

DDD und hexagonale Architektur in der Bank

JUGF - 25.05.2022

Andreas.Konrad@ •msg .group

Wer bin ich?

Andreas Konrad, Informatiker
Architekt, Softwareentwickler
15 Jahre Erfahrung (Java)



DDD & hexagonale Architektur

Andreas.Konrad@ •msg .group

Wer ist die msg 4 banking?

Firmengruppe msg, 9000 MA



msg 4 banking, 100 Jahre, 600 MA

München + Karlsruhe + Passau + weitere Standorte



DDD & hexagonale Architektur

Andreas.Konrad@ .msg .group

Wofür ist die Anwendung?

Annuitätendarlehen

- Nominalbetrag
- Auszahlungsdatum
- Laufzeit
- Tilgungsratenrhythmus
- p.a. Zins

Easy Credit 1-2-3

- Nominalbetrag: 10.000 €
- Auszahlungsdatum
- Laufzeit: 4 Jahre
- Tilgungsratenrhythmus: Monatlich
- ~~p.a. Zins~~



Tilgungsdarlehen

Name: Easy Credit 1-2-3



Termine

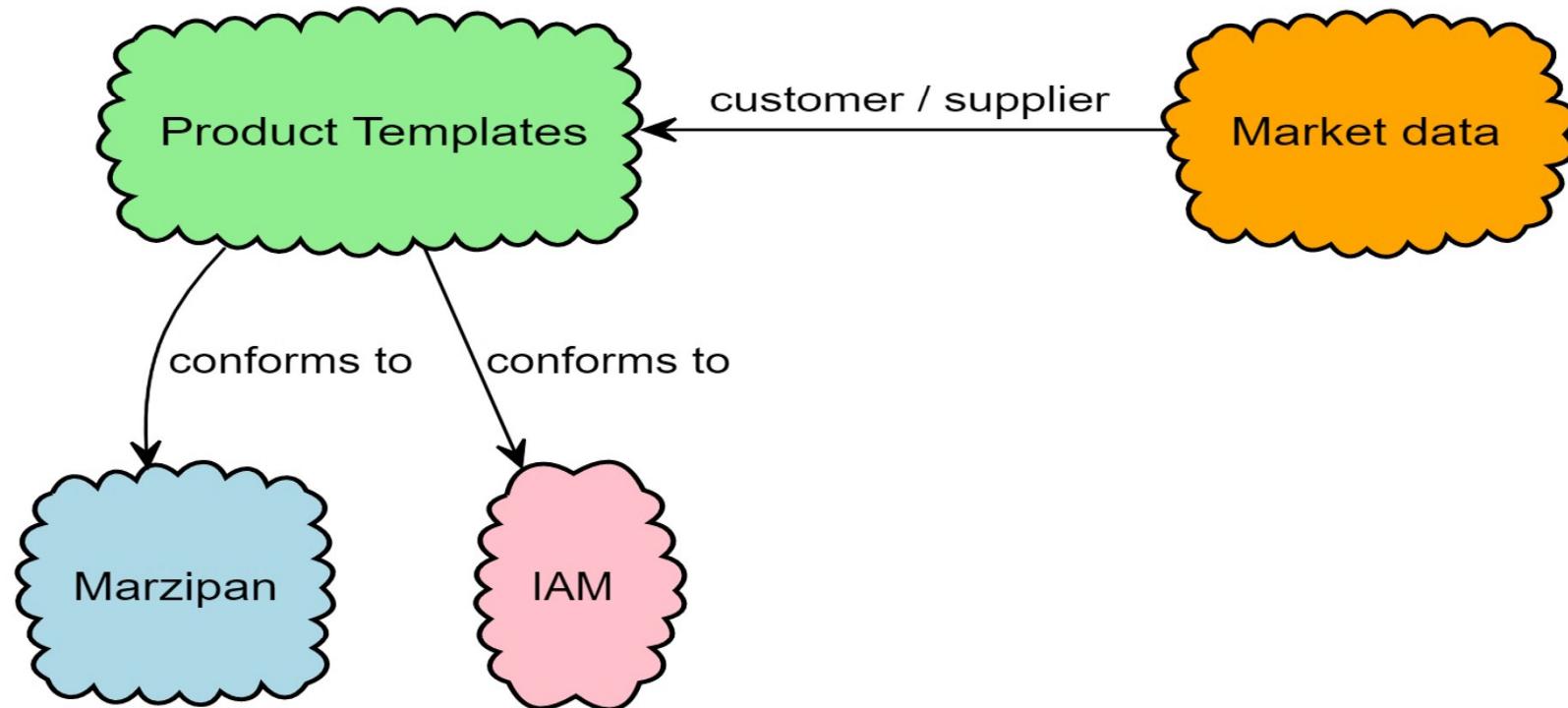
Anzeigemodus:

