

A) Theorie:

1. Was ist Softwarearchitektur?

Softwarearchitektur beschreibt die grundlegende Struktur einer Software und legt fest, wie verschiedene Komponenten miteinander interagieren. Sie definiert die Organisation des Systems, die Verteilung von Verantwortlichkeiten und die Zusammenarbeit der einzelnen Elemente. Dabei berücksichtigt sie funktionale und nicht-funktionale Anforderungen wie Performance, Skalierbarkeit, Wartbarkeit und Sicherheit.

2. Wie kann man Softwarearchitektur dokumentieren?

Die Dokumentation von Softwarearchitektur kann auf verschiedene Weisen erfolgen:

- **Architekturbeschreibungen:** Textuelle Beschreibungen der Architektur, einschließlich Diagrammen, Konzepten und Entscheidungen.
- **Architekturdiagramme:** Visuelle Darstellungen der Architektur, wie z. B. UML-Diagramme, Flussdiagramme oder Schichtenmodelle.
- **Dokumentation von Entscheidungen (Architecture Decision Records, ADRs):** Aufzeichnungen von Architekturentscheidungen, die den Kontext, die Probleme und die Lösungen dokumentieren.

3. Welches sind die wichtigsten Eigenschaften von Langlebigen Softwarearchitekturen (Lilienthal)?

Langlebige Softwarearchitekturen, wie von Michael Lilienthal beschrieben, weisen folgende Eigenschaften auf:

- **Robustheit:** Das System kann Änderungen und Störungen standhalten, ohne zusammenzubrechen.
- **Flexibilität:** Die Architektur erlaubt es, das System einfach anzupassen und zu erweitern.
- **Wartbarkeit:** Die Software ist leicht zu verstehen und zu ändern, was die Wartungskosten senkt.
- **Skalierbarkeit:** Die Architektur erlaubt es, das System bei steigenden Anforderungen zu erweitern, ohne dass es an Leistung verliert.
- **Testbarkeit:** Die Softwarearchitektur ermöglicht effektive Tests, um die Qualität des Systems sicherzustellen.

4. Was ist ein Modulith?

Ein Modulith ist eine Architektur, die Merkmale sowohl von monolithischen als auch von modularisierten Systemen vereint. Es handelt sich um eine monolithische Anwendung, die jedoch intern in gut abgegrenzte Module unterteilt ist. Dadurch können verschiedene Teile der Anwendung unabhängig voneinander entwickelt, getestet und skaliert werden, während sie als einzelne Einheit bereitgestellt werden.

5. Wie funktioniert die Ports and Adapters Architektur?

Die Ports and Adapters Architektur, auch bekannt als Hexagonal Architecture, trennt die Geschäftslogik von externen Einflüssen wie Datenbanken, Benutzerschnittstellen oder externen Services. Die Architektur besteht aus drei Hauptkomponenten:

- **Ports**: Definieren die Schnittstellen, über die die Anwendung mit der Außenwelt kommuniziert. Sie sind unabhängig von der Implementierung und werden von der Geschäftslogik genutzt.
- **Adapter**: Implementieren die konkrete Kommunikation mit externen Ressourcen über die Ports. Sie übersetzen die Anfragen und Daten in das für die Anwendung verwendete Format.
- **Geschäftslogik**: Enthält die eigentliche Funktionalität der Anwendung, die unabhängig von den externen Einflüssen ist und über die Ports aufgerufen wird.

6. DDD: Was sind die wesentlichen Bausteine des modellgetriebenen Entwurfs (Taktische Pattern) aus DDD?

Die wesentlichen Bausteine des modellgetriebenen Entwurfs (Taktische Pattern) aus Domain-Driven Design (DDD) sind:

