Refactoring eines Newsletter Addons von ConcreteCMS

Inhaltsverzeichnis

[Refactoring eines Newsletter Addons von ConcreteCMS 1](#_Toc84213076)

[Inhaltsverzeichnis 2](#_Toc84213077)

[Einleitung 3](#_Toc84213078)

[1. Abgabe Teil 1 5](#_Toc84213079)

[1.1 Theorie Teil 5](#_Toc84213080)

[1.2 Semesterarbeit 7](#_Toc84213081)

[1.2.1 Projektbeschreibung und Ziele 7](#_Toc84213082)

[1.2.2 Projektaufbau 7](#_Toc84213083)

[2. Abgabe Teil 2 8](#_Toc84213084)

[2.1 Theorie Teil 8](#_Toc84213085)

[2.2 Semesterarbeit 9](#_Toc84213086)

[2.2.1 Service Architektur 9](#_Toc84213087)

[2.2.1.1 Entitäten 9](#_Toc84213088)

[2.2.1.2 Subscribe Service 10](#_Toc84213089)

[2.2.1.3 Schlussendlich eingesetzte Services 11](#_Toc84213090)

[2.2.2 API-Design 12](#_Toc84213091)

[2.2.2.1 Newsletter-Sets 12](#_Toc84213092)

[2.2.2.2 Newsletters 13](#_Toc84213093)

[2.2.2.3 Weitere HTTP-Response-Codes 13](#_Toc84213094)

[2.2.3 Abfrage-Beispiele 13](#_Toc84213095)

[2.2.3.1 Newsletter-Alben Liste 13](#_Toc84213096)

[2.2.3.2 Newsletter-Album Update 14](#_Toc84213097)

[2.2.3.3 Löschen eines Newsletters 14](#_Toc84213098)

[2.2.4 Versionierungskonzept 15](#_Toc84213099)

[3. Abgabe Teil 3 16](#_Toc84213100)

[4. Abgabe Teil 4 17](#_Toc84213101)

[5. Abgabe Teil 5 18](#_Toc84213102)

[6. Literaturverzeichnis 19](#_Toc84213103)

[7. Abbildungsverzeichnis 19](#_Toc84213104)

[8. Tabellenverzeichnis 19](#_Toc84213105)

Einleitung

Das Content Management System (CMS) ConcreteCMS[[1]](#footnote-1) (PortlandLabs, 2021) ist ein Monolith, das in den Scriptsprachen PHP, MySQL und JavaScript geschrieben ist. Wie es bei solchen Systemen üblich ist, ist auch c5 durch Addons (Packages), erweiterbar. Diese Packages sind in der Regel so geschrieben, dass der ganze benötigte Code mitgeliefert wird, womit die Last der Server mit jedem installierten Addon zunimmt. Dies kann im schlimmsten Fall zu Ausfällen der Websiten führen.

Ein weiterer Nachteil kann sein, dass z.B. Konfigurationsdaten für den Zugriff auf eine externe Ressource auf jedem Hosting einzeln angepasst werden müssen. Im speziellen bietet sich hier das von Mesch betriebene Newsletter-Package an. Mit diesem Package können versionierte Newsletter erstellt und versendet werden, wobei der Versand über einen firmeneigenen E-Mail-Server abgewickelt wird. Wird nun eine Änderung des Ports nötig, über den die Versand-Aufträge von den Hostings auf E-Mail-Server laufen, so muss entweder direkt auf der Datenbank des jeweiligen Hostings oder aber im CMS-Dashboard die Änderung nachgetragen werden.

Weitere Nachteile solcher monolithischer Lösungen werden aus Architektonischer Sicht bei näherer Betrachtung ersichtlich:

* Ein schon riesiges System wird mit weiteren Funktionen angereichert, welche nicht zu ihren Kernkompetenzen (Domäne) gehören
* Vorhandene Datenkonstrukte werden vergewaltigt

Im Beispiel des Newsletter Packages werden, bezogen auf den obigen zweiten Punkt, Benutzergruppen dazu verwendet, ob jemand einen bestimmten Newsletter erhält oder nicht. Diese Gruppenzuweisungen werden vor allem aber auch dazu verwendet, ob ein Benutzer eine bestimmte Tätigkeit ausführen darf.