Name	Konfiguriert	Vorgabewert	Anzeige	Einstellungen
Suche nach Name				
Kalkulationsdatum	<input checked="" type="checkbox"/>			
Auszahlungsdatum	<input checked="" type="checkbox"/>			
Zinsbindung/Verrechnung bis (Datum)	<input checked="" type="checkbox"/>			
Erste Tilgung am	<input checked="" type="checkbox"/>			
Tag der Ratenzahlung	<input checked="" type="checkbox"/>	<input type="text"/>		
Tilgungsratenrhythmus	<input checked="" type="checkbox"/>	<input type="text"/>		
Anzahl voller Ratenzahlungsrhythmen	<input checked="" type="checkbox"/>	<input type="text"/>		
Erste Tilgungsanrechnung	<input checked="" type="checkbox"/>			
Tag der Tilgungsanrechnung	<input checked="" type="checkbox"/>	<input type="text"/>		
Tilgungsanrechnungsrhythmus	<input checked="" type="checkbox"/>	<input type="text"/>		
Anzahl voller Tilgungsanrechnungsrhythmen	<input checked="" type="checkbox"/>	<input type="text"/>		
Tilgungsbeginn gedeckter Teil	<input checked="" type="checkbox"/>			

Domain Driven Design

Ubiquitous language
Aggregates Bounded context Entities
Value Objects Repositories Services Strategic design
Tactical design Events

Context map



Strategic design 1

Use Cases?

Anzeige aktueller Marktdaten für den User

Datenmengen?

Daten pro Mandant < 10 KB

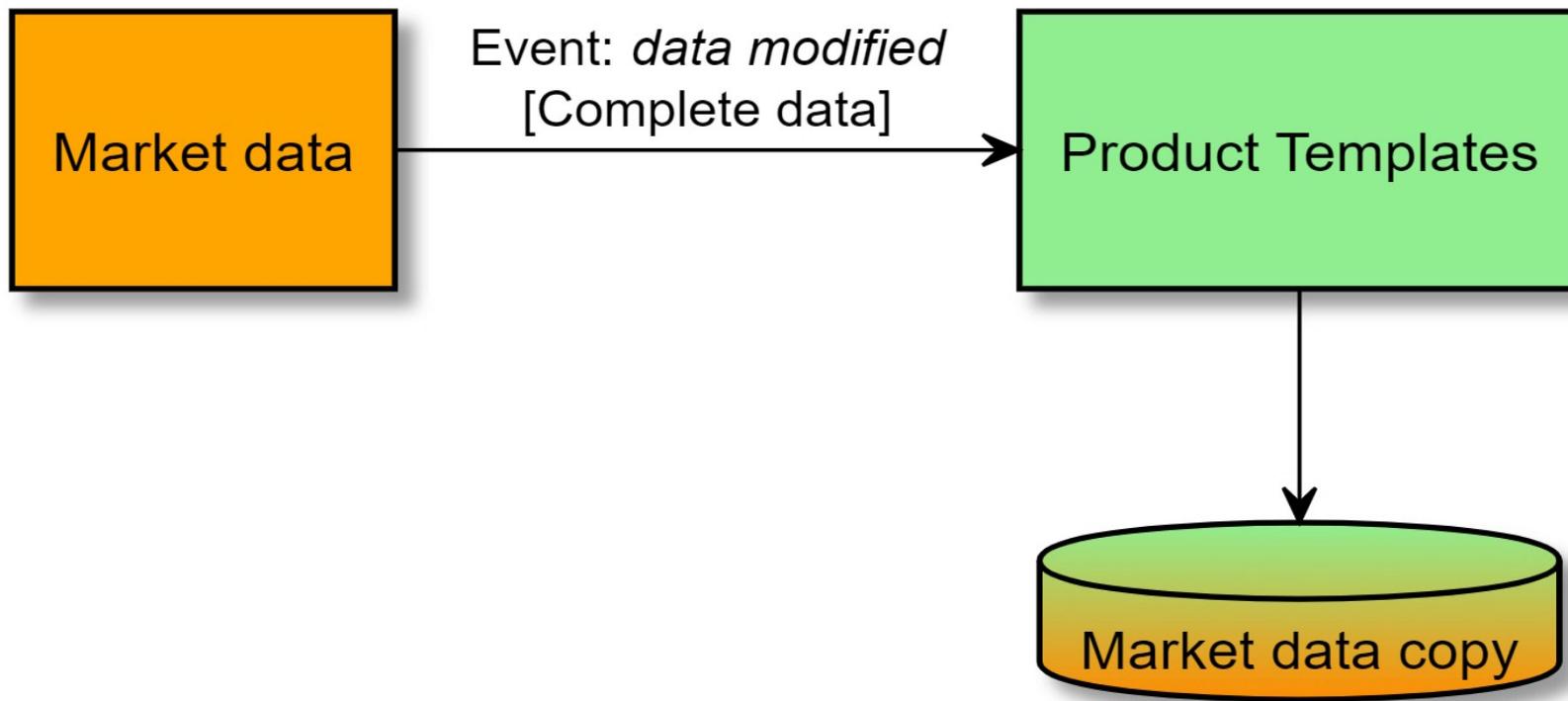
Anpassungsmöglichkeiten?

