Designbeschreibung

HOR-TINF2021

Gruppe

Tobias Hahn
Lars Holweger
Fabian Unger
Timo Zink
Frank Sadoine
Fabian Eilber

Betreuer

Prof. Dr. Phil. Antonius van Hoof

Dokumentverlauf

Version	Beschreibung/Änderung	Autor	Datum
1.0	Erstellung des Dokuments	Timo Zink, Fabian Eilber	23.03.2023
1.1	Korrektur und Finalisierung	Tobias Hahn	26.03.2023

Inhaltsverzeichnis

Dokumentverlauf 1
Inhaltsverzeichnis 1
0. Ziel des Dokuments 2
1. Produktübersicht 2
2. Struktur- und Entwurfsentscheidungen der Anwendung 2
2.1. Schichtenmodell 3
2.2. Angular 3
2.3. Spring Boot 3
2.3.1. Rest 4
2.4. MySQL 4
3. Struktur- und Entwurfsentscheidungen einzelner Pakete/Komponente 4
3.1. Eventmaster.Prototype.Core.Domain 4
3.2. Eventmaster.Prototype.Core.Interfaces 4
3.3. Eventmaster.Prototype.Core.Implementation 5
3.4. Eventmaster.Prototype.Angular 5
4. Quellen 5

0. Ziel des Dokuments

Ziel dieses Dokuments ist es, die softwareseitigen Designentscheidungen offenzulegen. Dabei wird auf die verwendete Technik eingegangen und der geplante Aufbau der Anwendung erläutert. Dies ist die Grundlage für den Aufbau des ersten Prototyps und folglich auch Grundstein für das spätere Produkt. Da die Entscheidung über das Design grundlegend ist, wurde diese mit dem ganzen Projektteam getroffen.

Für den ersten Prototypen können sich bestimmte Aspekte im Verlauf der Umsetzung ändern. Daher soll dieses Dokument als Grundlage, jedoch nicht als fest einzuhaltende Vorgabe zu sehen sein.

1. Produktübersicht

Die Anwendung soll im Wesentlichen die Möglichkeit einer Authentifizierung per E-Mail, das Erstellen und Verwalten von Events, das Suchen und Filtern nach Events und Datenverwaltung bieten. Für detaillierte Beschreibungen wird auf das Lastenheft verwiesen. In diesem Bericht geht es um die Struktur- und Entwurfsentscheidungen. Folgende Gesamtarchitektur ist zur Realisierung des Prototyps dargestellt:

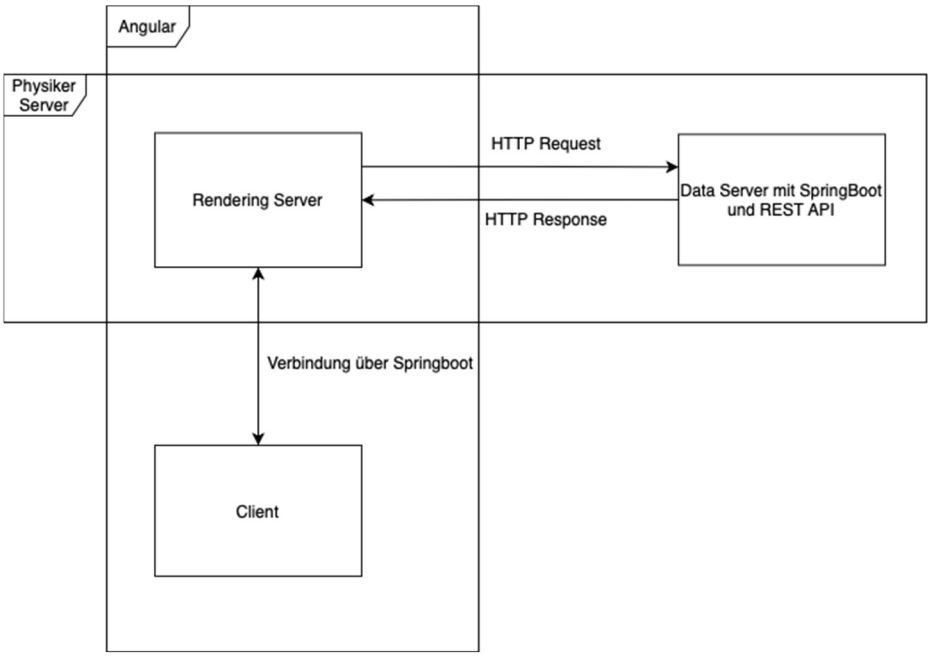

Abbildung 1 Gesamtarchitektur

2. Struktur- und Entwurfsentscheidungen der Anwendung

Im Folgenden wird das Architekturmuster der Anwendung, sowie die dafür eingesetzten Komponente genauer erläutert.

2.1. Schichtenmodell

Das Projektteam hat sich dazu entschieden, das Model-View-Controller (MVC) Entwurfsmuster anzuwenden. Indem die Anwendung in Schichten aufgeteilt wird, kann die Komplexität der Anwendung reduziert und gleichzeitig die Flexibilität erhöht werden. Demnach gliedert sich das Projekt in eine Präsentationsschicht, Geschäftsschicht und Datenhaltungsschicht.

Datenhaltungsschicht:
In der Datenhaltungsschicht befindet sich die Zugriffsmöglichkeit auf die Datenbank. Hier werden die Daten der Anwendung gespeichert und abgerufen.