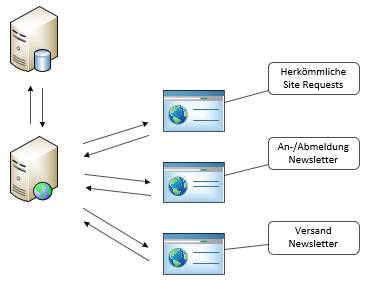


Abbildung 1: CMS wie ConcreteCMS bearbeiten sehr oft auch Anfragen, die Rechner intensiv sein können, aber mit den Kernaufgaben nichts zu tun haben.

Werden nun die Newsletterfunktionen in einen separaten Service ausgegliedert, so sendet, wie in Abbildung 2 zu sehen ist, der Hosting-Server weiterhin die Newsletter-Oberflächen, aber jegliche Funktionen, die die Newsletter betreffen, werden nun von einem anderen Server behandelt[[2]](#footnote-2).

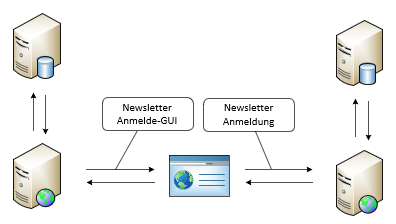
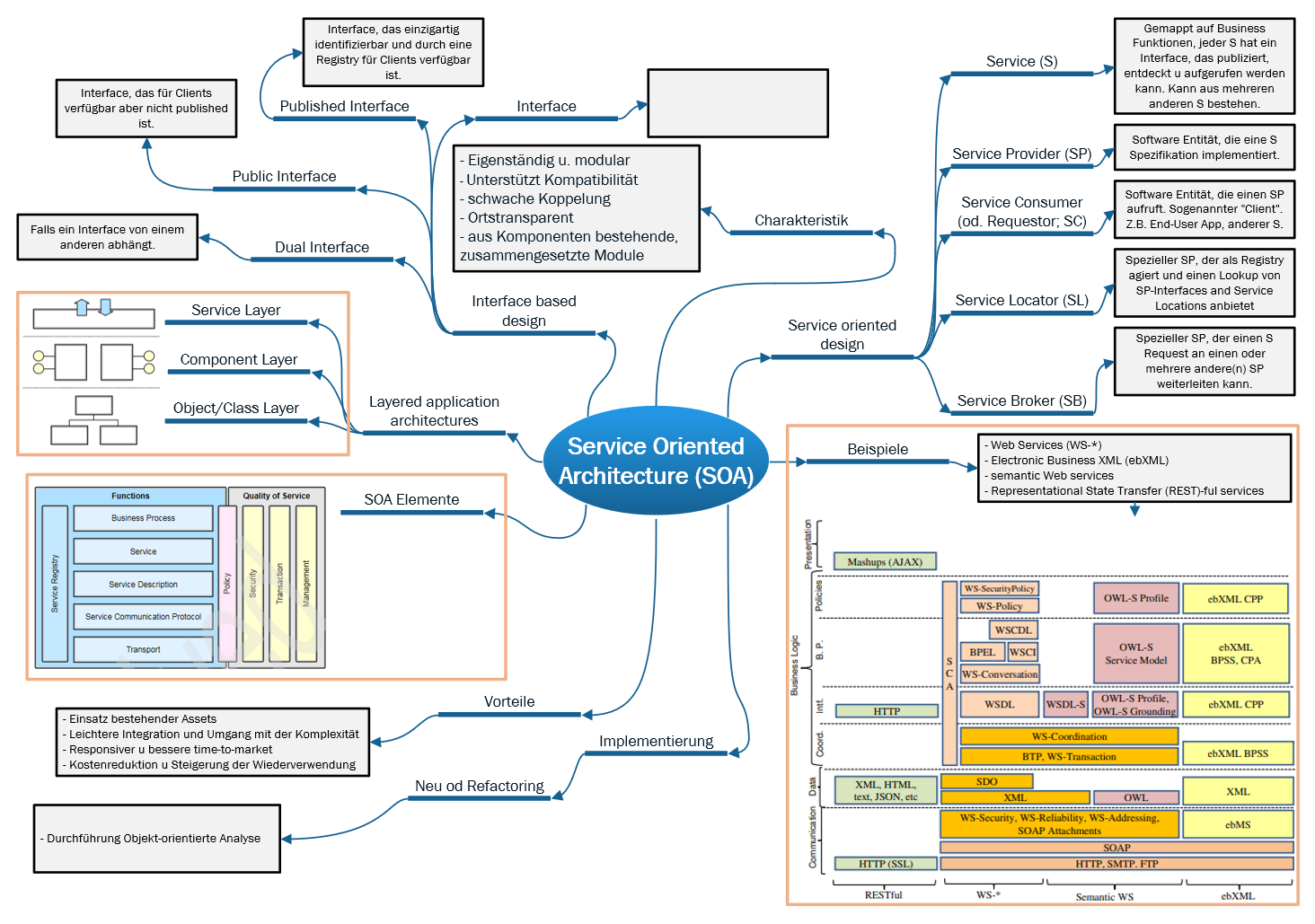


