Refactoring eines Newsletter Addons von ConcreteCMS

Inhaltsverzeichnis

[Refactoring eines Newsletter Addons von ConcreteCMS 1](#_Toc81784350)

[Inhaltsverzeichnis 2](#_Toc81784351)

[Einleitung 3](#_Toc81784352)

[1. Abgabe Teil 1 5](#_Toc81784353)

[1.1 Theorie Teil 5](#_Toc81784354)

[1.2 Semesterarbeit 7](#_Toc81784355)

[1.2.1 Projektbeschreibung und Ziele 7](#_Toc81784356)

[1.2.2 Projektaufbau 7](#_Toc81784357)

[2. Abgabe Teil 2 9](#_Toc81784358)

[3. Abgabe Teil 3 10](#_Toc81784359)

[4. Abgabe Teil 4 11](#_Toc81784360)

[5. Abgabe Teil 5 12](#_Toc81784361)

[6. Literaturverzeichnis 13](#_Toc81784362)

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.

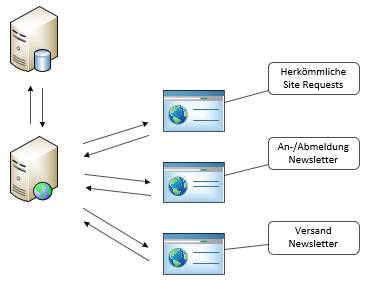


Abbildung 1: CMS wie ConcreteCMS bearbeiten sehr oft auch Anfragen, die Rechner intensiv sein können, aber mit den Kernaufgaben nichts zu tun haben.

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).

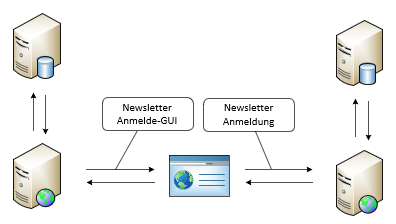
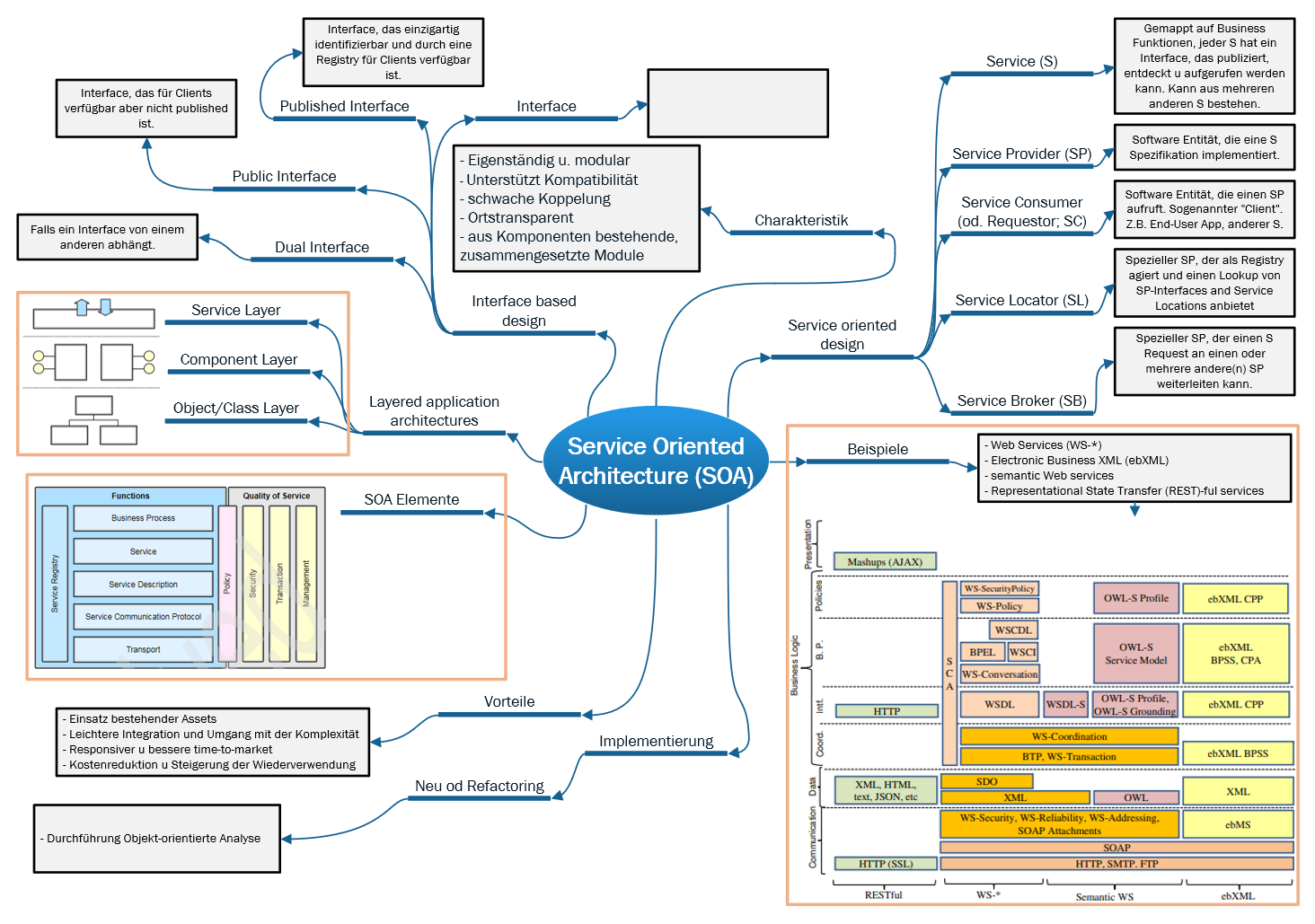