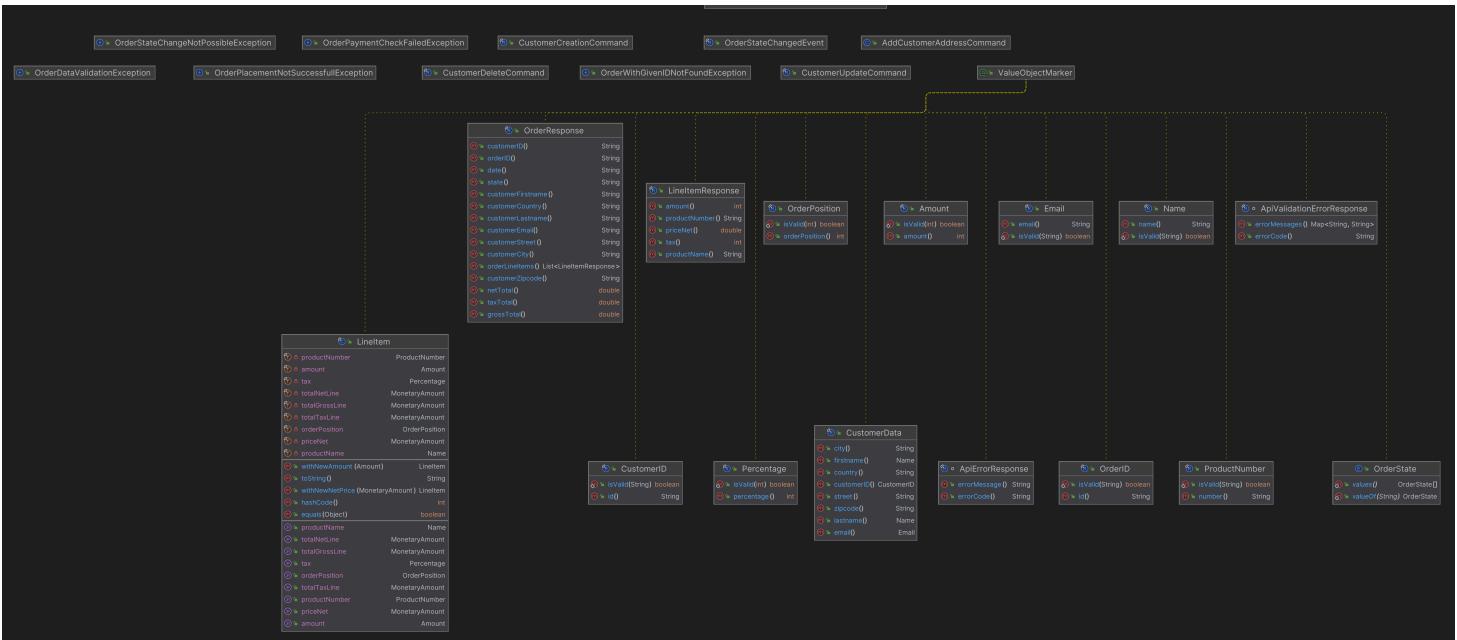
- **Entity**: Objekte mit einer eindeutigen Identität, deren Zustand sich im Laufe der Zeit ändern kann. Sie repräsentieren Dinge mit Lebenszyklen.
- **Value Object**: Objekte, die keinen eigenen Lebenszyklus haben und nur durch ihre Eigenschaften definiert sind. Sie werden für Konzepte verwendet, die durch ihre Attribute beschrieben werden.
- **Aggregate**: Gruppen von zusammengehörigen Objekten, die als eine Einheit behandelt werden. Sie haben eine Wurzel-Entität, die den Zugriff auf die anderen Objekte steuert.
- **Repository**: Ein Mechanismus zum Abrufen und Speichern von Aggregaten aus der Datenquelle. Es kapselt die Datenzugriffsdetails und bietet eine saubere Schnittstelle für die Anwendung.
- **Service**: Funktionen, die keine natürliche Zugehörigkeit zu einer Entität haben, aber dennoch Teil des Domänenkonzepts sind.
- **Factory**: Erzeugt komplexe Objekte oder Aggregat knoten und kapselt die Erstellungslogik.
- **Module**: Organisatorische Einheiten, die zusammengehörige Teile des Domänenwissens kapseln und voneinander isolieren.

- **Event:** Ereignisse, die relevante Änderungen im System beschreiben und für die Synchronisation zwischen verschiedenen Teilen des Systems verwendet werden können.