Marktdaten App kann angepasst werden

Fehlerbehandlung?

User soll keine Fehler sehen

Strategic design 2



Quiz

Was ist ein Nachteil der Lösung?

Eine initiale Datenbefüllung ist notwendig

Bounded context

Produkte, Produktvorlagen

~~Kreditgeschäfte~~

Variablen und -typen

~~Nominalbetrag, Laufzeit~~

Konfigurationsmöglichkeiten und Konzepte

~~Fachliche Beziehungen zwischen Variablen~~

Werte, Einheiten

~~Bedeutung der Konfiguration~~

Beispiel bounded context



Neues Konzept: Variablentyp abhängig von aktueller Konfiguration

Zusätzliche Abhängigkeit zu Marktdaten

Viel besser:

- i Über die Sichtbarkeitskonfiguration von „Zinsstruktur-Auswahl“ wird festgelegt, ob der Button bzw. die Buttons für die Auswahl von historischen Zinsstrukturen zur Verfügung stehen sollen. Zur Verdeutlichung, dass bei Deaktivierung der Zinsstruktur-Auswahl immer die aktuelle Zinsstruktur bzw. die aktuellen Zinsstrukturen bei der Berechnung gezogen werden, ist der Vorgabewert fest auf „Aktuelle Zinsstruktur“ eingestellt.