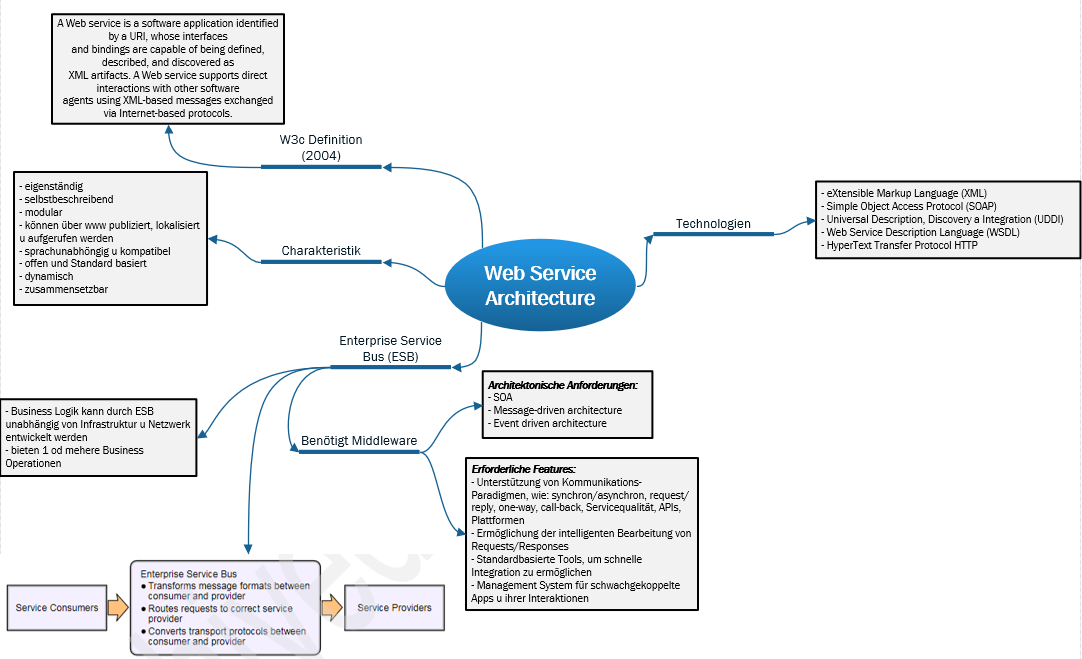
Abbildung 2: Ausgliederung der Newsletter Funktionen in einen eigenen Service, wobei links der Hosting- und rechts der Service-Server ist.

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
2. Abgabe Teil 3
3. Abgabe Teil 4
4. Abgabe Teil 5
5. 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/

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)