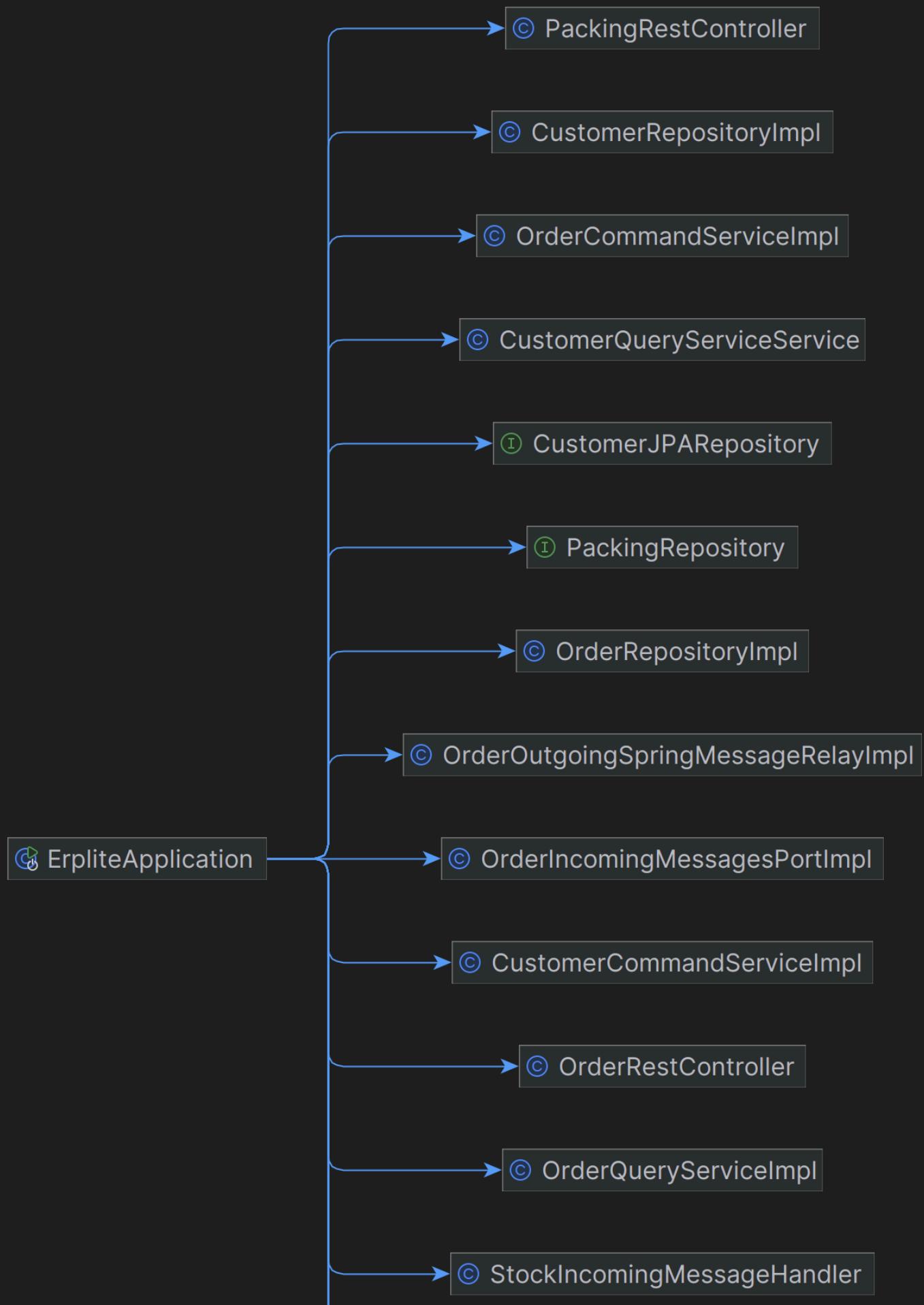
B) Architekturanalyse

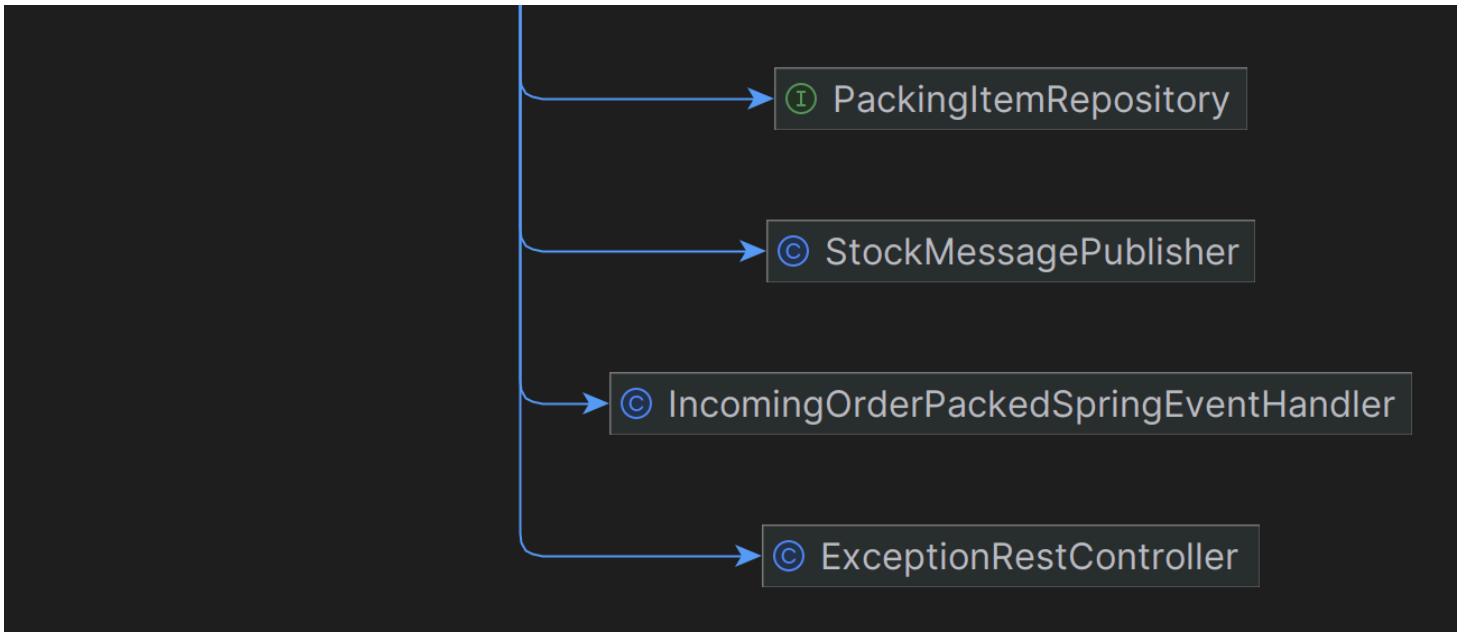
Modulith UML





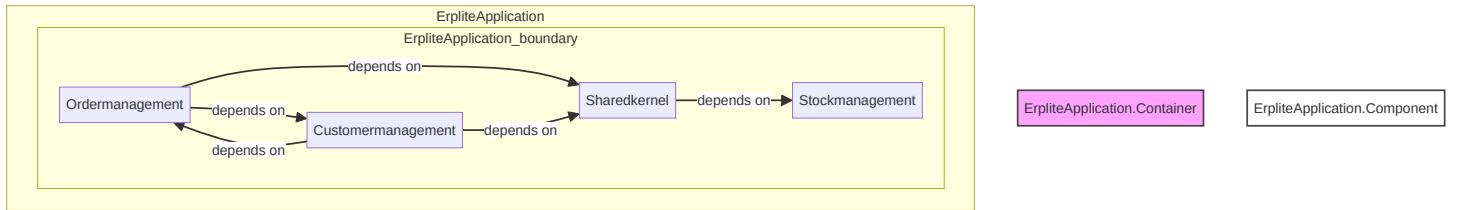
Modulith Spring dependency





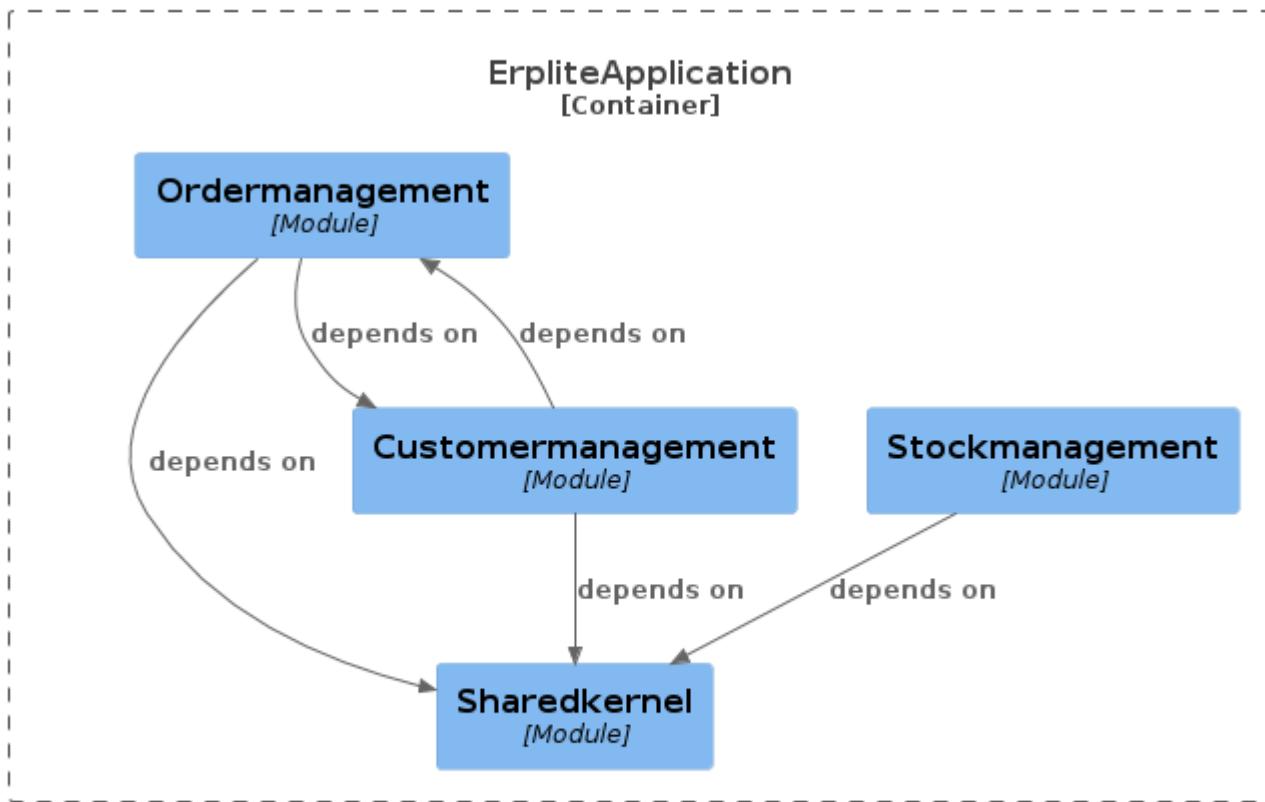
C) Modulith

