Inhalt des Workshops

In diesem Workshop wird die Theorie von Domain Driven Design und hexagonaler Architektur vorgestellt und diskutiert. Durch User Stories angeleitet, wird in Gruppen eine hexagonale Architektur implementiert und anschließend in der Gruppe diskutiert.

Einführung in die Theorie von Domain Driven Design und hexagonaler Architektur.

Umsetzung einer hexagonalen Architektur in einer beliebigen Programmiersprache.

Beantworten der Frage, wie solch ein strukturierter Monolith in Microservices überführt werden kann.

Best Practice (Dos and Don'ts)

Musterlösung

Die Musterlösung und sämtliche Dokumentationen sind öffentlich über Github zugänglich, aber nicht frei verwendbar. Wenn Du Teile der Inhalte nutzen möchtest, kontaktiere mich bitte.

Präsentation und Stories zum Workshop

Die Stories zum Workshop finden sich in dem Github Repository im misc-Ordner.

Link zum Code

Die Musterlösungen zu den einzelnen Storys befinden sich jeweils in einem eigenen Git-Branch.

https://github.com/larmic/workshop_ddd_implementing_hexagonal_architecture

Kontakt

Lars Michaelis
neusta software development GmbH
28217 Bremen
Konsul-Smidt-Straße 24
Mail: lmichaelis@neusta.de

Lars Michaelis @larmicDE

neusta software development

Domain Driven Design und die Umsetzung in einer hexagonalen Architektur

mit Lars Michaelis

Domain Driven Design

ist ein Ansatz zur Modellierung komplexer Software und stellt dabei die fachlichen Aspekte in den Vordergrund. Die hexagonale Architektur greift diesen Punkt auf und setzt diese Inhalte in den Kern der Anwendung. Die eigentliche Technik (Datenbank, MQ-System, ...) spielt dabei eine untergeordnete Rolle.

Building Blocks

Das taktische Design von DDD gibt uns Zugriff auf eine Menge von unterschiedlichen Begriffen oder Entwurfsmustern für die Programmierung. Diese Begriffe werden in DDD Building Blocks genannt.

Ziel ist es, in der Zusammenarbeit zwischen Expert:innen und Entwickler:innen eine gemeinsame (technische) Sprache zu sprechen.

Ubiquitous Language

Eine gemeinsame Sprache, die die Kommunikation zwischen Entwickler:innen und Fachleuten erleichtert. Diese sollte sich im Code wiederfinden.

Bounded Context

Ein begrenzter Bereich, der die Fachlichkeit von anderen Bereichen kapselt und eine eigene Ubiquitous Language festlegt. Die Zuständigkeit eines Contexts fällt immer genau in ein Team.

Entity

Ein Objekt, das im Wesentlichen nicht durch seine Eigenschaften, sondern durch die Identität (in der Regel eine ID) definiert ist. Eine Entität hat immer einen Lebenszyklus.

Domain Driven Design

Value Object

Ein unveränderliches Object, das durch seine Eigenschaften und Attribute definiert ist. Besitzt in der Regel keine ID.

Aggregat

Eine Root-Entity, welche andere Entitäten bündelt. Das Herz der Domäne dient als zentraler Einstiegspunkt in die Domäne.

Factory

Kann genutzt werden, um komplexe Objekte wie Entitäten oder Aggregate zu erstellen. Häufig genügt aber einfach ein Konstruktor.

Repository

Ein Interface, welches die Verbindung zur Infrastruktur kapselt. In der Regel die Verknüpfung zu einer Datenbank – was aber nicht verpflichtend ist. Im weiteren Sinne dient ein Repository zur Kommunikation mit der Infrastruktur, wie z.B. einem externen System.

Service

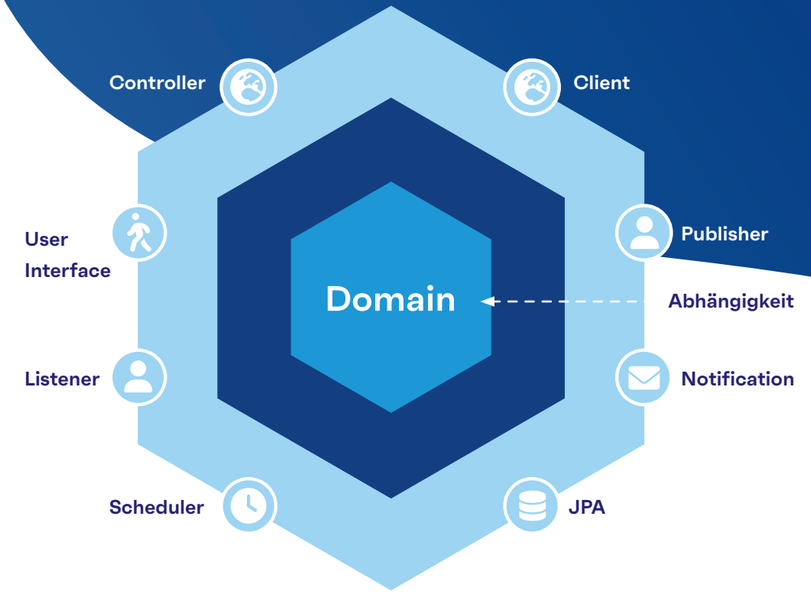
Ein Service (im Workshop auch Usecase genannt) beschreibt einen fachlichen Prozess. Wird in der Regel benötigt, wenn der Prozess mehr als nur ein Aggregat umfasst.

Domain Event

Ein fachliches Event zur Kommunikation zwischen verschiedenen Bounded Contexts.

Hexagonale Architektur

Die hexagonale Architektur ist ein Schichtmodell, das den fachlichen Kern in den Mittelpunkt stellt. Das Hexagon wird dabei genutzt, um die externen Schnittstellen thematisch sortiert zu gestalten und besteht in Verbindung mit DDD in der Regel aus drei Schichten. Die DDD Building Blocks sind einzelnen Schichten zugeordneten Bounded Contexts.



- **Domain:** Entity | Value Object | Aggregate | Repository | Domain Event
- **Application:** Service | Use Case
- **Infrastructure:** RepositoryImpl | REST-Schnittstelle | Sonstige Technik