C4 Modell

Nina Hartmann,
Clara Tschamon

Notiz:
Die Modelle wurden mit dem Tool „Drawio“ erstellt. Zusammenarbeit an der Datei war über Visual Studio LiveShare möglich.

Das **System Context Diagramm** zeigt die 4 voneinander unabhängigen Software Systeme:

- HotelBookingSystem
  - ➔ Abfragen von bookings, customers, free rooms und events
  - ➔ Erstellen von bookings, löschen von bookings, erstellen von customers, intialisieren der Datenbanken auf der readside und der writeside, löschen und reseten der Datenbanken auf der readside

- Externes QueryStorying System
  - ➔ Für jede eingegangene Query auf der readside wird in das QueryStoringSystem gespeichert, wann die Query abgesetzt wurde und was das Resultat der Qurey ist.

- Externes Protocolling System
  - ➔ Wird bei jedem abgesetzten Kommando auf der writeside benachrichtigt

Das **Container Diagram** zeigt die verschiedenen voneinander unabhängigen Container.
Alle sind voneinander unabhängig (sehr loose gekoppelt).

Der FX-Client kommuniziert über REST mir der Readside und der Writeside.
Die Writeside hat eine in-memory Database und verwendet die REST Schnittstelle der Eventside.

Die Readside ist für den FX-Client über eine REST-Schnittstelle zugänglich und macht eine subscription auf die Eventside und wird von dieser in Folge auch benachrichtigt. Die Readside liest aus und schreibt in eine in-memory Database.

Die Writeside Readside ist für den FX-Client über eine REST-Schnittstelle zugänglich. Sie schreibt in eine in-memory Database und verwendet die REST-Schnittstelle der Writeside.

Readside:

Im **Component Diagram** der Readside wird dargestellt, dass die Anfrage vom FX-Client an den Controller geht, welcher widerum die Projection verwendet, um die Queries zu handlen.
Die Projection speichert und liest aus einer in-memory Database und speichert informationen der Query ins externe QueryStoringSystem.
Außerdem verwendet der Controller den Communicator um mit der Eventside zu kommunizieren.
Außerdem verwendet die Eventside die RestSchnittstelle von dem Controller

Eventside:

Im **Component Diagram** der Eventside ist ersichtlich, dass die Writeside die RestSchnittstelle des EventsideRestControllers verwendet. Dieser verwendet Methoden des Eventservice. Der Eventservice schreibt in und liest aus der Eventstore Database. Außederem deligiert der das Benachrichtigen der subscribers an die EventSubscriber klasse. Diese verwendet für das „notify“ die REST-Schnittstelle der Readside.

Writeside:

Im **Component Diagram** der Writeside ist ersichtlich, dass der FX-Client kommandos an die REST-Schnittstelle im Controller sendet. Dieser deligiert Kommandos an das Aggregate, welches Funktionen der Datenbank verwendet, durch den EventPublisher an die EventSide sendet und außerdem das externe Protocolling System verwendet.

FX-Client:

Im **Component Diagram** des FX-Client ist ersichtlich, dass die Aktionen vom User auf der View vom MainController verwaltet werden. Dieser verwendet je nach Aktion einen anderen Adapter, welcher dann mit der Writeside oder Readside über REST kommuniziert.

FX-Client:

Im **Code Diagramm** der MainController Komponente und der Writeside Adapter Komponente kann erkannt werden, dass das Facade Design Pattern angewandt wurde und die Komponenten durch Verwendung von Dependency Injection loose gekoppelt sind.

Readside:

Im **Code Diagramm** der Projection ist dargestellt, dass die Projection

- Events verarbeitet,
- Objekte: Room, Customer und Booking erzeugt,
- Queries verarbeitet,
- und das interface IRepositoryFacade, welches die Projection durch Dependency Injection bereitgestellt bekommt, verwendet.

Im **Code Diagram** der in-memory Database ist dargestellt, dass die in-memory Database aus mehreren Repositories besteht, welche alle über eine Facade zugänglich gemacht werden.

Writeside:

Im **Code Diagramm** des Aggregates sieht man, dass das Aggregate

- Events erstellt,
- Bookings und Customers erstellt,
- das interface IRepositoryFacade, welches die Projection durch Dependency Injection bereitgestellt bekommt, verwendet
- und den EventPublisher, welchen die Projection durch Dependency Injection bereitgestellt bekommt, verwendet

Im **Code Diagramm** der in-memory Database ist ersichtlich, dass sie fast gleich aufgebaut ist wie die in-memory Database der Readside. Der Unterschied ist, dass die Writeside in-memory Database nicht über ein BookedRoomsReadRepository verfügt.

Eventside:

Es gibt kein Code Diagramm auf der Eventside, da die Eventside nur aus diesen 3 Klassen, welche ebenfalls Components sind, besteht.