|  |
| --- |
|  |

|  |
| --- |
|  |

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. |
| 1.0 | 06.04.2018 | Remo Kessler | Abgeschlossen | Fertige IPA Dokumentation |

Referenzierte Dokumente

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

Verteiler & Projektorganisation

|  |  |  |  |
| --- | --- | --- | --- |
| Name / Vorname | Kurzzeichen | Organisation | Rolle |
| Remo Kessler | KRE | CM Informatik AG | Lehrling Informatik |
| Martin Tinner | TMA | CM Informatik AG | IPA Betreuer |
| Matthias Hess | MHE | CM Informatik AG | IPA Fachbetreuer |
| Patrick Schättin | - | Noser Engineering | IPA Hauptexperte |
| Reto Loser | - | Swisscom | IPA Nebenexperte |

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. |

Inhaltsverzeichnis

[1 Zweck des Dokuments 12](#_Toc510789208)

[1.1 Thema und Zielsetzung 12](#_Toc510789209)

[1.2 Allgemeines zur Umgebung 12](#_Toc510789210)

[1.3 Endprodukt 12](#_Toc510789211)

[2 Rahmenbedingungen 13](#_Toc510789212)

[2.1 Titel der IPA 13](#_Toc510789213)

[2.2 Auslöser der Aufgabenstellung 13](#_Toc510789214)

[2.3 Grobanforderungen 13](#_Toc510789215)

[2.3.1 Parameterverwaltung 13](#_Toc510789216)

[2.3.1.1 Funktionale Anforderungen 13](#_Toc510789217)

[2.3.1.2 Nicht Funktionale Anforderungen 13](#_Toc510789218)

[2.3.2 «Getting Started» Dokumentation 14](#_Toc510789219)

[2.4 Vorkenntnisse 14](#_Toc510789220)

[2.5 Neue Lerninhalte 14](#_Toc510789221)

[2.6 Mittel & Methoden 14](#_Toc510789222)

[2.6.1 Eingesetzte Mittel 14](#_Toc510789223)

[2.6.2 Git Workflow 14](#_Toc510789224)

[2.6.3 Daily Business 14](#_Toc510789225)

[2.6.4 Dokumentablage 15](#_Toc510789226)

[2.6.5 Kontroll-Tasks 17](#_Toc510789227)

[3 Allgemeines 18](#_Toc510789228)

[3.1 Konventionen 18](#_Toc510789229)

[3.1.1 Dokumentation 18](#_Toc510789230)

[3.1.2 Code-Konventionen C# 18](#_Toc510789231)

[3.1.3 Code-Konventionen TypeScript 18](#_Toc510789232)

[3.1.4 Layout-Konventionen 19](#_Toc510789233)

[3.1.5 Unittests-Konventionen 19](#_Toc510789234)

[3.2 Vorgehensmodell 19](#_Toc510789235)

[3.2.1 IPERKA 19](#_Toc510789236)

[4 Umsysteme, Abhängigkeiten, Systemgrenzen und Schnittstellen 20](#_Toc510789237)

[4.1 Umsysteme 20](#_Toc510789238)

[4.2 Abhängigkeiten 20](#_Toc510789239)

[4.3 Systemgrenzen 21](#_Toc510789240)

[4.4 Schnittstellen 22](#_Toc510789241)

[5 Tagesjournale 23](#_Toc510789242)

[5.1 Tagesjournal vom 19.03.2018 23](#_Toc510789243)

[5.2 Tagesjournal vom 20.03.2018 24](#_Toc510789244)

[5.3 Tagesjournal vom 22.03.2018 25](#_Toc510789245)

[5.4 Tagesjournal vom 23.03.2018 26](#_Toc510789246)

[5.5 Tagesjournal vom 26.03.2018 27](#_Toc510789247)

[5.6 Tagesjournal vom 27.03.2018 28](#_Toc510789248)

[5.7 Tagesjournal vom 29.03.2018 29](#_Toc510789249)

[5.8 Tagesjournal vom 03.04.2018 30](#_Toc510789250)

[5.9 Tagesjournal vom 05.04.2018 31](#_Toc510789251)

[5.10 Tagesjournal vom 06.05.2018 32](#_Toc510789252)

[5.11 Allgemeines zum Tagesjournal 33](#_Toc510789253)

[6 Taskerstellung nach den Anforderungen 34](#_Toc510789254)

[6.1 Einzeltasks für die Dokumentation 35](#_Toc510789255)

[6.1.1 Getting Started Dokumentation 35](#_Toc510789256)

[6.2 Einzeltasks für die Realisierung 35](#_Toc510789257)

[6.2.1 Anzeige & Speichern der Parameter 35](#_Toc510789258)

[6.2.2 Implementation des Validierungsmechanismus 36](#_Toc510789259)

[6.2.3 Suchen eines Parameters 36](#_Toc510789260)

[6.2.4 Erstellung der Unit Tests und des Testkonzepts 36](#_Toc510789261)

[6.3 Einzeltaskerstellung der Kontrolltasks 37](#_Toc510789262)

[6.3.1 Durchführen der Tests 37](#_Toc510789263)

[7 Zeitplan & Burn down 38](#_Toc510789264)

[7.1 Zeitplan 38](#_Toc510789265)

[7.2 Burn down 40](#_Toc510789266)

[8 Entscheide der einzelnen Tasks 41](#_Toc510789267)

[8.1 Umsetzungsreihenfolge 41](#_Toc510789268)

[8.2 Erstellung des Testkonzepts 41](#_Toc510789269)

[8.2.1 Anforderung 41](#_Toc510789270)

[8.2.2 Mögliche Lösungen 41](#_Toc510789271)

[8.2.2.1 Unittests 41](#_Toc510789272)

[8.2.2.2 Integration Tests / e2e Tests 42](#_Toc510789273)

[8.2.2.3 User-Testing 42](#_Toc510789274)

[8.2.3 Umsetzungsbeschreibung 42](#_Toc510789275)

[8.3 Anzeige & Speichern der Parameter 43](#_Toc510789276)

[8.3.1 Anforderung 43](#_Toc510789277)

[8.3.2 Mögliche Lösungen Speichern 43](#_Toc510789278)

[8.3.2.1 Speichern der Parameter als Parametertyp im Json 43](#_Toc510789279)

[8.3.2.2 Speichern der Parameter als generischer Typ im Json 43](#_Toc510789280)

[8.3.3 Umsetzungsbeschreibung Speichern 44](#_Toc510789281)

[8.3.4 Mögliche Lösung Anzeige 44](#_Toc510789282)

[8.3.4.1 Ähnlich wie in Firefox / Waterfox die about:config 44](#_Toc510789283)

[8.3.4.2 Ähnlich wie die Chrome Settings 45](#_Toc510789284)

[8.3.4.3 Ähnlich wie in Visual Studio 46](#_Toc510789285)

[8.3.5 Umsetzungsbeschreibung Anzeige 47](#_Toc510789286)

[8.4 Implementation des Validierungsmechanismus 48](#_Toc510789287)

[8.4.1 Anforderung 48](#_Toc510789288)

[8.4.2 Mögliche Lösungen 48](#_Toc510789289)

[8.4.2.1 Nur serverseitig im Service den Parameter selbst testen lassen 48](#_Toc510789290)

[8.4.2.2 Regular Expression im Parameter 48](#_Toc510789291)

[8.4.3 Umsetzungsbeschreibung 48](#_Toc510789292)

[8.5 Suche eines Parameters 49](#_Toc510789293)

[8.5.1 Anforderung 49](#_Toc510789294)

[8.5.2 Mögliche Lösungen 49](#_Toc510789295)

[8.5.2.1 Volltextsuche über alles 49](#_Toc510789296)

[8.5.2.2 Suche auf die Parameternamen 49](#_Toc510789297)

[8.5.2.3 Suche auf Parameternamen und Parameterwert 50](#_Toc510789298)

[8.5.3 Umsetzungsbeschreibung 50](#_Toc510789299)

[8.6 Erstellung der Unittests 50](#_Toc510789300)

[8.6.1 Anforderung 50](#_Toc510789301)

[8.6.2 Umsetzungsbeschreibung 50](#_Toc510789302)

[8.7 Durchführen der Tests 50](#_Toc510789303)

[8.7.1 Anforderung 50](#_Toc510789304)

[8.7.2 Umsetzungsbeschreibung 50](#_Toc510789305)

[8.8 Getting Started Dokumentation 51](#_Toc510789306)

[8.8.1 Anforderung 51](#_Toc510789307)

[8.8.2 Umsetzungsbeschreibung 51](#_Toc510789308)

[9 Realisierung 52](#_Toc510789309)

[9.1 Erstellung des Testkonzepts 52](#_Toc510789310)

[9.2 Anzeige & Speichern der Parameter 53](#_Toc510789311)

[9.2.1 Frontend 53](#_Toc510789312)

[9.2.2 Backend 54](#_Toc510789313)

[9.2.3 Technische Umsetzung Backend 54](#_Toc510789314)

[9.2.4 Erstellte und Bearbeitete Klassen 55](#_Toc510789315)

[9.3 Implementation des Validierungsmechanismus 56](#_Toc510789316)

[9.3.1 Erstellte und bearbeitete Klassen 56](#_Toc510789317)

[9.4 Suche eines Parameters 57](#_Toc510789318)

[9.4.1 Erstellte und bearbeitete Klassen 57](#_Toc510789319)

[9.5 Schreiben der Unittests 58](#_Toc510789320)

[9.5.1 ParameterSerializerTests 58](#_Toc510789321)

[9.5.2 ParameterValidationTests 59](#_Toc510789322)

[9.5.3 Erstellte und bearbeitete Klassen 59](#_Toc510789323)

[10 Getting Started mit der Parameterverwaltung 60](#_Toc510789324)

[10.1 Einen eigenen Parameter schreiben 60](#_Toc510789325)

[10.1.1 Ein Setting existiert und nur ein Parameter soll hinzugefügt werden 60](#_Toc510789326)

[10.1.2 Kein Setting existiert 60](#_Toc510789327)

[10.2 Einen Parameter/ein Setting auslesen 61](#_Toc510789328)

[10.3 Beispielsparameter 61](#_Toc510789329)

[10.3.1 Unterstützte Parametertypen 61](#_Toc510789330)

[10.3.2 Unterstützte Attributtypen 61](#_Toc510789331)

[10.4 Resultat: 62](#_Toc510789332)

[11 Testing 63](#_Toc510789333)

[11.1 Testdurchgang nach Task Anzeige & Speichern der Parameter 63](#_Toc510789334)

[11.2 Testdurchgang nach Task Implementation des Validierungsmechanismus 64](#_Toc510789335)

[11.3 Testdurchgang nach Task Suche eines Parameters 65](#_Toc510789336)

[11.4 Testdurchgang nach Task Erstellung der Unittests 65](#_Toc510789337)

[12 Code Dokumentation Backend 66](#_Toc510789338)

[12.1 Assembly: CMI.Contract.Parameter 66](#_Toc510789339)

[12.1.1 Klasse: ParameterHelper 66](#_Toc510789340)

[12.1.1.1 Methode: GetParameterListFromSetting 66](#_Toc510789341)

[12.1.1.2 Methode: ValidateParameter 67](#_Toc510789342)

[12.1.1.3 Methode: SaveSetting 68](#_Toc510789343)

[12.1.1.4 Methode: GetSetting 69](#_Toc510789344)

[12.1.1.5 Methode: InitialSaveParameter 69](#_Toc510789345)

[12.1.1.6 Methode: GetJsonStringOfSetting 70](#_Toc510789346)

[12.1.1.7 Methode: GetSettingPath 70](#_Toc510789347)

[12.1.1.8 Methode: CreateParameter 71](#_Toc510789348)

[12.1.1.9 Methode: GetType 72](#_Toc510789349)

[12.1.2 Klasse: ParameterBusHelper 72](#_Toc510789350)

[12.1.2.1 Methode: SubscribeGetEvent 72](#_Toc510789351)

[12.1.2.2 Methode: SubscribeSaveEvent 73](#_Toc510789352)

[12.1.3 Klasse: Parameter 73](#_Toc510789353)

[12.1.3.1 Property: Name 73](#_Toc510789354)

[12.1.3.2 Property: Type 73](#_Toc510789355)

[12.1.3.3 Property: Description 74](#_Toc510789356)

[12.1.3.4 Property: Value 74](#_Toc510789357)

[12.1.3.5 Property: Default 74](#_Toc510789358)

[12.1.3.6 Property: RegexValidation 74](#_Toc510789359)

[12.1.3.7 Property: Mandatory 74](#_Toc510789360)

[12.1.4 Interface: ISetting 74](#_Toc510789361)

[12.1.5 Klasse: BusConfigurator 74](#_Toc510789362)

[12.1.6 Ordner: Attributes 75](#_Toc510789363)

[12.1.6.1 Klasse: DefaultAttribute 75](#_Toc510789364)

[12.1.6.2 Klasse: DescriptionAttribute 75](#_Toc510789365)

[12.1.6.3 Klasse: MandatoryAttribute 75](#_Toc510789366)

[12.1.6.4 Klasse: ValidationAttribute 76](#_Toc510789367)

[12.1.7 Ordner: GetParameter 76](#_Toc510789368)

[12.1.7.1 Klasse: GetParameterEvent 76](#_Toc510789369)

[12.1.7.2 Klasse: GetParameterEventResponse 76](#_Toc510789370)

[12.1.7.3 Klasse: GetParameterRequest 76](#_Toc510789371)

[12.1.7.4 Klasse: GetParameterResponse 77](#_Toc510789372)

[12.1.8 Ordner: SaveParameter 77](#_Toc510789373)

[12.1.8.1 Klasse: SaveParameterEvent 77](#_Toc510789374)

[12.1.8.2 Klasse: SaveParameterEventResponse 77](#_Toc510789375)

[12.1.8.3 Klasse: SaveParameterRequest 78](#_Toc510789376)

[12.1.8.4 Klasse: SaveParameterResponse 78](#_Toc510789377)

[12.2 Assembly: CMI.Host.ExampleServiceA 78](#_Toc510789378)

[12.3 Assembly: CMI.Host.ExampleServiceB 78](#_Toc510789379)

[12.4 Assembly: CMI.Host.Parameter 78](#_Toc510789380)

[12.5 Assembly: CMI.Manager.ExampleServiceA 79](#_Toc510789381)

[12.5.1 Klasse: ExampleServiceA 79](#_Toc510789382)

[12.5.1.1 Property: ParameterBus 79](#_Toc510789383)

[12.5.1.2 Methode: Start 79](#_Toc510789384)

[12.5.1.3 Methode: Stop 80](#_Toc510789385)

[12.5.2 Klasse: ExampleSettingA 80](#_Toc510789386)

[12.5.2.1 Property: Date 80](#_Toc510789387)

[12.5.2.2 Property: EMailAdress 80](#_Toc510789388)

[12.5.2.3 Property: ServiceOn 80](#_Toc510789389)

[12.6 Assembly: CMI.Manager.ExampleServiceB 81](#_Toc510789390)

[12.6.1 Klasse: ExampleServiceB 81](#_Toc510789391)

[12.6.1.1 Property: ParameterBus 81](#_Toc510789392)

[12.6.1.2 Methode: Start 81](#_Toc510789393)

[12.6.1.3 Methode: Stop 82](#_Toc510789394)

[12.6.2 Klasse: ExampleSettingB 82](#_Toc510789395)

[12.6.2.1 Property: EndeDatum 82](#_Toc510789396)

[12.6.2.2 Property: EMailAdress 82](#_Toc510789397)

[12.6.2.3 Property: FehlerVerstecken 82](#_Toc510789398)

[12.6.2.4 Property: Sekunden 82](#_Toc510789399)

[12.7 Assembly: CMI.Manager.Parameter 83](#_Toc510789400)

[12.7.1 Klasse: GetParameterEventResponseConsumer 83](#_Toc510789401)

[12.7.1.1 Methode: Consume 83](#_Toc510789402)

[12.7.2 Klasse: GetParameterRequestConsumer 83](#_Toc510789403)

[12.7.2.1 Methode: Consume 83](#_Toc510789404)

[12.7.3 Klasse: ParameterRequestResponseHelper 84](#_Toc510789405)

[12.7.3.1 Property: Parameters 84](#_Toc510789406)

[12.7.3.2 Property: SavedSuccessfully 84](#_Toc510789407)

[12.7.4 Klasse: ParameterService 84](#_Toc510789408)

[12.7.4.1 Property: ParameterBus 84](#_Toc510789409)

[12.7.4.2 Methode: Start 85](#_Toc510789410)

[12.7.4.3 Methode: Stop 85](#_Toc510789411)

[12.7.5 Klasse: SaveParameterEventResponseConsumer 86](#_Toc510789412)

[12.7.5.1 Methode: Consume 86](#_Toc510789413)

[12.7.6 Klasse: SaveParameterRequestConsumer 87](#_Toc510789414)

[12.7.6.1 Consume 87](#_Toc510789415)

[12.8 Assembly: CMI.Web.Management 88](#_Toc510789416)

[12.8.1 Ordner: Controllers 88](#_Toc510789417)

[12.8.1.1 Klasse: ParameterController 88](#_Toc510789418)

[12.8.1.1.1 Methode: GetAllParameters 88](#_Toc510789419)

[12.8.1.1.2 Methode: SaveParameter 88](#_Toc510789420)

[12.8.2 Ordner: Helpers 89](#_Toc510789421)

[12.8.2.1 Klasse: BusHelper 89](#_Toc510789422)

[12.8.2.1.1 Property: ParameterBus 89](#_Toc510789423)

[12.8.2.1.2 Klasse: BusHelper 89](#_Toc510789424)

[12.9 Assembly: CMI.Contract.Parameter.Tests 89](#_Toc510789425)

[12.9.1 Klasse: TestSetting 89](#_Toc510789426)

[12.9.1.1 Property: TestFlag 89](#_Toc510789427)

[12.9.1.1.1 Property: TestMailAdress 89](#_Toc510789428)

[12.9.1.2 Property: TestDate 90](#_Toc510789429)

[12.9.1.3 Property: TestNumber 90](#_Toc510789430)

[12.9.1.4 Property: TestUnknownType 90](#_Toc510789431)

[12.9.1.5 Property: TestMandatory 90](#_Toc510789432)

[12.9.2 ParameterSerializerTests 90](#_Toc510789433)

[12.9.2.1 Methode: An\_empty\_setting\_can\_be\_serialized 90](#_Toc510789434)

[12.9.2.2 Methode: Name\_can\_be\_serialized\_correclty 91](#_Toc510789435)

[12.9.2.3 Methode: Type\_can\_be\_serialized\_correctly 91](#_Toc510789436)

[12.9.2.4 Methode: Default\_can\_be\_serialized\_correctly 92](#_Toc510789437)

[12.9.2.5 Methode: Value\_can\_be\_serialized\_correctly 92](#_Toc510789438)

[12.9.2.6 Methode: Description\_can\_be\_serialized\_correctly 93](#_Toc510789439)

[12.9.2.7 Methode: Mandatory\_can\_be\_serialized\_correctly 93](#_Toc510789440)

[12.9.2.8 Methode: GetSettingPath 93](#_Toc510789441)

[12.9.2.9 Methode: Validation\_can\_be\_serialized\_correctly 94](#_Toc510789442)

[12.9.2.10 Methode: Can\_be\_serialized\_saved\_and\_can\_then\_be\_deserialized\_and\_got 95](#_Toc510789443)

[12.9.3 Klasse: ParameterValidationTests 96](#_Toc510789444)

[12.9.3.1 Methode: ParameterList\_Should\_be\_valid 96](#_Toc510789445)

[12.9.3.2 Methode: Empty\_value\_and\_mandatory\_should\_not\_be\_valid 96](#_Toc510789446)

[12.9.3.3 Methode: None\_empty\_value\_and\_mandatory\_should\_be\_valid 97](#_Toc510789447)

[12.9.3.4 Methode: None\_empty\_value\_that\_is\_not\_conform\_should\_not\_be\_valide 97](#_Toc510789448)

[12.9.3.5 Methode: None\_empty\_value\_that\_is\_conform\_should\_be\_valide 97](#_Toc510789449)

[12.9.3.6 Methode: Empty\_mandatory\_value\_with\_regex\_defined\_should\_not\_be\_valide 98](#_Toc510789450)

[12.9.3.7 Methode: None\_empty\_mandatory\_conform\_value\_with\_regex\_defined\_is\_valide 98](#_Toc510789451)

[12.9.3.8 Methode: None\_empty\_unconform\_mandatory\_value\_with\_regex\_defined\_is\_not\_valide 98](#_Toc510789452)

[13 Code Dokumentation Frontend 99](#_Toc510789453)

[13.1 Ordner: services 99](#_Toc510789454)

[13.1.1 Klasse: HttpService 99](#_Toc510789455)

[13.1.2 Klasse: ParameterService 99](#_Toc510789456)

[13.1.2.1 Methode: getAllParameters 99](#_Toc510789457)

[13.1.2.2 Methode: saveParameter 99](#_Toc510789458)

[13.1.2.3 Methode: \_createBaseUrl 100](#_Toc510789459)

[13.2 Ordner: highlight 100](#_Toc510789460)

[13.2.1 Klasse: HighlightComponent 100](#_Toc510789461)

[13.2.1.1 Property: highlight 100](#_Toc510789462)

[13.2.1.2 Property: text 100](#_Toc510789463)

[13.2.1.3 Methode: constructor 100](#_Toc510789464)

[13.2.1.4 Methode: getInnerHTML 101](#_Toc510789465)

[13.2.2 HTML: HighlightComponent 101](#_Toc510789466)

[13.3 Ordner: parameterManager 101](#_Toc510789467)

[13.3.1 Klasse: Parameter 101](#_Toc510789468)

[13.3.1.1 Property: name 101](#_Toc510789469)

[13.3.1.2 Property: value 102](#_Toc510789470)

[13.3.1.3 Property: type 102](#_Toc510789471)

[13.3.1.4 Property: mandatory 102](#_Toc510789472)

[13.3.1.5 Property: description 102](#_Toc510789473)

[13.3.1.6 Property: regexValidation 102](#_Toc510789474)

[13.3.1.7 Property: default 102](#_Toc510789475)

[13.4 Ordner: parameter 103](#_Toc510789476)

[13.4.1 Klasse: ParameterComponent 103](#_Toc510789477)

[13.4.1.1 Property: parameter 103](#_Toc510789478)

[13.4.1.2 Property: validationEvent 103](#_Toc510789479)

[13.4.1.3 Property: searchString 103](#_Toc510789480)

[13.4.1.4 Property: active 103](#_Toc510789481)

[13.4.1.5 Property: value 103](#_Toc510789482)

[13.4.1.6 Property: checked 104](#_Toc510789483)

[13.4.1.7 Property: validationError 104](#_Toc510789484)

[13.4.1.8 Property: \_onFocusChange 104](#_Toc510789485)

[13.4.1.9 Methode: constructor 104](#_Toc510789486)

[13.4.1.10 Methode: ngOnInit 105](#_Toc510789487)

[13.4.1.11 Methode: onValueChanged 105](#_Toc510789488)

[13.4.1.12 Methode: onFocus 106](#_Toc510789489)

[13.4.1.13 Methode: saveParameter 106](#_Toc510789490)

[13.4.1.14 Methode: cancelEdit 106](#_Toc510789491)

[13.4.1.15 Methode: \_isValid 107](#_Toc510789492)

[13.4.1.16 Methode: \_validateString 107](#_Toc510789493)

[13.4.1.17 Methode: getErrorClass 108](#_Toc510789494)

[13.4.1.18 Methode: getInputClass 108](#_Toc510789495)

[13.4.2 HTML: ParameterComponent 109](#_Toc510789496)

[13.4.3 Less: ParameterComponent 110](#_Toc510789497)

[13.5 Ordner: ParameterList 111](#_Toc510789498)

[13.5.1 Klasse: ParameterListComponent 111](#_Toc510789499)

[13.5.1.1 Property: loading 111](#_Toc510789500)

[13.5.1.2 Property: filteredParameters 111](#_Toc510789501)

[13.5.1.3 Property: \_allParameters 111](#_Toc510789502)

[13.5.1.4 Property: validationEvent 111](#_Toc510789503)

[13.5.1.5 Property: searchString 111](#_Toc510789504)

[13.5.1.6 Property: searchedStringUpToDate 112](#_Toc510789505)

[13.5.1.7 Methode: constructor 112](#_Toc510789506)

[13.5.1.8 Methode: getAllParameters 112](#_Toc510789507)

[13.5.1.9 Methode: onValueChanged 112](#_Toc510789508)

[13.5.1.10 Methode: emitValidationEvent 113](#_Toc510789509)

[13.5.1.11 Methode: searchParam 113](#_Toc510789510)

[13.5.2 HTML: ParameterListComponent 114](#_Toc510789511)

[13.5.3 Less: ParameterListComponent 115](#_Toc510789512)

[14 Schlusswort 116](#_Toc510789513)

[14.1 Ausblick auf nach der Arbeit 116](#_Toc510789514)

[14.2 Reflexion 116](#_Toc510789515)

[14.3 Danksagung 116](#_Toc510789516)

[15 Verzeichnisse 117](#_Toc510789517)

[15.1 Glossar 117](#_Toc510789518)

[15.2 Tabellenverzeichnis 119](#_Toc510789519)

[15.3 Bildverzeichnis 121](#_Toc510789520)

[15.4 Quellenverzeichnis 122](#_Toc510789521)

# 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.

## Allgemeines zur Umgebung

Das Projekt wird in einem eigenen Repository unabhängig vom restlichen Kundenprojekt 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 Visual Studio respektive Webstorm 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 die hohen und komplexen Anforderungen an den Backendbereich, entschloss man sich dieses mit einer Microservicearchitektur umzusetzen. Dies hat zur Folge, dass die Parametrierung dieser Services nicht zentral gespeichert werden kann, da sonst die Eigenständigkeit der Services nicht mehr gewährleistet wird, welches entgegen des Grundsatzes der Microservices-Architektur wäre, dass jeder Service unabhängig von einem anderen laufen kann. Wenn dies der Fall ist, hat man einen der grössten Vorteile der Microservices-Architektur.