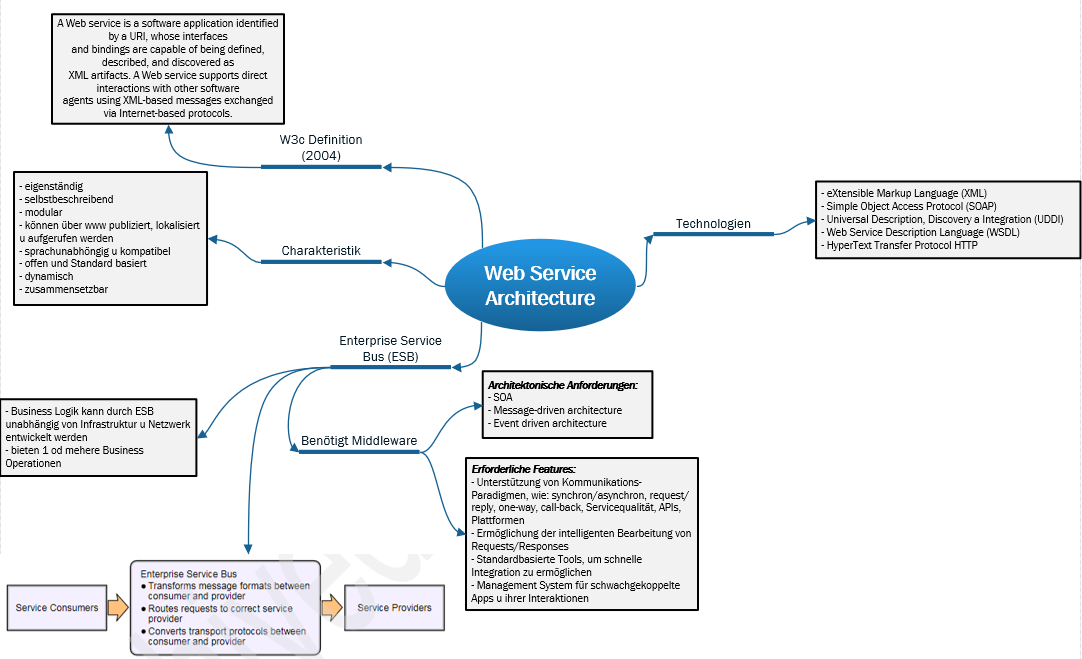
Abbildung 2: Ausgliederung der Newsletter Funktionen in einen eigenen Service, wobei links der Hosting- und rechts der Service-Server ist.

Diese Arbeit behandelt das Refactoring des zuvor vorgestellten Newsletter Packages. Dabei wird ein Dienst gemäss SOA geplant.