Geschäftsschicht:
In der Geschäftsschicht befindet sich die Logik der Anwendung. Hier sollen die Kernfunktionalitäten ausgeführt werden. Diese beinhalten Datenvalidierung, Datenverarbeitung sowie Datenzugriff.

Präsentationsschicht:
In der Präsentationsschicht befindet sich die Benutzeroberfläche der Anwendung. Es handelt sich hier um die Schicht, die dem Benutzer direkt zugänglich ist und es ihm ermöglicht, mit der Anwendung zu interagieren.

2.2. Angular

Angular ist ein Framework zu Entwicklung von clientseitigen Webanwendungen mit TypeScript und JavaScript. Vorteile die für die Verwendung von Angular für die clientseitige Webentwicklung sind:

- Schreiben von Code in HTML und JavaScript bzw. TypeScript
- Angular-Anwendungen bestehen aus Komponenten, Diensten, Modulen und sind dadurch sehr modular.
- Angular bietet eine integrierte Routing-Funktionalität, mit der die Navigation zwischen verschiedenen Ansichten der Anwendung verwaltet werden können.

Angular Anwendungen basieren auf Komponenten. Eine Komponente in Angular ist ein Element der Benutzeroberfläche, beispielsweise eine Seite, ein Dialogfeld oder ein Formular für Dateneingaben. Weitere Informationen zu Angular finden sich unter folgender Quelle: (angular.io, 2023)

2.3. Spring Boot

Spring Boot ist ein Framework, das auf dem Spring Framework aufbaut. Es ist für die Entwicklung von Java-Anwendungen angedacht und erleichtert das Erstellen von RESTful Web Services. Spring Boot bietet dabei eine Reihe von vorgefertigten Bibliotheken, Tools und Konventionen, die es Entwicklern ermöglichen, sich auf die Geschäftslogik der Anwendung zu

konzentrieren. Weitere Informationen zu Spring Boot finden sich unter folgender Quelle: (Spring, 2023)

2.3.1. Rest

Die REST-API macht den Austausch von Informationen zwischen Benutzeroberfläche und Datenbank möglich. Besonders im Zeitalter von Computer und mobilen Endgeräten ist eine Benutzung von REST-APIs notwendig. Bei einer REST-API handelt es sich um eine Maschine-Maschine-Kommunikation. REST-APIs ermöglichen einen einheitlichen Informationsaustausch durch das zusammenbringen von Systemen und Geräten. Die Informationen werden mithilfe von HTTP-Requests ausgetauscht. Ein solcher HTTP-Request setzt sich zusammen aus einem definierten Endpoint und entsprechenden Parametern. Ein solcher Endpoint wird in der Spring Boot Applikation definiert. Über diesen können dann Informationen ausgetauscht werden.

2.4. MySQL

MySQL ist eine Open-Source Relationale Datenbankverwaltungssystemsoftware, die eine zuverlässige Lösung für die Speicherung, Abfrage und Verwaltung von Daten bietet. Dem Anwender wird eine breite Palette von Funktionen. Besonders die Skalierbarkeit ist hierbei ein erwähnenswerter Faktor. Von Vorteil ist zudem, dass MySQL auf vielen Plattformen betrieben werden kann und bereits in vielen Anwendungen und Web-Hosting-Umgebungen verbreitet ist. Weitere Informationen zu MySQL finden sich unter folgender Quelle: (MySQL, 2023)

3. Struktur- und Entwurfsentscheidungen einzelner Pakete/Komponente

Der Prototyp besteht aus einer Solution, welche sich wiederum in mehrere Pakete aufteilt. Jedes dieser Pakete stellt eine gekapselte Einheit dar. Nachfolgend werden die wichtigsten Pakete benannt und ihre Funktion erklärt. Dabei handelt es sich um eine Gedankliche Aufteilung der Pakete und ist so nicht eins zu eins in der Solution zu finden.

3.1. Eventmaster.Prototype.Core.Domain

Das Paket Core.Domain stellt die Schnittstelle zwischen dem Frontend und dem „Data Server“ (Backend) dar. Durch verschiedene Klassen oder diversen Hubs, wird die Kommunikation und Weiterleitung der Anfragen des Clients ermöglicht.

3.2. Eventmaster.Prototype.Core.Interfaces

Durch die Schnittstellen, die das Paket Core.Interfaces bereitstellt, wird die Zusammenarbeit erleichtert und die Abgrenzung der Funktionalitäten begünstigt. Außerdem bieten die

Interfaces die Möglichkeit, selbst im Prototyp Implementierungen auszutauschen. Dies geschieht durch Klassen mit gleicher Basisklasse oder Interface.

3.3. Eventmaster.Prototype.Core.Implementation

Die bereits im Pakte Core.Interfaces vorhandenen Schnittstellen werden im Paket Core.Implementation in konkreten Klassen implementiert. Diese Klassen werden als sogenannte „Beans“ verwendet. Sie besitzen wenig, bis keine funktionale Logik und dienen als reine „Datenspeicher“-Klassen, welche zur Laufzeit verwendet werden.

3.4. Eventmaster.Prototype.Angular

Im Paket Angular befindet sich die Komponenten für Angular und somit des Frontends. Die Komponenten teilen sich in die verschiedenen Teile der Webseite auf und wurden wie bereits erwähnt in HTML, JavaScript und TypeScript erstellt.

4. Quellen

angular.io. (22. 03 2023). Von Angular: https://angular.io/docs abgerufen

MySQL. (22. 03 2023). *dev.mysql.com*. Von MySQL: https://dev.mysql.com/doc/ abgerufen

Spring. (22. 03 2023). Von Spring: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/ abgerufen