 25 |
| **Total** | 50 / 100 |

#### User-Testing

|  |  |
| --- | --- |
| Kriterium | Punktzahl |
| Einfach aus zu führen | 25 / 25 |
| Einfach zu schreiben / definieren | 25 / 25 |
| Zeitaufwand des Tests | 5 / 25 |
| Zeitaufwand des Schreibens der Test | 25 / 25 |
| **Total** | 80 / 100 |

### Umsetzungsbeschreibung

Für das Testkonzept werde ich eine Mischung aus den zwei Siegern machen. Im Frontend sind die Anforderungen, die getestet werden müssen nicht mit Unit-Tests abdeckbar. Jedoch sind e2e Tests viel zu aufwendig. Im Backend macht ein User-Testing jedoch keinen Sinn, weshalb das Backend mit Unittests abgedeckt wird.

## Anzeige & Speichern der Parameter

### Anforderung

Die Parameter dürfen nicht zentral gespeichert werden. Die Microservices selbst müssen den ganzen Aufbau und die Konfiguration der Parameter selbst wissen. Der Parameterservice trägt diese dann zusammen. So muss jeder Service seine Parameter selbständig lesen und schreiben können.

### Mögliche Lösungen Speichern

#### Speichern der Parameter als Parametertyp im Json

|  |  |
| --- | --- |
| Kriterium | Punktzahl |
| Einfach aus zu lesen | 25 / 25 |
| Einfach zu schreiben | 25 / 25 |
| Einfach erweiterbar | 5 / 25 |
| Fix definierte Form | 0 / 25 |
| **Total** | 55 / 100 |

#### Speichern der Parameter als generischer Typ im Json

|  |  |
| --- | --- |
| Kriterium | Punktzahl |
| Einfach aus zu lesen | 10 / 25 |
| Einfach zu schreiben | 15 / 25 |
| Einfach erweiterbar | 25 / 25 |
| Fix definierte Form | 25 / 25 |
| **Total** | 75 / 100 |

### Umsetzungsbeschreibung Speichern

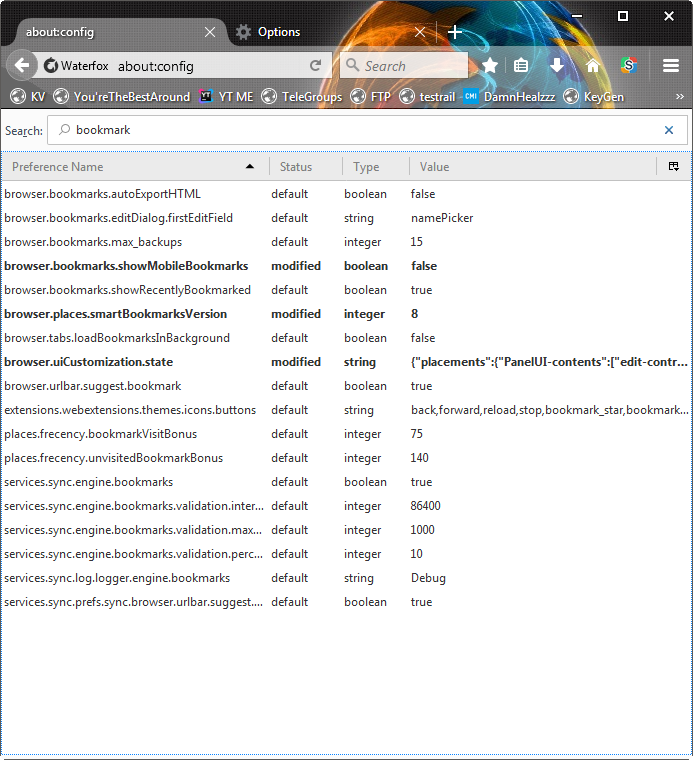
Das Speichern der Parameter als generischer Typ im Json scheint die bessere Option zu sein, da man klar definiert wie das Json aus zu sehen hat. Da jeder Service seine Parameter selbst speichert, ist es natürlich dem Service selbst überlassen. Jedoch ist zur Versendung der Parameter eine Vereinheitlichung notwendig. Der Aufbau der Versendungsklasse ist dann auch idealerweise der Aufbau des Jsons.

Das Json muss folgende Informationen des Parameters beinhalten:

* Name
* Value
* Typ
* Beschreibung
* Pflichtfeld
* Wie der Parameter zu validieren ist.