Pro Hauptkapitel, die sich den periodischen Abgaben nach den PVAs widmen, befindet sich einerseits ein Theorieteil, der entsprechende Themen behandelt, andererseits die eigentliche Semesterarbeit.

1. Abgabe Teil 1
   1. Theorie Teil





* 1. Semesterarbeit
     1. Projektbeschreibung und Ziele

Ziel dieses Projekts ist es, die Business Logik des Newsletter Packages von Mesch für c5 in einen Service auszulagern, der den bekannten SOA Pattern folgt. Das umgebaute Package soll mittels RESTfull API Requests auf diesen Service zugreifen[[3]](#footnote-3). Eine neu zu entwickelnde Oberfläche soll das zentrale Konfigurieren des Services ermöglichen, wobei deren Zugriff ebenfalls mittels RESTfull API Requests ablaufen muss.

|  |  |  |
| --- | --- | --- |
| Komponente | Neu | Tätigkeiten |
| Newsletter Package | Nein | * Auslagerung der Business-Logik * Entfernung der Konfigurationsseite * Versand neu mittels RESTful API Requests * Statistiks neu über RESTful API Requests |
| An-/Abmelde Package | Nein | * Infos der Newsletter mittels REST Requests * An-/Abmeldungen über REST Requests |
| Newsletter Microservice | Ja | * Aufbau der RESTful API Schnittstelle * Handling Newsletters pro Hosting * Handling Newsletter Versand über ein Queue-System * Handling Konfigurationsanfragen |
| Newsletter Konfigurations-UI | Ja | * Leicht aufsetzbares Hosting * GUI für die Konfiguration des Newsletterversands * Queue-Administratoren Aufgaben |

Das Projekt wird nach dem agilen Vorgehensmodell durchgeführt, wobei einige Tools von Scrum (z.B. User Stories, Story Board und Backlog) verwendet werden. Die Versionskontrolle findet auf GitHub statt (Meister & Plüss, 1stthomas / ffhs-soa-work, 2021), wo auch ein vereinfachtes Projektmanagement betrieben wird (Meister & Plüss, PVA 1 - KickOff, 2021).

* + 1. Projektaufbau

Vom Auftrag vordefinierte Meilensteine geben die Projektstruktur vor. Sie sind in der Tabelle 1 aufgeführt.

Tabelle 1: Meilensteine des Projekts.

|  |  |  |
| --- | --- | --- |
| Meilenstein | Auftrag | Datum |
| 1 | Projektstruktur | 05.09.2021 |
| 2 | Service Architektur | 03.10.2021 |
| 3 | Deployment- und Testkonzept | 31.10.2021 |
| 4 | Monitoring- und Security-Konzept | 26.11.2021 |
| 5 | Projektabgabe | 24.12.2021 |

1. Abgabe Teil 2
   1. Theorie Teil
   2. Semesterarbeit
      1. Service Architektur
         1. Entitäten

Eine Analyse der Geschäftsprozesse ergibt die in Tabelle 2 dargestellten Hauptentitäten.

Tabelle 2: Hauptentitäten der Newsletter-Anwendung.

|  |  |  |
| --- | --- | --- |
| # | Entität | Bemerkung |
| E1 | NewsletterSet | Zusammengesetzte Entität |
| E2 | Newsletter | Komplexere, zusammengesetzte Entität |
| E3 | User | Einfache Entität, v.a. E-Mail |
| E4 | Account | Zusammengesetzte Entität |
| E5 | Log | Einfache Entität |
| E6 | Queue | Zusammengesetzte Entität |

Die Beziehung der Entitäten zueinander ist in Abbildung 3 dargestellt.