## 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
* Defaultwerte 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 Parameterverwaltung 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 nicht läuft
* Die PV funktioniert rechnerübergreifend, ohne File-Sharing
* Es werden sprechende Namen für Klassen / Methoden verwendet
* Die Einbindung der Parameterverwaltung soll mit der Angabe des Parametertyps und einem Minimum von weiterem Code möglich sein.
* 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

## Vorkenntnisse

* C# Programmierung während der Lehre bei der Greenshare AG / CM Informatik AG
* 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

## Mittel & Methoden

### Eingesetzte Mittel

* Visual Studio für die Backendprogrammierung in C#
* Webstorm für die Frontendprogrammierung in TypeScript
* Microsoft Office für die Erstellung der Dokumentation
* officeatwork Vorlagen, um die Layoutrichtlinien der Dokumentation einzuhalten
* 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 Datensicherung wird dennoch jeder Commit auch gleich gepusht. Zudem wird es jeden Tag eine Version geben.

### Daily Business

Am Ende jedes Tages gibt es den Tagesabschluss. Bei diesem wird das Arbeits- und Tagesjournal geschrieben. Nachdem das Journal fertig verfasst ist, wird es Martin Tinner, dem IPA Betreuer, gegeben. Er unterschreibt das Tagesjournal und gibt es mir wieder retour. Anschliessend mache ich ein Rundmail, in dem ich das Tagesjournal an Martin Tinner, Matthias Hess (technischer Betreuer) und Marco Zollinger (Projektleiter des Viaduc Projekts) schicke.

### Dokumentablage

|  |  |
| --- | --- |
| Abbildung 1 - Struktur der Dokumentablage | Client  In diesem Ordner befindet sich das Webstorm Projekt für den Client-Teil der IPA. Dieser liegt in derselben Struktur wie das Livesystem. |
| **Doc**  Hier drin befindet sich die Dokumentation der IPA, der Zeitplan, die Kriterien und die Vorlage für das Tagesjournal. |
| **Arbeits- & Tagesjournal**  Der Ordner beinhaltet sämtliche ausgefüllte Arbeits- und Tagesjournale. |
| **Diagramme**  Der Ordner enthält eine Kopie jedes Diagramms als PNG. |
| **Screenshot**  Sämtliche Screenshots, die in der Doku verwendet werden, werden auch in diesem Ordner abgelegt. |
| **Server**  In diesem Ordner liegt das ganze Visual Studio Projekt des Backend. |

Tabelle 1 – Dokumentablage

|  |  |
| --- | --- |
| Webstorms Projekt (Frontend) | Visual Studio Projekt (Backend) |
| C:\Users\kre.CMIAG\Downloads\Frontend Folders (1).png  **Abbildung 2 - Frontend Ordnerstruktur** | C:\Users\kre.CMIAG\Downloads\Backend Assemblies (1).png  **Abbildung 3 - Backend Ordnerstruktur** |

Tabelle 2 - Source Code Ordnerstruktur

### Kontroll-Tasks

* Nach den IPERKA Phasen planen, entscheiden und realisieren wird das Dokument zum Gegenlesen gegeben. Am anschliessenden Tag wird jeweils eine Stunde eingerechnet, um zu besprechen und allfällige Fehler zu beheben.
* Jeweils am Ende einer Woche, wird die Anforderungsliste mit dem Ist-Stand verglichen.

# Allgemeines

## Konventionen

### Dokumentation

|  |  |
| --- | --- |
| Bereich | Vorgabe |
| Seitenumbruch | Alle Überschriften erster Stufe beginnen auf einer neuen Seite. Keine Überschrift steht ganz unten auf einer Seite. |
| Dokumentvorlage | Von officeatwork zu entnehmen |

Tabelle 3 - Konventionen Dokumentation

### Code-Konventionen C#

|  |  |
| --- | --- |
| Bereich | Vorgaben |
| Namensgebung Methoden/Variablen | Methoden- und Variablennamen sollen 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 |

Tabelle 4 - Code-Konventionen C#

### Code-Konventionen TypeScript

|  |  |
| --- | --- |
| Bereich | Vorgaben |
| Namensgebung Methoden/Variablen | Methoden- und Variablennamen sollen 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 anzulegen  KomponentenName.component.(ts | html | less)  Services sind im Services Verzeichnis abzulegen und via Dependency Injection mit Angular anzusteuern. |

Tabelle 5 - Code Konventionen TypeScript

### Layout-Konventionen

|  |  |
| --- | --- |
| Bereich | Vorgabe |
| Layoutrichtlinien | Richtlinien des Bundes sind einzuhalten. Sagt dieses nichts aus, ist auf Bootstrap zurückzugreifen. Ist da ebenfalls nichts definiert, so muss selbst ein Vorschlag gemacht werden. |
| Barrierefreiheit | Es muss alles mit der Tastatur bedienbar sein. |

Tabelle 6 - Layout-Konventionen

### Unittests-Konventionen

|  |  |
| --- | --- |
| Bereich | Vorgabe |
| Assemblyname | Die Testassembly muss gleich wie die normale Assembly heissen mit der Endung «.Tests» als Suffix. |
| Methodennamen | Upper Snake Case |

Tabelle 7 - Unittests-Konventionen

## 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

Abbildung 4 - IPERKA

Das gewählte Arbeitsmodell IPERKA bietet sich für diese Projektarbeit an, da sie einen grossen Wert auf die Dokumentation legt. Dadurch, dass die Dokumentation bei dieser Arbeit im Vordergrund steht, kommt diese Stärke hier sehr gut zum Tragen. Die sechs Phasen von IPERKA gehen ineinander 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 statt. Da diese aber in der IPA nicht enthalten sind, kommuniziert das System mit keinem Umsystem. In untenstehendem Diagramm sehen wir die Microservices, welche im Rahmen der IPA umgesetzt werden. Sie sind Dummy-Services und bieten verschiedene Parameter an.

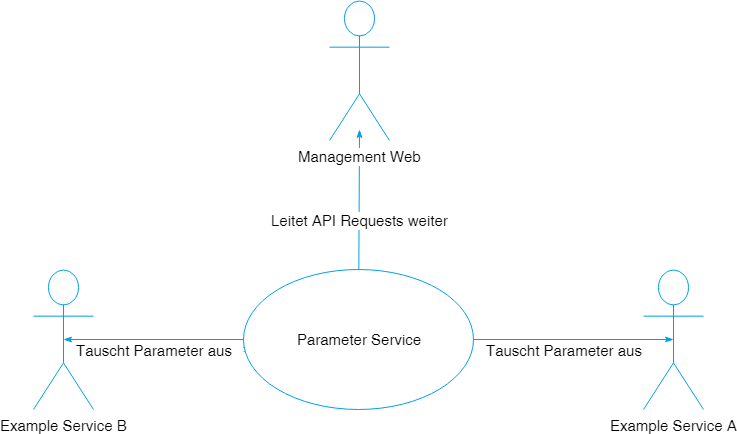


Abbildung 5 - Umsysteme

## 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 Ort, an welchem sich die APIs befinden |
| Parameter Service | Der Service, um welchen sich die IPA dreht |
| Example Service A & B | Diese zwei Services stellen zwei beliebige Viaduc Microservices dar |
| 1 | API Call und Response |
| 2, 3, 4 | RabbitMQ / MassTransit Kommunikation |

Tabelle 8 - Systemgrenze Erklärungen Diagramm

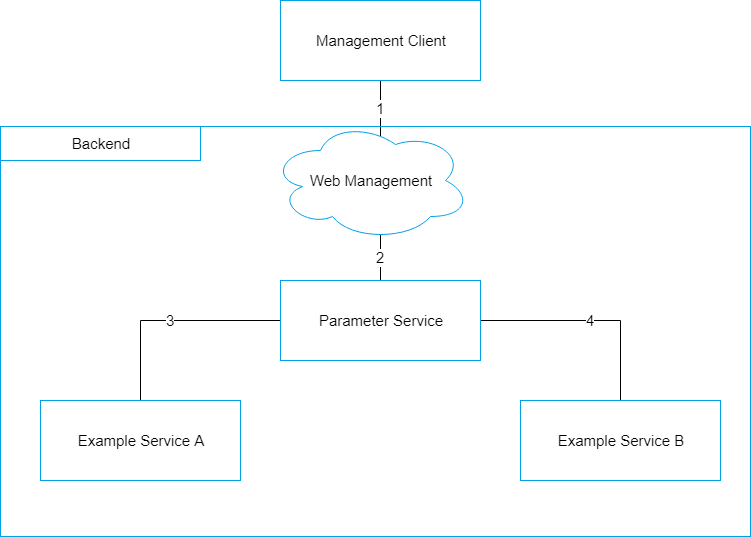


Abbildung 6 - Systemgrenzen

## 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 vorzustellen:

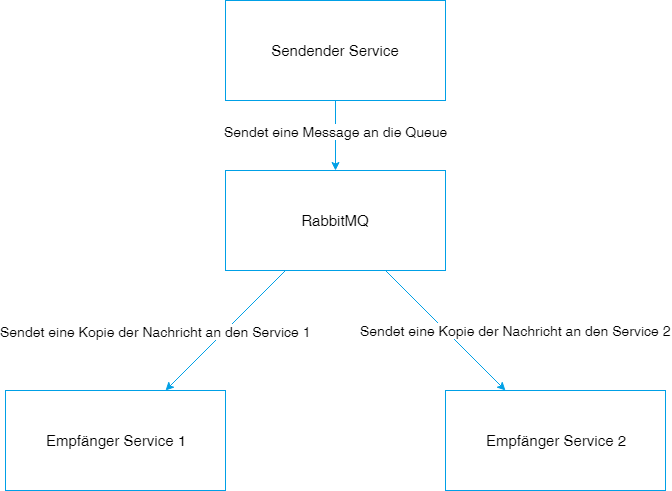


Abbildung 7 - RabbitMQ Event Kommunikation

Dieses Diagramm zeigt die Event-Kommunikation von RabbitMQ und MassTransit auf. Der Sender Service gibt Messages in die Queue und von der Queue aus gelangt dann eine Kopie der Nachricht an jeden Subscriber(Empfänger Service) gesendet. Einmal erfolgreich abgearbeitet, werden die Messages aus der Queue entfernt.

# Tagesjournale

## Tagesjournal vom 19.03.2018

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Geplante Tagesziele | | | | | |
| **Nr.** | **Ziel** | **Soll** | **Ist** | **Abweichung** | **Abgeschlossen** |
| 1 | Aufgabenbeschreibung und Allgemeines beschreiben | 4 | 4 | 0 | Ja |
| 2 | Umsystem und Abhängigkeiten beschreiben | 1.5 | 1.5 | 0 | Nein |
| 3 | Zeitplanerstellung / Taskerstellung | 2 | 2 | 0 | Ja |

Tabelle 9 - Geplante Tagesziele Tag 1

|  |  |  |  |
| --- | --- | --- | --- |
| Tätigkeit | Zeit | Aufwand | Ziel |
| Dokument erstellt, Aufgabenbeschreibungen erstellt | 8.15 – 11.45  12.15-12.45 | 4 | 1 |
| Umsysteme und Abhängigkeiten beschrieben | 12.45 – 14.15 | 1.5 | 2 |
| Tasks erstellt | 14.30 – 15.30 | 1 | 3 |
| Zeitplan erstellt | 15.30 – 16.30 | 1 | 3 |
| Tagesabschluss | 16.30 – 17.00 | 0.5 | - |

Tabelle 10 - Tätigkeiten Tag 1

|  |
| --- |
| Probleme |
| Kriterium «eine Beispielimplementierung im Projekt Viaduc ist vorhanden und einsehbar» warf noch eine Frage auf. Siehe Hilfestellung. Des Weiteren scheint unsere Office at Work Vorlage noch einen kleinen Bug zu haben, welcher verhindert, dass Überschriften erster Stufe im Inhaltsverzeichnis erscheinen. Habe diesen Fehler unserer Expertin Melanie Müller geschickt und hoffe, ihn mit ihr diese Woche noch beheben zu können. |

Tabelle 11 - Probleme Tag 1

|  |
| --- |
| Hilfestellung |
| Ich habe Matthias Hess gefragt, ob ich das Kriterium «eine Beispielimplementierung im Projekt Viaduc ist vorhanden und einsehbar» etwas komisch formuliert ist, denn es lässt vermuten, dass die IPA dann auch im Livesystem umgesetzt werden müsste. Dem ist jedoch nicht der Fall. Ich habe mir notiert, dies noch explizit mit Herr Schättin am Donnerstag an zu schauen, wenn er einen Besuch machen kommt. |

Tabelle 12 - Hilfestellung Tag 1

|  |
| --- |
| Tagesreflexion |
| Ich bin überraschend gut vorangekommen. Für die zwei Probleme, denen ich heute begegnet bin, konnte ich schnell eine Lösung finden. |

Tabelle 13 - Tagesreflexion Tag 1

**Ich bin:**

* Voraus
* **Im Plan**
* In Verzug

## Tagesjournal vom 20.03.2018

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Geplante Tagesziele | | | | | |
| **Nr.** | **Ziel** | **Soll** | **Ist** | **Abweichung** | **Abgeschlossen** |
| 1 | Umsysteme und Abhängigkeiten beschreiben | 0.5 | 0.5 | 0 | Ja |
| 2 | Anforderungsanalyse pro Task | 2 | 2 | 0 | Ja |
| 3 | Vorgehensweise und Pattern-Erklärung | 4 | 1.5 | -2.5 | Nein |
| 4 | Korrektur des Gegenlesens | 1 | 0.25 | - 0.75 | Nein |

Tabelle 14 - Geplante Tagesziele Tag 2

|  |  |  |  |
| --- | --- | --- | --- |
| Tätigkeit | Zeit | Aufwand | Ziel |
| RabbitMQ Beschreibung der Kommunikation | 8.15 – 8.45 | 0.5 | 1 |
| Hilfe von Melanie Müller mit OAW Vorlage Inhaltsverzeichnis | 8.45 – 9.00 | 0.25 | - |
| Anforderungsanalyse und Umsetzungsreihenfolge definieren | 9.00 – 11.00 | 2 | 2 |
| Vorgehensweise und Pattern-Erklärung | 11.00 – 12.00  12.30 – 13.00 | 1.5 | 3 |
| Umsetzungserklärungen der einzelnen Tasks | 13.00 – 16.00 | 3 | - |
| Nachfrage nach der gegengelesenen Doku bei Matthias Hess, leider noch nicht dazu gekommen. | 16.00 – 16.15 | 0.25 | 4 |
| Tagesabschluss | 16.15 – 16.30 | 0.25 | - |

Tabelle 15 - Tätigkeiten Tag 2

|  |
| --- |
| Probleme |
| Ich bin heute auf das Problem gestossen, dass ich während der Vorgehensweise und Pattern-Erklärung bemerkte, dass ich zuerst die Tasks definiert haben möchte, bevor ich genau die einzelnen Patterns etc. erkläre. Dies führt dazu, dass sich die Reihenfolge dieser zwei Tasks austauscht. So werde ich mich morgen vor allem mit der Pattern-Erklärung auseinandersetzen. |

Tabelle 16 - Probleme Tag 2

|  |
| --- |
| Hilfestellung |
| Ich habe heute das Problem der OAW Vorlage mit Melanie Müller behoben. |

Tabelle 17 - Hilfestellung Tag 2

|  |
| --- |
| Tagesreflexion |
| Der Tag verlief gut, nicht ganz wie geplant, aber ich habe durch das Schieben der Tasks keine Zeit verloren. |

Tabelle 18 - Tagesreflexion Tag 2

**Ich bin:**

* Voraus
* **Im Plan**
* In Verzug

## Tagesjournal vom 22.03.2018

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Geplante Tagesziele | | | | | |
| **Nr.** | **Ziel** | **Soll** | **Ist** | **Abweichung** | **Abgeschlossen** |
| 1 | Umsetzungserklärung der einzelnen Tasks | 4 | 4 | 0 | Ja |
| 2 | Expertenbesuch | 1 | 1 | 0 | Nein |
| 3 | Speichern der Parameter | 2.5 | 2 | -0.5 | Ja |
| 4 | Erstellung der Unit Tests & des Testkonzept | 1 | 0.5 | -0.5 | Nein |

Tabelle 19 - Geplante Tagesziele Tag 3

|  |  |  |  |
| --- | --- | --- | --- |
| Tätigkeit | Zeit | Aufwand | Ziel |
| Korrektur des Gegenlesens von Matthias Hess | 7.30 – 8.00 | 0.5 | - |
| Vorgehensweiseerklärung dokumentiert | 8.00 – 9.00  11.00 – 12.30 | 2.5 | - |
| Expertenbesuch | 9.00 – 10.00 | 1 | - |
| Umsetzungserklärung verfeinert / Negative Punkte von Herr Schättin verbessert. | 10.00 -11.00 | 1 |  |
| Erstellung des Testkonzepts | 13.30 – 14.00 | 0.5 | 4 |
| Speichern der Parameter | 14.00 – 16.00 | 2 | 3 |
| Tagesjournal | 16.00 – 16.30 | 0.5 | - |

Tabelle 20 - Tätigkeiten Tag 3

|  |
| --- |
| Probleme |
| Heute habe ich das Problem des einen Bewertungskriteriums mit Herr Schättin besprochen. Wir haben nun ein neues Kriterium an Herr Schöpflin geschickt. Dies wurde am selben Tag noch bestätigt und in der Dokumentation angepasst. |

Tabelle 21 - Probleme Tag 3

|  |
| --- |
| Hilfestellung |
| Melanie Müller, unsere Office-Spezialistin hat mir gezeigt, wie man bei unseren Vorlagen ein Tabellenverzeichnis generieren kann. |

Tabelle 22 - Hilfestellung Tag 3

|  |
| --- |
| Tagesreflexion |
| Da mir nicht klar war, ob der Expertenbesuch Teil der IPA-Zeit ist oder nicht, stehe ich momentan etwas im Verzug. Dies sollte ich morgen wieder einholen können. |

Tabelle 23 - Tagesreflexion Tag 3

**Ich bin:**

* Voraus
* Im Plan
* **In Verzug**

## Tagesjournal vom 23.03.2018

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Geplante Tagesziele | | | | | |
| **Nr.** | **Ziel** | **Soll** | **Ist** | **Abweichung** | **Abgeschlossen** |
| 1 | Speichern der Parameter | 6 | 7 | +1 | Nein |
| 2 | Korrektur des Gegenlesens | 1 | 0 | -1 | Nein |
| 3 | Kriterien F06 & Aufgabenstellung mit Stand vergleichen | 0.5 | 0.5 | 0 | Nein |

Tabelle 24 - Geplante Tagesziele Tag 4

|  |  |  |  |
| --- | --- | --- | --- |
| Tätigkeit | Zeit | Aufwand | Ziel |
| Json Serialisierung eines Parameters | 8.15 – 9.30 | 1.25 | 1 |
| API-Erweiterung, dass ein Parameter retour kommt und dies im Client anzeigen | 9.30 – 11.30 | 2 | 1 |
| API parametriert, dass Lower Camel Case Json zurückgegeben wird. | 11.30 – 11.45 | 0.25 | 1 |
| Kriterien F06 & Aufgabenstellung mit Stand vergleichen | 12.30 – 13.00 | 0.5 | 3 |
| Backend Helper-Klassen schreiben für Json Serialisierung / Deserialisierung | 13.00 – 16.30 | 3.5 | 1 |
| Tagesjournal | 16.30 – 17.00 | 0.5 | - |

Tabelle 25 - Tätigkeiten Tag 4

|  |
| --- |
| Probleme |
| Ein Problem, dass ich am Morgen hatte, war dass die API ein Upper Camel Case Json Objekt retour lieferte. Dies konnte ich dann nicht serialisieren. Mit der Hilfe von Benjamin Schäublin, einem Entwicklerkollegen von mir, konnte ich das Problem schnell lösen, es fehlte nur ein Attribut bei der API. Zudem bin ich lange an einem Punkt hängen geblieben, bei der das Binding einer Checkbox sich nicht updatet, wenn ich es via Code setze. Dies konnte ich mit der Angular Dokumentation von Google lösen. |

Tabelle 26 - Probleme Tag 4

|  |
| --- |
| Hilfestellung |
| Google Angular Dokumentation, Benjamin Schäublin. |

Tabelle 27 - Hilfestellung Tag 4

|  |
| --- |
| Tagesreflexion |
| Der Tag verlief gut. Ich bin schnell vorangekommen, leider konnte ich die Reflexion des Gegenlesens nicht machen, da ich die Dokumentation nicht wieder zurück bekommen habe heute. Sie sollte aber montags vorliegen. Das Programmieren der API gestaltet sich als schwieriger als erwartet. Der Einsatz von viel Reflection macht das ganze Arbeiten ziemlich Algorithmus-Lastig. Ich tüftle aber gerne, weshalb es mir dennoch Spass macht. Ich bin noch eine halbe Stunde in Verzug. Dies wird sich aber sicherlich im Verlaufe der nächsten Woche bessern. |

Tabelle 28 - Tagesreflexion Tag 4

**Ich bin:**

* Voraus
* Im Plan
* **In Verzug**

## Tagesjournal vom 26.03.2018

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Geplante Tagesziele | | | | | |
| **Nr.** | **Ziel** | **Soll** | **Ist** | **Abweichung** | **Abgeschlossen** |
| 1 | Speichern der Parameter | 4.5 | 3.5 | -1 | Ja |
| 2 | Implementation des Validierungsmechanismus | 3 | 2.5 | -0.5 | Nein |
| 3 | Durchführen der Tests | 1 | 1 | 0 | Nein |

Tabelle 29 - Geplante Tagesziele Tag 5

|  |  |  |  |
| --- | --- | --- | --- |
| Tätigkeit | Zeit | Aufwand | Ziel |
| Korrektur des Gegenlesens von Melanie Müller. | 8.30 – 9.00 | 0.5 | - |
| Speichern der Parameter fertiggestellt | 9.00 – 11.45  12.30 – 13.15 | 3.5 | 1 |
| Testing von Ziel 1, Fehlerbehebung der gefundenen Tests | 13.15 – 14.15 | 1 | 1 |
| Frontendseitige Validierung verbaut | 14.15 – 16.45 | 2.5 | 2 |
| Tagesabschluss | 16.45 – 17.00 | 0.25 | - |

