[Achtung: Verwenden Sie einen Sperrvermerk nur in sehr gut begründeten Fällen!]

[evtl. Sperrvermerk]

Auf Wunsch der Firma [FIRMA] ist die vorliegende Arbeit bis zum [DATUM] für die öffentliche Nutzung zu sperren.

Veröffentlichung, Vervielfältigung und Einsichtnahme sind ohne ausdrückliche Genehmigung der oben genannten Firma und der/dem Verfasser/in nicht gestattet. Der Titel der Arbeit sowie das Kurzreferat/Abstract dürfen jedoch veröffentlicht werden.

Dornbirn, Unterschrift der Verfasserin/des Verfassers

Firmenstempel

FH Vorarlberg
University of Applied Sciences

[Titel der Arbeit]

[Untertitel der Arbeit]

Bachelorarbeit I
zur Erlangung des akademischen Grades

Bachelor of Science in Engineering (BSc)

Fachhochschule Vorarlberg
Informatik – Software and Information Engineering

Betreut von
Prof. (FH) Dipl. Inform. Thomas Feilhauer

Vorgelegt von
Dominic Luidold
Dornbirn, November 2020

[Widmung]

[Text der Widmung]

Kurzreferat

[Deutscher Titel Ihrer Arbeit]

[Text des Kurzreferats]

Abstract

[English Title of your thesis]

[text of the abstract]

Vorwort

Der Verfasser der vorliegenden Arbeit bekennt sich zu einer geschlechtergerechten Sprachverwendung.

Um die Lesbarkeit zu gewährleisten und zugunsten der Textökonomie werden die verwendeten Personen beziehungsweise Personengruppen fix männlich oder weiblich zugeordnet. Zum Beispiel wird immer „die Entwicklerin“ und „der Benutzer“ verwendet. Es wurde besonders darauf geachtet, stereotype Rollenbeschreibungen zu vermeiden. Die insgesamt eventuell dadurch hervorgerufene Irritation bei den Lesenden ist gewünscht und soll dazu beitragen, eine Bewusstheit für die bestehende, Frauen diskriminierende Sprachgewohnheit (generelle Verwendung der männlichen Begriffe für beide Geschlechter) zu wecken beziehungsweise zu stärken.

Inhaltsverzeichnis

Abbildungsverzeichnis

Quellcodeverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

BLS Better Life System

CI Continuous Integration

CD Continuous Deployment

CQRS Command Query Responsibility Segregation

CRUD Create, Read, Update, Delete

DTO Data Transfer Object

1 Einleitung

Diese Bachelorarbeit verfolgt das Ziel, einen Einblick in die Implementierung und Erweiterung des bereits bestehenden Backend-Systems des „Better Life System“ (kurz „BLS“) der Viterma Handels GmbH mit dem sogenannten „CQRS“-Pattern zu geben.

Das Better Life System - welches von Fusonic GmbH entwickelt wird und mittels einer Weboberfläche und gängigen Webbrowsern bedient werden kann - ermöglicht es Endkunden, mithilfe einer Handelsvertreterin der Firma Viterma (beziehungsweise mit einer Vertreterin eines ihrer Franchise-Partner) ein Badezimmer anhand der jeweiligen Bedürfnisse auszusuchen und zu konfigurieren, um sich schlussendlich ein entsprechendes Angebot dafür erstellen zu lassen.

1.1 Arbeitgeber Fusonic GmbH

Die Fusonic GmbH hat ihren Firmensitz in Götzis, Vorarlberg, und besteht aus einem Team von aktuell mehr als 25 Angestellten, Softwareentwicklerinnen und Projektleiterinnen. Diese sind aufgeteilt in diverse kleinere Teams, die intern unter anderem „Duck-Team“ beziehungsweise „Parrot-Team“ genannt werden. Die Teams arbeiten dabei an jeweils eigenständigen Projekten und setzen diverse Technologien ein. Zu den verwendeten Technologien gehören unter anderem C# mit .NET, PHP mit Symfony und JavaScript/TypeScript mit Angular. (vgl. Fusonic GmbH 2020, "Übersicht aller Technologien") Während es regelmäßigen Austausch unter den Teams gibt, besteht jedes aus Frontend- sowie Backend-Entwicklern, da die meisten Projekte - aufgrund der zugrundeliegenden Anwendungsfälle - aus einem Web-Frontend sowie serverseitgen Backend bestehen.

1.2 Nutzung & Umfeld des „Better Life System“

Die Hauptaufgabe des Better Life System liegt darin, als computergestütztes und plattformunabhängiges Tool, Vertreterinnen rund um die Vitmera GmbH bei der Konfiguration und Zusammenstellung eines Badezimmers - zusammen mit dem Endkunden (meist Haus- beziehungsweise Wohnungsbesitzer oder Hotels) - zu unterstützen und diesen Vorgang zu erleichtern. Das BLS wir des Weiteren dazu verwendet, aktuelle und abgeschlossene Angebote zu verwalten, Produkte, Produktinformationen und dazugehörige Preisaufschläge zu pflegen und Kundendaten abzulegen. Zudem dient es als Kontrollinstanz für in Auftrag gegebene Sanierungen, um sicherzustellen, dass alle benötigten Materialen und Teile in entsprechenden Mengen vorhanden und kompatibel sind.