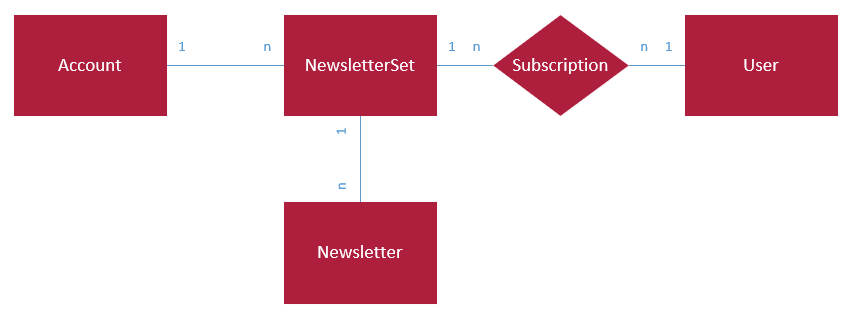


Abbildung 3: Die Hauptentitäten der Newsletter-Anwendung und ihre Beziehungen zueinander. Die Log Entität findet sich hier nicht, da sie keine Beziehung zu einer anderen hat.

Werden die Anwendungsfälle der Newsletter-Anwendung als Nanoservice betrachtet und ihnen die Service-Typen gegenübergestellt (s. Tabelle 3), kann dies als Grundlage für die Einteilung in einzelne Services verwendet werden.

Tabelle 3: Einzelnen Tätitgkeiten der Newsletter-Anwendung.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Anwendungsfall | Service-Typ | Entität | Bemerkung |
| UC1 | Anmelden an NewsletterSet | Entity Service | Subscription |  |
| UC2 | Abmelden von NewsletterSet | Entity Service | Subscription |  |
| UC3 | Newsletterset erstellen | Entity Service | NewsletterSet |  |
| UC4 | Newsletterset bearbeiten | Entity Service | NewsletterSet |  |
| UC5 | NewsletterSet löschen | Entity Service | NewsletterSet |  |
| UC6 | Newsletter erstellen | Entity Service | Newsletter |  |
| UC7 | Newsletter bearbeiten | Entity Service | Newsletter |  |
| UC8 | Newsletter löschen | Entity Service | Newsletter |  |
| UC9 | Versandauftrag in Queue | Utility Service | Subscription |  |
| UC10 | Abarbeiten Queue | Task Service | Subscription |  |
| UC11 | Newsletter Config | Entity Service | Account |  |
| UC12 | Newsletter Statistik | Utility Service | Account, Subscription |  |
| UC13 | Konto Verwaltung | Entity Service | User, Subscription | Div. Konto-Typen |
| UC14 | Services Config | Entity Service | Service |  |
| UC15 | Newsletter Logging | Utility Service | Account, Subscription |  |
| UC16 | Queue Admin | Utility Service |  |  |

Die einfachste Lösung wäre nun, Services anhand der Typen zu definieren. Dann wäre jedoch einerseits eine Vermischung der Domänen zu gross und andererseits wäre mit dem Task Service ein Nanoservice, was gemäss (Rotem-Gal-Oz, 2012) ein Antipattern darstellt.

* + - 1. Subscribe Service

Somit werden die Grenzen der Typen durchbrochen werden müssen. Eine Möglichkeit stellt ein Subscribe Service dar, der nur für An- und Abmeldungen zuständig ist. Eine Zuteilung, die diese Art von Service verwendet, findet sich in der nachfolgenden Tabelle.

Tabelle 4: Einteilung unter Verwendung eines Subscribe Services.

|  |  |  |
| --- | --- | --- |
| Service Name | # |  |
| Subscribe | UC1 | Anmelden an NewsletterSet |
| UC2 | Abmelden von NewsletterSet |
| Admin | UC3 | Newsletterset erstellen |
| UC4 | Newsletterset bearbeiten |
| UC5 | NewsletterSet löschen |
| UC6 | Newsletter erstellen |
| UC7 | Newsletter bearbeiten |
| UC8 | Newsletter löschen |
| UC11 | Newsletter Config |
| UC13 | Konto Verwaltung |
| UC14 | Services Config |
| Utility | UC9 | Versandauftrag in Queue |
| UC12 | Newsletter Statistik |
| UC15 | Newsletter Logging |
| UC16 | Queue Admin |

Diese Architektur bietet sich vor allem an, wenn ein geosses Aufkommen an An- und Abmeldungen zu erwarten ist und eine sehr gute Skalierung gewährleistet werden muss. Dieses Bedürfnis kann einerseits von einer hohen Verwendung dieser Newsletter-Anwendung oder von einer weiteren Verwendung für andere Anwendungen kommen. Da beide Punkte bei diesem Projekt in nächster Zeit nicht eintreten dürften, wird eine Aufteilung, wie in Tabelle 5 zu sehen, eher den Bedürfnissen gerecht.