Ubiquitous language

Die Domäne bestimmt die Begriffe

Dev gibt Feedback

Klärt alle Begriffe mit BA

Dokumentieren

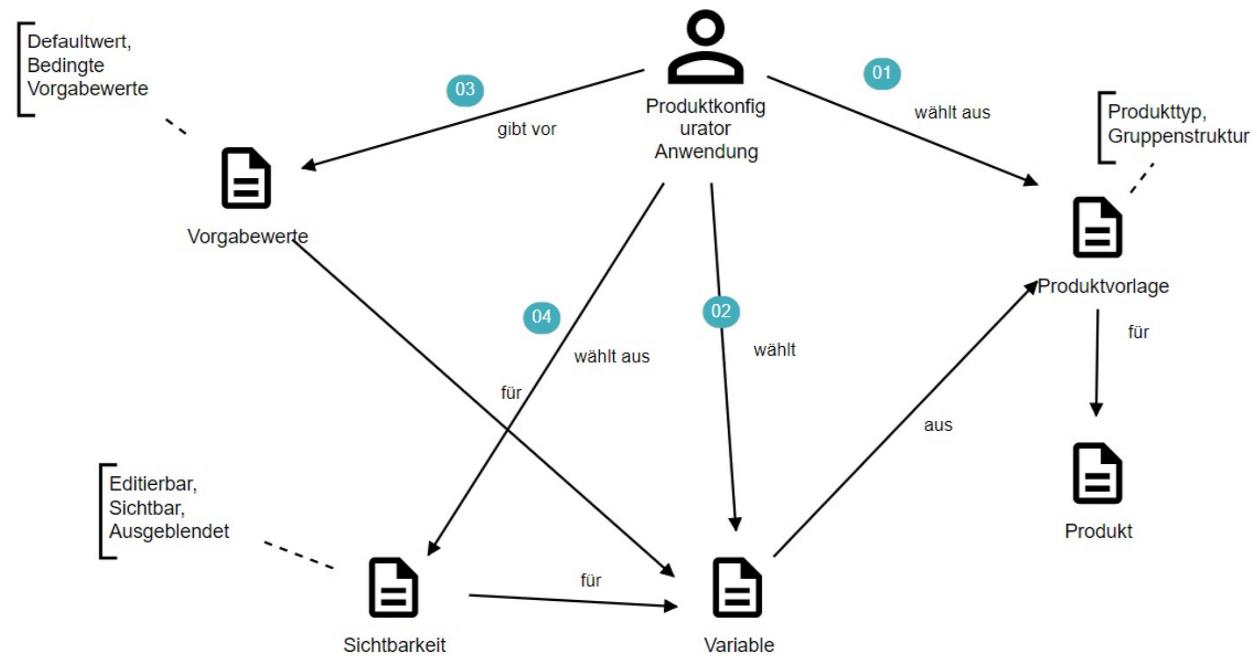
German	English	Explanation
Sichtbarkeit	Visibility	Variables and groups have a property visibility. The possible values are listed below
editierbar	editable	The variable can be edited by the user (in Marzipan)
		Can be configured for variables of category input
schreibgeschützt	read only	The variable is visible for the user but cannot be edited (in Marzipan)
		Can be configured for variables of category input
sichtbar	visible	The variable or group is visible for the user (in Marzipan)
		Can be configured for groups and variables of category output
ausgeblendet	hidden	The variable or group is not visible for the user (in Marzipan)
		Can be configured for groups and variables of type input and output

Domain story telling

Passt das Thema?

Lass den BA erzählen

wps.de/modeler



DDD & hexagonale Architektur

Andreas.Konrad@ .msg .group

Tactical design

Aggregate Entity
Value Object Repository
Service Event

Aggregate

```
public class Product extends PersistedEntity<ProductId> implements AggregateRoot {

    private final ProductTemplateId templateId;
    private final List<ProductGroup> groups;
    private final ProductStatusInfo statusInfo;
    private ProductName name;
    private Set<VariableConfiguration> variables;
    private String description;

    public Product(PersistenceInfo<ProductId> persistenceInfo, ProductTemplateId templateId,
                  String productType, List<ProductGroup> groups, Set<VariableConfiguration> variables,
                  ProductStatusInfo statusInfo) {
        super(persistenceInfo);
        notNull(persistenceInfo, "persistenceInfo");
        notNull(persistenceInfo.getIdentifier(), "ProductId");
        this.templateId = notNull(templateId, "templateId");
        this.productType = notNull(productType, "productType");
        this.groups = Collections.unmodifiableList(notEmpty(groups, "productGroups"));
        this.variables = Collections.unmodifiableSet(notEmpty(variables, "productVariables"));
        this.statusInfo = notNull(statusInfo, "statusInfo");
        validate();
    }
}
```

Aggregate: Kommunikation 1

ProductTemplate

TemplateGroup

VariableDefinition



Product

ProductGroup

VariableConfiguration

Domain Service

Aggregate: Kommunikation 2

Set<ListOfCompetenceOfTerms>

String name

ListOfPredefinedValues



List<String> values

Event

Entity

```
public class ListOfPredefinedValues extends IdentifiedObject<ListOfPredefinedValuesId>
    implements AggregateRoot {

    private final boolean mutable;
    private List<String> values;

    public ListOfPredefinedValues(String name, boolean mutable, List<String> values) {
        super(new ListOfPredefinedValuesId(notEmpty(name, "ListOfPredefinedValues name")));
        this.values = values == null ? new ArrayList<>() : new ArrayList<>(values);
        this.mutable = mutable;
    }

    public void update(List<String> values) {
        if (mutable) {
            this.values = values == null ? new ArrayList<>() : new ArrayList<>(values);
        } else {
            throw new ListOfPredefinedValuesCanNotBeUpdatedException(getName());
        }
    }
}
```