Tabelle 30 - Tätigkeiten Tag 5

|  |
| --- |
| Probleme |
| Ich hatte heute Schwierigkeiten mit RabbitMQ. Meine Service reagierten nicht. Dies konnte ich aber mit einem PC Neustart beheben. |

Tabelle 31 - Probleme Tag 5

|  |
| --- |
| Hilfestellung |
| Heute habe ich keine Hilfestellung gebraucht. |

Tabelle 32 - Hilfestellung Tag 5

|  |
| --- |
| Tagesreflexion |
| Ich bin sehr gut vorangekommen. Insbesondere das Speichern der Parameter fertig zu machen und die Frontendvalidierung gingen schneller als erwartet. Ich bin nun leicht dem Plan Voraus. Ich rechne damit, dass ich Morgen die Validierung früher fertig stellen kann als erwartet. Die eingesparte Zeit werde ich zum Feinschliff meiner Dokumentation einsetzen. Das Gegenlesen von Melanie Müller hat viel gebracht, da so einige Rechtschreibfehler korrigiert werden konnten. |

Tabelle 33 - Tagesreflexion Tag 5

**Ich bin:**

* **Voraus**
* Im Plan
* In Verzug

## Tagesjournal vom 27.03.2018

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Geplante Tagesziele | | | | | |
| **Nr.** | **Ziel** | **Soll** | **Ist** | **Abweichung** | **Abgeschlossen** |
| 1 | Implementation des Validierungsmechanismus | 5 | 5.5 | +0.5 | Ja |
| 2 | Suche eines Parameters | 1.5 | 1 | -0.5 | Nein |
| 3 | Durchführen der Tests | 1 | 1 | 0 | Nein |

Tabelle 34 - Geplante Tagesziele Tag 6

|  |  |  |  |
| --- | --- | --- | --- |
| Tätigkeit | Zeit | Aufwand | Ziel |
| Fertigstellung der serverseitigen Validierung. | 8.30 – 12.00  12.45 – 14.45 | 5.5 | 1 |
| Durchführen der Tests | 14.45 – 15.45 | 1 | 3 |
| Suche eines Parameters | 15.45 – 16.45 | 1 | 2 |
| Arbeitsjournal | 16.45 – 17.15 | 0.5 | - |

Tabelle 35 - Tätigkeiten Tag 6

|  |
| --- |
| Probleme |
| Ich bin bei der serverseitigen Validierung lange stecken geblieben, da ich null in die API bekam. Nachdem Visual Studio abgestürzt ist ging es. Ich habe die ganze Zeit das Problem beim Inputbinding gesucht. Dadurch habe ich viel Zeit verloren. |

Tabelle 36 - Probleme Tag 6

|  |
| --- |
| Hilfestellung |
| Ich habe Martin Tinner gefragt, wie ich es in der Dokumentation mit den vielen englischen Fachbegriffen regeln soll, da ich versucht habe alle zu übersetzen. Auf Rat von ihm, werde ich einfach all jene, die nicht geläufig sind ins Glossar aufnehmen. |

Tabelle 37 - Hilfestellung Tag 6

|  |
| --- |
| Tagesreflexion |
| Ich habe viel Zeit verloren mit dem Problem von Visual Studio, konnte jedoch sehr schnell eine Möglichkeit fürs Suchen erarbeiten, was mich einiges an Zeit aufholen liess. |

Tabelle 38 - Tagesreflexion Tag 6

**Ich bin:**

* Voraus
* **Im Plan**
* In Verzug

## Tagesjournal vom 29.03.2018

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Geplante Tagesziele | | | | | |
| **Nr.** | **Ziel** | **Soll** | **Ist** | **Abweichung** | **Abgeschlossen** |
| 1 | Suche eines Parameters | 6.5 | 7.5 | +1 | Ja |
| 2 | Testen | 1 | 0 | -1 | Nein |
| 3 | Kriterien F06 & Aufgabenstellung mit Stand vergleichen | 0.5 | 0.5 | 0 | Nein |

Tabelle 39 - Geplante Tagesziele Tag 7

|  |  |  |  |
| --- | --- | --- | --- |
| Tätigkeit | Zeit | Aufwand | Ziel |
| Suche eines Parameters fertigstellen | 7.00 – 9.00 | 2 | 1 |
| Dokumentation der Realisierung Verfeinert, Codedokumentation Backend fertig und Frontend angefangen. | 9.00 – 12.30  13.00 – 15.00 | 5.5 | 1 |
| Kriterien | 15.00 – 15.30 | 0.5 | 3 |
| Tagesjournal | 15.30 – 16.00 | 0.5 | - |

Tabelle 40 - Tätigkeiten Tag 7

|  |
| --- |
| Probleme |
| Keine Probleme |

Tabelle 41 - Probleme Tag 7

|  |
| --- |
| Hilfestellung |
| Keine Hilfestellungen |

Tabelle 42 - Hilfestellung Tag 7

|  |
| --- |
| Tagesreflexion |
| Ich bin gut vorangekommen. Die Codedokumentation sieht bereits sehr gut aus. |

Tabelle 43 - Tagesreflexion Tag 7

**Ich bin:**

* Voraus
* **Im Plan**
* In Verzug

## Tagesjournal vom 03.04.2018

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Geplante Tagesziele | | | | | |
| **Nr.** | **Ziel** | **Soll** | **Ist** | **Abweichung** | **Abgeschlossen** |
| 1 | Erstellung der Unittests | 3 | 4 | +1 | Ja |
| 2 | Korrektur des Gegenlesens | 1 | 2 | +1 | Ja |
| 3 | Getting Started Doku | 2 | 0 | -2 | Nein |
| 4 | Durchführen der Tests | 1 | 1 | 0 | Ja |

Tabelle 44 - Geplante Tagesziele Tag 8

|  |  |  |  |
| --- | --- | --- | --- |
| Tätigkeit | Zeit | Aufwand | Ziel |
| Korrektur des Gegenlesens meiner Schwester Shana Kessler | 8.15 – 10.15 | 2 | 2 |
| Erstellung der Unittests | 10.15 -11.45  12.30 – 15.00 | 4 | 1 |
| Durchführen der Tests | 15.00 – 16.00 | 1 | 4 |
| Tagesjournal | 16.00 – 16.30 | 0.5 | - |

Tabelle 45 - Tätigkeiten Tag 8

|  |
| --- |
| Probleme |
| Heute hatte ich keine Probleme. |

Tabelle 46 - Probleme Tag 8

|  |
| --- |
| Hilfestellung |
| Keine Hilfestellungen. |

Tabelle 47 - Hilfestellung Tag 8

|  |
| --- |
| Tagesreflexion |
| Ich konnte den ganzen Tag durch gut arbeiten, die Unittests haben jedoch länger gebraucht als erwartet. Ich nehme nicht an, dass ich noch Zusatzfunktionalität umsetzen kann im Rahmen der IPA. |

Tabelle 48 - Tagesreflexion Tag 8

**Ich bin:**

* Voraus
* **Im Plan**
* In Verzug

## Tagesjournal vom 05.04.2018

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Geplante Tagesziele | | | | | |
| **Nr.** | **Ziel** | **Soll** | **Ist** | **Abweichung** | **Abgeschlossen** |
| 1 | Korrektur der gefundenen Fehler | 2 | 2 | 0 | Ja |
| 2 | Konzept und Umsetzungsvergleich | 4 | 5.5 | + 1.5 | Ja |

Tabelle 49 - Geplante Tagesziele Tag 9

|  |  |  |  |
| --- | --- | --- | --- |
| Tätigkeit | Zeit | Aufwand | Ziel |
| Getting Started Dokumentation | 7.00 – 9.00 | 2 | - |
| Korrektur der gefundenen Fehler | 9.00 – 11.00 | 2 | 1 |
| Konzept und Umsetzungsvergleich | 11.00 – 12.00  13.00 – 15.30 | 3.5 | 2 |
| Tagesjournal | 15.30 – 16.00 | 0.5 | - |

Tabelle 50 - Tätigkeiten Tag 9

|  |
| --- |
| Probleme |
| Heute ist aufgefallen, dass ein Klemmheft etwas wenig für ca. 100 Seiten ist. Deshalb habe ich den Experten Patrick Schättin gefragt, ob ein Ordner für die Abgabe auch in Ordnung wäre |

Tabelle 51 - Probleme Tag 9

|  |
| --- |
| Hilfestellung |
| Keine Hilfestellung |

Tabelle 52 - Hilfestellung Tag 9

|  |
| --- |
| Tagesreflexion |
| Der Tag verlief gut. Ich konnte viel an meiner Dokumentation arbeiten und einige Fehler noch beheben. |

Tabelle 53 - Tagesreflexion Tag 9

**Ich bin:**

* Voraus
* **Im Plan**
* In Verzug

## Tagesjournal vom 06.05.2018

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Geplante Tagesziele | | | | | |
| **Nr.** | **Ziel** | **Soll** | **Ist** | **Abweichung** | **Abgeschlossen** |
| 1 | Korrektur des Gegenlesens | 1 | 1 | 0 | Ja |
| 2 | Kriterien F06 & Aufgabenstellung mit Stand vergleichen | 0.5 | 1 | +0.5 | Ja |
| 3 | Expertenbesuch | 0.5 | 0 | 0.5 | Ja |
| 4 | Reflexion | 1 | 1 | 0 | Ja |
| 5 | Zeitpuffer | 4 | 4.5 | + 0.5 | Ja |

Tabelle 54 - Geplante Tagesziele Tag 10

|  |  |  |  |
| --- | --- | --- | --- |
| Tätigkeit | Zeit | Aufwand | Ziel |
| Korrekturen des Gegenlesens von Melanie Müller | 8.00 - 9.00 | 1 | 1 |
| Kriterien F06 & Aufgabenstellung mit Stand vergleichen | 9.00 – 10.00 | 1 | 2 |
| Zeitpuffer verwendet für einfügen des Zeitplans und dessen Beschreibung | 10.00- 12.00 | 2 | 5 |
| Reflexion | 13.00 – 14.00 | 1 | 4 |
| Zeitpuffer für Reflexion | 14.00 – 14.30 | 0.5 | 4 |
| Tagesjournal | 14.30 – 15.00 | 0.5 | - |
| Zeitpuffer für Drucken und IPA Versand | 15.00 – 17.00 | 2 | - |

Tabelle 55 - Tätigkeiten Tag 10

|  |
| --- |
| Probleme |
| Keine Probleme |

Tabelle 56 - Probleme Tag 10

|  |
| --- |
| Hilfestellung |
| Keine Hilfestellungen |

Tabelle 57 - Hilfestellung Tag 10

|  |
| --- |
| Tagesreflexion |
| Rückblickend auf die Ganze IPA hat mir das Projekt viel Spass gemacht. Das komplett selbständige Arbeiten war zwar herausfordernd aber auch schön. Ich bin zum Glück mit allem fertig geworden, weshalb ich nun mit gutem Gewissen drucken und anschliessend abgeben kann. |

Tabelle 58 - Tagesreflexion Tag 10

**Ich bin:**

* Voraus
* **Im Plan**
* In Verzug

## Allgemeines zum Tagesjournal

Das Tagesjournal wurde entweder in ausgedruckt oder elektronisch Martin Tinner gezeigt. Er hat diese jeweils am Ende des Tages gesehen.

|  |  |  |
| --- | --- | --- |
| Datum | Kandidat | Verantwortlicher |
|  |  |  |

Tabelle 59 - Tagesjournale gesehen

# Taskerstellung nach den Anforderungen

Die Einzeltasks sind eine Abbildung der Use-Cases und der Kriterien. Sie formulieren spezifische Umsetzungspunkte. Die Use-Cases der Anwendung sehen wie folgt aus:

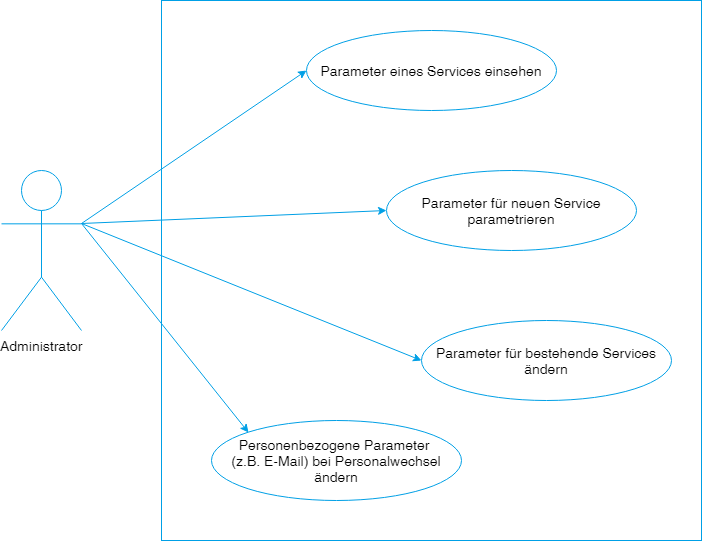


Abbildung 8 - Use-Case Diagramm

Das Use-Case Diagramm stellt den Akteur, in diesem Fall den Systemadministrator, und die Anwendungsfälle dar. Da diese Applikation für einen Administrator gedacht ist, steht die Funktionalität im Vordergrund. Die Use-Cases sind nach den bisherigen Anforderungen im Viaduc aufgestellt. Die Parameter sind technischer (z.B. ein DB-Connection-String), fachlicher (z.B. verschiedene Dauern und Daten für bestimmte Vorgänge) oder menschlicher (z.B. eine E-Mailadresse oder Personenverweise in Texten) Natur.

## Einzeltasks für die Dokumentation

### Getting Started Dokumentation

|  |
| --- |
| Zu erfüllende 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 |

Tabelle 60 - Getting Started Dokumentation Ziele

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

Tabelle 61 - Getting Started Dokumentation Anforderungen

**Geschätzter Aufwand**: 2h

## Einzeltasks für die Realisierung

### Anzeige & Speichern der Parameter

|  |
| --- |
| Zu erfüllende Ziele |
| Ein Hilfetext kann für jeden Parameter hinterlegt werden |
| Defaultwerte 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 |
| Die Einbindung der Parameterverwaltung soll mit der Angabe des Parametertyps und einem Minimum von weiterem Code mögl. sein. |

Tabelle 62 - Anzeige & Speichern der Parameter Ziele

|  |
| --- |
| 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. |

Tabelle 63 - Anzeige & Speichern der Parameter Anforderungen

**Geschätzter Aufwand**: 12h

### Implementation des Validierungsmechanismus

|  |
| --- |
| Zu erfüllende 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 |

Tabelle 64 - Implementation des Validierungsmechanismus Ziele

|  |
| --- |
| 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 Defaultwerte genügen. |

Tabelle 65 - Implementation des Validierungsmechanismus Anforderungen

**Geschätzter Aufwand**: 8h

### Suchen eines Parameters

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

Tabelle 66 - Suchen eines Parameters Ziele

|  |
| --- |
| 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. |

Tabelle 67 - Suchen eines Parameters Anforderungen

**Geschätzter Aufwand**: 8h

### Erstellung der Unit Tests und des Testkonzepts

|  |
| --- |
| Anforderung |
| Für Funktionen, welche Daten verarbeiten oder validieren, muss mindestens ein Unit-Test geschrieben werden. |

Tabelle 68 - Erstellung der Unit Tests und des Testkonzepts Anforderungen

**Gesamter geschätzter Aufwand:** 4h

## Einzeltaskerstellung der Kontrolltasks

### Durchführen der Tests

|  |
| --- |
| Zu erfüllende 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 |
| Es werden sprechende Namen für Klassen / Methoden verwendet |
| Ist es nachvollziehbar, warum gerade diese Lösung gewählt wurde? Was waren die Kriterien? |

Tabelle 69 - Durchführen der Tests Ziele

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

Tabelle 70 - Durchführen der Tests Anforderungen

**Geschätzter Aufwand pro Durchgang**: 1h

Dieser Task muss 1 Mal nach jedem realisierten Realisierung-Punkt umgesetzt werden.

**Gesamter geschätzter Aufwand:** 4h

# Zeitplan & Burn down

Der Zeitplan ist nach Tagen respektive Halbtagen gegliedert. AM ist der Vormittag PM der Nachmittag.

## Zeitplan

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Phase** | **Kurzbeschreibung** | **Zeit- schätzung** | **Montag 19.03.2018** | | | | **Dienstag 20.03.2018** | | | | **Donnerstag 22.03.2018** | | | | **Freitag 23.03.2018** | | | | **Montag 26.03.2018** | | | |
| AM | | PM | | AM | | PM | | AM | | PM | | AM | | PM | | AM | | PM | |
| Soll | Ist | Soll | Ist | Soll | Ist | Soll | Ist | Soll | Ist | Soll | Ist | Soll | Ist | Soll | Ist | Soll | Ist | Soll | Ist |
| I | Aufgabenstellung und Allgemeines beschreiben | 4 | 4 | 3.5 |  | 0.5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P | Umsysteme und Abhängigkeiten beschreiben | 2 |  |  | 1.5 | 1.5 | 0.5 | 0.5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Zeitplanerstellung / Taskerstellung | 2 |  |  | 2 | 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| E | Anforderungsanalyse pro Task | 2 |  |  |  |  | 2 | 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Vorgehensweise und Paternerklährung | 4 |  |  |  |  | 1 | 1 | 3 | 0.5 |  | 2.5 |  |  |  |  |  |  |  |  |  |  |
| Umsetzungserklärungen der einzelnen Tasks | 4 |  |  |  |  |  |  |  | 3 | 4 | 1 |  |  |  |  |  |  |  |  |  |  |
| R | Speichern der Parameter | 13 |  |  |  |  |  |  |  |  |  |  | 2.5 | 2 | 3 | 3.5 | 3 | 3.5 | 4.5 | 3.5 |  |  |
| Implementation des Validierungsmechanismus | 8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 3 | 2.5 |
| Suche eines Parameters | 8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Erstellung der Unit Tests & des Testkonzept | 4 |  |  |  |  |  |  |  |  |  |  | 1 | 0.5 |  |  |  |  |  |  |  |  |
| Zeitpuffer / Zusatzfunktionen | 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| K | Durchführen der Tests | 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 | 1 |
| Korrektur des Gegenlesens | 4 |  |  |  |  | 1 |  |  | 0.5 |  | 0.5 |  |  | 1 |  |  |  |  | 0.5 |  |  |
| Kriterien F06 & Aufgabenstellung mit Stand vergleichen | 1.5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 0.5 | 0.5 |  |  |  |  |
| Expertenbesuch | 1.5 |  |  |  |  |  |  |  |  | 1 | 1 |  |  |  |  |  |  |  |  |  |  |
| Korrektur der gefundenen Fehler | 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| A | Tagesjournal & Tages-Versionserstellung | 5 |  |  | 0.5 | 0.5 |  |  | 0.5 | 0.5 |  |  | 0.5 | 0.5 |  |  | 0.5 | 0.5 |  |  | 0.5 | 0.5 |
| Reflexion | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Getting Started | 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Konzept und Umsetzungsvergleich | 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Total |  | 80 | 4 | 3.5 | 4 | 4.5 | 4.5 | 3.5 | 3.5 | 4.5 | 5 | 5 | 4 | 3 | 4 | 3.5 | 4 | 4.5 | 4.5 | 4 | 4.5 | 4 |
| Burn down | | 80 | 76 | 77 | 72 | 72 | 68 | 69 | 64 | 64 | 59 | 59 | 55 | 56 | 51 | 53 | 47 | 48 | 43 | 44 | 38 | 40 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Phase** | **Kurzbeschreibung** | **Zeit- schätzung** | **Dienstag 27.03.2018** | | | | **Donnerstag 29.03.2018** | | | | **Dienstag 03.04.2018** | | | | **Donnerstag 05.04.2018** | | | | **Freitag 06.04.2018** | | | |
| AM | | PM | | AM | | PM | | AM | | PM | | AM | | PM | | AM | | PM | |
| Soll | Ist | Soll | Ist | Soll | Ist | Soll | Ist | Soll | Ist | Soll | Ist | Soll | Ist | Soll | Ist | Soll | Ist | Soll | Ist |
| I | Aufgabenstellung und Allgemeines beschreiben | 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P | Umsysteme und Abhängigkeiten beschreiben | 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Zeitplanerstellung / Taskerstellung | 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| E | Anforderungsanalyse pro Task | 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Vorgehensweise und Paternerklährung | 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Umsetzungserklärungen der einzelnen Tasks | 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| R | Speichern der Parameter | 13 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Implementation des Validierungsmechanismus | 8 | 4 | 3.5 | 1 | 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Suche eines Parameters | 8 |  |  | 1.5 | 1 | 4 | 5.5 | 2.5 | 2 |  |  |  |  |  |  |  |  |  |  |  |  |
| Erstellung der Unit Tests & des Testkonzept | 4 |  |  |  |  |  |  |  |  | 3 | 1.5 |  | 2.5 |  |  |  |  |  |  |  |  |
| Zeitpuffer / Zusatzfunktionen | 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 2 | 2 | 2 | 2.5 |
| K | Durchführen der Tests | 4 |  |  | 1 | 1 |  |  | 1 |  |  |  | 1 | 1 |  |  |  |  |  |  |  |  |
| Korrektur des Gegenlesens | 4 |  |  |  |  |  |  |  |  | 1 | 2 |  |  |  |  |  |  | 1 | 1 |  |  |
| Kriterien F06 & Aufgabenstellung mit Stand vergleichen | 1.5 |  |  |  |  |  |  | 0.5 | 0.5 |  |  |  |  |  |  |  |  | 0.5 | 1 |  |  |
| Expertenbesuch | 1.5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 0.5 |  |  |  |
| Korrektur der gefundenen Fehler | 2 |  |  |  |  |  |  |  |  |  |  |  |  | 2 |  |  |  |  |  |  |  |
| A | Tagesjournal & Tages-Versionserstellung | 5 |  |  | 0.5 | 0.5 |  |  | 0.5 | 0.5 |  |  | 0.5 | 0.5 |  |  | 0.5 | 0.5 |  |  | 0.5 | 0.5 |
| Reflexion | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 | 1 |
| Getting Started | 2 |  |  |  |  |  |  |  |  |  |  | 2 |  |  | 2 |  |  |  |  |  |  |
| Konzept und Umsetzungsvergleich | 4 |  |  |  |  |  |  |  |  |  |  |  |  | 1 | 2.5 | 3 | 3 |  |  |  |  |
| Total |  | 80 | 4 | 3.5 | 4 | 4.5 | 4 | 5.5 | 4.5 | 3 | 4 | 3.5 | 3.5 | 4 | 3 | 4.5 | 3.5 | 3.5 | 4 | 4 | 3.5 | 4 |
| Burn down | | 80 | 34 | 37 | 30 | 32 | 26 | 27 | 22 | 24 | 18 | 20 | 14 | 16 | 11 | 12 | 7.5 | 8 | 3.5 | 4 | 0 | 0 |

Erklärungen zum Zeitplan

Der Morgen und Nachmittag zusammen ergeben jeweils einen Tag. Der Morgen ist jeweils weiss und der Nachmittag grau. Die Tasks sind auf eine halbe Stunde genau geschätzt. Die Einteilung der Zeitachse ist jedoch auf Halbtage (vier Stunden) genau. Die Reihenfolge ist möglichst so gewählt, dass man bis und mit Realisierungsphase von oben nach unten sich durcharbeiten kann. Die Korrektur- und Auswertungsphase passiert fortlaufend, weshalb diese Tasks so verteilt sind, wie sie fällig sind. Die Meilensteine sind jeweils nach der Planungs-, Entscheidungs- und Realisationsphase und bei Projektende nach der Auswertung. Die Kontrollphase ist über das ganze Projekt verteilt, weshalb sie erst mit Abgabe wirklich zu Ende ist.

## Burn down

Das Burn down Chart zeigt grafisch auf, wie viele Stunden für welchen Schritt benötigt wurden.

# Entscheide der einzelnen 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 und die konkrete Umsetzung benötigt, 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 |

Tabelle 71 - Umsetzungsreihenfolge

## 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**

Unittests sind klar definierte Tests, welche eine bestimmte Funktion testen. Sie sind einfach zu schreiben und schnell ausgeführt. Sie zeigen einem auf, wenn der Code etwas falsch gemacht hat. Man kann mit diesen Tests herausfinden, ob erwartete, unerwartete oder falsche Übergabewerte übergeben werden.