* + - 1. Schlussendlich eingesetzte Services

Tabelle 5: Die endgültige Version der Services und ihrer Use Cases.

|  |  |  |
| --- | --- | --- |
| Service Name | # |  |
| Newsletter-Operations | UC1 | Anmelden an NewsletterSet |
| UC2 | Abmelden von NewsletterSet |
| UC3 | Newsletterset erstellen |
| UC4 | Newsletterset bearbeiten |
| UC5 | NewsletterSet löschen |
| UC6 | Newsletter erstellen |
| UC7 | Newsletter bearbeiten |
| UC8 | Newsletter löschen |
| UC9 | Versandauftrag in Queue |
| Newsletter-Admin | UC11 | Newsletter Config |
| UC13 | Konto Verwaltung |
| UC14 | Services Config |
| UC16 | Queue Admin |
| UC10 | Abarbeiten Queue |
| Newsletter-Utility | UC12 | Newsletter Statistik |
| UC15 | Newsletter Logging |

Mit dieser Einteilung der Anwendungsfälle sind nun folgende Punkte sichergestellt

* der Utility Service ist nur für lesende Zugriffe zuständig
* der Newsletter-Operations Service ist der einzige, der die Newsletter und NewsletterSet Entitäten liest und schreibt
* Alle weiteren Anwendungsfälle können als Administrationsaufgabe aufgefasst werden und finden sich im passenden Admin Service wieder

Die Zuordnung Service <-> Entität ist in Abbildung 4 zu sehen. Wie ersichtlich ist, gteift jeder Service nur auf seine ihm zugeordneten Entitäten zu.

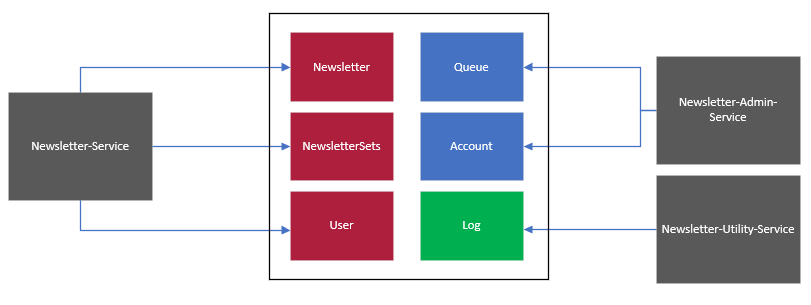


Abbildung 4: Zuordnung der Entitäten zu den Services. Die grauen Kästen sind die Services, die farbigen die Entitäten.

* + 1. API-Design

Im Folgenden werden die Designs für einzelne Entitäten aufgeführt. Dabei bedeutet die Aktion create ein Erstellen einer Entität, ein update ein Aktualisieren, ein get entweder der Erhalt einer oder mehrerer Entitäten. Schlussendlich wird mit einem delete eine Entität gelöscht. Mit {id} wird jeweils angedeutet, dass der Identifier der gewünschten Entität mitgegeben werden muss.

* + - 1. Newsletter-Sets

Mit diesen Abfragen werden die Newsletter-Alben betreut.

Tabelle 6: URI-Design für die Newsletter-Alben Entität.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| http-Methode | URI-Design | Aktion | Response OK | Response nOK |
| POST | /v1/newsletter-sets/ | create | 201 | 422 |
| PUT | /v1/newsletter-sets/{id} | update | 200 | 422 |
| GET | /v1/newsletter-sets/{id} | get | 200 | 404 |
| GET | /v1/newsletter-sets/ | get | 200 | 204 |
| DELETE | /v1/newsletter-sets/{id} | delete | 200 | 404 |

Es wird darauf verzichtet, bei Anfragen einer id, für die keine Berechtigung besteht, einen 403 Response Code zurückzugeben. Dies um keine unnötige (oder je nachdem gefährliche) Informationen mitzuteilen.