Generiert mittels spring-modulith-docs, dann in mermaid konvertiert



Direkt von PUML

ErpliteApplication

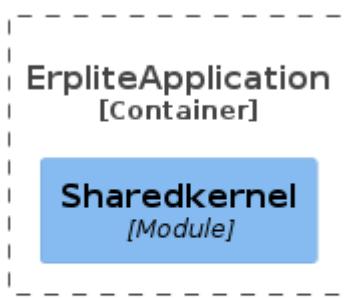


Legend

component

container boundary (dashed)

Sharedkernel

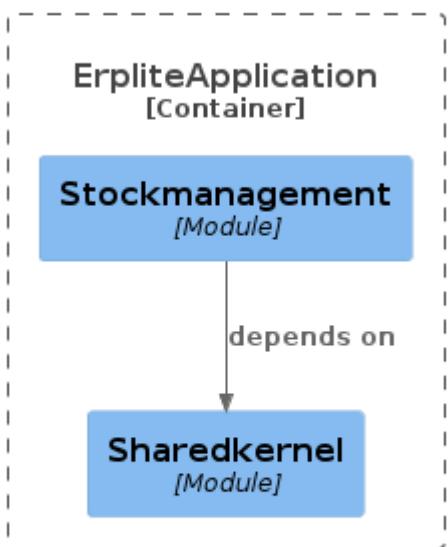


Legend

component

container boundary (dashed)