Sie sind technisch einfach, können aber GUI-Tests übernehmen.

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

Tabelle 72 - Lösungsmatrix Unittests

#### Integration Tests / e2e Tests

Integrationstests sind automatisierte Tests, die Businesslogik testen. Sie können verschiedene Teile des Systems testen. Sie sind relativ komplex, da man sich zuerst die ganze Datenstruktur aufbauen muss. Sie sind sehr Zeitaufwendig zu schreiben. E2e Tests ist das Äquivalent zu den Integrationstests für GUIs. Man kann mit ihnen einen Systemzustand simulieren, und schauen, ob dann das Richtige angezeigt werden würde. Für e2e Test fehlt mir jedoch die Erfahrung, da ich noch keine selbst geschrieben habe und mir nach kurzer Recherche auffiel, dass sie den Rahmen dieses Projektes übersteigen würden.

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

Tabelle 73 - Lösungsmatrix Integration Tests / e2e Tests

#### User-Testing

Das User-Testing ist das Testen eines Benutzers von Hand, ob die Applikation, das macht was von ihr erwartet wird. Es sind zeitaufwendige Tests, die besonders für Layout and Anzeige gut zu gebrauchen sind, da ein Computer nicht alles testen kann (z.B. sieht die Webseite optisch gut aus).

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

Tabelle 74 - Lösungsmatrix User-Testing

### 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 nur 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

Mit der Newtonsoft Json Drittkomponente z.B., kann man eine C# Klasse zu einem Json serialisieren, dies mit wenigen Zeilen Code. Ein solches Json File zu erweitern, heisst, dass man es clientseitig auch anpassen muss. Somit muss für eine Parameteranpassung auch eine Layoutanpassung erfolgen.

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

Tabelle 75 - Lösungsmatrix Speichern der Parameter als Parametertyp im Json

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

Das Speichern der Parameter als generischer Typ bietet den Vorteil, dass man ein klar definiertes Json Schema hat. Dies ist dann in jedem Service gleich abgelegt und kann im Frontend dann für jeden Parameter gleichbehandelt werden. Der Nachteil dieser Variante ist, dass es komplizierter zum Schreiben ist, als ein Framework zu verwenden. Via C# Reflection kann ein Parameter generisch ausgelesen werden.

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

Tabelle 76 - Lösungsmatrix Speichern der Parameter als generischer Typ im Json

### Umsetzungsbeschreibung Speichern

Das Speichern der Parameter als generischer Typ im Json scheint die bessere Option zu sein, da man klar definiert wie das Json auszusehen 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

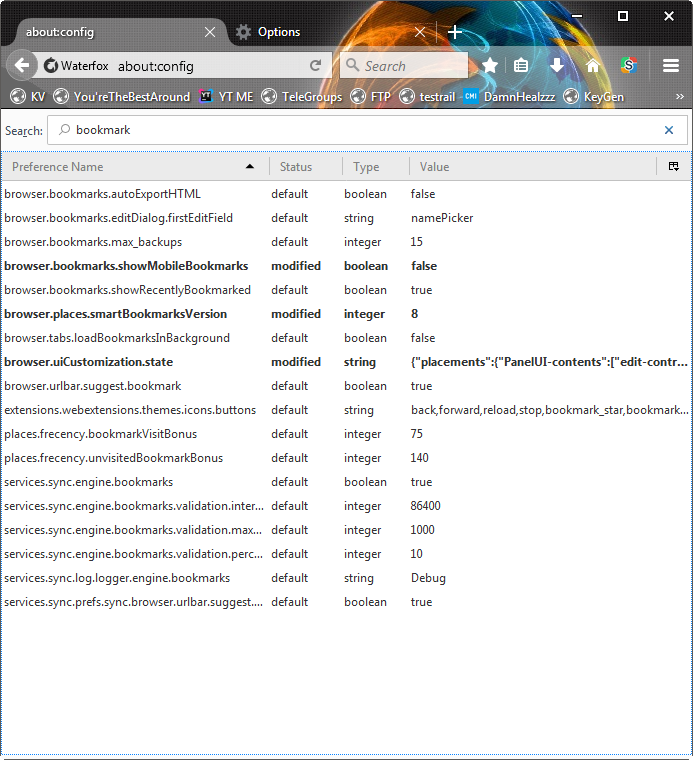


Abbildung 9 - Firefox/Waterfox Einstellungen

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

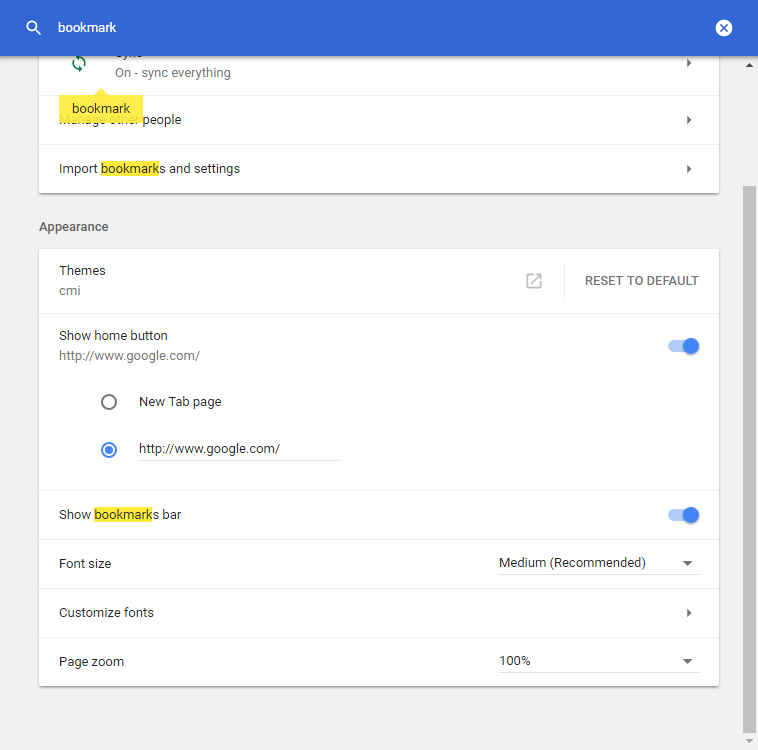


Abbildung 10 - Chrome Einstellungen

Die Chrome Settings Page erlaubt ebenfalls das Suchen nach einem Begriff. Es wird alles hervorgehoben, was gefunden wurde. Die Page ist strukturiert aufgebaut, jedoch nicht ganz so einfach um etwas zu finden, wenn man nicht genau weiss wo. Es ist modern aufgebaut und besitzt die Funktion alles auf den Defaultwert zu setzen. Der Standardwert wird oftmals mit (Recommended) gekennzeichnet.

#### Ähnlich wie in Visual Studio

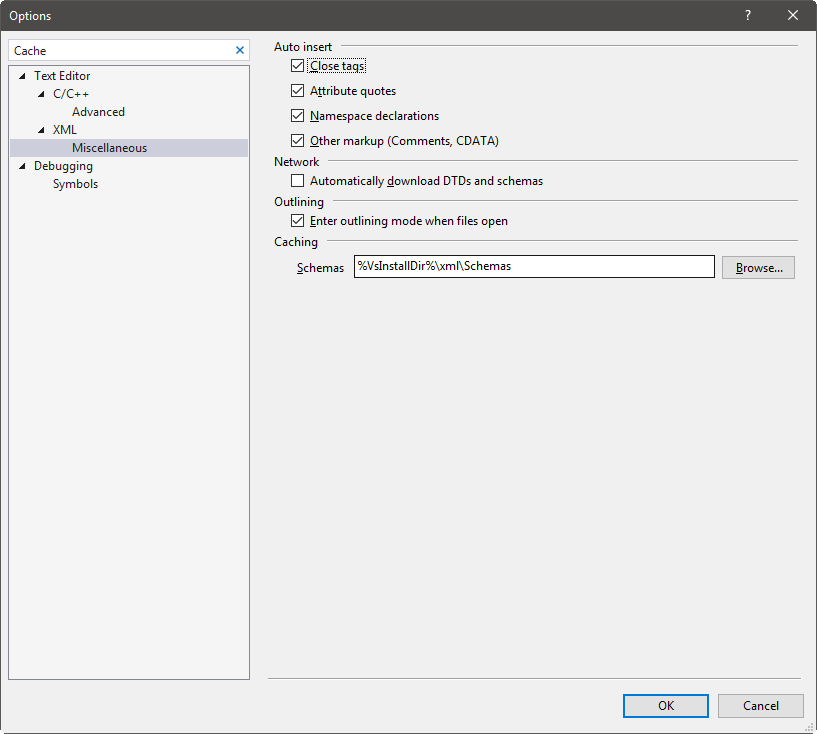


Abbildung 11 - Visual Studio Einstellungen

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

Damit eine ideale Lösung gestaltet werden kann, wird die Parameternamensgebung von Firefox, die Gruppierung von Visual Studio via den Parameternamen und die Suche wie in Chrome verwendet. Wobei die Gruppierung optional vorgenommen werden kann, wenn noch Zeit dafür ist.

Ein grobes Konzept könnte wie folgt aussehen:

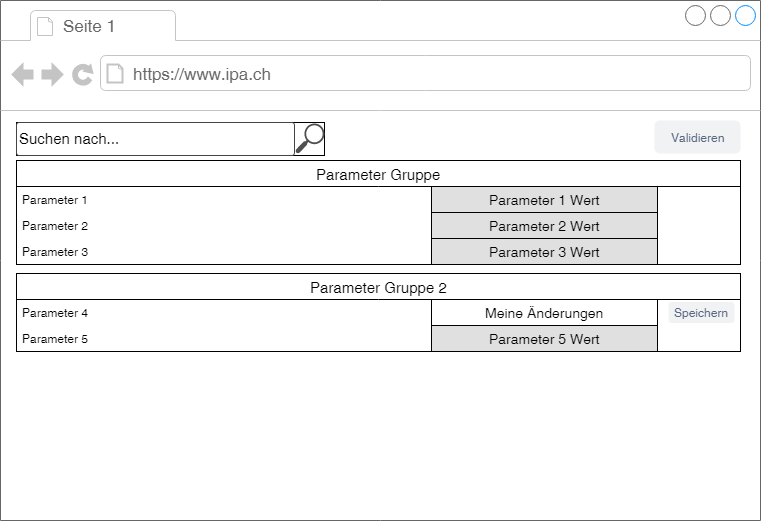


Abbildung 12 - Mockup

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 Parameterwert 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 Parametername 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 |

Tabelle 77 - Lösungsmatrix nur serverseitig im Service den Parameter selbst testen lassen

#### 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 |

Tabelle 78 - Lösungsmatrix Regular Expression im Parameter

### 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 alle formatbezogenen und wertbezogenen Validierungen damit machen (z.B. liegt die E-Mailadresse in einer gültigen Form vor, oder ist der Wert nicht grösser als 100).

## Suche eines Parameters

### Anforderung

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

### Mögliche Lösungen

#### Volltextsuche über alles

Die Volltextsuche schaut jeden Parameter als Ganzes durch. Wird eine Übereinstimmung auf irgendeine Parameter-Property festgestellt, wird der Parameter in der Suche angezeigt.

|  |  |
| --- | --- |
| 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 |

Tabelle 79 - Lösungsmatrix Volltextsuche über alles

#### Suche auf die Parameternamen

Mit der Suche nur nach Parameternamen, kann jemand, der genau weiss, nach was er suchen muss, sehr schnell fündig werden.

|  |  |
| --- | --- |
| 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 |

Tabelle 80 - Lösungsmatrix Suche auf die Parameternamen

#### Suche auf Parameternamen und Parameterwert

Mit der Suche nach Parameternamen und Parameterwert, behält man die Vorteile der Suche nach dem Parameternamen, fügt aber noch die Möglichkeit hinzu, z.B. bei Personalwechsel eine E-Mailadresse in den Parametern schnell zu finden.

|  |  |
| --- | --- |
| 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 |

Tabelle 81 - Lösungsmatrix Suche auf Parameternamen und Parameterwert

### 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 Defaultwert 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. Ansonsten werden nur die einzelnen Parameter in einer Liste angezeigt.

## Erstellung der Unittests

### Anforderung

Die Unittests müssen die wichtigsten Funktionalitäten des Programmes abdecken um Fehler zu vermeiden. Die Weitergabe von Daten via RabbitMQ/MassTransit kann mit dem zeitlich engen Rahmen dieses Projektes leider nicht getestet werden.

### Umsetzungsbeschreibung

Nur serverseitig müssen Unittests geschrieben werden, da clientseitig nur anzeigende Logik verbaut wird, bräuchte diese e2e Tests. Unittests sind zu wenig aussagekräftig für diesen Anwendungsfall, weshalb sich der Aufwand nicht lohnt. Die einzige clientseitig mit Unittests überprüfte Logik ist die der Clientservices. Da im Rahmen dieser Projektarbeit aber keine neuen Services dazugekommen sind, gibt es keine mit Tests abzudeckenden Clientlogik.

## Durchführen der Tests

### Anforderung

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

### Umsetzungsbeschreibung

Das Kapitel Testing wird geschrieben als Testprotokoll.

## Getting Started Dokumentation

### Anforderung

Die Anforderungen an diese Dokumentation sind aus der Aufgabenstellung her genau definiert. Es gilt eine «Step by Step» Anleitung zu schreiben. Diese soll einen konzeptionellen Überblick verschaffen und mindestens 2 Screenshots des GUIs und 2 Beispiele für die Validierung enthalten.

### Umsetzungsbeschreibung

Das Getting Started ist sinnvollerweise in Word geschrieben. Für diese Projektarbeit wird die Getting Started in dieser Dokumentation eingefügt.

# Realisierung

Das Fehlerhandling während der gesamten Realisierung ist nach dem Fail-Fast Pattern aufgebaut. So soll ein unerwarteter Fehler definitiv zum Absturz führen. Allgemein wurde vor allem Null Refference Exceptions abgefangen. Speichern ist nur nach der Validierung möglich. Falls ein falscher Wert z.B. über XSS übergeben wird, so passiert am Client nichts und der Parameter wird nicht gespeichert, da die Validierung fehlschlägt.

## Erstellung des Testkonzepts

|  |
| --- |
| Testfall |
| Ein Benutzer kann einen Parameter einsehen |
| Ein Benutzer kann einen Parameter speichern |
| Ein Benutzer kann nach einem Parameter suchen und findet den entsprechenden Parameter |
| Die gefundenen Parameter sind hervorgehoben |
| Die Webapplikation ist mit der Tastatur bedienbar |
| Der Titel des Parameters ist klar ersichtlich |
| Der Wert des Parameters ist ersichtlich |
| Ein allfälliger Standardwert des Parameters ist ersichtlich |
| Parameter, die über ihn verfügen, können einen Hilfetext anzeigen. |
| Die Unittests laufen fehlerfrei durch. |
| Layoutrichtlinien wurden eingehalten |
| Es können keine falschen Parameter gespeichert werden |

Tabelle 82 - Testfälle

Die Testfälle müssen jeweils ausgeführt werden. Ist die Funktionalität hinter einem Testfall noch nicht vorhanden, so wird dies im Testprotokoll festgehalten. Die Testfälle sind Use-Case Tests der Applikation. Tritt während des Testens ein Fehler auf, so ist dieser gleich zu beheben.

## Anzeige & Speichern der Parameter

### Frontend

Die Umsetzung im Frontend ist einfach aufgebaut und intuitiv. Man hat einen Parameter pro Zeile. Wird der Fokus auf das Inputfeld gelegt, so wird der Bearbeitungsmodus gesetzt und der «Speichern»- respektive «Abbrechen»-Knopf erscheint. Diese verfügen über die vermutete Funktionalität. Speichern speichert den Parameter, Abbrechen bricht ab und setzt den Wert des Parameters auf den vom Server geladenen Wert. Dieses Ergebnis wird erreicht, indem das Inputfeld an eine separate Variable gebunden wird. So wird das Serverobjekt nur verändert, wenn gespeichert wurde. Springt man aus dem Inputfeld heraus, so wird ebenfalls der ungespeicherte Text zurückgesetzt. Der Hinweistext des Parameters ist als Title-Attribut des Parameters ausgegeben. So wird der Hinweistext beim Mouseover angezeigt. Es kann jeweils nur genau ein Parameter im Bearbeitungsmodus sein. Dies wird via einem «static Subject» erreicht. Dies erlaubt das Senden einer Nachricht an alle Parameter. In dem Empfänger wird der Bearbeitungsmodus auf jedem deaktiviert. Der Event wird auf dem OnFocus Event des Inputfelds ausgelöst. Nach dem Auslösen des Events, wird der aktuelle Parameter in Bearbeitung gesetzt.

Die Gruppierung nach Service ist weggelassen, da es eine Zusatzanforderung ist und die Zeit dafür nicht ausreicht.

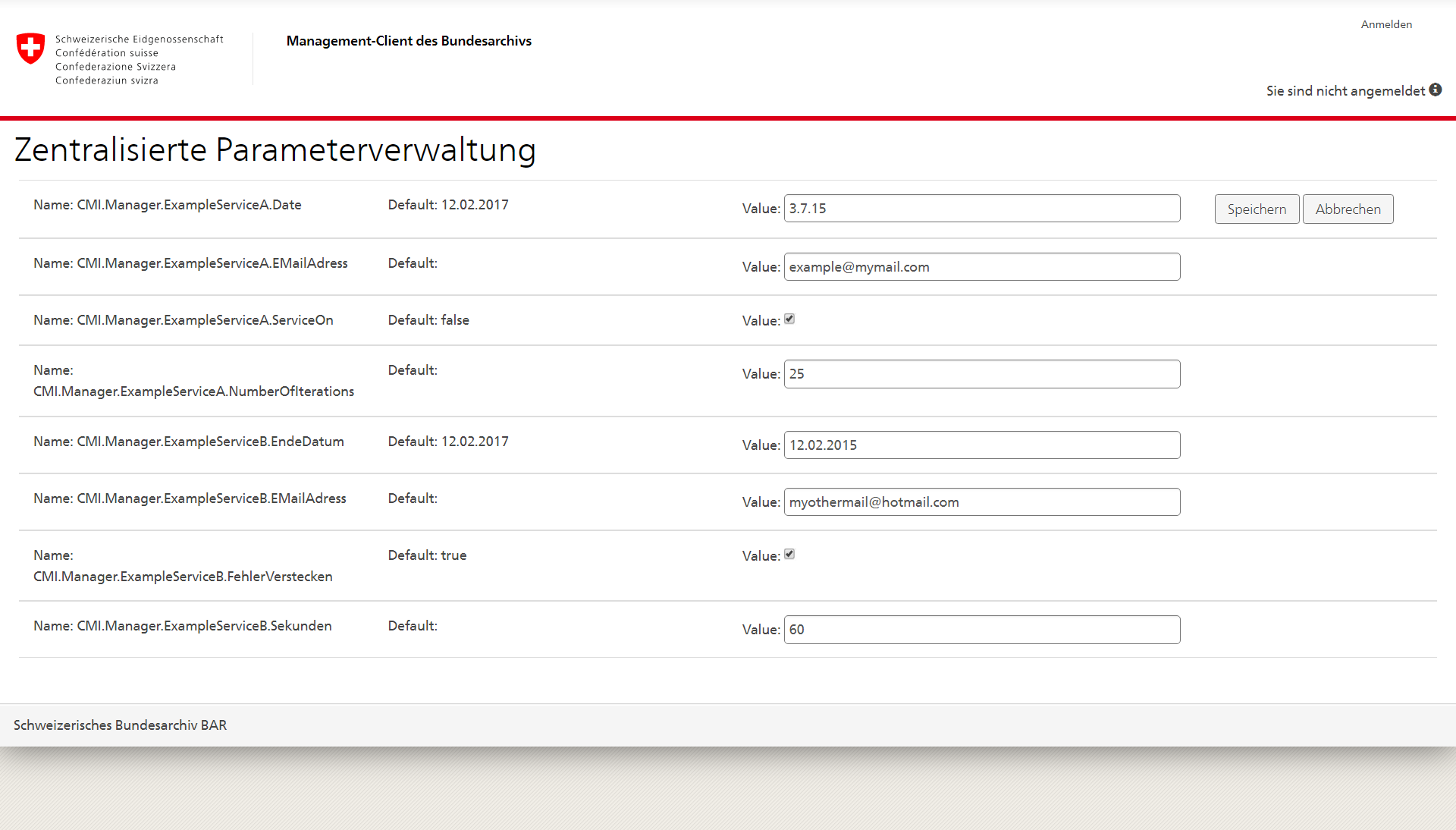


Abbildung 13 - Screenshot umgesetzte Lösung Anzeige & Speichern der Parameter

### Backend

Der Backendbereich ist Zuständig für das Speichern und Verwalten der Parameter. Es können nur alle Parameter geladen werden und nur ein Parameter auf einmal gespeichert werden. Die ganze Kommunikation sieht wie folgt aus:

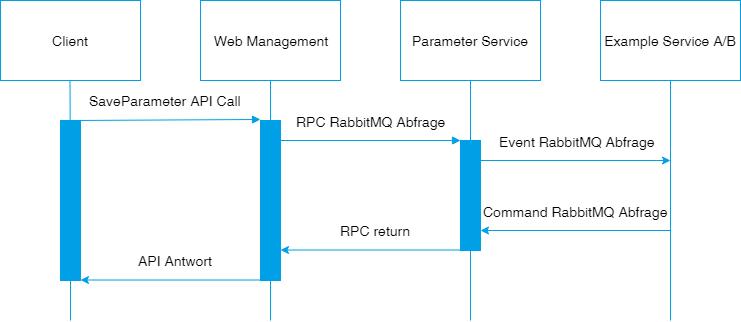


Abbildung 14 - Sequenzdiagramm Servicekommunikation beim «Save»

Example Service A und B werden beide Asynchron angesprochen. Sie melden sich beide beim Parameter Service. Dieser wartet zwischenzeitlich. Nach dem der Parameterservice 400 Millisekunden gewartet hat, gibt dieser alle gesammelten Parameter weiter an den API Controller im Web Management. Anschliessend wird die Liste von Parametern an den Client zurückgesendet.

### Technische Umsetzung Backend

Wird ein Parameter geladen, wird die API aufgerufen. Die API holt sich anschliessend via RabbitMQ alle Parameter. Dies geschieht, mit einem selbstgebautem Serialisierer. Dies ist alles in eine statische Helper Klasse ausgelagert, im CMI.Contract.Parameter. So hat jeder Service, der einen Parameter zum Parametrieren hat, auf die Helperklasse Zugriff.

### Erstellte und Bearbeitete Klassen

**Backend**

* CMI.Contract.Parameter
  + Attributes
    - DefaultAttribute
    - DescriptionAttribute
    - MandatoryAttribute
    - ValidationAttribute
  + GetParameter
    - GetParameterEventResponse
    - GetParameterResponse
  + SaveParameter
    - SaveParameterEvent
    - SaveParameterEventResponse
    - SaveParameterRequest
    - SaveParameterResponse
  + ISetting
  + Parameter
  + ParameterHelper
* CMI.Manager.ExampleServiceA
  + ExampleSettingA
  + ExampleServiceA
* CMI.Manager.ExampleServiceB
  + ExampleSettingB
  + ExampleServiceB
* CMI.Manager.Parameter
  + GetParameterEventResponseConsumer
  + GetParameterRequestConsumer
  + ParameterRequestResponseHelpers
  + ParameterService.cs
  + SaveParameterEventResponseConsumer
  + SaveParameterRequestConsumer
* CMI.Web.Management
  + Controllers
    - ParameterController

**Frontend**

* parameter
  + HTML: ParameterComponent
  + Less: ParameterComponent
  + Klasse: ParameterComponent
* parameterList
  + HTML:ParameterListComponent
  + Less: ParameterListComponent
  + Klasse: ParameterListComponent
* Klasse: parameterEntity
* services
  + Klasse: HttpService
  + Klasse ParameterService

## Implementation des Validierungsmechanismus

Der Validierungsmechanismus ist client- und serverseitig umgesetzt. Es wird beides Mal die gleiche Validierung vorgenommen. Dies ist so umgesetzt um unnötigen Traffic auf den Server zu vermeiden, da kein Request abgesetzt wird, solange es noch Fehler hat. Ist ein Fehler gefunden, wird dieser rot hinterlegt. Die Validierung wird beim Druck des Validieren-Buttons ausgelöst über alle Parameter, oder nachdem auf speichern gedrückt wird der gespeicherte Parameter.