### Mögliche Lösung Anzeige

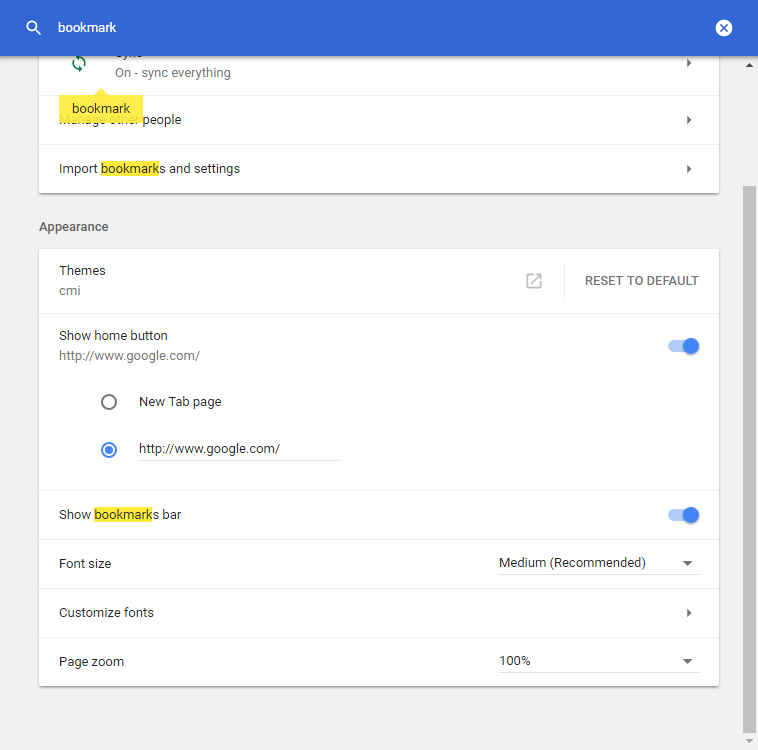
#### Ähnlich wie in Firefox / Waterfox die about:config



Man Listet einfach alle Parameter. Mit einem Doppelklick auf den Parameter kann man diesen editieren und speichern.

Dies hat den Vorteil, dass es einfach zu programmieren ist, da jeder Parameter gleich aussieht. Es gibt keine Möglichkeit die Parameter zu validieren, was man noch hinzufügen müsste. Die Parameter sind nicht Gruppiert, es gibt aber eine Suchmöglichkeit, die das ganze ziemlich angenehm zu bedienen macht.

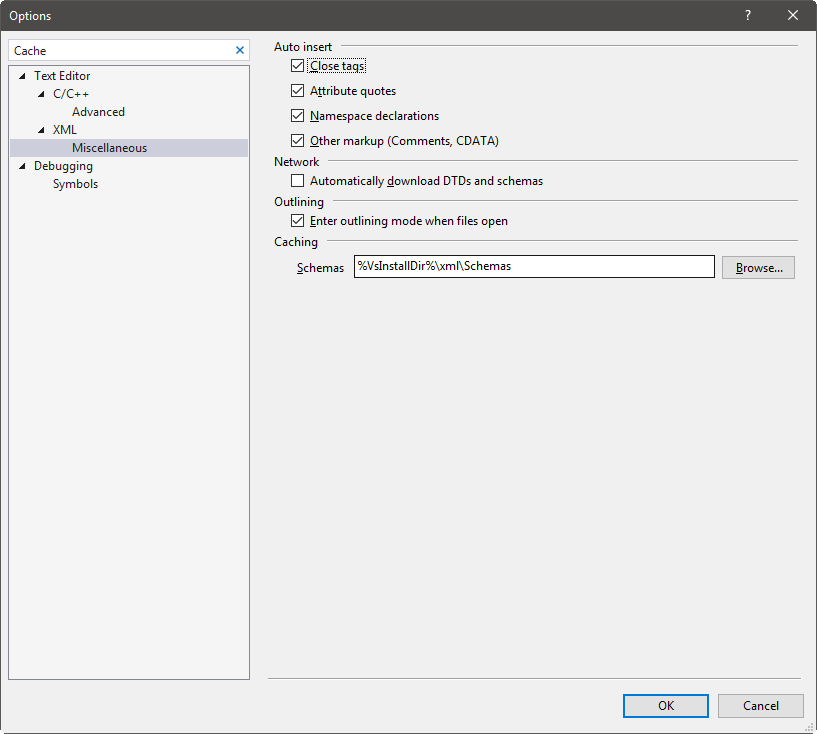
#### Ähnlich wie die Chrome Settings



Die Chrome Settings Page erlaubt ebenfalls das Suchen nach einem begriff. Es wird alles hervorgehoben wurde. Es ist strukturiert aufgebaut, jedoch nicht ganz so simpel um etwas zu finden, wenn man nicht genau weiss wo.