Die Nutzung des BLS findet zu großen Teilen auf mobilen Rechnern beziehungsweise Laptops statt, die während der Beratung und Betreuung von Kunden zum Einsatz kommen. Da nicht immer gewährleistet werden kann, dass eine aufrechte Internetverbindung besteht, ist die Offline-Fähigkeit bei gleichbleibender Nutzung im Webbrowser ein wichtiger Bestandteil des Web-Frontends.

Aufgrund der Anforderungen der Viterma GmbH, dass das Better Life System offline ebenso wie online verwendet werden kann, basiert das Frontend auf dem TypeScript Web-Framework Angular[1], welches solch eine Funktionalität ohne eine Installation oder zusätzliche Voraussetzungen am Endgerät unterstützt. Im Gegensatz zu weniger komplex gehaltenen Frontends - die primär Daten darstellen, die vom Backend eingehen - ist dieses beim BLS für einen Großteil der Ablauflogik, Berechnung von Preisen und Generierung von diversen PDFs zuständig. Das Frontend wird, da der Fokus dieser Arbeit auf der Entwicklung im Backend der Anwendung liegt, in weiterer Folge jedoch nicht genauer beleuchtet und als gegeben angenommen. Es kann davon ausgegangen werden, dass eingehende Anfragen an das Backend aus Benutzereingaben beziehungsweise zeitlich gesteuerten Events resultieren.

[1] Angular (https://angular.io)

Das Backend, welches alle API-Anfragen bearbeitet und als Schnittstelle zur Datenbank dient, ist in der Programmiersprache PHP geschrieben und verwendet das Framework Symfony[2] sowie diverse „Bundles“, die darauf aufbauen. Ein detaillierterer Überblick über den Aufbau, bisher angewandte Konzepte sowie technische Begebenheiten werden in Kapitel 2 auf Seite 21 behandelt.

Während die Nutzung des Better Life System auf allen Plattformen - die einen modernen Webbrowser anbieten - möglich ist, findet die Entwicklung primär auf der Linux-Distribution Ubuntu statt. Der Grund für diese Betriebssystemwahl ist damit zu erklären, dass sowohl die lokale Entwicklung als auch die Continuous Integration Umgebung (genutzt für automatisierte Tests) und das Continuous Deployment mittels Docker[3] Containern stattfindet. Für die Entwicklung im Backend-Bereich der Anwendung wird die IDE PhpStorm verwendet, während im Frontend auf Visual Studio Code gesetzt wird.

1.3 Problemstellung

Die Entwicklung des BLS hat im Herbst 2018 begonnen und die Anwendung ist seit dem stetig weiterentwickelt worden. Über die Zeit haben sich entsprechend sowohl die Anforderungen und Wünsche des Kunden als auch die technischen Begebenheiten, Best Practices und Möglichkeiten verändert.

Die bisher bei der Umsetzung des Projekts verwendete Struktur sowie der Aufbau wurde stark an die empfohlene Herangehensweise von Symfony angelehnt und die Code-Basis entsprechend darauf ausgelegt (siehe dazu Symfony 2020, „The Symfony Framework Best Practices“). Durch wachsende Anforderungen und entsprechend benötigten Programmcode, der diese erfüllt, hat sich jedoch ein komplexes System ergeben, dessen *Controller*, *Entities*, *Services* und *Data Transfer Objects (DTOs)* auf viele Verzeichnisse und Unterverzeichnisse verteilt sind. Die Abbildung 1.1 auf Seite 17 zeigt einen Ausschnitt der Ordner, die im Projekt vorkommen und teilweise weitere Ordner mit diversen Klasen, Traits etc. beinhalten.

[2]Symfony (https://symfony.com)
[3]Docker (https://docker.com)