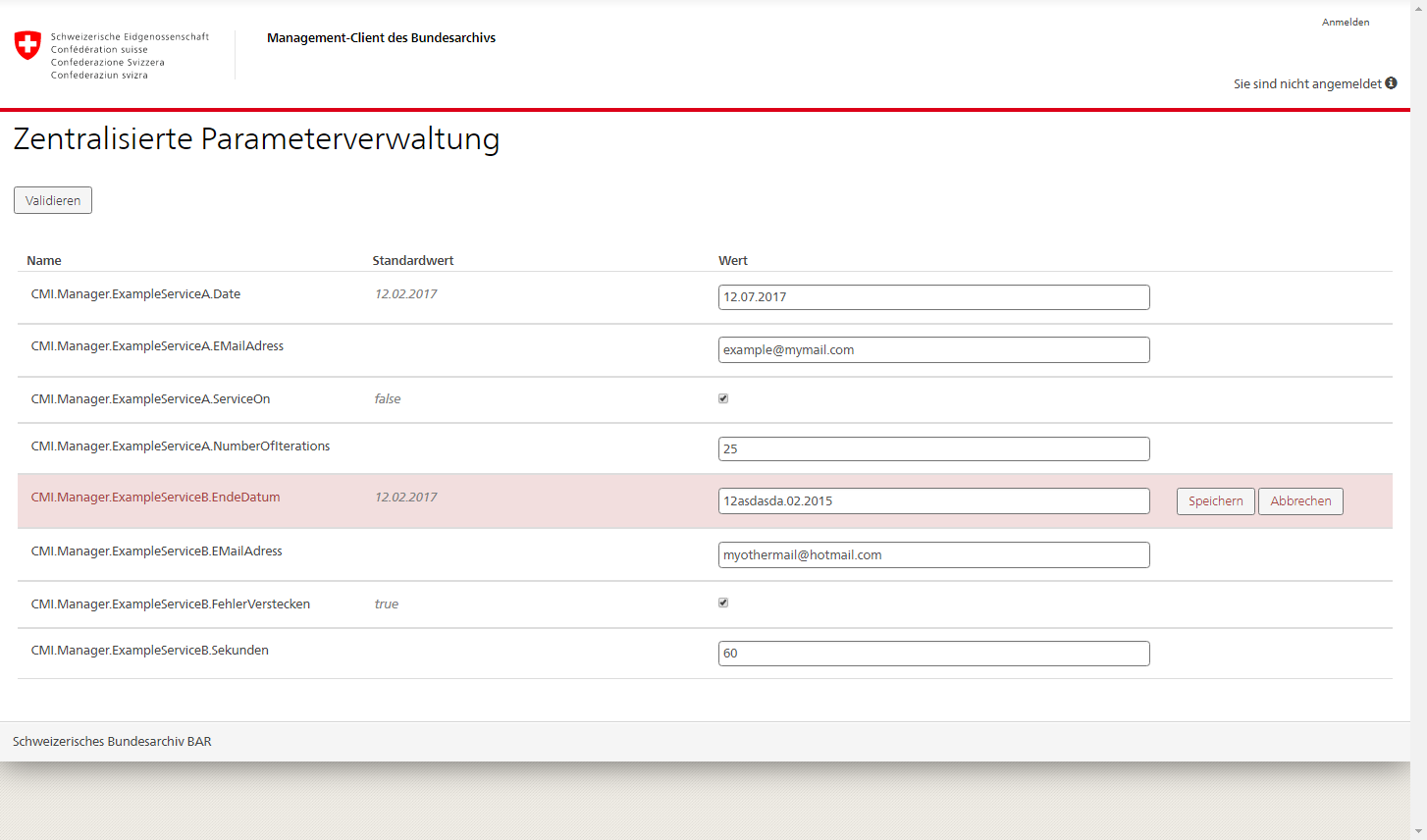


Abbildung 15 – Screenshot umgesetzte Lösung Implementation des Validierungsmechanismus

### Erstellte und bearbeitete Klassen

**Backend**

* CMI.Contract.Parameter
  + ParameterHelper.cs

**Frontend**

* parameter
  + HTML: ParameterComponent
  + Less: ParameterComponent
  + Klasse: ParameterComponent
* parameterList
  + HTML:ParameterListComponent
  + Less: ParameterListComponent
  + Klasse: ParameterListComponent

## ****Suche eines Parameters****

Die Suche funktioniert rein clientseitig, da kein Paging eingesetzt wird. Somit erzeugt die Suche kein Traffic. Die Suche geht auf den Parameternamen und den Parameterwert. Sind Treffer im Namen gefunden, wird diese Stelle gelb hervorgehoben. Wird ein Treffer im Wert gefunden, wird das ganze Inputfeld hervorgehoben. Die Suche ist nicht Case-Sensitiv.

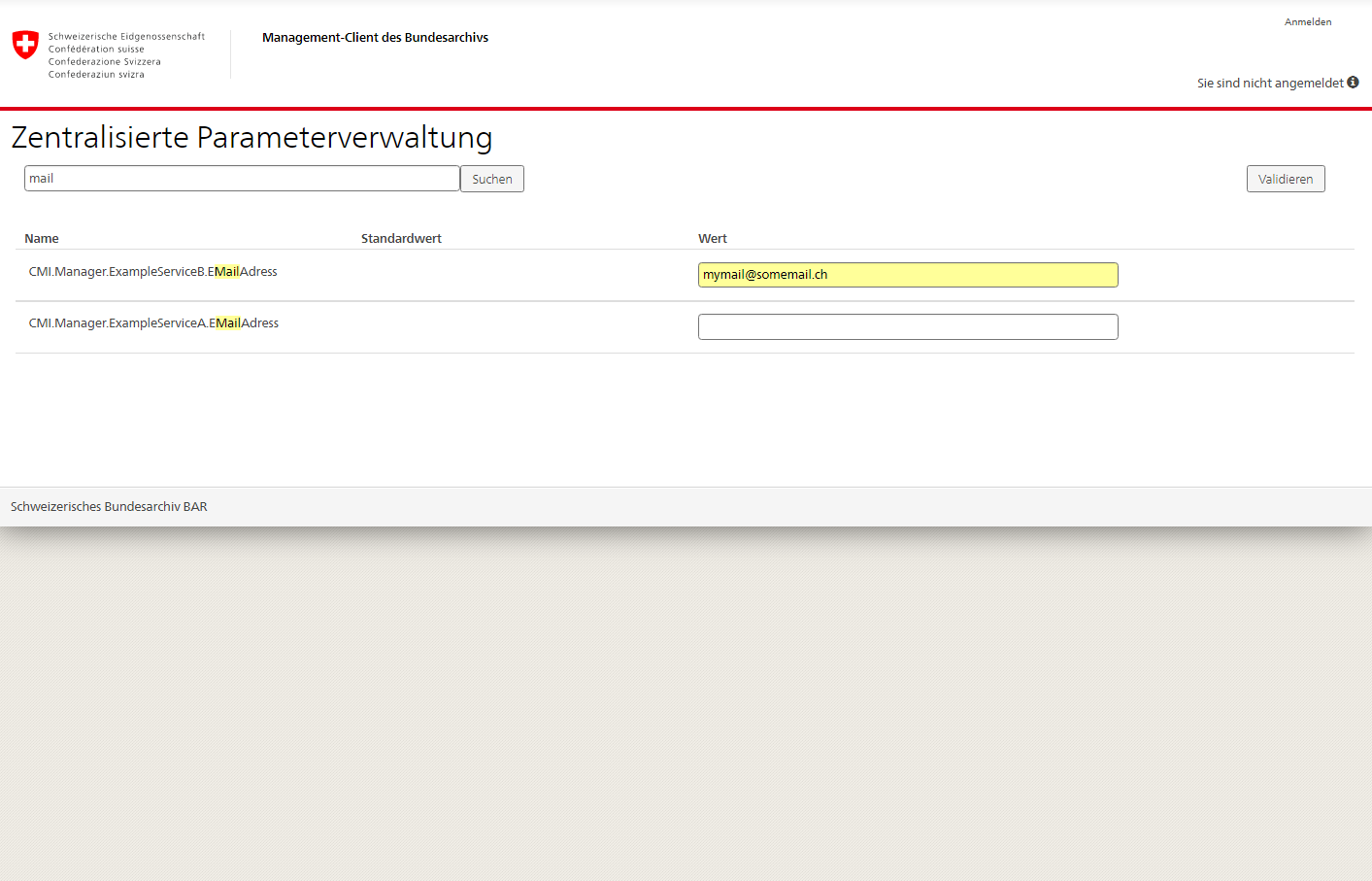


Abbildung 16 - Screenshot umgesetzte Lösung Suche eines Parameters

### Erstellte und bearbeitete Klassen

**Frontend**

* parameter
  + HTML: ParameterComponent
  + Less: ParameterComponent
  + Klasse: ParameterComponent
* parameterList
  + HTML:ParameterListComponent
  + Less: ParameterListComponent
  + Klasse: ParameterListComponent
* highlight
  + HTML: HighlightComponent
  + Less: HighlightComponent
  + Klasse: HighlightComponent

## Schreiben der Unittests

Die Unittests testen nur den CMI.Contract.Parameter Assembly. Darin wird hauptsächlich die ParameterHelper Klasse getestet, da diese alle Datenmanipulationen vornimmt. Die Unittests sind in zwei Klassen geschrieben. Eine um die Validierung zu testen, und eine um die Serialisierung und das Speichern zu testen. Für die Unittests wurden die Klassen ParameterSerializerTests und ParameterValidationTests geschrieben. Als Unittest-Setting wurde die Klasse TestSetting geschrieben. Während des Schreibens der Unittests ist mir ebenfalls noch ein kleiner Fehler in der Validierung aufgefallen, diese konnte noch eine Nullrefference Exception werfen. Dieser Fehler wurde noch kurzerhand gefixt.

### ParameterSerializerTests

|  |  |
| --- | --- |
| Test | Aussage |
| An\_empty\_setting\_can\_be\_serialized | Das Serialisieren eines neuen Settings funktioniert. |
| Value\_can\_be\_serialized\_correctly | Die Serialisierung des Werts des Settings funktioniert. |
| Default\_can\_be\_serialized\_correctly | Die Serialisierung des Default-Attribut funktioniert. |
| Type\_can\_be\_serialized\_correctly | Die Serialisierung des Typen funktioniert. |
| Description\_can\_be\_serialized\_correctly | Die Serialisierung des Description-Attribut funktioniert. |
| Mandatory\_can\_be\_serialized\_correctly | Die Serialisierung des Mandatory-Attribut funktioniert. |
| Validation\_can\_be\_serialized\_correctly | Die Serialisierung des Validation-Attribut funktioniert. |
| Name\_can\_be\_serialized\_correclty | Die Serialisierung des Namens funktioniert. |
| Can\_be\_serialized\_saved\_and\_can\_then\_be\_deserialized\_and\_got | Das Speichern und Auslesen eines validen Parameters funktioniert. |

Tabelle 83 - Aussagen ParameterSerializer Tests

### ParameterValidationTests

|  |  |
| --- | --- |
| Test | Aussage |
| ParameterList\_Should\_be\_valid | Ein valider Parameter wird als richtig validiert. |
| Empty\_value\_and\_mandatory\_should\_not\_be\_valid | Ein leerer Pflichtparameter wird als falsch validiert. |
| None\_empty\_value\_and\_mandatory\_should\_be\_valid | Ein ausgefüllter Pflichtparameter wird als richtig validiert. |
| None\_empty\_value\_that\_is\_not\_conform\_should\_not\_be\_valide | Ein falsch ausgefüllter Parameter wird als falsch validiert. |
| None\_empty\_value\_that\_is\_conform\_should\_be\_valide | Ein richtig ausgefüllter Parameter wird als richtig validiert. |
| Empty\_value\_that\_is\_mandatory\_and\_has\_a\_regex\_defined\_should\_not\_be\_valide | Ein leer ausgefüllter Pflichtparameter mit Validierung wird als falsch validiert. |
| None\_empty\_conform\_value\_that\_is\_mandatory\_and\_has\_a\_regex\_defined\_should\_be\_valide | Ein richtig ausgefüllter Pflichtparameter mit Validierung wird als richtig validiert. |
| None\_empty\_none\_conform\_value\_that\_is\_mandatory\_and\_has\_a\_regex\_defined\_should\_not\_be\_valide | Ein falsch ausgefüllter Pflichtparameter mit Validierung wird als falsch validiert. |

Tabelle 84 - Aussage ParameterValidation Tests

### Erstellte und bearbeitete Klassen

**Backend**

* CMI.Contract.Parameter.Tests
  + TestSetting
  + ParameterValidationTests
* CMI.Contract.Parameter
  + ParameterHelper

# Getting Started mit der Parameterverwaltung

## Einen eigenen Parameter schreiben

Um einen eigenen Parameter einem Service hinzuzufügen gibt es 2 Möglichkeiten.

### Ein Setting existiert und nur ein Parameter soll hinzugefügt werden

Im <MyService>Setting.cs im Service ist eine Klasse definiert, die ISetting implementiert.

1. Füge in diesem Setting den Parameter als eine Property hinzu.
2. Füge der Property die benötigten Attribute hinzu (Siehe Kapitel: «Unterstützte Attributtypen»).
3. Der Parameter kann nun parametriert werden.

### Kein Setting existiert

1. Erstelle eine neue C# Klasse <MyService>Setting.cs im Service. Lass die Klass ISettings implementieren
2. Das Interface befindet sich im CMI.Parameter.Contract, dieses muss noch im Service referenziert werden.
3. Füge in diesem Setting den Parameter als eine Property hinzu.
4. Füge der Property die benötigten Attribute hinzu (Siehe Kapitel: «Unterstützte Attributtypen»).
5. Geh zu der Start-Methode im <MyService>. Hier müssen noch die <MyService> durch den Servicenamen ersetzt werden. Die Console.Out.WriteLineAsync können durch andere Logmethoden ersetzt werden. Ändere das Skript demzufolge ab und füge in der Start-Methode folgende zwei RabbitMQ Endpunkte ein:

BusConfigurator.ConfigureBus((cfg, host) =>

{

cfg.ReceiveEndpoint(host, "GetAllParameters<MyService>", ep =>

{

ep.Handler<GetParameterEvent>(context =>

{

ParameterBusHelper.SubscribeGetEvent<<MyService>Setting>(ParameterBus);

return Console.Out.WriteLineAsync("Get Parameters");

});

});

cfg.ReceiveEndpoint(host, "SaveParameters<MyService>", ep =>

{

ep.Handler<SaveParameterEvent>(context =>

{

ParameterBusHelper.SubscribeSaveEvent<<MyService>Setting>(ParameterBus, context.Message.Parameter);

return Console.Out.WriteLineAsync("Saved Parameter");

});

});

});

## Einen Parameter/ein Setting auslesen

Ein Parameter respektive ein Setting kann wie folgt ausgelesen werden.

var setting = ParameterHelper.GetSetting(new <MySetting>Setting());

var param = setting.<MyParameter>;

## Beispielsparameter

Hier sind vier Beispiele für häufige Parameter.

public class ExampleSetting : ISetting

{

[Default("false")]

public bool ExampleBool;

[Default("max.muster@supermail.ch")]

[Validation(@"([a-zA-Z0-9\_.]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.])")]

public string ExampleEMail;

[Validation(@"((0[1-9]|[12][0-9]|3[01])[- /.](0[1-9]|1[012])[- /.]\d\d\d\d)")]

public string ExampleDate;

[Mandatory]

[Description("You can here describe your number!")]

public int ExampleNumber;

}

### Unterstützte Parametertypen

Es können String, Boolean, Int, Long, Float und Double verwendet werden. Es sind keine anderen Typen unterstützt.

### Unterstützte Attributtypen

Mandatory (Muss Feld, wenn gesetzt), Default (Standardwert Hinweis für den Benutzer), Validation (Regex für die Validierung), Description (Beschreibung des Parameters für den Hilfetext)

## Resultat:

Alle Parameter werden angezeigt

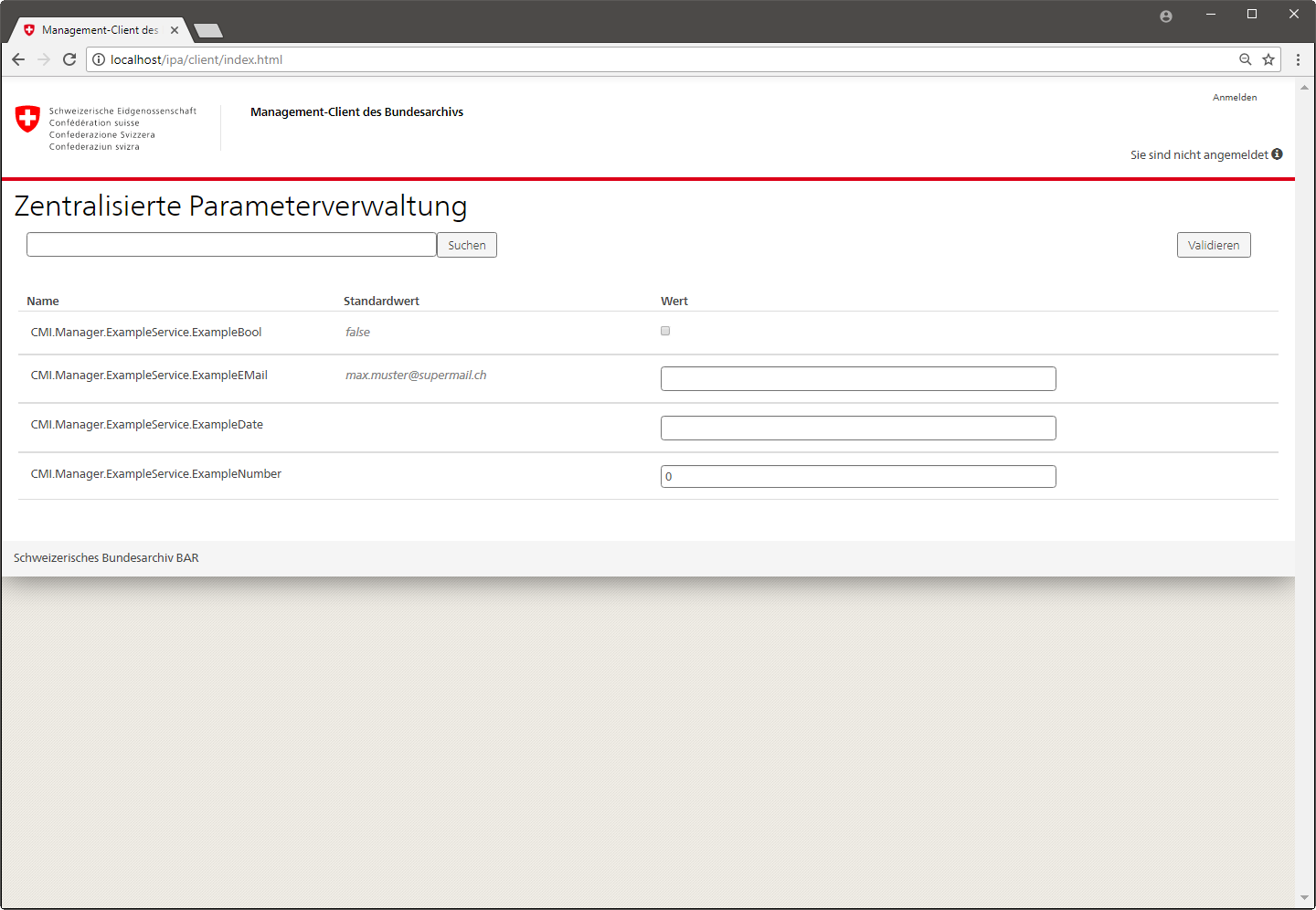


Abbildung 17 - Screenshot Getting Started

Die Validierung verhindert das Speichern eines falschen Parameters

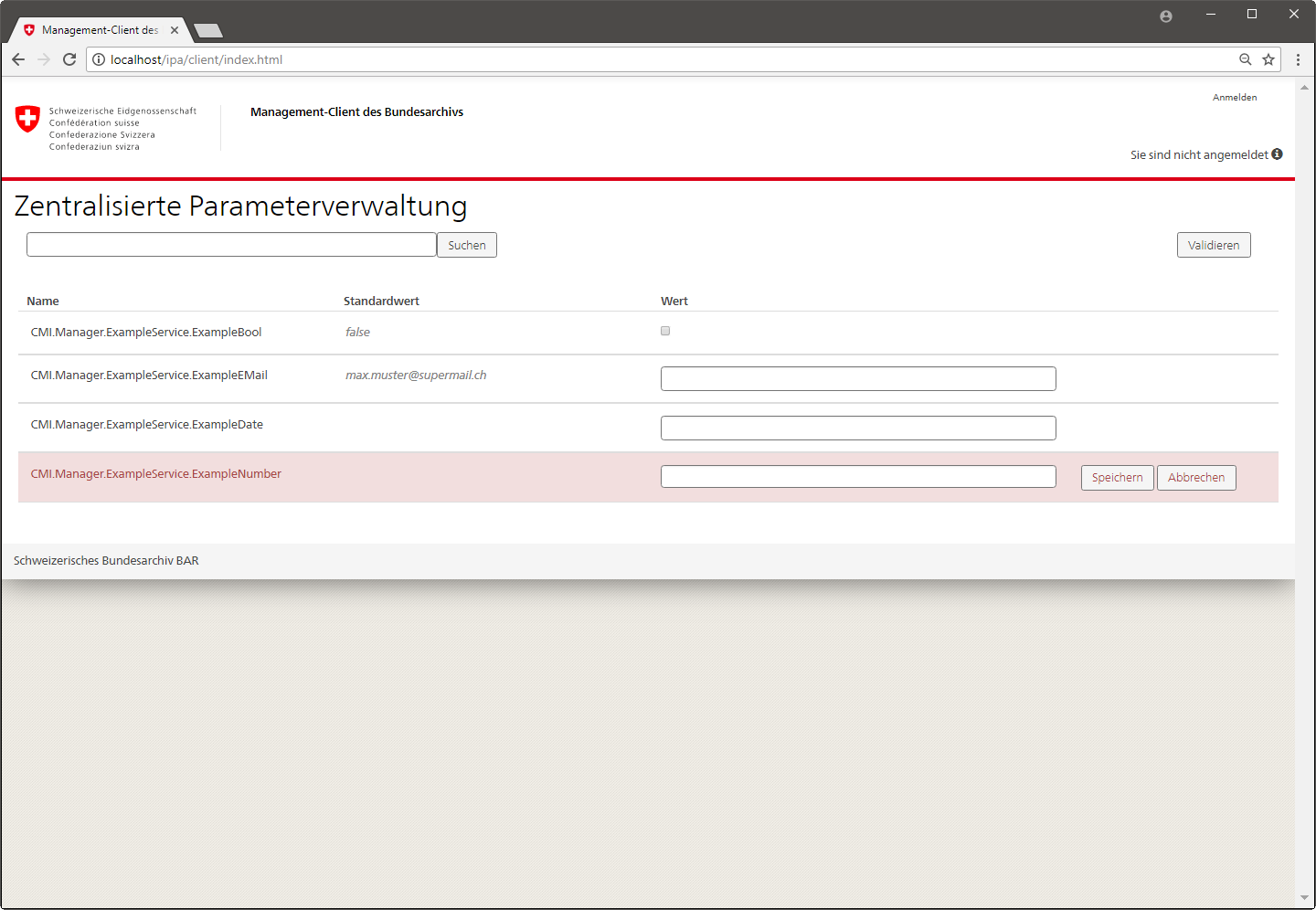


Abbildung 18 - Screenshot Getting Started Validierung

# Testing

## Testdurchgang nach Task Anzeige & Speichern der Parameter

Durch diesen Testdurchgang wurde noch ein Bug gefunden, welcher bei Booleanwerten auftrat. Wegen der Gross-/Kleinschreibung waren alle Checkboxen nie angekreuzt. Der Fehler wurde kurzerhand gefixt. Die Anpassung am CSS wurde ebenfalls noch im Rahmen der Korrekturen dieses Testes gemacht.