Value Object

```
public class ProductName implements ValueObject {  
    private final String value;  
  
    public ProductName(String value) {  
        this.value = notEmpty(value, "ProductName");  
    }  
  
    public String value() {  
        return value;  
    }  
}
```

Repository

```
public interface ProductTemplateRepository extends AggregateRepository {  
    void save(ProductTemplate productTemplate);  
    Collection<ProductTemplate> findAll();  
    Optional<ProductTemplate> findByIdentifier(ProductTemplateId identifier);  
    Collection<ProductTemplate> findByName(String name);  
    Collection<ProductTemplate> findByProductType(String productType);  
    void deleteAll();  
}
```

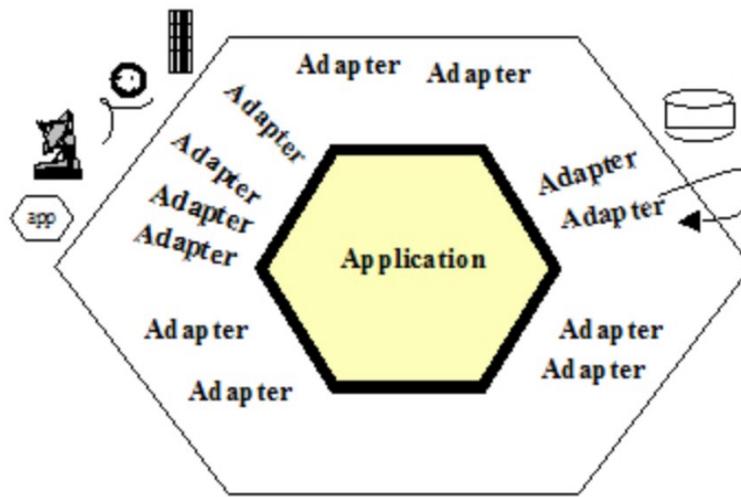
Command query separation

```
public class Product extends PersistedEntity<ProductId> implements AggregateRoot {  
  
    private final ProductTemplateId templateId;  
    private final List<ProductGroup> groups;  
    private final ProductStatusInfo statusInfo;  
    private ProductName name;  
    private Set<VariableConfiguration> variables;  
    private String description;
```



```
public interface ProductMetaDataQuery extends UseCaseQuery {  
  
    List<ProductMetaDataDto> findAllProductMetaData();  
    List<ProductMetaDataDto> findProductMetaDataByIds(Collection<ProductId> identifiers);  
    String findTypeOfProduct(ProductId productId);  
}
```