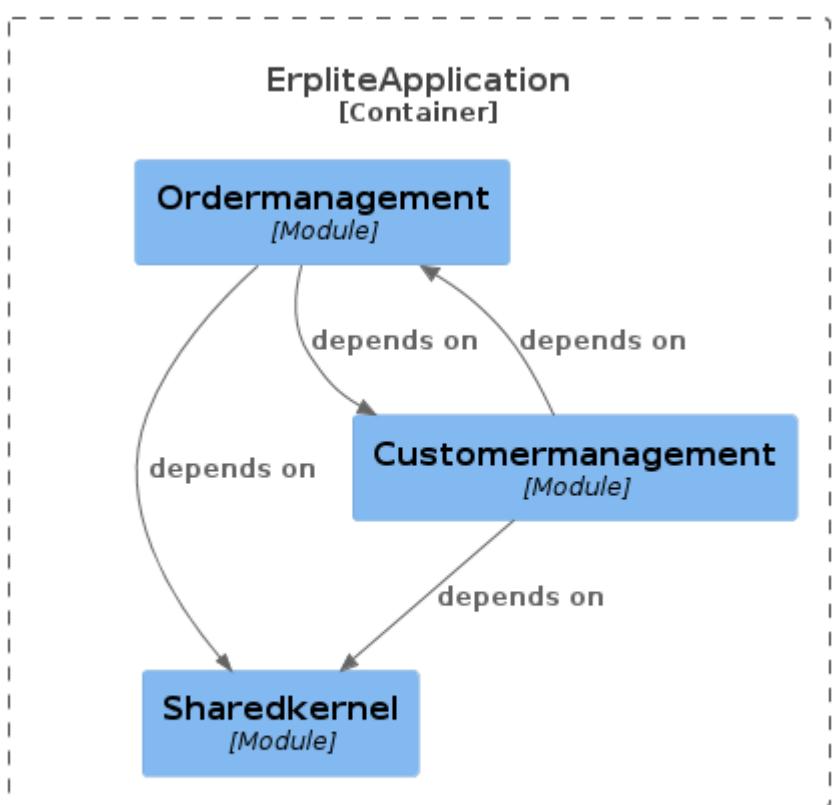
Stockmanagement



Legend

- component
- container boundary (dashed)

Ordermanagement



Legend

- component
- container boundary (dashed)

Die Library kann bei dem Testen und Erkennen von Zyklischen dependencies und genereller Analyse helfen.

Microservices

Projektaufbau

Dieses Projekt besteht aus 5 MicroServices, die komplett eigenständig sind.

Dockerfiles für die Erstellung der notwendigen Container sind im Dockerfile Ordner.

<input type="checkbox"/>	 erpliteinfrastructuresimple		Running (3/3)	21.23%	1 hour ago	  
<input type="checkbox"/>	 mariadb 9ac178cc76e4 	mariadb:10.3	Running	12.71%	3306:3306 	1 hour ago
<input type="checkbox"/>	 rabbitmq 3e361f26f8b9 	rabbitmq:3-management-alpine	Running	8.52%	15672:15672  Show all ports (2)	1 hour ago
<input type="checkbox"/>	 phpmyadmin2 6f5d9507bb1d 	phpmyadmin	Running	0%	8090:80 	1 hour ago

Project

- erplite-microservices-main C:\FSE\FSE_NEUN
 - .idea
 - apigateway [erpliteapigateway]
 - delivery
 - dockerfiles
 - frontend
 - orders [erplite]
 - servicediscovery [erpliteservicedisc]
 - stock [erplitestock]
 - .gitignore
 - doc.md
 - img.png
 - img_1.png
 - img_2.png
 - img_3.png
 - img_4.png
 - img_5.png
 - img_6.png

Services

- Spring Boot
 - Running
 - ApiGateway :9999/
 - DeliverMS :8082/
 - OrdersMS [devtools] :8080/
 - ServiceDiscovery :8761/
 - StockMS [devtools] :8081/
- Docker

RabbitMQ

RabbitMQ ist eine Messaging-Software, die das Nachrichtenvermittlungsprotokoll AMQP (Advanced Message Queuing Protocol) implementiert.

Die Queue enthält dabei beispielsweise eine Id und Prioritäten und gibt so die Meldungen weiter. RabbitMQ dient der Kommunikation zwischen verschiedenen Teilen einer Anwendung oder zwischen verschiedenen Anwendungen.

The screenshot shows the RabbitMQ Management Console interface. At the top, there's a navigation bar with tabs: Overview, Connections, Channels, Exchanges, **Queues and Streams** (which is currently selected), and Admin. Below the navigation bar, the title "Queues" is displayed, followed by a sub-section title "All queues (7)". A "Pagination" section includes a dropdown for "Page" set to 1, a "Filter" input field, and a "Regex" checkbox. The main content area is a table titled "Overview" with the following columns: Virtual host, Name, Type, Features, State, Ready, Unacked, Total, incoming, deliver / get, and ack. The table lists seven queues:

Virtual host	Name	Type	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack
/	q.initiate_delivery	classic	D	running	0	0	0	0.00/s	0.00/s	0.00/s
/	q.order_delivered	classic	D	running	0	0	0	0.00/s	0.00/s	0.00/s
/	q.order_in_delivery	classic	D	running	0	0	0	0.00/s	0.00/s	0.00/s
/	q.order_paymentchecked	classic	D	running	0	0	0	0.00/s	0.00/s	0.00/s
/	q.order_placed	classic	D	running	4	0	4	0.00/s		
/	q.orderpacked1	classic	D	running	0	0	0	0.00/s	0.00/s	0.00/s
/	q.orderpacked2	classic	D	running	0	2	2	0.00/s	267/s	0.00/s

Below the table, there's a link "▶ Add a new queue". At the bottom of the page, there's a footer with links: HTTP API, Documentation, Tutorials, New releases, Commercial edition, Commercial support, Discussions, Discord, Plugins, and GitHub.