|  |  |
| --- | --- |
| Testfall | Resultat |
| Ein Benutzer kann einen Parameter einsehen | Funktioniert nachdem ein kleiner Bug mit Booleanwerten gefixt wurde. |
| Ein Benutzer kann einen Parameter speichern | Funktioniert |
| Ein Benutzer kann nach einem Parameter suchen und findet den entsprechenden Parameter | Noch nicht umgesetzt |
| Die gefundenen Parameter sind hervorgehoben | Noch nicht umgesetzt |
| Die Webapplikation ist mit der Tastatur bedienbar | Funktioniert |
| Der Titel des Parameters ist klar ersichtlich | Funktioniert |
| Der Wert des Parameters ist ersichtlich | Funktioniert |
| Ein allfälliger Standardwert des Parameters ist ersichtlich | Funktioniert |
| Parameter, die über ihn verfügen, können einen Hilfetext anzeigen. | Funktioniert |
| Die Unittests laufen fehlerfrei durch. | Noch nicht umgesetzt |
| Layoutrichtlinien wurden eingehalten | Noch nicht ganz, es klebt alles noch am oberen Rand. Ansonsten ist es eingehalten. |
| Es können keine falschen Parameter gespeichert werden | Noch nicht umgesetzt |

Tabelle 85 - Resultate Testdurchgang 1

## Testdurchgang nach Task Implementation des Validierungsmechanismus

Während der heutigen Entwicklungsarbeiten wurde eine zwischenzeitliche Änderung an dem Fokusevent vorgenommen. Dies musste ich nun wieder zurückbauen, da sonst das Fokussetzen via Tastatur nicht geht.

|  |  |
| --- | --- |
| Testfall | Resultat |
| Ein Benutzer kann einen Parameter einsehen | Funktioniert |
| Ein Benutzer kann einen Parameter speichern | Funktioniert |
| Ein Benutzer kann nach einem Parameter suchen und findet den entsprechenden Parameter | Noch nicht umgesetzt |
| Die gefundenen Parameter sind hervorgehoben | Noch nicht umgesetzt |
| Die Webapplikation ist mit der Tastatur bedienbar | Funktioniert nicht, der Bearbeitungsmodus wird nicht gesetzt. |
| Der Titel des Parameters ist klar ersichtlich | Funktioniert |
| Der Wert des Parameters ist ersichtlich | Funktioniert |
| Ein allfälliger Standardwert des Parameters ist ersichtlich | Funktioniert |
| Parameter, die über ihn verfügen, können einen Hilfetext anzeigen. | Funktioniert |
| Die Unittests laufen fehlerfrei durch. | Noch nicht umgesetzt |
| Layoutrichtlinien wurden eingehalten | Funktioniert |
| Es können keine falschen Parameter gespeichert werden | Funktioniert |

Tabelle 86 - Resultate Testdurchgang 2

## Testdurchgang nach Task Suche eines Parameters

Es wurden in diesem Testdurchgang keine Fehler gefunden.

|  |  |
| --- | --- |
| Testfall | Resultat |
| Ein Benutzer kann einen Parameter einsehen | Funktioniert |
| Ein Benutzer kann einen Parameter speichern | Funktioniert |
| Ein Benutzer kann nach einem Parameter suchen und findet den entsprechenden Parameter | Funktioniert |
| Die gefundenen Parameter sind hervorgehoben | Funktioniert |
| Die Webapplikation ist mit der Tastatur bedienbar | Funktioniert |
| Der Titel des Parameters ist klar ersichtlich | Funktioniert. |
| Der Wert des Parameters ist ersichtlich | Funktioniert |
| Ein allfälliger Standardwert des Parameters ist ersichtlich | Funktioniert |
| Parameter, die über ihn verfügen, können einen Hilfetext anzeigen. | Funktioniert |
| Die Unittests laufen fehlerfrei durch. | Noch nicht umgesetzt |
| Layoutrichtlinien wurden eingehalten | Funktioniert |
| Es können keine falschen Parameter gespeichert werden | Funktioniert |

Tabelle 87 - Resultate Testdurchgang 3

## Testdurchgang nach Task Erstellung der Unittests

Es wurden in diesem Testdurchgang keine Fehler gefunden.

|  |  |
| --- | --- |
| Testfall | Resultat |
| Ein Benutzer kann einen Parameter einsehen | Funktioniert |
| Ein Benutzer kann einen Parameter speichern | Funktioniert |
| Ein Benutzer kann nach einem Parameter suchen und findet den entsprechenden Parameter | Funktioniert |
| Die gefundenen Parameter sind hervorgehoben | Funktioniert |
| Die Webapplikation ist mit der Tastatur bedienbar | Funktioniert |
| Der Titel des Parameters ist klar ersichtlich | Funktioniert |
| Der Wert des Parameters ist ersichtlich | Funktioniert |
| Ein allfälliger Standardwert des Parameters ist ersichtlich | Funktioniert |
| Parameter, die über ihn verfügen, können einen Hilfetext anzeigen. | Funktioniert |
| Die Unittests laufen fehlerfrei durch. | Funktioniert, |
| Layoutrichtlinien wurden eingehalten | Funktioniert |
| Es können keine falschen Parameter gespeichert werden | Funktioniert |

Tabelle 88 - Resultate Testdurchgang 4

# Code Dokumentation Backend

## Assembly: CMI.Contract.Parameter

Die CMI.Contract.Parameter Assembly hat die Aufgabe sämtliche relevanten Informationen und Funktionen zur Verfügung zu stellen, welches die CMI.Manager.Parameter Assembly und die Assembly auf der anderen Seite der RabbitMQ brauchen.

### Klasse: ParameterHelper

Die Klasse bietet Helpermethoden zur Verarbeitung von Parametern zur Verfügung. Die Klasse ist generisch aufgebaut, sodass jede Klasse, die von ISetting erbt, hier verarbeitet, gespeichert und ausgelesen werden kann.

#### Methode: GetParameterListFromSetting

Diese Methode wandelt einen ISetting in eine Parameterliste um.

public static Parameter[] GetParameterListFromSetting (ISetting setting)

{

var paramList = new List<Parameter>();

var namePrefix = setting.GetType().Namespace;

foreach (var fieldInfo in setting.GetType().GetFields())

{

var param = CreateParameter(fieldInfo, setting, namePrefix);

if (param.Name != null)

{

paramList.Add(param);

}

else

{

throw new NullReferenceException();

}

}

return paramList.ToArray();

}

#### Methode: ValidateParameter

Diese Methoden validieren einen Parameter. Sie geben true zurück, wenn der Parameter gültig ist und false, falls er nicht gültig ist. Die Methode kommt mit zwei Overloads, einen für einen einzelnen Parameter und einen für eine Liste von Parametern.

public static bool ValidateParameter(Parameter parameter)

{

if (string.IsNullOrEmpty(parameter.Value) && parameter.Mandatory)

{

return false;

}

if (parameter.RegexValidation == null || parameter.Value == null)

{

return true;

}

var regex = new Regex(parameter.RegexValidation);

var match = regex.IsMatch(parameter.Value);

return match;

}

public static bool ValidateParameter(Parameter[] parameters)

{

return parameters.All(ValidateParameter);

}

#### Methode: SaveSetting

Die Methode speichert ein Setting auf dem Filesystem ab, nachdem alle Parameter validiert wurden. Der Rückgabewert ist, ob es geklappt hat oder nicht.

public static bool SaveSetting(ISetting setting, Parameter[] parameters)

{

var path = GetSettingPath(setting);

var jsonString = string.Empty;

if (parameters == null)

{

if (!ValidateParameter(GetParameterListFromSetting(setting))) return false;

jsonString = GetJsonStringOfSetting(setting);

}

if (parameters != null)

{

if (!ValidateParameter(parameters)) return false;

jsonString = Newtonsoft.Json.JsonConvert.SerializeObject(parameters);

}

try

{

System.IO.File.WriteAllText(path, jsonString);

return true;

}

catch

{

return false;

}

}

#### Methode: GetSetting

Diese Methode gibt ein ISetting mit seinen Werten zurück. So ist der Aufruf mit new ExampleSettingA() z.B. möglich, um den ExampleSettingA zu bekommen im ExampleServiceA.

public static ISetting GetSetting(ISetting setting)

{

var path = GetSettingPath(setting);

if (!System.IO.File.Exists(path))

{

InitialSaveSetting(setting);

}

var jsonString = System.IO.File.ReadAllText(path);

var paramList = Newtonsoft.Json.JsonConvert.DeserializeObject<Parameter[]>(jsonString);

var namePrefix = setting.GetType().Namespace;

foreach (var fieldInfo in setting.GetType().GetFields())

{

var value = paramList.First(p => p.Name == namePrefix + "." + fieldInfo.Name)?.Value;

if (value != null)

{

fieldInfo.SetValue(setting, Convert.ChangeType(value, fieldInfo.FieldType));

}

}

return setting;

}

#### Methode: InitialSaveParameter

Diese Methode initialisiert einen Parameter, der noch nicht vorhanden ist, indem ein leereres Setting eines ISettingtyps im entsprechenden Service gespeichert wird.

private static void InitialSaveSetting(ISetting setting)

{

var path = GetSettingPath(setting);

var jsonString = GetJsonStringOfSetting(setting);

System.IO.File.WriteAllText(path, jsonString);

}

#### Methode: GetJsonStringOfSetting

Diese Methode wandelt ein Setting in einen JSON-String um.

private static string GetJsonStringOfSetting(ISetting setting)

{

var paramList = GetParameterListFromSetting(setting);

var jsonString = Newtonsoft.Json.JsonConvert.SerializeObject(paramList);

if (jsonString == null)

{

throw new NullReferenceException();

}

return jsonString;

}

#### Methode: GetSettingPath

Diese Methode gibt den Pfad eines Parameters zurück, um ihn so in die richtige Assembly zu lesen, respektive abzulegen.

private static string GetSettingPath(ISetting s)

{

var fullPath = s.GetType().Assembly.CodeBase;

var path = fullPath.Replace(fullPath.Split('/').Last(), "parameters.json");

var uri = new UriBuilder(path);

return Uri.UnescapeDataString(uri.Path);

}

#### Methode: CreateParameter

Diese Methode erstellt einen Parameter aufgrund eines ISetting und dessen FieldInfo. Der Prefix-String ist für die Assemblyzuordnung im Frontend.

private static Parameter CreateParameter(FieldInfo fieldInfo, ISetting setting, string prefix)

{

var param = new Parameter

{

Name = prefix + "." + fieldInfo.Name,

Value = fieldInfo.GetValue(setting)?.ToString(),

Type = GetType(fieldInfo.FieldType)

};

if (param.Name == null || param.Type == null) return null;

var attributes = fieldInfo.GetCustomAttributes(true);

foreach (var attribute in attributes)

{

var mandatoryAttribute = attribute as MandatoryAttribute;

var defaultAttribute = attribute as DefaultAttribute;

var validationAttribute = attribute as ValidationAttribute;

var descriptionAttribute = attribute as DescriptionAttribute;

if (mandatoryAttribute != null)

{

param.Mandatory = true;

}

if (defaultAttribute != null)

{

param.Default = defaultAttribute?.Default;

}

if (validationAttribute != null)

{

param.RegexValidation = validationAttribute?.Regex;

}

if (descriptionAttribute != null)

{

param.Description = descriptionAttribute?.Description;

}

}

return param;

}

#### Methode: GetType

Diese Methode gibt den HTML Input-Typ für den Client zurück, damit man sich nicht darum kümmern muss, wie der Parameter im Client angezeigt wird.

private static string GetType(Type type)

{

if (type.Name == "Boolean")

{

return "checkbox";

}

if (type.Name == "Int32" || type.Name == "Double" || type.Name == "Float" || type.Name == "Int64" || type.Name == "Long")

{

return "number";

}

return "text";

}

### Klasse: ParameterBusHelper

Diese Klasse stellt Hilfsfunktionen für die RabbitMQ-Kommunikation zur Verfügung. Sie beinhaltet eine Funktion für das Subscriben am Get-Event und eine für das Subscriben am Save-Event.

#### Methode: SubscribeGetEvent

Stellt die Funktionalität zur Verfügung, was man im Get-Event machen muss, damit ein Parameter vom Parameter Service ausgelesen werden kann.

public static void SubscribeGetEvent<T>(IBus parameterBus) where T:ISetting

{

var paramInstance = Activator.CreateInstance(typeof(T)) as ISetting;

paramInstance = ParameterHelper.GetSetting(paramInstance);

parameterBus.Publish(new GetParameterEventResponse { Parameters = ParameterHelper.GetParameterListFromSetting(paramInstance) });

}

#### Methode: SubscribeSaveEvent

Stellt die Funktionalität zur Verfügung, was man im Save-Event machen muss, damit ein Parameter vom Parameter Service gespeichert werden kann.

public static void SubscribeSaveEvent<T>(IBus parameterBus, Parameter paramToSave) where T:ISetting

{

var paramInstance = Activator.CreateInstance(typeof(T)) as ISetting;

paramInstance = ParameterHelper.GetSetting(paramInstance);

var paramAsParamList = ParameterHelper.GetParameterListFromSetting(paramInstance);

if(paramAsParamList.Any(p => p.Name == paramToSave.Name))

{

paramAsParamList.First(p => p.Name == paramToSave.Name).Value = paramToSave.Value;

parameterBus.Publish(new SaveParameterEventResponse { Success = ParameterHelper.SaveSetting(paramInstance, paramAsParamList) });

}

}

### Klasse: Parameter

Diese Klasse zeigt, wie ein Parameter auszusehen hat.

#### Property: Name

Der Name des Parameters.

public string Name { get; set; }

#### Property: Type

Der Typ des Parameters.

public string Type { get; set; }

#### Property: Description

Ein Hilfetext des Parameters.

public string Description { get; set; }

#### Property: Value

Der parametrierte Wert des Parameters.

public string Value { get; set; }

#### Property: Default

Die Standardeinstellung des Parameters.

public string Default { get; set; }

#### Property: RegexValidation

Der Validierungs-String des Parameters.

public string RegexValidation { get; set; }

#### Property: Mandatory

Gibt zurück, ob der Parameter zwingend ausgefüllt sein muss oder nicht.

public bool Mandatory { get; set; } = false;

### Interface: ISetting

Dieses Interface dient als gemeinsamer Nenner aller Settings. Es verfügt über keine Member.

### Klasse: BusConfigurator

Aus dem Projekt Viaduc kopiert. Diese Klasse erstellt den RabbitMQ Bus. Die Klasse enthält die ConfigureBus Methode, mit welcher man einen neuen RabbitMQ Bus erstellen kann.

### Ordner: Attributes

Dies ist ein Ordner er enthält sämtliche erstellten Attribute. Diese haben keine Logik, weshalb sie hier einfach als ganze Klassen aufgelistet werden.

#### Klasse: DefaultAttribute

Dieses Attribut ist für die Deklaration eines Standardwerts im Parameter.

[AttributeUsage(AttributeTargets.Field)]

public class DefaultAttribute : Attribute

{

public string Default { get; set; }

public DefaultAttribute(string value)

{

Default = value;

}

}

#### Klasse: DescriptionAttribute

Mit diesem Attribut kann ein Hinweistext dem Parameter mitgegeben werden.

[AttributeUsage(AttributeTargets.Field)]

public class DescriptionAttribute : Attribute

{

public string Description { get; set; }

public DescriptionAttribute(string description)

{

Description = description;

}

}

#### Klasse: MandatoryAttribute

Dieses Attribut zeigt an, wenn gesetzt, dass dies ein Pflichtfeld ist.

[AttributeUsage(AttributeTargets.Field)]

public class MandatoryAttribute : Attribute

{

}

#### Klasse: ValidationAttribute

Mit diesem Attribut kann eine Regular Expression dem Parameter zur Validierung gegeben werden.

[AttributeUsage(AttributeTargets.Field)]

public class ValidationAttribute : Attribute

{

public string Regex { get; set; }

public ValidationAttribute(string regex)

{

Regex = regex;

}

}

### Ordner: GetParameter

Dies ist ein Ordner mit allen Interfaces für die Kommunikation von RabbitMQ für den Get-Befehl. Die Klassen waren ohne Members bereits vorhanden. Sie dienen nur als Deklaration der Schnittstelle der einzelnen Microservices. Die Member werden jeweils mitgeschickt. Die Klassennamen sind sprechend für sich.

#### Klasse: GetParameterEvent

public class GetParameterEvent

{

}

#### Klasse: GetParameterEventResponse

public class GetParameterEventResponse

{

public Parameter[] Parameters { get; set; }

}

#### Klasse: GetParameterRequest

public class GetParameterRequest

{

}

#### Klasse: GetParameterResponse

public class GetParameterResponse

{

public Parameter[] Parameters { get; set; }

}

### Ordner: SaveParameter

Dies ist ein Ordner mit allen Interfaces für die Kommunikation von RabbitMQ für den Save-Befehl. Sie dienen nur als Deklaration der Schnittstelle der einzelnen Microservices. Die Member werden jeweils mitgeschickt. Die Klassennamen sind so gewählt, dass klar sein sollte, was sie tun.

#### Klasse: SaveParameterEvent

public class SaveParameterEvent

{

public Parameter Parameter { get; set; }

public SaveParameterEvent(Parameter parameter)

{

Parameter = parameter;

}

}

#### Klasse: SaveParameterEventResponse

public class SaveParameterEventResponse

{

public bool Success { get; set; } = false;

}

#### Klasse: SaveParameterRequest

public class SaveParameterRequest

{

public Parameter Parameter { get; set; }

public SaveParameterRequest(Parameter parameter)

{

Parameter = parameter;

}

}

#### Klasse: SaveParameterResponse

public class SaveParameterResponse

{

public bool Success { get; set; } = false;

}

## Assembly: CMI.Host.ExampleServiceA

Dieses Projekt dient dazu den Windowsdienst zu starten und zu beenden. Es beinhaltet keine Logik und ist in der Vorbereitung auf die IPA bereits vorhanden gewesen.

## Assembly: CMI.Host.ExampleServiceB

Dieses Projekt dient dazu den Windowsdienst zu starten und zu beenden. Es beinhaltet keine Logik und ist in der Vorbereitung auf die IPA bereits vorhanden gewesen.

## Assembly: CMI.Host.Parameter

Dieses Projekt dient dazu den Windowsdienst zu starten und zu beenden. Es beinhaltet keine Logik und ist in der Vorbereitung auf die IPA bereits vorhanden gewesen.

## Assembly: CMI.Manager.ExampleServiceA

### Klasse: ExampleServiceA

Diese Klasse steuert das Startverhalten des Example Services A. In der Startfunktion werden sämtliche Endpoints für RabbitMQ initialisiert.

#### Property: ParameterBus

private IBusControl ParameterBus { get; set; }

#### Methode: Start

Diese Methode definiert das Startverhalten des Service.

public void Start()

{

ParameterBus = BusConfigurator.ConfigureBus((cfg, host) =>

{

cfg.ReceiveEndpoint(host, "GetAllParametersA", ep =>

{

ep.Handler<GetParameterEvent>(context =>

{

ParameterBusHelper.SubscribeGetEvent<ExampleSettingA>(ParameterBus);

return Console.Out.WriteLineAsync("Get Parameters");

});

});

cfg.ReceiveEndpoint(host, "SaveParametersA", ep =>

{

ep.Handler<SaveParameterEvent>(context =>

{

ParameterBusHelper.SubscribeSaveEvent<ExampleSettingA>(ParameterBus, context.Message.Parameter);

return Console.Out.WriteLineAsync("Saved Parameter");

});

});

});

ParameterBus.Start();

}

#### Methode: Stop

public void Stop()

{

}

### Klasse: ExampleSettingA

Diese Klasse erbt vom ISetting Interface damit man den Parameter serialisieren kann. Dies ist ein Testparameter zur Demonstration und zum Testen.

#### Property: Date

[Mandatory]

[Default("12.02.2017")]

public string Date;

#### Property: EMailAdress

[Description("Diese Email dient nur zu Demonstrationszwecken.")]

[Validation(@"([a-zA-Z0-9\_.]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.])")]

public string EMailAdress;

#### Property: ServiceOn

[Default("false")]

public bool ServiceOn;

NumberOfIterations

[Validation(@"[0-9]|[0-9][0-9]")]

[Mandatory]

public int NumberOfIterations;

## Assembly: CMI.Manager.ExampleServiceB

### Klasse: ExampleServiceB

Diese Klasse steuert das Starverhalten des Example Services B. In der Startfunktion werden sämtliche Endpoints für RabbitMQ initalisiert.

#### Property: ParameterBus

Businstanz für die RabbitMQ Kommunikation.

private IBusControl ParameterBus { get; set; }

#### Methode: Start

Diese Methode definiert das Startverhalten des Service.

public void Start()

{

ParameterBus = BusConfigurator.ConfigureBus((cfg, host) =>

{

cfg.ReceiveEndpoint(host, "GetAllParametersB", ep =>

{

ep.Handler<GetParameterEvent>(context =>

{

ParameterBusHelper.SubscribeGetEvent<ExampleSettingB>(ParameterBus);

return Console.Out.WriteLineAsync("Get Parameters");

});

});

cfg.ReceiveEndpoint(host, "SaveParametersB", ep =>

{

ep.Handler<SaveParameterEvent>(context =>

{

ParameterBusHelper.SubscribeSaveEvent<ExampleSettingB>(ParameterBus, context.Message.Parameter);

return Console.Out.WriteLineAsync("Saved Parameter");

});

});

});

ParameterBus.Start();

}

#### Methode: Stop

public void Stop()

{

}

### Klasse: ExampleSettingB

Diese Klasse erbt vom ISetting Interface damit man den Parameter serialisieren kann. Dies ist ein Testparameter zur Demonstration und zum Testen.

#### Property: EndeDatum

[Mandatory]

[Default("12.02.2017")]

[Validation(@"((0[1-9]|[12][0-9]|3[01])[- /.](0[1-9]|1[012])[-/.]\d\d\d\d)")]

public string EndeDatum;

#### Property: EMailAdress

[Description("Diese Email dient nur zu Demonstrationszwecken.")]

[Validation(@"([a-zA-Z0-9\_.]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.])")]

public string EMailAdress;

#### Property: FehlerVerstecken

[Default("true")]

public bool FehlerVerstecken;

#### Property: Sekunden

[Validation(@"[0-9]|[0-5][0-9]")]

[Mandatory]

public int Sekunden;

## Assembly: CMI.Manager.Parameter

### Klasse: GetParameterEventResponseConsumer

Dies Klasse enthält die Logik, welche im Parameter Service ausgeführt wird, wenn eine Event-Antwort eines Get-Events im Parameter Service ankommt.

#### Methode: Consume

public async Task Consume(ConsumeContext<GetParameterEventResponse> context)

{

ParameterRequestResponseHelper.Parameters.AddRange(context.Message.Parameters);

await Console.Out.WriteLineAsync("Parameter Recived");

}

### Klasse: GetParameterRequestConsumer

Diese Klasse enthält die Logik, welche im Parameter Service ausgeführt wird, wenn ein GetAllParameter Request ausgelöst wird.

#### Methode: Consume

public Task Consume(ConsumeContext<GetParameterRequest> context)

{

ParameterRequestResponseHelper.Parameters = new List<Contract.Parameter.Parameter>();

ParameterService.ParameterBus.Publish(new GetParameterEvent());

Console.Out.WriteLineAsync("Get Event started");

Thread.Sleep(400);

if (ParameterRequestResponseHelper.Parameters.Count == 0)

{

Thread.Sleep(6000);

}

context.RespondAsync(new GetParameterResponse()

{

Parameters = ParameterRequestResponseHelper.Parameters.ToArray()

});

return Console.Out.WriteLineAsync("Get Event response sent");

}

}

### Klasse: ParameterRequestResponseHelper

Dies ist eine Helperklasse für die Eventkommunikation. In RabbitMQ gibt es keine Möglichkeit, eine Antwort eines Events für jeden Subscriber zu erhalten. Deshalb gibt einen Event vom zu parametrierenden Service aus einen Event zurück, welcher dann die Antworten dieser Services im Parameter Service speichern.