* + - 1. Newsletters

Mit diesen Abfragen werden die Newsletter-Alben betreut.

Tabelle 7: URI-Design für die Newsletter Entität.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| http-Methode | URI-Design | Aktion | Response OK | Response nOK |
| POST | /v1/newsletter/ | create | 201 | 422 |
| PUT | /v1/newsletter/{id} | update | 200 | 422 |
| GET | /v1/newsletter/{id} | get | 200 | 404 |
| GET | /v1/newsletter/ | get | 200 | 204 |
| DELETE | /v1/newsletter/{id} | delete | 200 | 404 |

* + - 1. Weitere HTTP-Response-Codes

Im folgenden die Response-Codes für ausgewählte (fehlerhafte) Abfragen.

|  |  |
| --- | --- |
| HTTP Response Code | Wert |
| 400 | Falsche Abfragedaten |
| 401 | Falls eine nicht Person auf einen nicht authorisierten Service zugreifen möchte |
| 403 | Abfragen eines anderen Accounts, auf den keine Zugriffsberechtigungen bestehen |
| 404 | Abfragen z.B. mit fehlerhaften Pfädern |
| 405 | Falls für einen Request eine falsche HTTP-Methode verwendet wurde |
| 408 | Falls der Server die Anfrage nicht innerhalb einer definierten Zeitspanne antworten kann |
| 409 |  |
| 422 | Fehler beim Erstellen oder aktualisieren einer Entität |
| 500 |  |

* + 1. Abfrage-Beispiele
       1. Newsletter-Alben Liste

Mit diesem Beispiel werden alle Newsletter-Alben des verbundenen Accounts aufgelistet.

Tabelle 8: Request Informationen für die Newsletter-Album Auflistung.

|  |  |
| --- | --- |
| Bezeichnung | Wert |
| Protokoll | HTTP |
| HTTP-Methode | GET |
| Bsp.-URL | https://domain.ch/v1/newsletter-sets/ |

Tabelle 9: Response Informationen für die Newsletter-Album Auflistung.

|  |  |
| --- | --- |
| Bezeichnung | Wert |
| Datentyp | JSON |
| Wert | {  "newsletter\_sets":  [  {  "id": 1,  "name": "Set 1",  "createdAt":"2021-06-13 04:02:04",  "updatedAt": "2021-10-02 01:25:07",  },  {…}  ]  } |
| HTTP Response Code | 200 |

* + - 1. Newsletter-Album Update

Mit diesem Beispiel wird das Newsletter-Album mit der id 1 von «Set 1» auf «wonderful set» umbenannt.

Tabelle 10: Request Informationen für das Newsletter-Album Update.

|  |  |
| --- | --- |
| Bezeichnung | Wert |
| Protokoll | HTTP |
| HTTP-Methode | PUT |
| Bsp.-URL | https://domain.ch/v1/newsletter-sets/1 |
| Formular-Daten | [name="wonderful set"] |

Tabelle 11: Response Informationen für das Newsletter-Album Update.

|  |  |
| --- | --- |
| Bezeichnung | Wert |
| Datentyp | JSON |
| HTTP Response Code | 200 |

* + - 1. Löschen eines Newsletters

Die Abfrage aus diesem Beispiel löscht den Newsletter mit der id 1 mit all seinen Verbindungen.

Tabelle 12: Request Informationen für das Löschen eines Newsletters.

|  |  |
| --- | --- |
| Bezeichnung | Wert |
| Protokoll | HTTP |
| HTTP-Methode | DELETE |
| Bsp.-URL | https://domain.ch/v1/newsletters /1 |

Tabelle 13: Response Informationen für das Newsletter-Album Update.

|  |  |
| --- | --- |
| Bezeichnung | Wert |
| Datentyp | JSON |
| HTTP Response Code | 200 |

* + 1. Versionierungskonzept

Wie im Kapitel 2.2.2 bei den URI-Designs zu sehen ist, werden den Pfädern /v1 vorangestellt, womit die jeweilige API-Version festgelegt wird. Damit die Versionen zentral verwaltet werden können, werden alle Requests, bevor sie auf den jeweiligen Service treffen, über ein API Gateway geroutet.

