Die Package-Struktur des Entwicklungs-Projekts

Schriftliche Ausarbeitung

vorgelegt von

Bjarne Christel
aus Mönchengladbach
geboren am: 29.03.1999
Matrikelnr.: 1344413

Tobias Piepers
aus Mönchengladbach
geboren am: 19.01.1998
Matrikelnr.: 1359763

Louisa Schmitz
aus Krefeld
geboren am: 11.07.2000
Matrikelnr.: 1319646

Hochschule Niederrhein
Fachbereich Wirtschaftswissenschaften
Studiengang M. Sc. Wirtschaftsinformatik

Wintersemester 2023/ 2024

Prüfer: Prof. Dr. Schekelmann

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Die beiden in Aufgabe 1 erstellten Bounded Contexts wurden in Aufgabe 2 mit der Programmiersprache Java praktisch umgesetzt und implementiert. Die Package-Struktur innerhalb der IDE Eclipse sollte dabei sowohl die Konzepte aus dem Domain-Driven-Design als auch aus dem Architekturmuster „Hexagonale Architektur“ berücksichtigen.
Dabei wurde sich an der Implementierung des Web-Shops aus der Vorlesung orientiert (vgl. Abbildung 1).

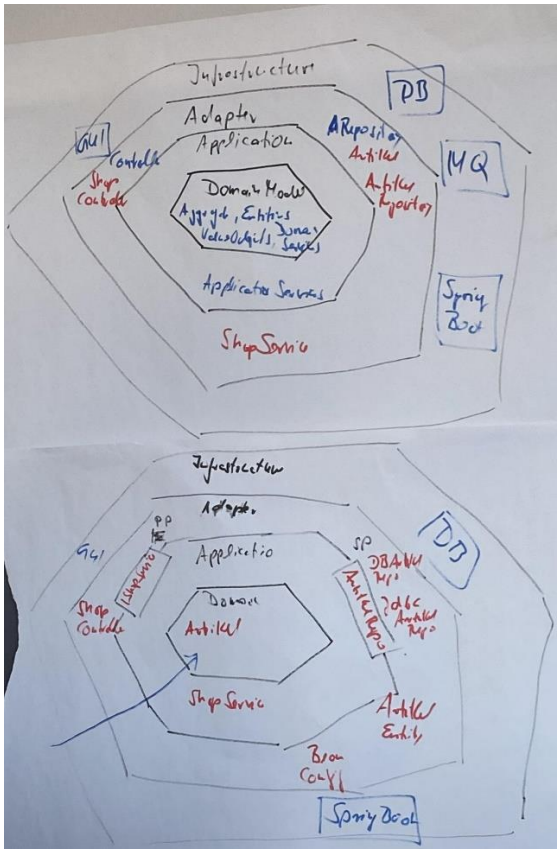

Abbildung 1 Beispiel einer hexagonalen Architektur[1]

2 Vorüberlegungen

Im Vorfeld der Implementierung hat sich die Projektgruppe Gedanken zu einer sinnvollen Package-Aufteilung gemacht, die die in der Vorlesung besprochenen Konzepte berücksichtigt. Dabei ist folgende Aufteilung entstanden (vgl. Abbildung 2):

[1] Vorlesung „Fortgeschrittene Softwareentwicklung“, WS 23/24.

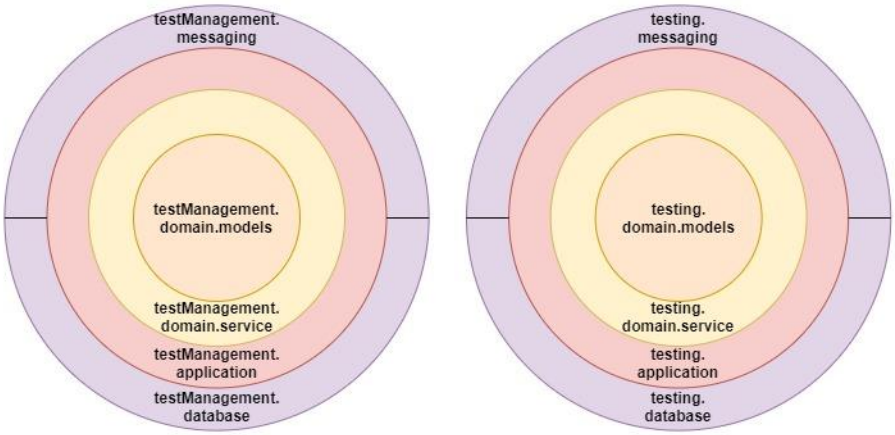

Abbildung 2 Onion-Architektur der beiden behandelten Bounded Contexts[2]

Bei der Erstellung der Paketstruktur wurde die Trennung von Fachlichkeit und Infrastruktur berücksichtigt, indem die fachliche Implementierung der Fachdomäne (fachliche Entitäten, Objekte und Methoden) innerhalb der dargestellten Onion-Architektur klar von den externen Implementierungen der Infrastruktur (Pakete: „database“ und „messaging“) getrennt ist.