#### Property: Parameters

public static List<Contract.Parameter.Parameter> Parameters { get; set; } = new List<Contract.Parameter.Parameter>();

#### Property: SavedSuccessfully

public static bool? SavedSuccessfully { get; set; }

### Klasse: ParameterService

Diese Klasse regelt das Startverhalten vom Parameterservice

#### Property: ParameterBus

Businstanz für die RabbitMQ Kommunikation.

public static IBusControl ParameterBus { get; set; }

#### Methode: Start

Diese Methode definiert das Startverhalten des Service.

public void Start()

{

ParameterBus = BusConfigurator.ConfigureBus((cfg, host) =>

{

cfg.ReceiveEndpoint("GetParameterQueue", ec =>

{

ec.Consumer(() => new GetParameterRequestConsumer());

});

cfg.ReceiveEndpoint("GetResponseParameterEventQueue", ec =>

{

ec.Consumer(() => new GetParameterEventResponseConsumer());

});

cfg.ReceiveEndpoint("SaveParameterQueue", ec =>

{

ec.Consumer(() => new SaveParameterRequestConsumer());

});

cfg.ReceiveEndpoint("SaveResponseParameterEventQueue", ec =>

{

ec.Consumer(() => new SaveParameterEventResponseConsumer());

});

});

ParameterBus.Start();

}

#### Methode: Stop

public void Stop()

{

ParameterBus.Stop();

}

### Klasse: SaveParameterEventResponseConsumer

Diese Klasse enthält die Logik, welche im Parameter Service ausgeführt wird, wenn eine Event-Antwort eines Save-Events im Parameter Service ankommt.

#### Methode: Consume

public async Task Consume(ConsumeContext<SaveParameterEventResponse> context)

{

ParameterRequestResponseHelper.SavedSuccessfully = context.Message.Success;

if (context.Message.Success)

{

await Console.Out.WriteLineAsync("Saved Successfully");

}

else

{

await Console.Out.WriteLineAsync("An Error has occured while saving!");

}

}

### Klasse: SaveParameterRequestConsumer

Dies ist eine Helperklasse für die Eventkommunikation. In RabbitMQ gibt es keine Möglichkeit, eine Antwort eines Events für jeden Subscriber zu erhalten. Deshalb gibt der Save-Event vom zu parametrierenden Service aus einen Event zurück, welcher dann die Antworten dieser Services im Parameter Service speichern.

#### Consume

public Task Consume(ConsumeContext<SaveParameterRequest> context)

{

ParameterRequestResponseHelper.SavedSuccessfully = null;

ParameterService.ParameterBus.Publish(new SaveParameterEvent(context.Message.Parameter));

Console.Out.WriteLineAsync("Save Event started");

Thread.Sleep(400);

if (ParameterRequestResponseHelper.SavedSuccessfully == null)

{

Thread.Sleep(6000);

}

context.RespondAsync(new SaveParameterResponse()

{

Success = ParameterRequestResponseHelper.SavedSuccessfully == true

});

return Console.Out.WriteLineAsync("Save Event response sent");

}

## Assembly: CMI.Web.Management

### Ordner: Controllers

#### Klasse: ParameterController

Diese Klasse enthält die API Controllers.

##### Methode: GetAllParameters

[Route(@"~/Controllers/GetAllParameters")]

[HttpGet]

public IHttpActionResult GetAllParameters()

{

var uri = new Uri(BusHelper.ParameterBus.Address, "GetParameterQueue");

var requestClient = BusHelper.ParameterBus.CreateRequestClient<GetParameterRequest, GetParameterResponse>(uri, TimeSpan.FromSeconds(20));

var result = requestClient.Request(new GetParameterRequest()).GetAwaiter().GetResult();

return Ok(result.Parameters);

}

##### Methode: SaveParameter

[Route(@"~/Controllers/SaveParameter")]

[HttpPost]

public IHttpActionResult SaveParameter(Parameter parameter)

{

var uri = new Uri(BusHelper.ParameterBus.Address, "SaveParameterQueue");

var requestClient = BusHelper.ParameterBus.CreateRequestClient<SaveParameterRequest, SaveParameterResponse>(uri, TimeSpan.FromSeconds(20));

var result = requestClient.Request(new SaveParameterRequest(parameter)).GetAwaiter().GetResult();

return Ok(result.Success);

}

### Ordner: Helpers

#### Klasse: BusHelper

Diese Klasse stellt die RabbitMQ Bus Instanz zur Kommunikation mit dem Parameterservice zur Verfügung.

##### Property: ParameterBus

Businstanz für die RabbitMQ Kommunikation.

public static IBusControl ParameterBus { get; }

##### Klasse: BusHelper

static BusHelper()

{

ParameterBus = BusConfigurator.ConfigureBus((cfg, host) =>

{

});

ParameterBus.Start();

}

## Assembly: CMI.Contract.Parameter.Tests

### Klasse: TestSetting

Die Klasse erbt von ISetting. Sie dient zum Testen mit Unittests.

#### Property: TestFlag

[Default("false")]

[Description("Some Test")]

public bool TestFlag;

##### Property: TestMailAdress

[Default("max.muster@supermail.ch")]

[Mandatory]

[Validation(@"([a-zA-Z0-9\_.]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.])")]

public string TestMailAdress;

#### Property: TestDate

[Validation(@"((0[1-9]|[12][0-9]|3[01])[- /.](0[1-9]|1[012])[- /.]\d\d\d\d)")]

public string TestDate;

#### Property: TestNumber

[Description("Testwert")]

public int TestNumber;

#### Property: TestUnknownType

public char TestUnknownType;

#### Property: TestMandatory

[Mandatory]

public string TestMandatory;

### ParameterSerializerTests

Diese Klasse testet das Serialisieren, Deserialisieren, Speichern und Lesen der Parameter.

#### Methode: An\_empty\_setting\_can\_be\_serialized

[TestMethod]

public void An\_empty\_setting\_can\_be\_serialized()

{

var testSetting = new TestSetting();

var parameterList = ParameterHelper.GetParameterListFromSetting(testSetting);

Assert.AreEqual(parameterList.Length, 6);

}

#### Methode: Name\_can\_be\_serialized\_correclty

[TestMethod]

public void Name\_can\_be\_serialized\_correclty()

{

var testSetting = new TestSetting();

var pl = ParameterHelper.GetParameterListFromSetting(testSetting);

Assert.IsTrue(pl.Any(p => p.Name == "CMI.Contract.Parameter.Tests.TestDate"));

Assert.IsTrue(pl.Any(p => p.Name == "CMI.Contract.Parameter.Tests.TestMailAdress"));

Assert.IsTrue(pl.Any(p => p.Name == "CMI.Contract.Parameter.Tests.TestNumber"));

Assert.IsTrue(pl.Any(p => p.Name == "CMI.Contract.Parameter.Tests.TestMandatory"));

Assert.IsTrue(pl.Any(p => p.Name == "CMI.Contract.Parameter.Tests.TestFlag"));

Assert.IsTrue(pl.Any(p => p.Name == "CMI.Contract.Parameter.Tests.TestUnknownType"));

}

#### Methode: Type\_can\_be\_serialized\_correctly

[TestMethod]

public void Type\_can\_be\_serialized\_correctly()

{

var testSetting = new TestSetting();

var pl = ParameterHelper.GetParameterListFromSetting(testSetting);

Assert.IsTrue(pl.Any(p => p.Name.EndsWith("TestFlag") && p.Type == "checkbox"));

Assert.IsTrue(pl.Any(p => p.Name.EndsWith("TestMailAdress") && p.Type == "text"));

Assert.IsTrue(pl.Any(p => p.Name.EndsWith("TestDate") && p.Type == "text"));

Assert.IsTrue(pl.Any(p => p.Name.EndsWith("TestUnknownType") && p.Type == "text"));

Assert.IsTrue(pl.Any(p => p.Name.EndsWith("TestNumber") && p.Type == "number"));

}

#### Methode: Default\_can\_be\_serialized\_correctly

[TestMethod]

public void Default\_can\_be\_serialized\_correctly()

{

var testSetting = new TestSetting();

var pl = ParameterHelper.GetParameterListFromSetting(testSetting);

Assert.IsTrue(pl.Any(p => p.Name.EndsWith("TestFlag") && p.Default == "false"));

Assert.IsTrue(pl.Any(p => p.Name.EndsWith("TestMailAdress") && p.Default == "max.muster@supermail.ch"));

Assert.AreEqual(pl.Count(p => p.Default != null), 2);

Assert.AreEqual(pl.Count(p => p.Default == null), 4);

}

#### Methode: Value\_can\_be\_serialized\_correctly

[TestMethod]

public void Value\_can\_be\_serialized\_correctly()

{

var testSetting = new TestSetting

{

TestDate = "03.04.2018",

TestFlag = true,

TestMailAdress = "testmail@mail.ch",

TestNumber = 20,

TestMandatory = "Test"

};

var pl = ParameterHelper.GetParameterListFromSetting(testSetting);

Assert.IsTrue(pl.Any(p => p.Name.EndsWith("TestDate") && p.Value == "03.04.2018"));

Assert.IsTrue(pl.Any(p => p.Name.EndsWith("TestFlag") && p.Value == "True"));

Assert.IsTrue(pl.Any(p => p.Name.EndsWith("TestMailAdress") && p.Value == "testmail@mail.ch"));

Assert.IsTrue(pl.Any(p => p.Name.EndsWith("TestNumber") && p.Value == "20"));

Assert.IsTrue(pl.Any(p => p.Name.EndsWith("TestMandatory") && p.Value == "Test"));

Assert.IsTrue(pl.Any(p => p.Name.EndsWith("TestUnknownType") && p.Value == new char().ToString()));

}

#### Methode: Description\_can\_be\_serialized\_correctly

[TestMethod]

public void Description\_can\_be\_serialized\_correctly()

{

var testSetting = new TestSetting();

var pl = ParameterHelper.GetParameterListFromSetting(testSetting);

Assert.IsTrue(pl.Any(p => p.Name.EndsWith("TestFlag") && p.Description == "Some Test"));

Assert.IsTrue(pl.Any(p => p.Name.EndsWith("TestNumber") && p.Description == "Testwert"));

Assert.AreEqual(pl.Count(p => p.Description != null), 2);

Assert.AreEqual(pl.Count(p => p.Description == null), 4);

}

#### Methode: Mandatory\_can\_be\_serialized\_correctly

[TestMethod]

public void Mandatory\_can\_be\_serialized\_correctly()

{

var testSetting = new TestSetting();

var pl = ParameterHelper.GetParameterListFromSetting(testSetting);

Assert.IsTrue(pl.Any(p => p.Name.EndsWith("TestMailAdress") && p.Mandatory));

Assert.IsTrue(pl.Any(p => p.Name.EndsWith("TestMandatory") && p.Mandatory));

Assert.AreEqual(pl.Count(p => p.Mandatory), 2);

Assert.AreEqual(pl.Count(p => !p.Mandatory), 4);

}

#### Methode: GetSettingPath

private static string GetSettingPath(ISetting s)

{

var fullPath = s.GetType().Assembly.CodeBase;

var path = fullPath.Replace(fullPath.Split('/').Last(), "setting.json");

var uri = new UriBuilder(path);

return Uri.UnescapeDataString(uri.Path);

}

#### Methode: Validation\_can\_be\_serialized\_correctly

[TestMethod]

public void Validation\_can\_be\_serialized\_correctly()

{

var testSetting = new TestSetting();

var pl = ParameterHelper.GetParameterListFromSetting(testSetting);

Assert.IsTrue(pl.Any(p => p.Name.EndsWith("TestMailAdress") && p.RegexValidation == @"([a-zA-Z0-9\_.]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.])"));

Assert.IsTrue(pl.Any(p => p.Name.EndsWith("TestDate") && p.RegexValidation == @"((0[1-9]|[12][0-9]|3[01])[- /.](0[1-9]|1[012])[- /.]\d\d\d\d)"));

Assert.AreEqual(pl.Count(p => p.RegexValidation != null), 2);

Assert.AreEqual(pl.Count(p => p.RegexValidation == null), 4);

}

#### Methode: Can\_be\_serialized\_saved\_and\_can\_then\_be\_deserialized\_and\_got

[TestMethod]

public void Can\_be\_serialized\_saved\_and\_can\_then\_be\_deserialized\_and\_got()

{

var testSetting = new TestSetting

{

TestDate = "03.04.2018",

TestFlag = true,

TestMailAdress = "testmail@mail.ch",

TestNumber = 20,

TestMandatory = "Test"

};

try

{

Assert.IsTrue(ParameterHelper.SaveSetting(testSetting, null));

var newTestSetting = (TestSetting) ParameterHelper.GetSetting(new TestSetting());

Assert.AreEqual(newTestSetting.TestDate, testSetting.TestDate);

Assert.AreEqual(newTestSetting.TestFlag, testSetting.TestFlag);

Assert.AreEqual(newTestSetting.TestMailAdress, testSetting.TestMailAdress);

Assert.AreEqual(newTestSetting.TestNumber, testSetting.TestNumber);

Assert.AreEqual(newTestSetting.TestMandatory, testSetting.TestMandatory);

Assert.AreEqual(newTestSetting.TestUnknownType, testSetting.TestUnknownType);

}

finally

{

var path = GetSettingPath(testSetting);

if (System.IO.File.Exists(path))

{

System.IO.File.Delete(path);

}

}

}

### Klasse: ParameterValidationTests

Diese Klasse testet die Parametervalidation.

#### Methode: ParameterList\_Should\_be\_valid

[TestMethod]

public void ParameterList\_Should\_be\_valid()

{

var testSetting = new TestSetting

{

TestDate = "03.04.2018",

TestFlag = true,

TestMailAdress = "testmail@mail.ch",

TestNumber = 20,

TestMandatory = "Test"

};

var parameterList = ParameterHelper.GetParameterListFromSetting(testSetting);

Assert.IsTrue(ParameterHelper.ValidateParameter(parameterList));

}

#### Methode: Empty\_value\_and\_mandatory\_should\_not\_be\_valid

[TestMethod]

public void Empty\_value\_and\_mandatory\_should\_not\_be\_valid()

{

var testSetting = new TestSetting();

var parameterList = ParameterHelper.GetParameterListFromSetting(testSetting);

Assert.IsFalse(ParameterHelper.ValidateParameter(parameterList.First(p => p.Name.EndsWith("TestMandatory"))));

}

#### Methode: None\_empty\_value\_and\_mandatory\_should\_be\_valid

[TestMethod]

public void None\_empty\_value\_and\_mandatory\_should\_be\_valid()

{

var testSetting = new TestSetting();

testSetting.TestMandatory = "Test";

var parameterList = ParameterHelper.GetParameterListFromSetting(testSetting);

Assert.IsTrue(ParameterHelper.ValidateParameter(parameterList.First(p => p.Name.EndsWith("TestMandatory"))));

}

#### Methode: None\_empty\_value\_that\_is\_not\_conform\_should\_not\_be\_valide

[TestMethod]

public void None\_empty\_value\_that\_is\_not\_conform\_should\_not\_be\_valide()

{

var testSetting = new TestSetting();

testSetting.TestDate = "This is no Date!";

var parameterList = ParameterHelper.GetParameterListFromSetting(testSetting);

Assert.IsFalse(ParameterHelper.ValidateParameter(parameterList.First(p => p.Name.EndsWith("TestDate"))));

}

#### Methode: None\_empty\_value\_that\_is\_conform\_should\_be\_valide

[TestMethod]

public void None\_empty\_value\_that\_is\_conform\_should\_be\_valide()

{

var testSetting = new TestSetting();

testSetting.TestDate = "03.04.2018";

var parameterList = ParameterHelper.GetParameterListFromSetting(testSetting);

Assert.IsTrue(ParameterHelper.ValidateParameter(parameterList.First(p => p.Name.EndsWith("TestDate"))));

}

#### Methode: Empty\_mandatory\_value\_with\_regex\_defined\_should\_not\_be\_valide

[TestMethod]

public void Empty\_mandatory\_value\_with\_regex\_defined\_should\_not\_be\_valide()

{

var testSetting = new TestSetting();

var parameterList = ParameterHelper.GetParameterListFromSetting(testSetting);

Assert.IsFalse(ParameterHelper.ValidateParameter(parameterList.First(p => p.Name.EndsWith("TestMailAdress"))));

}

#### Methode: None\_empty\_mandatory\_conform\_value\_with\_regex\_defined\_is\_valide

[TestMethod]

public void None\_empty\_mandatory\_conform\_value\_with\_regex\_defined\_is\_valide()

{

var testSetting = new TestSetting();

testSetting.TestMailAdress = "max.muster@supermail.ch";

var parameterList = ParameterHelper.GetParameterListFromSetting(testSetting);

Assert.IsTrue(ParameterHelper.ValidateParameter(parameterList.First(p => p.Name.EndsWith("TestMailAdress"))));

}

#### Methode: None\_empty\_unconform\_mandatory\_value\_with\_regex\_defined\_is\_not\_valide

[TestMethod]

public void None\_empty\_unconform\_mandatory\_value\_with\_regex\_defined\_is\_not\_valide()

{

var testSetting = new TestSetting();

testSetting.TestMailAdress = "This is not a mail adress!";

var parameterList = ParameterHelper.GetParameterListFromSetting(testSetting);

Assert.IsFalse(ParameterHelper.ValidateParameter(parameterList.First(p => p.Name.EndsWith("TestMailAdress"))));

}

# Code Dokumentation Frontend

## Ordner: services

Der Servicesordner beinhaltet sämtliche Services. Die Services werden via Angular Dependency Injection geladen.

### Klasse: HttpService

Dieser Service ist aus dem Viaduc Projekt übernommen. Der Service stellt die «get» und «post» Methode zur Verfügung.

### Klasse: ParameterService

Dieser Service ist für die API Aufrufe der Parameterverwaltung zuständig.

#### Methode: getAllParameters

Diese Methode gibt alle Parameter zurück.

public async getAllParameters() {

let url = this.\_createBaseUrl() + '/GetAllParameters';

return await this.\_http.get<Parameter[]>(url, this.\_http.noCaching).toPromise();

}

#### Methode: saveParameter

Diese Methode speichert einen Parameter. Die Validierung geschieht in der ParameterComponent und im Backend.

public async saveParameter(param: Parameter) {

let url = this.\_createBaseUrl() + '/SaveParameter';

return await this.\_http.post<void>(url, param, this.\_http.noCaching).toPromise();

}

#### Methode: \_createBaseUrl

Diese Methode stellt die URL der Webseite zur Verfügung. Diese Methode gibt es so auch im Projekt Viaduc. Sie wurde nur leicht abgeändert, um die URL des IPA Systems zurückzugeben.

private \_createBaseUrl(): string {

let loc = window.location;

let port = isNaN(parseInt(loc.port, 10)) ? undefined : parseInt(loc.port, 10);

let baseUrl = '' + loc.protocol + '//' + loc.hostname + (port ? ':' + port : '') + '/ipa/Controllers';

return baseUrl;

}

## Ordner: highlight

### Klasse: HighlightComponent

Diese Component stellt die Funktionalität zur Verfügung, einen Text in einem anderen Text gelb hervorzuheben.

#### Property: highlight

Der Text, welcher hervorgehoben werden soll.

@Input()

public highlight: string;

#### Property: text

Der Text, der angezeigt werden soll.

@Input()

public text: string;

#### Methode: constructor

public constructor() {}

#### Methode: getInnerHTML

Diese Methode gibt den anzuzeigenden Text zurück.

public getInnerHTML(): string {

if (this.text && this.highlight) {

let position = this.text.toLowerCase().indexOf(this.highlight.toLowerCase());

if (position !== -1) {

let innerHTML: string = '',

replaceString = this.text.substr(position, this.highlight.length),

split = this.text.split(replaceString),

last = split.pop();

for (let s of split) {

innerHTML += s;

innerHTML += '<ins>' + replaceString + '</ins>';

}

innerHTML += last;

return innerHTML;

}

}

return this.text;

}

### HTML: HighlightComponent

<span [innerHTML]="getInnerHTML()">

</span>

## Ordner: parameterManager

### Klasse: Parameter

Diese Component ist für das Mapping des Parameter Typens mit dem des Backends.

#### Property: name

name: string;

#### Property: value

value: string;

#### Property: type

type: string;

#### Property: mandatory

mandatory: boolean;

#### Property: description

description: string;

#### Property: regexValidation

regexValidation: string;

#### Property: default

default: string;

## Ordner: parameter

### Klasse: ParameterComponent

Diese Component ist zuständig für die Anzeige, Speichern und das Validieren eines Parameters.

#### Property: parameter

Diese Property ist das Parameterobjekt.

@Input()

public parameter: Parameter;

#### Property: validationEvent

Dies ist der Event für die Validierungsfunktion via Button. Der Button befindet sich auf der ParameterListComponent.

@Input()

public validationEvent: EventEmitter<void> = new EventEmitter<void>();

#### Property: searchString

Dies ist der Suchtext vom Suchfeld.

@Input()

public searchString: string;

#### Property: active

Dieses Flag zeigt an, ob das Feld in Bearbeitung ist oder nicht.

public active: boolean = false;

#### Property: value

Dies ist der Momentan geschriebene Wert im Parameterwertefeld. Es wird für die Abbrechen-Funktion benötigt.

public value: string;

#### Property: checked

Dieses Property ist das Value-Property für Boolean-Werte.

public checked: boolean;

#### Property: validationError

Dieses Flag steuert, ob der Parameter rot hinterlegt wird oder nicht.

public validationError: boolean;

#### Property: \_onFocusChange

Dies ist der Event, welcher es ermöglicht, dass nur ein Parameter gleichzeitig in Bearbeitung sein kann.

private static \_onFocusChange: Subject<string> = new Subject();

#### Methode: constructor

constructor (private \_paramService: ParameterService) {

}

#### Methode: ngOnInit

Die ngOnInit Methode enthält die Logik, welche gleich nach dem Konstruktor folgt. Hier wird der Parameter befüllt und die Events subscribed.

public ngOnInit() {

ParameterComponent.\_onFocusChange.subscribe((name) => {

if (name !== this.parameter.name) {

this.cancelEdit();

}

});

this.validationEvent.subscribe(() => {

if (this.value) {

this.validationError = !this.\_validateString(this.value);

} else {

this.validationError = !this.\_isValid();

}

});

this.value = this.parameter.value;

this.checked = this.parameter.value === 'True';

}

#### Methode: onValueChanged

Dies ist der Eventhandler des Value-Changed Event. Hier wird das Value- respektive Checked-Property gesetzt.

public onValueChanged(event: any) {

if (this.parameter.type === 'checkbox') {

this.checked = event.target.checked;

if (this.checked === (this.parameter.value === 'True')) {

this.active = false;

}

} else {

this.value = event.target.value;

}

}

#### Methode: onFocus

Dies ist der Eventhandler, der das Feld beim Fokus in Bearbeitung setzt. Die anderen Parameter werden zurückgesetzt.

public onFocus() {

ParameterComponent.\_onFocusChange.next(this.parameter.name);

this.active = true;

}

#### Methode: saveParameter

Diese Methode validiert und speichert den Parameter. Schlägt die Validierung fehl, wird der Parameter rot hervorgehoben und der Parameter wird nicht gespeichert.

public saveParameter() {

this.validationError = !this.\_validateString(this.value);

if (this.validationError === false) {

if (this.parameter.type === 'checkbox') {

this.parameter.value = this.checked.toString();

} else {

this.parameter.value = this.value;

}

this.\_paramService.saveParameter(this.parameter).then( success => this.validationError = !success);

}

}

#### Methode: cancelEdit

Diese Methode bricht das Bearbeiten ab und setzt den Parameter wieder auf den Stand, der vom Server geholt wurde.

public cancelEdit() {

this.value = this.parameter.value;

if (this.parameter.type === 'checkbox') {

this.checked = this.parameter.value === 'True';

} else {

this.value = this.parameter.value;

}

this.active = false;

}

#### Methode: \_isValid

Diese Methode validiert den Parameter.

private \_isValid(): boolean {

return this.\_validateString(this.parameter.value);

}

#### Methode: \_validateString

Diese Methode validiert einen Text mit den Validierungen des Parameters.

private \_validateString(value: string): boolean {

if (!value && this.parameter.mandatory === true) {

return false;

}

if (this.parameter && this.parameter.regexValidation && value) {

let matches = value.match(this.parameter.regexValidation);

return (matches && matches[0] !== null);

} else {

return true;

}

}

#### Methode: getErrorClass

Diese Methode gibt die CSS Klasse zurück für den Parameter, ob der Parameter einen Fehler hat oder nicht.

public getErrorClass(): string {

return this.validationError ? 'parameter-list row alert-danger' : 'parameter-list row';

}

#### Methode: getInputClass

Diese Methode gibt die CSS Klasse für einen gefundenen Suchtreffer im Inputfeld zurück.

public getInputClass(): string {

if (this.value && this.searchString) {

if (this.value.toLowerCase().indexOf(this.searchString.toLowerCase()) !== -1) {

return 'form-control highlighted';

}

}

return 'form-control';

}

### HTML: ParameterComponent

<div \*ngIf="parameter" [class]="getErrorClass()" [title]="parameter.description ? parameter.description : ''">

<div class="col-md-3">

<cmi-viaduc-highlight [text]="parameter.name" [highlight]="searchString"></cmi-viaduc-highlight>

</div>

<var class="col-md-3">

**{{**parameter.default**}}**

</var>

<div class="col-md-4">

<input [class]="getInputClass()" [type]="parameter.type" (focus)="onFocus()" (change)="onValueChanged($event)" [checked]="checked" [value]="value">

</div>

<div \*ngIf="active" class="col-md-2">

<input type="button" class="btn" value="Speichern" (click)="saveParameter()">

<input type="button" class="btn" value="Abbrechen" (click)="cancelEdit()">

</div>

</div>

### Less: ParameterComponent

.parameter-list {

border-bottom: 1px solid #ddd;

border-top: 1px solid #ddd;

margin-left: 5px;

margin-right: 5px;

div, var {

.form-control[type='text'], .form-control[type='number'] {

display: initial;

}

.form-control[type='checkbox'] {

display: initial;

width: auto;

}

.highlighted {

background: #ff9;

color: #000;

text-decoration: none;

}

}

div[class\*=col-], var[class\*=col-]{

margin-top: 1em;

margin-bottom: 1em;

}

}

## Ordner: ParameterList

### Klasse: ParameterListComponent

Diese Component dient als Liste für die Parameter. Sie enthält das Suchenfeld und den Validierungsbutton.