Hier ein Beispiel für incoming and outgoing Messages im Ordermanagement.

Gezeigt im Code ist die Rabbit Klasse.

```
package at.kolleg.erplite.ordermanagement.messaging.rabbitmq;

import org.springframework.amqp.core.*;
import org.springframework.amqp.support.converter.Jackson2JsonMessageConverter;
import org.springframework.amqp.support.converter.MessageConverter;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class RabbitMQConfig {

    //https://www.rabbitmq.com/tutorials/tutorial-one-spring-amqp.html
    //https://springhow.com/spring-boot-rabbitmq/

    @Bean
    public Declarables createPostRegistrationSchema() {
        return new Declarables(
            new FanoutExchange("x.erplitefanout"),
            new TopicExchange("x.erplitetopic"),
            new Queue("q.orderpacked2"),
            new Queue("q.order_paymentchecked"),
            new Queue("q.order_placed"),
            new Queue("q.initiate_delivery"),
            new Queue("q.order_in_delivery"),
            new Queue("q.order_delivered"),
            new Binding("q.order_paymentchecked", Binding.DestinationType.QUEUE, "x.erplitefanout"),
            new Binding("q.order_placed", Binding.DestinationType.QUEUE, "x.erplitefanout"),
            new Binding("q.initiate_delivery", Binding.DestinationType.QUEUE, "x.erplitefanout"),
            new Binding("q.order_in_delivery", Binding.DestinationType.QUEUE, "x.erplitefanout"),
            new Binding("q.orderpacked2", Binding.DestinationType.QUEUE, "x.erplitefanout"),
            new Binding("q.order_delivered", Binding.DestinationType.QUEUE, "x.erplitefanout")
        );
    }

    @Bean
    public OrderIncomingRabbitMessageRelay receiver() {
        return new OrderIncomingRabbitMessageRelay();
    }

    @Bean
    MessageConverter messageConverter() {
        return new Jackson2JsonMessageConverter();
    }
}
```



```
package at.kolleg.erplite.ordermanagement.messaging.rabbitmq;

import at.kolleg.erplite.ordermanagement.marker.AdapterMarker;
import at.kolleg.erplite.ordermanagement.ports.out.OrderOutgoingMessageRelay;
import at.kolleg.erplite.sharedkernel.events.OrderInitiateDeliveryEvent;
import at.kolleg.erplite.sharedkernel.events.OrderPaymentValidatedEvent;
import at.kolleg.erplite.sharedkernel.events.OrderPlacedEvent;
import org.springframework.amqp.rabbit.core.RabbitTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.logging.Level;
import java.util.logging.Logger;

@Service
class OrderOutgoingRabbitMessageRelayImpl implements OrderOutgoingMessageRelay {

    @Autowired
    private RabbitTemplate template;

    /*
     * @Qualifier("orderPlaced")
     * @Autowired
     * private Queue orderPlacedQueue;
     *
     * @Qualifier("orderPaymentCheck")
     * @Autowired
     * private Queue orderPaymentCheckQueue; */

    @Override
    public void publish(final OrderPlacedEvent orderPlacedEvent) {
        Logger.getLogger(this.getClass().getName()).log(Level.INFO, "Publishing order placed event");
        this.template.convertAndSend("q.order_placed", orderPlacedEvent);
        Logger.getLogger(this.getClass().getName()).log(Level.INFO, "Order placed event published");
    }

    @Override
    public void publish(final OrderPaymentValidatedEvent orderPaymentValidatedEvent) {
        Logger.getLogger(this.getClass().getName()).log(Level.INFO, "Publishing order payment validated event");
        this.template.convertAndSend("q.order_paymentchecked", orderPaymentValidatedEvent);
        Logger.getLogger(this.getClass().getName()).log(Level.INFO, "Order payment validated event published");
    }
}
```

```
@Override  
public void publish(final OrderInitiateDeliveryEvent orderInitiateDeliveryEvent) {  
    Logger.getLogger(this.getClass().getName()).log(Level.INFO, "Publishing orderIn:  
this.template.convertAndSend("q.initiate_delivery", orderInitiateDeliveryEvent);  
    Logger.getLogger(this.getClass().getName()).log(Level.INFO, "OrderInitiateDelive  
}  
}  
}
```

```
package org.springframework.amqp.rabbit.annotation;

import java.lang.annotation.Documented;
import java.lang.annotation.ElementType;
import java.lang.annotation.Repeatable;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;
import org.springframework.messaging.handler.annotation.MessageMapping;

@Target({ElementType.TYPE, ElementType.METHOD, ElementType.ANNOTATION_TYPE})
@Retention(RetentionPolicy.RUNTIME)
@MessageMapping
@Documented
@Repeatable(RabbitListeners.class)
public @interface RabbitListener {
    String id() default "";

    String containerFactory() default "";

    String[] queues() default {};

    Queue[] queuesToDeclare() default {};

    boolean exclusive() default false;

    String priority() default "";

    String admin() default "";

    QueueBinding[] bindings() default {};

    String group() default "";

    String returnExceptions() default "";

    String errorHandler() default "";

    String concurrency() default "";

    String autoStartup() default "";

    String executor() default "";
}
```

```
String ackMode() default "";
String replyPostProcessor() default "";
String messageConverter() default "";
String replyContentType() default "";
String converterWinsContentType() default "true";
}
```