1. Abgabe Teil 3
2. Abgabe Teil 4
3. Abgabe Teil 5
4. Literaturverzeichnis

Meister, S., & Plüss, T. (05. 09 2021). *1stthomas / ffhs-soa-work*. Abgerufen am 05. 09 2021 von GitHub: https://github.com/1stthomas/ffhs-soa-work

Meister, S., & Plüss, T. (05. 09 2021). *PVA 1 - KickOff*. Abgerufen am 05. 09 2021 von GitHub: https://github.com/1stthomas/ffhs-soa-work/projects/1

PortlandLabs. (2. 9 2021). *Open Source Content Management System for Teams*. (PortlandLabs, Herausgeber) Abgerufen am 2. 9 2021 von concretecms.com: https://www.concretecms.com/

Rotem-Gal-Oz, A. (2012). Nanoservice antipattern. In A. Rotem-Gal-Oz, *SOA Patterns* (S. 195-202). Shelter Island, NY 11964: Manning. Abgerufen am 1. 10 2021

1. Abbildungsverzeichnis

[Abbildung 1: CMS wie ConcreteCMS bearbeiten sehr oft auch Anfragen, die Rechner intensiv sein können, aber mit den Kernaufgaben nichts zu tun haben. 3](#_Toc84213106)

[Abbildung 2: Ausgliederung der Newsletter Funktionen in einen eigenen Service, wobei links der Hosting- und rechts der Service-Server ist. 4](#_Toc84213107)

[Abbildung 3: Die Hauptentitäten der Newsletter-Anwendung und ihre Beziehungen zueinander. Die Log Entität findet sich hier nicht, da sie keine Beziehung zu einer anderen hat. 9](#_Toc84213108)

[Abbildung 4: Zuordnung der Entitäten zu den Services. 12](#_Toc84213109)

1. Tabellenverzeichnis

[Tabelle 1: Meilensteine des Projekts. 7](#_Toc84213110)

[Tabelle 2: Hauptentitäten der Newsletter-Anwendung. 9](#_Toc84213111)

[Tabelle 3: Einzelnen Tätitgkeiten der Newsletter-Anwendung. 10](#_Toc84213112)

[Tabelle 4: Einteilung unter Verwendung eines Subscribe Services. 11](#_Toc84213113)

[Tabelle 5: Die endgültige Version der Services und ihrer Use Cases. 11](#_Toc84213114)

[Tabelle 6: URI-Design für die Newsletter-Alben Entität. 13](#_Toc84213115)

[Tabelle 7: URI-Design für die Newsletter Entität. 13](#_Toc84213116)

[Tabelle 8: Request Informationen für die Newsletter-Album Auflistung. 13](#_Toc84213117)

[Tabelle 9: Response Informationen für die Newsletter-Album Auflistung. 14](#_Toc84213118)

[Tabelle 10: Request Informationen für das Newsletter-Album Update. 14](#_Toc84213119)

[Tabelle 11: Response Informationen für das Newsletter-Album Update. 14](#_Toc84213120)

[Tabelle 12: Request Informationen für das Löschen eines Newsletters. 15](#_Toc84213121)

[Tabelle 13: Response Informationen für das Newsletter-Album Update. 15](#_Toc84213122)

1. Vor 2021 hiess das CMS concrete5, weshalb im weiteren Verlauf des Dokuments die weitläufig bekannte Abkürzung c5 verwendet wird. [↑](#footnote-ref-1)
2. Die vom zusätzlichen Service benötigte Oberfläche kann auch, anders als hier im Beispiel dargestellt, vom Service selbst angeboten werden. [↑](#footnote-ref-2)
3. In dieser Arbeit wird nur die RESTful API Technologie verwendet, weshalb im weiteren Verlauf der Arbeit nur noch von REST Requests geschrieben wird. [↑](#footnote-ref-3)