#### Property: loading

Diese Property verhindert, dass die Page vor dem Eintreffen der Daten geladen wird und solange ein Ladebalken angezeigt wird.

public loading: boolean = true;

#### Property: filteredParameters

Diese Property enthält alle momentan angezeigten Parameter.

public filteredParameters: Parameter[] = [];

#### Property: \_allParameters

Diese Property enthält alle vom Server geladenen Parameter.

private \_allParameters: Parameter[] = [];

#### Property: validationEvent

Dies ist der Validierungsevent.

public validationEvent: EventEmitter<void> = new EventEmitter<void>();

#### Property: searchString

Dies ist das Inputbindig des Suchfelds.

public searchString: string = '';

#### Property: searchedStringUpToDate

Dieses Flag zeigt an, ob der Suchstring der aktuellste ist oder ob er seit der letzten Suche verändert wurde.

public searchedStringUpToDate: boolean;

#### Methode: constructor

Im Konstruktor werden alle Parameter geladen.

constructor(private \_params: ParameterService) {

this.getAllParameters();

}

#### Methode: getAllParameters

Diese Methode holt alle Parameter vom Server und setzt die Liste aller Parameter und der angezeigten Parameter auf den Stand des Servers.

public async getAllParameters() {

this.\_params.getAllParameters().then(response => {

this.\_allParameters = response;

this.filteredParameters = this.\_allParameters;

this.loading = false;

});

}

#### Methode: onValueChanged

Dies ist der Event, welcher ausgelöst wird, wenn der Text im Suchfeld angepasst wurde. Der Eventhandler setzt das searchString- und searchStringUpToDate-Attribut.

public onValueChanged(event: any) {

this.searchString = event.target.value;

if (this.searchString) {

this.searchedStringUpToDate = false;

}

}

#### Methode: emitValidationEvent

Dies ist der Eventauslöser für die Validierung.

public emitValidationEvent() {

this.validationEvent.emit();

}

#### Methode: searchParam

Diese Methode filtert die Parameter nach der Eingabe im Suchfeld. Name und Value des Parameters werden überprüft.

public searchParam() {

this.filteredParameters = [];

this.searchedStringUpToDate = true;

if (this.searchString !== '') {

this.filteredParameters = this.\_allParameters.filter((param) =>

param.name.toLowerCase().indexOf(this.searchString.toLowerCase()) !== -1 || param.value && param.value.toLowerCase().indexOf(this.searchString.toLowerCase()) !== -1

);

} else {

this.filteredParameters = this.\_allParameters;

}

}

### HTML: ParameterListComponent

<div \*ngIf="!loading">

<h1>Zentralisierte Parameterverwaltung</h1>

<div [class]="getClass">

Der Parameter wurde erfolgreich gespeichert.

</div>

<div class="row">

<div class="col-md-4">

<input type="text" class="form-control" [value]="searchString" (change)="onValueChanged($event)"/>

</div>

<div class="col-md-1">

<input type="button" (click)="searchParam()" value="Suchen" class="btn"/>

</div>

<div class="col-md-6">

</div>

<div class="col-md-1">

<input type="button" (click)="emitValidationEvent()" value="Validieren" class="btn"/>

</div>

</div>

<div class="row">

<b class="col-md-3">Name</b>

<b class="col-md-3">Standardwert</b>

<b class="col-md-4">Wert</b>

</div>

<div \*ngFor="let param of filteredParameters">

<cmi-viaduc-parameter [searchString]="searchedStringUpToDate ? searchString : ''" [parameter]="param" [validationEvent]="validationEvent"></cmi-viaduc-parameter>

</div>

</div>

<div \*ngIf="loading">

<cmi-blocker class="cmi-visible cmi-fixed cmi-center cmi-shadow">

<cmi-spinner></cmi-spinner>

</cmi-blocker>

</div>

### Less: ParameterListComponent

.btn {

margin-bottom: 1em;

}

.no-display {

display: none;

}

.row {

margin: 0;

.col-md-1 {

padding: 0;

}

.col-md-4 {

padding-right: 0;

}

}

cmi-blocker {

cmi-spinner {

width: 50%;

height: 50%;

overflow: auto;

margin: auto;

position: fixed;

left: 0;

right: 0;

bottom: 0;

top: 40%;

}

}

# Schlusswort

## Ausblick auf nach der Arbeit

In der Woche nach Abschluss der Arbeit, muss ich das Ganze noch ins Livesystem einbinden. Während diesem Schritt könnte man noch die Gruppierung nach Services einbauen, die zwar geplant war aber als Kann-Anforderung keinen Platz mehr fand. Etwas anderes, was man noch einbauen könnte, wäre eine «Speichern erfolgreich» Meldung. Die IPA könnte auch als Grundlage für eine zentrale Verwaltungsmöglichkeit für das CMIAXIOMA und CMISTAR sein. Dies sind unsere Geschäfts- und Protokollverwaltungslösungen respektive unsere Archivlösungen. Wir haben vor diese von dem einen grossen Service, den wir momentan noch haben, in mehrere kleine aufzuteilen. Da können die gesammelten Erfahrungen dieser IPA sicherlich wieder zum Einsatz kommen.

## Reflexion

Ich habe die IPA mit grosser Nervosität angefangen. Diese Nervosität konnte ich aber schnell ablegen und mich voll und ganz auf die Arbeit konzentrieren.

Das Dokumentieren ging besser als erwartet. Ich rechnete damit, dass ich grosse Mühe haben werde mit der Dokumentation, denn ich war bisher in Schulprojekten eher gut im Programmieren und eher weniger gut im Dokumentieren. Durch das IPERKA Vorgehensmodell konnte ich mich während der Planung- und Entscheidungsphasen wirklich auf das Planen und Entscheiden konzentrieren. Ich habe mich an einigen Orten während dieser Phase zurückerinnern können an das erste und zweite Lehrjahr, in denen ich die verschiedensten Diagramme kennengelernt habe. Dies half mir sehr einen Anhaltspunkt für das Ganze zu finden. In der Lehre konnte ich oftmals konzeptionell Probleme analysieren. Ich denke, dass mir dies enorm half um die grosse Aufgabe gut zu analysieren und zu zerlegen.

Wegen des Programmierens habe ich überhaupt diese Lehre angefangen. Deshalb verwundert es wohl nicht gross, dass mir das Programmieren am meisten Spass gemacht hat. Ich glaube mir ist der generische Ansatz gut gelungen in der Umsetzung. Ich konnte hier auf das Wissen zurückgreifen, welches ich während knapp vier Jahren bei der CM Informatik AG und der Greenshare AG aufgebaut habe. Das Programmieren der Reflection in der ParameterHelper Klasse habe ich sehr spannend gefunden. Die Zeit ist während der Realisierungsphase gefühlt extrem schnell vorbeigegangen.

Das Korrigieren und Auswerten zum Schluss zog sich noch ein wenig. Durch das Gegenlesen sind jeweils einige Rechtschreibfehler zum Vorschein gekommen.

Ich denke diese Arbeit wiederspiegelt gut, was ich in den letzten Jahren gelernt habe. Während dem Schreiben dieser Arbeit konnte ich viel Gelerntes nochmals vertiefen. Ich konnte viel neues Wissen über die Funktionsweise von RabbitMQ, Masstransit und vielen anderen Aspekten vom Viaduc Projekt gewinnen. Ich bin da hauptsächlich im Frontend unterwegs, wo ich selbst meine Stärke sehe. Es war jedoch ein sehr spannender Einblick ins Backend und die Problemstellungen, die sich da stellen. Ich gebe die Arbeit mit verschiedene Emotionen ab. Ich bin froh, dass diese Prüfung endlich vorbei ist und ich bald das EFZ Zeugnis in den Händen halten darf, schaue aber auch zurück und sehe all die schönen Erfahrungen, die ich während der Lehre gemacht habe. Ich freue mich aber auf den noch vor mir stehenden Berufsalltag und alles was dieser mit sich bringen mag.

## Danksagung

Besonderen Dank gilt Martin Tinner, welcher mich mit vollem Engagement durch die IPA begleitet hat. Melanie Müller, für das Gegenlesen der Dokumentation und der Bereitschaft mir bei jedem Wordproblem zu helfen. Matthias Hess und Benjamin Schäublin, welche mir die letzten Jahre viel über das Programmieren beibrachten und ich während der IPA bei Fragen angehen durfte. Zuletzt Doris Fux, sie hat mir durch die Lehre hindurch gezeigt, dass der Informatiker-Beruf nicht nur programmieren ist, sondern auch noch viel Interesse für die Materie hinter dem Programm braucht.

# Verzeichnisse

## Glossar

|  |  |
| --- | --- |
| Begriff | Erklärung |
| Angular | Clientframework für «Single Page Applikationen». |
| API | Serverseitige Funktion. |
| Backend | Serverteil des Programms, ist für die Datenverarbeitung im Hintergrund verantwortlich |
| Bootstrap | CSS Framework |
| Branch | Einen Stand des Source Codes, der unabhängig vom restlichen Entwicklungsstand ist zur Implementierung eines Features ohne Einfluss von aussen. |
| Camel-Case | Eine Möglichkeit etwas zu benennen. DieserSatzStehtInCamelCase |
| CMIAXIOMA | Geschäfts- und Protokollverwaltungslösung der CM Informatik AG. |
| CMISTAR | Archivlösung der CM Informatik AG. |
| Commit | Änderungen ins Git übernehmen. |
| Contributer | Englisch der Beiträger. Im Kontext von Git / Github ein Entwickler |
| Dependency Injection | Methode um ein die Abhängigkeit eines Objekts zu übergeben. So kann ein Service z.B. von allen genutzt werden, obwohl es nur eine Instanz gibt. |
| E2E Tests | End-to-End Tests, womit das Frontend getestet werden kann. |
| Flag | Ein Ja/Nein Wert |
| Framework | Eine Programmierhilfe mit gewisser Funktionalität, sodass das Grundgerüst nicht selbst geschrieben werden muss. |
| Frontend | Benutzeroberfläche und alles, was dafür berechnet werden muss. Interagiert mit dem Backend |
| Git / Github | Git ist ein Versionsverwaltungssystem und Github ist ein Cloud-Speicherort dafür. |
| GUI | Grafical User Interface, die Benutzerüberfläche |
| Inputbindig | Das binden einer Variable an ein Inputfeld |
| Lower-Camel-Case | Gleich wie Camel-Case nur kleiner Buchstabe am Anfang |
| MassTransit | C#.Net Anbindung für RabbitMQ |
| Mergen | Zusammenführen von 2 Branchen |
| Message Queue | Ein Stapel von Nachrichten von einem Service an den anderen. |
| Mouseover | GUI Event beim drüberfahren der Maus über das Objekt. |
| officeatwork | Tool für die Vorlagenverwaltung |
| OnFocus | GUI Event beim Anklicken / Anspringen mit Tab eines Inputelements. |
| Overload | Gleiche Methode mit verschiedenen Übergabewerten |
| Paging | Ein Software design Pattern, in welchem man eine Liste von Daten in mehreren Schritten lädt. Z.B. bei einer Google-Suche kriegt man die Ergebnisse «gepaged» zurück. Sprich auf mehreren Seiten. |
| Prefix | Ein voranstehender Text. |
| Queue | Englisch für «Reihe». Eine Kommunikation-Queues im Kontext dieser Arbeit ist ein «Stapel» voller Anweisungen von RabbitMQ. |
| RabbitMQ | Message Queue Tool zur Kommunikation der Microservices |
| Regex | Abkürzung für Regular Expression, ist ein Muster um etwas in einem Text zu finden. |
| Repository | Repository ist ein Github Projekt |
| Scrum | Agile Projektmethode |
| Snake Case | Dieser\_satz\_ist\_in\_snake\_case\_geschrieben |
| Traffic | Datenverkehr zwischen Frontend und Backend |
| Visual Studio | Programmierumgebung für C# in diesem Projekt |
| Webstorm | Programmierumgebung für TypeScript, HTML und Less in diesem Projekt |

Tabelle 89 - Glossar

## Tabellenverzeichnis

[Tabelle 1 – Dokumentablage 15](#_Toc510789118)

[Tabelle 2 - Source Code Ordnerstruktur 16](#_Toc510789119)

[Tabelle 3 - Konventionen Dokumentation 18](#_Toc510789120)

[Tabelle 4 - Code-Konventionen C# 18](#_Toc510789121)

[Tabelle 5 - Code Konventionen TypeScript 18](#_Toc510789122)

[Tabelle 6 - Layout-Konventionen 19](#_Toc510789123)

[Tabelle 7 - Unittests-Konventionen 19](#_Toc510789124)

[Tabelle 8 - Systemgrenze Erklärungen Diagramm 21](#_Toc510789125)

[Tabelle 9 - Geplante Tagesziele Tag 1 23](#_Toc510789126)

[Tabelle 10 - Tätigkeiten Tag 1 23](#_Toc510789127)

[Tabelle 11 - Probleme Tag 1 23](#_Toc510789128)

[Tabelle 12 - Hilfestellung Tag 1 23](#_Toc510789129)

[Tabelle 13 - Tagesreflexion Tag 1 23](#_Toc510789130)

[Tabelle 14 - Geplante Tagesziele Tag 2 24](#_Toc510789131)

[Tabelle 15 - Tätigkeiten Tag 2 24](#_Toc510789132)

[Tabelle 16 - Probleme Tag 2 24](#_Toc510789133)

[Tabelle 17 - Hilfestellung Tag 2 24](#_Toc510789134)

[Tabelle 18 - Tagesreflexion Tag 2 24](#_Toc510789135)

[Tabelle 19 - Geplante Tagesziele Tag 3 25](#_Toc510789136)

[Tabelle 20 - Tätigkeiten Tag 3 25](#_Toc510789137)

[Tabelle 21 - Probleme Tag 3 25](#_Toc510789138)

[Tabelle 22 - Hilfestellung Tag 3 25](#_Toc510789139)

[Tabelle 23 - Tagesreflexion Tag 3 25](#_Toc510789140)

[Tabelle 24 - Geplante Tagesziele Tag 4 26](#_Toc510789141)

[Tabelle 25 - Tätigkeiten Tag 4 26](#_Toc510789142)

[Tabelle 26 - Probleme Tag 4 26](#_Toc510789143)

[Tabelle 27 - Hilfestellung Tag 4 26](#_Toc510789144)

[Tabelle 28 - Tagesreflexion Tag 4 26](#_Toc510789145)

[Tabelle 29 - Geplante Tagesziele Tag 5 27](#_Toc510789146)

[Tabelle 30 - Tätigkeiten Tag 5 27](#_Toc510789147)

[Tabelle 31 - Probleme Tag 5 27](#_Toc510789148)

[Tabelle 32 - Hilfestellung Tag 5 27](#_Toc510789149)

[Tabelle 33 - Tagesreflexion Tag 5 27](#_Toc510789150)

[Tabelle 34 - Geplante Tagesziele Tag 6 28](#_Toc510789151)

[Tabelle 35 - Tätigkeiten Tag 6 28](#_Toc510789152)

[Tabelle 36 - Probleme Tag 6 28](#_Toc510789153)

[Tabelle 37 - Hilfestellung Tag 6 28](#_Toc510789154)

[Tabelle 38 - Tagesreflexion Tag 6 28](#_Toc510789155)

[Tabelle 39 - Geplante Tagesziele Tag 7 29](#_Toc510789156)

[Tabelle 40 - Tätigkeiten Tag 7 29](#_Toc510789157)

[Tabelle 41 - Probleme Tag 7 29](#_Toc510789158)

[Tabelle 42 - Hilfestellung Tag 7 29](#_Toc510789159)

[Tabelle 43 - Tagesreflexion Tag 7 29](#_Toc510789160)

[Tabelle 44 - Geplante Tagesziele Tag 8 30](#_Toc510789161)

[Tabelle 45 - Tätigkeiten Tag 8 30](#_Toc510789162)

[Tabelle 46 - Probleme Tag 8 30](#_Toc510789163)

[Tabelle 47 - Hilfestellung Tag 8 30](#_Toc510789164)

[Tabelle 48 - Tagesreflexion Tag 8 30](#_Toc510789165)

[Tabelle 49 - Geplante Tagesziele Tag 9 31](#_Toc510789166)

[Tabelle 50 - Tätigkeiten Tag 9 31](#_Toc510789167)

[Tabelle 51 - Probleme Tag 9 31](#_Toc510789168)

[Tabelle 52 - Hilfestellung Tag 9 31](#_Toc510789169)

[Tabelle 53 - Tagesreflexion Tag 9 31](#_Toc510789170)

[Tabelle 54 - Geplante Tagesziele Tag 10 32](#_Toc510789171)

[Tabelle 55 - Tätigkeiten Tag 10 32](#_Toc510789172)

[Tabelle 56 - Probleme Tag 10 32](#_Toc510789173)

[Tabelle 57 - Hilfestellung Tag 10 32](#_Toc510789174)

[Tabelle 58 - Tagesreflexion Tag 10 32](#_Toc510789175)

[Tabelle 59 - Tagesjournale gesehen 33](#_Toc510789176)

[Tabelle 60 - Getting Started Dokumentation Ziele 35](#_Toc510789177)

[Tabelle 61 - Getting Started Dokumentation Anforderungen 35](#_Toc510789178)

[Tabelle 62 - Anzeige & Speichern der Parameter Ziele 35](#_Toc510789179)

[Tabelle 63 - Anzeige & Speichern der Parameter Anforderungen 35](#_Toc510789180)

[Tabelle 64 - Implementation des Validierungsmechanismus Ziele 36](#_Toc510789181)

[Tabelle 65 - Implementation des Validierungsmechanismus Anforderungen 36](#_Toc510789182)

[Tabelle 66 - Suchen eines Parameters Ziele 36](#_Toc510789183)

[Tabelle 67 - Suchen eines Parameters Anforderungen 36](#_Toc510789184)

[Tabelle 68 - Erstellung der Unit Tests und des Testkonzepts Anforderungen 36](#_Toc510789185)

[Tabelle 69 - Durchführen der Tests Ziele 37](#_Toc510789186)

[Tabelle 70 - Durchführen der Tests Anforderungen 37](#_Toc510789187)

[Tabelle 71 - Umsetzungsreihenfolge 41](#_Toc510789188)

[Tabelle 72 - Lösungsmatrix Unittests 41](#_Toc510789189)

[Tabelle 73 - Lösungsmatrix Integration Tests / e2e Tests 42](#_Toc510789190)

[Tabelle 74 - Lösungsmatrix User-Testing 42](#_Toc510789191)

[Tabelle 75 - Lösungsmatrix Speichern der Parameter als Parametertyp im Json 43](#_Toc510789192)

[Tabelle 76 - Lösungsmatrix Speichern der Parameter als generischer Typ im Json 43](#_Toc510789193)

[Tabelle 77 - Lösungsmatrix nur serverseitig im Service den Parameter selbst testen lassen 48](#_Toc510789194)

[Tabelle 78 - Lösungsmatrix Regular Expression im Parameter 48](#_Toc510789195)

[Tabelle 79 - Lösungsmatrix Volltextsuche über alles 49](#_Toc510789196)

[Tabelle 80 - Lösungsmatrix Suche auf die Parameternamen 49](#_Toc510789197)

[Tabelle 81 - Lösungsmatrix Suche auf Parameternamen und Parameterwert 50](#_Toc510789198)

[Tabelle 82 - Testfälle 52](#_Toc510789199)

[Tabelle 83 - Aussagen ParameterSerializer Tests 58](#_Toc510789200)

[Tabelle 84 - Aussage ParameterValidation Tests 59](#_Toc510789201)

[Tabelle 85 - Resultate Testdurchgang 1 63](#_Toc510789202)

[Tabelle 86 - Resultate Testdurchgang 2 64](#_Toc510789203)

[Tabelle 87 - Resultate Testdurchgang 3 65](#_Toc510789204)

[Tabelle 88 - Resultate Testdurchgang 4 65](#_Toc510789205)

[Tabelle 89 - Glossar 118](#_Toc510789206)

[Tabelle 90 – Quellenverzeichnis 122](#_Toc510789207)

## Bildverzeichnis

[Abbildung 1 - Struktur der Dokumentablage 15](#_Toc510789010)

[**Abbildung 2 - Frontend Ordnerstruktur** 16](#_Toc510789011)

[**Abbildung 3 - Backend Ordnerstruktur** 16](#_Toc510789012)

[Abbildung 4 - IPERKA 19](#_Toc510789013)

[Abbildung 5 - Umsysteme 20](#_Toc510789014)

[Abbildung 6 - Systemgrenzen 21](#_Toc510789015)

[Abbildung 7 - RabbitMQ Event Kommunikation 22](#_Toc510789016)

[Abbildung 8 - Use-Case Diagramm 34](#_Toc510789017)

[Abbildung 9 - Firefox/Waterfox Einstellungen 44](#_Toc510789018)

[Abbildung 10 - Chrome Einstellungen 45](#_Toc510789019)

[Abbildung 11 - Visual Studio Einstellungen 46](#_Toc510789020)

[Abbildung 12 - Mockup 47](#_Toc510789021)

[Abbildung 13 - Screenshot umgesetzte Lösung Anzeige & Speichern der Parameter 53](#_Toc510789022)

[Abbildung 14 - Sequenzdiagramm Servicekommunikation beim «Save» 54](#_Toc510789023)

[Abbildung 15 – Screenshot umgesetzte Lösung Implementation des Validierungsmechanismus 56](#_Toc510789024)

[Abbildung 16 - Screenshot umgesetzte Lösung Suche eines Parameters 57](#_Toc510789025)

[Abbildung 17 - Screenshot Getting Started 62](#_Toc510789026)

[Abbildung 18 - Screenshot Getting Started Validierung 62](#_Toc510789027)

## Quellenverzeichnis

|  |  |
| --- | --- |
| Quelle | Information |
| https://angular.io/ | TypeScript Funktionalitätsfragen von TypeScript |
| http://masstransit-project.com/ | Wie man RabbitMQ anspricht / Buskonfiguration |
| https://msdn.microsoft.com/en-us/ | C# Reflection Syntaxfragen |
| https://stackoverflow.com/ | Syntaxfragen von TypeScript und C# |
| Projekt Viaduc Code | Grundgerüst der IPA |

Tabelle 90 – Quellenverzeichnis