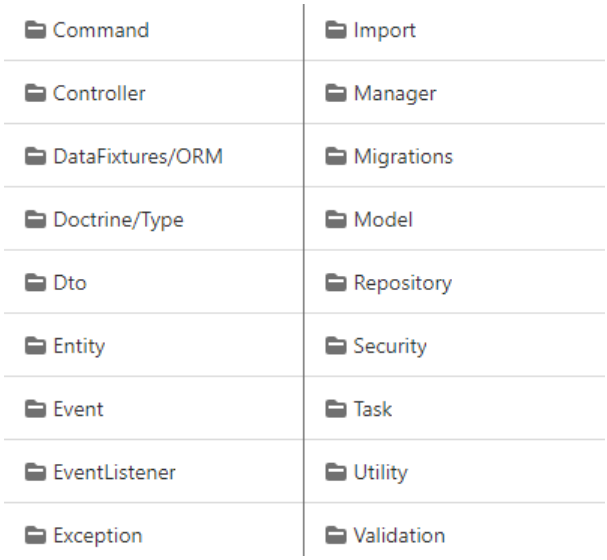

Abbildung 1.1: Teilausschnitt der BLS-Ordnerstruktur

Die aktuelle Herangehensweise hat im weiteren Verlauf das Problem, dass Controller - die für die Definition der Routen der API Endpunkte, das Zusammentragen von Daten und der Erstellung einer *Response* zuständig sind - schnell unübersichtlich und komplex werden können. Der Quellcode 1 auf Seite 18 zeigt ein Beispiel für ein Controller, der exemplarisch Routen für einen API Endpunkt mit CRUD-Funktionalität definiert. Es ist hierbei bereits zu erkennen, dass dieser trotz fehlender Implementierungen und weiterer Funktionen aufgebläht wirkt. Erschwerend kommt hinzu, dass benötigte Abhängigkeiten (dazu zählen beispielsweise Services, Repositories, Entities und DTOs) jeweils so abstrakt wie möglich gehalten werden müssen, damit diese für alle im Controller definierten Abläufe verwendet werden können. Eine effektive Kapselung zusammengehöriger Funktionalitäten, die entsprechend getrennt voneinander austauschbar und mittels Unit Tests testbar sein sollten, ist somit schwer zu erreichen ist.

```php
<?php

[...]

final class SampleController extends AbstractController
{
    [...]

    /**
     * @Route("/samples", name="get_samples", methods={"GET"})
     */
    public function cgetAction(Request $request): Response { // Implementation }

    /**
     * @Route("/samples/{id}", name="get_sample", methods={"GET"})
     */
    public function getAction(int $id): Response { // Implementation }

    /**
     * @Route("/samples", name="post_sample", methods={"POST"})
     */
    public function postAction(Request $request): Response { // Implementation }

    /**
     * @Route("/samples/{id}", name="patch_sample", methods={"PATCH"})
     */
    public function patchAction(int $id, Request $request): Response { // Implementation }

    /**
     * @Route("/samples/{id}", name="delete_sample", methods={"DELETE"})
     */
    public function deleteAction(int $id): Response { // Implementation }

    [...]
}
```

Quellcode 1: Beispiel Controller mit definierten Routen für CRUD-Funktionalität

1.4 Anforderungen

Die in Abschnitt 1.3 angeführten Umstände und den daraus resultierenden Herangehensweisen, die nicht mehr in vollem Maße zufriedenstellend sind, führen zu neuen Anforderungen an das Better Life System. Diese Anforderungen sollen entsprechend umgesetzt werden, um ein zukunftssicheres Backend für die gesamte Anwendung garantieren zu können und die Entwicklung neuer sowie bestehender Funktionalitäten zu erleichtern und zu vereinheitlichen.

Zum Start des Berufspraktikums Anfang Juli 2020 stand bereits fest, dass in weiterer Folge das CQRS-Pattern (was für „Command Query Responsibility Segregation“ steht) zum Einsatz kommen wird, da andere Projekte der Fusonic GmbH - die primär auf C# basieren - bereits gute Erfahrungen damit gemacht haben.

Die Anforderungen, die im Zuge des Berufspraktikums und folglich dieser Bachelorarbeit zu erfüllen sind, bestehen darin, das aktuell bestehende System neben dem „Tagesgeschäft“ (sprich Bugfixes, Feature Requests etc.) laufend umzustellen und neue Funktionalitäten entsprechend mit der neuen Struktur und dem Pattern umzusetzen.

1.5 Zielsetzung

Die in den Abschnitten 1.3 und 1.4 angeführten Punkte ergeben folgende Ziele, die im Verlauf des Berufspraktikums so weit wie möglich umzusetzen sind. Aufgrund von Budget- und Zeitlimitierungen findet das, wie im Abschnitt zuvor bereits erwähnt wurde, laufend statt und entspricht keiner Komplettumstellung in einem Zug.
Als Ziele einzustufen sind:

- Vorbereitung der bestehenden Infrastruktur auf CQRS in Zusammenarbeit mit Mitarbeitern der Fusonic GmbH
- Umstellung der bestehenden API Endpunkte und deren Controller auf CQRS bei Beibehaltung von möglichst viel Programmlogik

- Umsetzung neuer Funktionalitäten mittels CQRS
- Weiterhin Abwickeln des Tagesgeschäftes

2 Stand der Technik

TODO

Literatur

Fusonic GmbH (2020). *Web, Mobile, Desktop-Anwendungen*. Fusonic, Vorarlberg. URL: https://www.fusonic.net/de/leistungen/ (besucht am 04. 08. 2020).

Symfony (2020). *The Symfony Framework Best Practices (Symfony Best Practices)*. Library Catalog: symfony.com. URL: https://symfony.com/doc/current/best_practices.html (besucht am 06. 08. 2020).

[evtl. Anhang]

Formatvorlage für den Fließtext.

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Bachelorarbeit I selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Stellen sind als solche kenntlich gemacht. Die Arbeit wurde bisher weder in gleicher noch in ähnlicher Form einer anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Dornbirn, am [Tag. Monat Jahr anführen] Dominic Luidold