Frontend und Datenbank

Hier sind die Frontendeingaben und die Datenbankeinträge mit Screenshots dokumentiert.

The screenshot shows a web browser window with several tabs open. The tabs include "Portal Tirol", "FSE LAND NEUN", "itkolleg-imst / FS", "localhost:8090 / c", and "PMA". The main content area shows a search bar with the URL "http://127.0.0.1:5500/webshop.html". Below the search bar, there is a row of links: "Portal Tirol", "Global NetAcad Inst...", "Apple Developer", "W3Schools Online...", "Discord | #announc...", and a small orange icon.

[erplite] Frontend Prototype - WEBSHOP

[webshop]---[stock]---[delivery]

Orders

Testbestellung aufgeben

Order #ONR9981bcf for customer Caesar FRANKLIN. Status [PLACED]. Ordered products:

1. MacBook Pro 2022
2. Ipad Pro 2021

payment received

Order #ONRd7f9406 for customer Caesar FRANKLIN. Status [DELIVERED]. Ordered products:

1. MacBook Pro 2022
2. Ipad Pro 2021

The screenshot shows a database table with two rows of data. The columns are: orderid, date, gross_total, net_total, city, country, customer_id, email, firstname, lastname, street, zipcode, state, tax_total, and version. The first row (order #ONR9981bcf) has values: ONR9981bcf, 2024-05-08 09:31:56, 2299.89, 1999.90, LA, USA, CUS1d34e56, a.b@c.de, Caesar, Franklin, Hollywood Boulevard 3452, 299.99, 0. The second row (order #ONRd7f9406) has values: ONRd7f9406, 2024-05-08 09:10:34, 2299.89, 1999.90, LA, USA, CUS1d34e56, a.b@c.de, Caesar, Franklin, Hollywood Boulevard 3452, 299.99, 4.

	orderid	date	gross_total	net_total	city	country	customer_id	email	firstname	lastname	street	zipcode	state	tax_total	version
<input type="checkbox"/>	ONR9981bcf	2024-05-08 09:31:56	2299.89	1999.90	LA	USA	CUS1d34e56	a.b@c.de	Caesar	Franklin	Hollywood Boulevard 3452	299.99	0		
<input type="checkbox"/>	ONRd7f9406	2024-05-08 09:10:34	2299.89	1999.90	LA	USA	CUS1d34e56	a.b@c.de	Caesar	Franklin	Hollywood Boulevard 3452	299.99	4		

Operationen für das Abfrageergebnis

Alles anzeigen | Anzahl der Datensätze: 25 | Zeilen filtern: Diese Tabelle durchsuchen | Nach Schlüssel sortieren: keine

Zusätzliche Optionen

order_db_entity_orderid	amount	order_position	price_net	product_name	product_number	tax	total_gross_line	total_net_line	total_tax_line
ONRd7f9406	1	1	1000.00	MacBook Pro 2022	P123RE123D	20	1200.00	1000.00	200.00
ONRd7f9406	10	2	99.99	Ipad Pro 2021	O12345RE12	10	1099.89	999.90	99.99
ONR9981bcf	1	1	1000.00	MacBook Pro 2022	P123RE123D	20	1200.00	1000.00	200.00
ONR9981bcf	10	2	99.99	Ipad Pro 2021	O12345RE12	10	1099.89	999.90	99.99

Alles anzeigen | Anzahl der Datensätze: 25 | Zeilen filtern: Diese Tabelle durchsuchen | Nach Schlüssel sortieren: keine

Operationen für das Abfrageergebnis

[erplite] Frontend Prototype - STOCK

[\[webshop\]](#)---[\[stock\]](#)---[\[delivery\]](#)

Packings

Packing id 1 for order #ONR487403e. Packing items:

- [packing item #2 for product MacBook Pro 2022] [packed]
- [packing item #3 for product Ipad Pro 2021] [packed]

Packing id 4 for order #ONR7df5d4c. Packing items:

- [packing item #5 for product MacBook Pro 2022] [packed]
- [packing item #6 for product Ipad Pro 2021] [packed]

Packing id 7 for order #ONRd7f9406. Packing items:

- [packing item #8 for product MacBook Pro 2022] [packed]
- [packing item #9 for product Ipad Pro 2021] [packed]

Packing id 10 for order #ONR9981bcf. Packing items:

- [packing item #11 for product MacBook Pro 2022] mark packed
- [packing item #12 for product Ipad Pro 2021] mark packed