Es ist modern aufgebaut und besitzt die Funktion alles auf den Default wert zu setzen. Der Standardwert wird oftmals mit (Recommended) gekennzeichnet.

### Ähnlich wie in Visual Studio



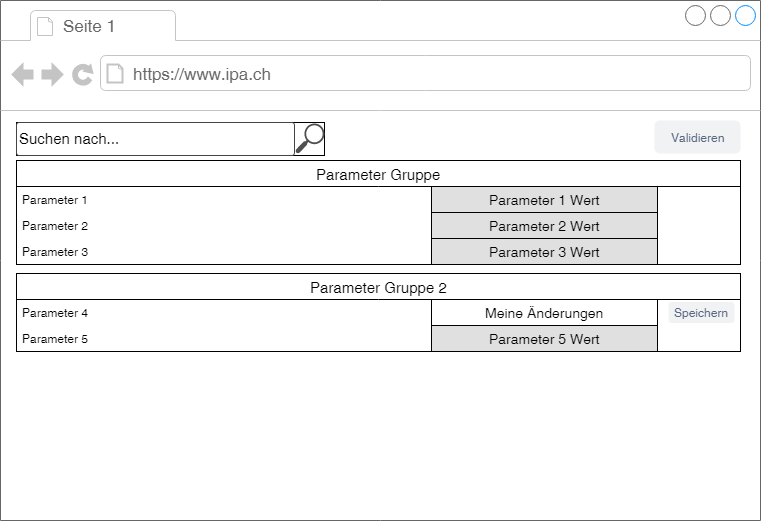
Visual Studio hat ebenfalls viele Parameter zu verwalten genau wie die Browser. In Visual Studio jedoch gibt es viele verschiedene Parameter, welche verschiedene Editoren braucht. Oftmals sind es auch nur Checkboxen. Es unterstützt das Auffinden von Parametern, jedoch nur schlecht. Das gute an dieser Lösung ist, die hierarchische Gliederung der Parameter. Denn es kann so schnell eingegrenzt werden wo etwas sein sollte. Diese ist gut, wenn man sich mit dem Tool nicht so auskennt. Und nicht weiss, wie welcher Parameter genau heisst.

### Umsetzungsbeschreibung Anzeige

Alle der drei Vorgestellen Möglichkeiten haben Vor- und Nachteile. Da im Projekt Bedingungen herrschen, denen keine der Drei Lösungen entspricht wird hier von allem das beste genommen.

So wird die Parameterbezeichnung von Firefox genommen. Via dieser eine Gruppierung wie in Visual Studio vorgenommen. Und die Suche wie in Chrome, dass es hervorgehoben wird. Wobei die Gruppierung Optional vorgenommen werden kann, wenn noch Zeit dafür ist.

Ein grobes Konzept könnte wie folgt aussehen:



Klickt man in ein Feld, so wird der Bearbeitungsmodus aktiviert und der Speichern-Button erscheint. Ist man mit den Änderungen fertig, klickt man auf «Speichern» und der Parameter wird gespeichert. Dies ist das gleiche Verhalten wie bei Firefox, nur dass in Firefox ein Pop-Up aufgeht. Dies ist jedoch für dieses Projekt nicht von Vorteil, falls man viele Parameter auf einmal editieren muss.

Der Hilfetext ist via Mouseover angezeigt und der Standardwert wird vor dem Parameter Wert angezeigt.

Ist bei der Validierung ein Parameter nicht ok. So soll dieser rot hervorgehoben werden. Wird er mit Hilfe der Suche gefunden, so wird der Parameter Name gelb hervorgehoben.

## Implementation des Validierungsmechanismus

### Anforderung

Der Validierungsmechanismus muss mehrere verschiedene Formate zulassen. Zudem muss der Validierungsmechanismus auch separat, nicht beim Speichern aufgerufen werden können.

### Mögliche Lösungen

#### Nur Serverseitig im Service den Parameter selbst testen lassen.

|  |  |
| --- | --- |
| Kriterium | Punktzahl |
| Sicher gegen Angriffe (z.B. XSS) | 25 / 25 |
| Clientseitige Validierung möglich | 0 / 25 |
| Serverseitige Validierung möglich | 25 / 25 |
| Verschieden Formate unterstützt | 25 / 25 |
| **Total** | 75 / 100 |

#### Regular Expression im Parameter

|  |  |
| --- | --- |
| Kriterium | Punktzahl |
| Sicher gegen Angriffe (z.B. XSS) | 25 / 25 |
| Clientseitige Validierung möglich | 25 / 25 |
| Serverseitige Validierung möglich | 25 / 25 |
| Verschieden Formate unterstützt | 25 / 25 |
| **Total** | 100 / 100 |

### Umsetzungsbeschreibung

Indem der Parameter einen Regex Ausdruck kennt, kann er sich jeweils selbst Validieren. Hierbei spielt es keine Rolle, ob es auf dem Client oder dem Server validiert werden muss. Zudem ist Regex sehr flexibel und man kann fast alles damit validieren.