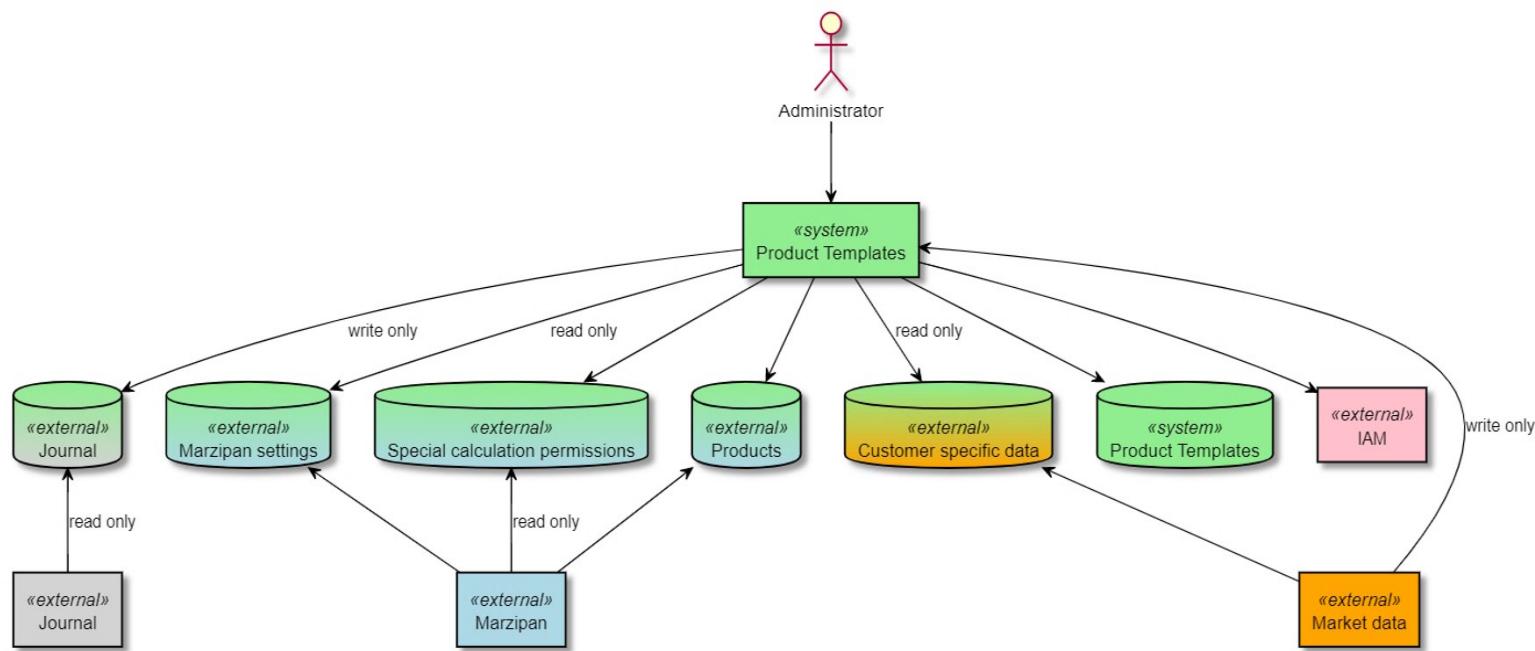
Hexagonale Architektur



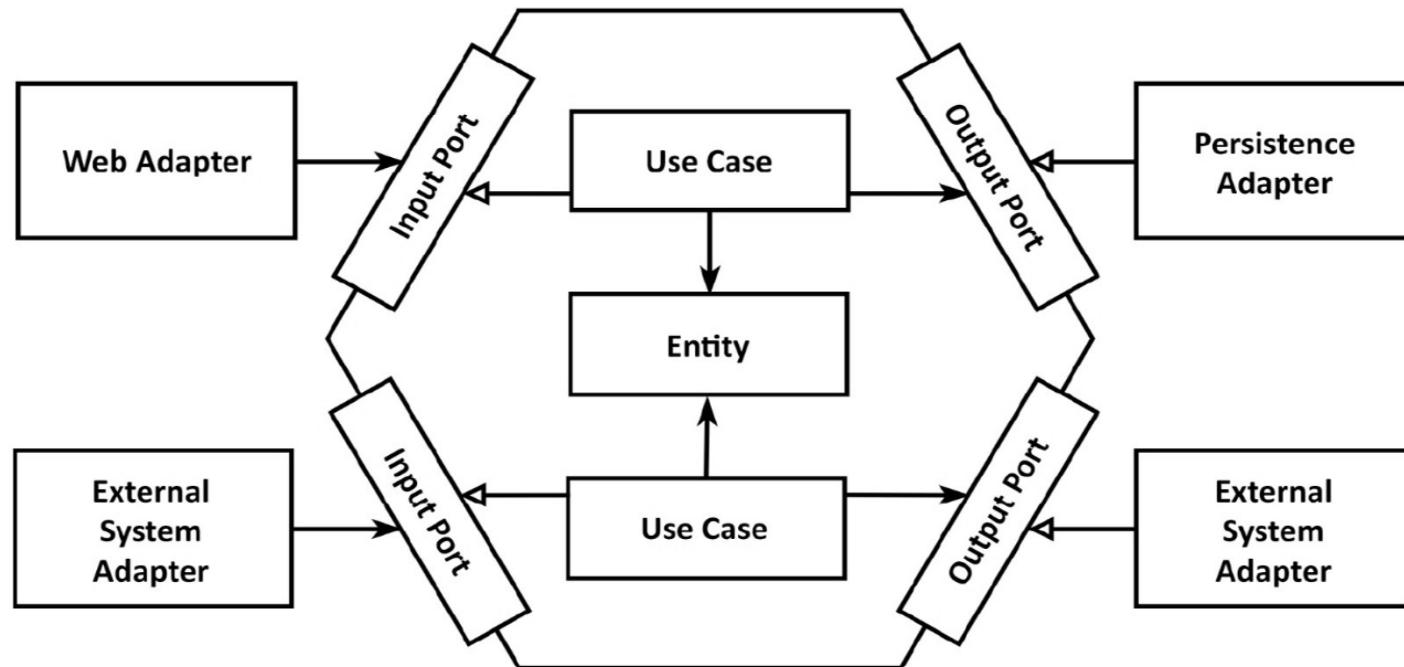
DDD & hexagonale Architektur

Andreas.Konrad@ •msg .group

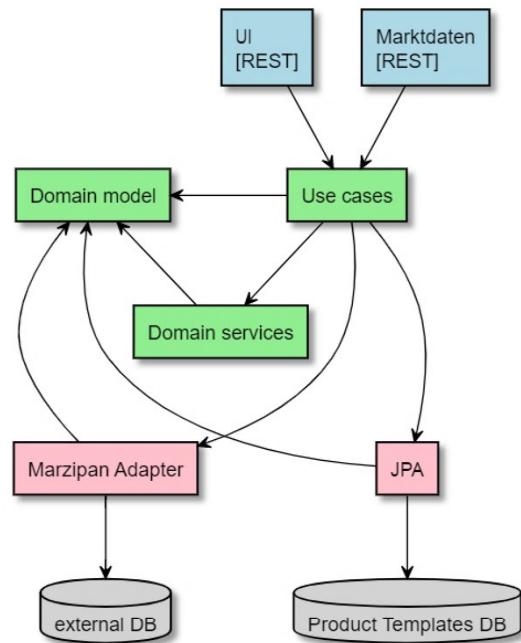
Motivation



Ports and adapters



Functional view



Spring application module

Cross cutting concerns

- IAM
- Tenants
- Exceptions

Vorteile

Die Domain ist frei von Technik

- REST
- Spring
- JPA

Die Domain ist frei von Abhängigkeiten

- Restliche Anwendung
- Repositories
- Externe Systeme

Domain pom.xml

```
<dependencies>
    <dependency>
        <groupId>de.msggillardon.producttemplates</groupId>
        <artifactId>producttemplates-exceptions</artifactId>
        <version>${project.version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-log4j2</artifactId>
    </dependency>
</dependencies>
```

Auswirkungen

Use Cases sind von zentraler Bedeutung

- Querschnittssaspekte
- Persistenz Interaktionen
- Transformationen

Ausnahmen: Logging, Lombok, Exceptions