Messen [Inline bearbeiten] [Bearbeiten] [SQL erklären] [PHP-Code erzeugen] [Aktualisieren]

Alles anzeigen | Anzahl der Datensätze: 25 | Zeilen filtern: Diese Tabelle durchsuchen | Nach Schlüssel sortieren: kein

Zusätzliche Optionen

		orderid	date	gross_total	net_total	city	country	customer_id	email	firstname	lastname	street	zipcode	state	tax_total	ver
<input type="checkbox"/>	Bearbeiten Kopieren Löschen	ONR9981bcf	2024-05-08 09:31:56	2299.89	1999.90	LA	USA	CUS1d34e56	a.b@c.de	Caesar	Franklin	Hollywood Boulevard 2	3452	PAYMENT_VERIFIED	299.99	
<input type="checkbox"/>	Bearbeiten Kopieren Löschen	ONRd7f9406	2024-05-08 09:10:34	2299.89	1999.90	LA	USA	CUS1d34e56	a.b@c.de	Caesar	Franklin	Hollywood Boulevard 2	3452	DELIVERED	299.99	

Alle auswählen markierte: Bearbeiten Kopieren Löschen Exportieren

Portal Tirol Global NetAcad Inst... Apple Developer W3Schools Online... Discord | #announc... GitLab.org Englisch ⇔ Deutsch... Chat-Gpt Wire GAMP_GIT_SAP Alle Lesezeichen

phpMyAdmin

Server db » Datenbank deliverymgmt » Tabelle order_delivery

Anzeigen Struktur SQL Suche Einfügen Exportieren Importieren Rechte Operationen Trigger

Zeige Datensätze 0 - 1 (insgesamt, Die Abfrage dauerte 0.0001 Sekunden.)

```
SELECT * FROM `order_delivery`
```

Messen [Inline bearbeiten] [Bearbeiten] [SQL erklären] [PHP-Code erzeugen] [Aktualisieren]

Alles anzeigen | Anzahl der Datensätze: 25 | Zeilen filtern: Diese Tabelle durchsuchen | Nach Schlüssel sortieren: keine

Zusätzliche Optionen

		delivery_id	customer_city	customer_country	customer_email	customer_firstname	customerid	customer_lastname	customer_street	customer_zipcode	delivered	order_id
<input type="checkbox"/>	Bearbeiten Kopieren Löschen	1	LA	USA	a.b@c.de	Caesar	CUS1d34e56	Franklin	Hollywood Boulevard 2	3452	1	0
<input type="checkbox"/>	Bearbeiten Kopieren Löschen	2	LA	USA	a.b@c.de	Caesar	CUS1d34e56	Franklin	Hollywood Boulevard 2	3452	0	0

Alle auswählen markierte: Bearbeiten Kopieren Löschen Exportieren

Operationen für das Abfrageergebnis

Drucken In Zwischenablage kopieren Exportieren Diagramm anzeigen Erzeuge View

Konsole

Portal Tirol | FSE LAND NEUN | itkolleg-imst / FS | localhost:8090 / | RabbitMQ Manager | Neuer Tab

http://127.0.0.1:5500/delivery.html

Portal Tirol Global NetAcad Inst... Apple Developer W3Schools Online... Discord | #announc... GitLab.org Englisch ⇔ Deutsch...

[erplite] Frontend Prototype - DELIVERY

[webshop]---[stock]---[delivery]

Registered deliveries

1. Delivery id 1 for order #ONRd7f9406 [delivered]
2. Delivery id 2 for order #ONR9981bcf [delivered]

http://localhost:8090/index.php?route=/sql&pos=0&db=ordermgmt&table=order_db_entity

phpMyAdmin

Server: db » Datenbank: ordermgmt » Tabelle: order_db_entity

Anzeigen Struktur SQL Suche Einfügen Exportieren Importieren Rechte Operationen Trigger

Zeige Datensätze 0 - 1 (2 insgesamt, Die Abfrage dauerte 0.0003 Sekunden.)

SELECT * FROM `order_db_entity`

Messen [Inline bearbeiten] [Bearbeiten] [SQL erklären] [PHP-Code erzeugen] [Aktualisieren]

Alles anzeigen Anzahl der Datensätze: 25 Zeilen filtern: Diese Tabelle durchsuchen Nach Schlüssel sortieren: keine

orderid	date	gross_total	net_total	city	country	customer_id	email	firstname	lastname	street	zipcode	state	tax_total	version
ONR9981bcf	2024-05-08 09:31:56	2299.89	1999.90	LA	USA	CUS1d34e56	a.b@c.de	Caesar	Franklin	Hollywood Boulevard	3452	DELIVERED	299.99	4
ONRd7f9406	2024-05-08 09:10:34	2299.89	1999.90	LA	USA	CUS1d34e56	a.b@c.de	Caesar	Franklin	Hollywood Boulevard	3452	DELIVERED	299.99	4

Alle auswählen markierte: Bearbeiten Kopieren Löschen Alle auswählen Anzahl der Datensätze: 25 Zeilen filtern: Diese Tabelle durchsuchen Nach Schlüssel sortieren: keine

Operationen für das Abfrageergebnis

Drucken In Zwischenablage kopieren Exportieren Diagramm anzeigen Erzeuge View