## Suche eines Parameters

### Anforderung

Die Parameterverwaltung muss das Auffinden eines Parameters unterstützen. Die gefundenen Treffer sollen visuell hervorgehoben werden.

### Mögliche Lösungen

#### Volltextsuche über alles

|  |  |
| --- | --- |
| Kriterium | Punktzahl |
| Findet den Parameter, wenn man dessen Namen kennt | 25 / 25 |
| Findet den Parameter, wenn man dessen Wert kennt | 25 / 25 |
| Findet den Parameter, wenn man weiss, auf welchem Service definiert ist | 25 / 25 |
| Geringe Komplexität der Suche | 15 / 25 |
| **Total** | 90 / 100 |

#### Volltextsuche auf die Parameternamen

|  |  |
| --- | --- |
| Kriterium | Punktzahl |
| Findet den Parameter, wenn man dessen Namen kennt | 25 / 25 |
| Findet den Parameter, wenn man dessen Wert kennt | 0 / 25 |
| Findet den Parameter, wenn man weiss, auf welchem Service definiert ist | 25 / 25 |
| Geringe Komplexität der Suche | 25 / 25 |
| **Total** | 75 / 100 |

#### Volltextsuche auf Parameternamen und Parameterwert

|  |  |
| --- | --- |
| Kriterium | Punktzahl |
| Findet den Parameter, wenn man dessen Namen kennt | 25 / 25 |
| Findet den Parameter, wenn man dessen Wert kennt | 25 / 25 |
| Findet den Parameter, wenn man weiss, auf welchem Service definiert ist | 25 / 25 |
| Geringe Komplexität der Suche | 15 / 25 |
| **Total** | 90 / 100 |

### Umsetzungsbeschreibung

Die Volltextsuche auf Parameternamen und Parameterwert scheint am Sinnvollsten, da man mit der Volltextsuche über alles nach «max.muster» z.B. auch einen Parameter bekommt, bei dem der Default Wert die E-Mailadresse «max.muster@example.com» wäre. Dies möchte man aber wahrscheinlich nicht finden. Eine Serverseitige suche ist hier nicht notwendig, da kein Paging existiert. So kann man einfach auf dem Client die Daten neu anzeigen lassen. Optional kann bei genügend Zeit noch die ganze Parametergruppe angezeigt werden. Dies setzt jedoch voraus, dass genug Zeit für die Parametergruppe bereits gefunden wurde. Ansonst werden nur die Einzelnen Parameter in einer Liste angezeigt.

## Erstellung der Unittests

### Anforderung

Die Unittests müssen sämtliche Methoden testen, die eine Datenmanipulation vornehmen. Die Weitergabe von Daten via RabbitMQ/MassTransit kann leider nicht getestet werden.

### Umsetzungsbeschreibung

Nur Serverseitig müssen Unittests geschrieben werden, da Clientseitig nur e2e Tests Sinn machen. Unittests sind zu wenig aussagekräftig weshalb sich der Aufwand nicht lohnt. Die einzige Clientseitig mit Unittests überprüfte Logik ist die der Clientservices. Da aber keine neuen Services dazugekommen sind im Rahmen dieser Projektarbeit, gibt es keine mit Tests abzudeckenden Clientlogik.

## Durchführen der Tests

### Anforderung

Die Tests müssen nach jedem beendeten Realisierungstask durchgeführt werden und Protokolliert werden.

### Umsetzungsbeschreibung

Siehe Kapitel Testergebnisse.

## Getting Started Dokumentation

### Anforderung

Die Anforderungen an diese Dokumentation sind aus der Aufgabenstellung her genau definiert.

|  |
| --- |
| Anforderungen |
| Schritt für Schritt Anleitung für Einbindung |
| Mind. 2 Beispiele für die Validierung |
| Mind. 2 Screenshots der resultierenden Darstellung im GUI |
| Bietet einen konzeptionellen Überblick |

### Umsetzungsbeschreibung

Das Getting Started ist sinnvollerweise ein separates Word, welches man dann so ablegen kann. Für die Projektdokumentation wird eine exakte Kopie des Getting Started auch noch in die Dokumentation eingefügt. Die Getting Started Dokumentation ist.