Beispiel Use Case

```
@Override
public ProductSaveResponse handle(ProductSaveRequest request) {
    ProductContext productContext = createProductContext(request.getTemplateId());

    Product product = isReferringToExistingProduct(request) ?
        loadProduct(request, productContext)
        : createProduct(request.getProductId(), productContext);

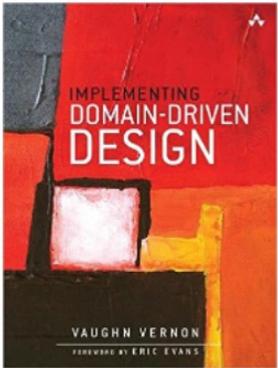
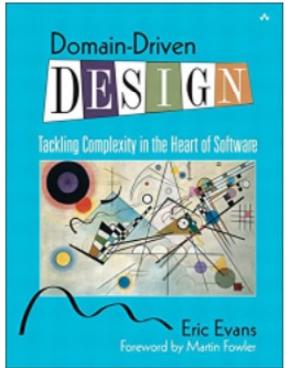
    validateAndUpdateProduct(request, product, productContext);

    product = validateAndSaveProduct(product, productContext);

    createJournalEntry(product, isReferringToExistingProduct(request));

    return createResponse(product);
}
```

Quellen



<https://alistair.cockburn.us/hexagonal-architecture/>

DDD & hexagonale Architektur

Andreas.Konrad@ .msg .group

Zusammenfassung

Anwendung zur Konfiguration von Produkten einer Bank

Domain Driven Design

Ports and adapters Architektur



DDD & hexagonale Architektur

Andreas.Konrad@ .msg .group