3 Konkrete Umsetzung

Die konkrete Umsetzung der Struktur ist in Abbildung 3 zu sehen. Das Package „domain“ ist in das Package „model“ und das Package „service“ unterteilt. Im Paket „model“ sind alle Aggregates, Entities und Value Objects sowie die Transport-Objects vereint. Das Paket „service“ umfasst den Domain-Service „changeStatus“ mit dem dazugehörigen Domain-Event, das zum zweiten Microservice verschickt wird. Damit sind die beiden inneren Schichten der Onion-Architektur realisiert.

Die darüberliegende Applikationsschicht ist im Paket „application“ realisiert. Dieses Paket stellt die Schnittstellen für die außenstehende Infrastruktur bereit, die auf diese bereitgestellten Methoden zugreifen kann, ohne dabei die konkrete Implementierung zu kennen.
Die äußerste Schicht befasst sich mit der Infrastruktur und wird im Paket „adapter“ implementiert. Der erste Teil beschäftigt sich mit den CRUD-Operationen im Bereich „database“. Das Repository bietet Methoden zur Verarbeitung der Datenbankdaten an. Mit dem JDBC-Framework wird dieser Zugriff vereinfacht und muss nicht manuell implementiert werden. Das Paket „messaging“ ist für die konkrete Umsetzung der asynchronen,

[2] Eigene Darstellung.

ereignisbasierten Kommunikation zwischen den beiden Microservices verantwortlich. Das Interface stellt die Methoden für den Domain-Service bereit. Die konkrete Implementierung findet in den anderen Klassen statt.
Der Controller für die API-Schnittstelle und die Konfigurationsdatei für die Dependency Injection liegen ebenfalls in der Adapter-Schicht.

Auch im weiteren Verlauf des Projekts könnte diese Implementierung von Vorteil sein, falls eine Umstellung der Event-Abwicklung von RabbitMQ auf Kafka notwendig wird. Der Aufbau des Microservices ermöglicht eine unkomplizierte Umsetzung dieser Änderung, da die fachliche Domäne unabhängig vom Dienst ist, der das Event sendet oder empfängt, solange die Fachlichkeit innerhalb des Events korrekt ist.

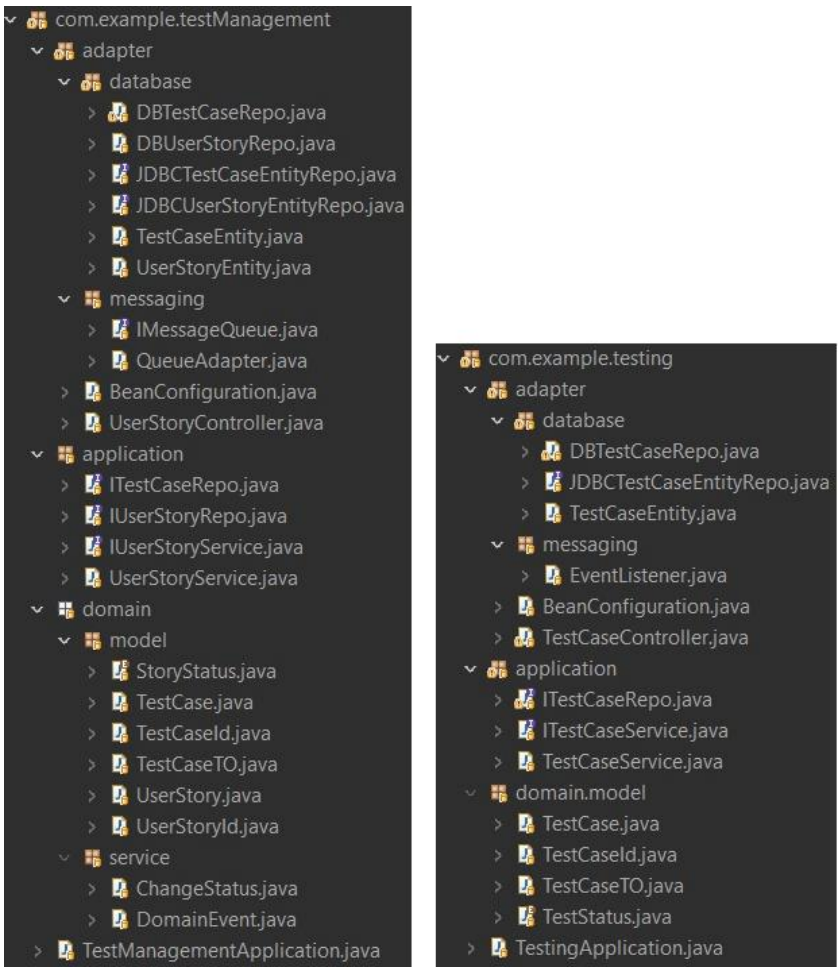

Abbildung 3 Package-Struktur in Eclipse[3]

[3] Eigene Darstellung.