

# Entwicklerdokumentation

Version	Bearbeitungsdatum	Autor
1.0.0	17.11.2019	Till Große, Tobias Köllner, Anna Lopatkina, Kien Dang Tran, Viktor Reusch

## 1. Einführung und Ziele

### 1.1. Aufgabenstellung

Für *Miss-Mint Mending Points* soll für die Mitarbeiter eine Software bereitgestellt werden. Mitarbeiter (*Staff*) können über den Admin registriert (*registerStaff*) und entlassen (*deleteUser*) werden. Unauthorisierte Personen sollen keinen Zugriff auf betriebsinterne Bereiche bekommen (*@PreAuthorize("isAuthenticated()")*). Der Betrieb stellt folgende Dienstleistungen (*MissMintService*) in Kategorien (*ServiceCategory*) bereit:

- eine Flickschusterei (*KLUDGE*)
- einen Nähservice (*SEWING*)
- einen Schlüsseldienst (*LOCKSMITH*)
- eine Schnellreinigung (*CLEANING*)
- eine Elektrowerkstatt (*ELECTRONICS*)
- eine Scherenschleiferei (*GRINDERY*)

Nachdem der Kunde eine Dienstleistung ausgewählt und im Voraus bezahlt hat, wird ein Auftrag (*MissMintOrder*) mit einer Auftragsnummer (*OrderIdentifier*) im System angelegt. Zusätzlich wird ein erwartetes Fertigstellungsdatum (*getExpectedFinished*) berechnet, indem ein Bearbeitungstermin (*TimeTableEntry*) im Zeitplan registriert wird (*createEntry*). Der Auftrag kann sich in verschiedenen Zuständen (*OrderState*) befinden.

Für die Bearbeitung der Aufträge wird eine Raumplanung inklusive Zeitplan zur Verfügung gestellt. Dementsprechend werden Räume (*Room*) passend zur Dienstleistung gebucht. Für die Dienste wird i. A. Material (*Material*) benötigt. Diese können, falls nötig, nachbestellt werden und die Kosten werden automatisch vom System erfasst (*checkAndConsume*).

Ab dem Fertigstellungstermin kann die Ware eine Woche lang unter Vorlage des Auftragszettels abgeholt werden (*FINISHED*). Danach wandert die Ware für drei Monate in eine Aufbewahrungsstelle (*STORED*), wo sie unter Vorlage des Auftragszettels durch Zahlung einer Aufbewahrungsgebühr von 0,50€ pro Woche ausgelöst werden kann. Waren, die auch in diesem Zeitraum nicht abgeholt werden, fallen anschließend an eine karitative Organisation (*CHARITABLE\_USED*). Umgekehrt wird für jeden ganzen verspäteten Tag Kosten in Höhe von 10% (bis maximal 100%) erstattet (*calculateCharge*).

Jegliches Einkommen bzw. Ausgaben sind für den Filialleiter sichtbar (*showFinancePage*). Dabei ist

auch eine Übersicht über den letzten Monat zur Abbuchung der Gewinne an den Mutterkonzern möglich (*showLastMonth*).

## 1.2. Qualitätsziele

### Wartbarkeit

Dieses Merkmal gibt den Grad an Effektivität und Effizienz an, mit dem ein Produkt oder System modifiziert werden kann, um es zu verbessern, zu korrigieren oder an Änderungen in der Umgebung und in den Anforderungen anzupassen.

### Nutzerfreundlichkeit

Grad, in dem ein Produkt oder System von bestimmten Benutzern verwendet werden kann, um bestimmte Ziele mit Effektivität, Effizienz und Zufriedenheit in einem bestimmten Verwendungskontext zu erreichen.

### Sicherheit

Grad des Schutzes von Informationen und Daten durch ein Produkt oder System, sodass Personen oder andere Produkte oder Systeme über den Datenzugriff verfügen, der für ihre Art und ihre Berechtigungsstufe angemessen ist.

### Design

Gestaltung der Benutzeroberfläche, um die visuelle Wahrnehmung des Kunden zu beeinflussen, und so die Informationsvermittlung und Nutzerfreundlichkeit zu steigern. Dies beinhaltet Optimierungen für verschiedene Auflösungen und verschiedene Browser. Wichtig ist auch die visuelle Differenzierung von Inhalten, um die Informationsaufnahme des Kunden zu lenken.

1 = Nicht wichtig .. 5 = Sehr wichtig

Qualitätsanspruch	1	2	3	4	5
Wartbarkeit				x	
Benutzerfreundlichkeit					x
Sicherheit				x	
Design		x			

## 2. Randbedingungen

### 2.1. Hardware-Vorgaben

Eine Liste von Geräten/Hardware, die benötigt werden um die Software zu benutzen:

- Server
- Computer
- Tastatur
- Maus

## 2.2. Software-Vorgaben

Die Anwendung setzt folgende Software voraus:

- Java 11 oder neuer
- Mozilla Firefox 70+
- Chromium 78+

## 2.3. Vorgaben zum Betrieb des Software

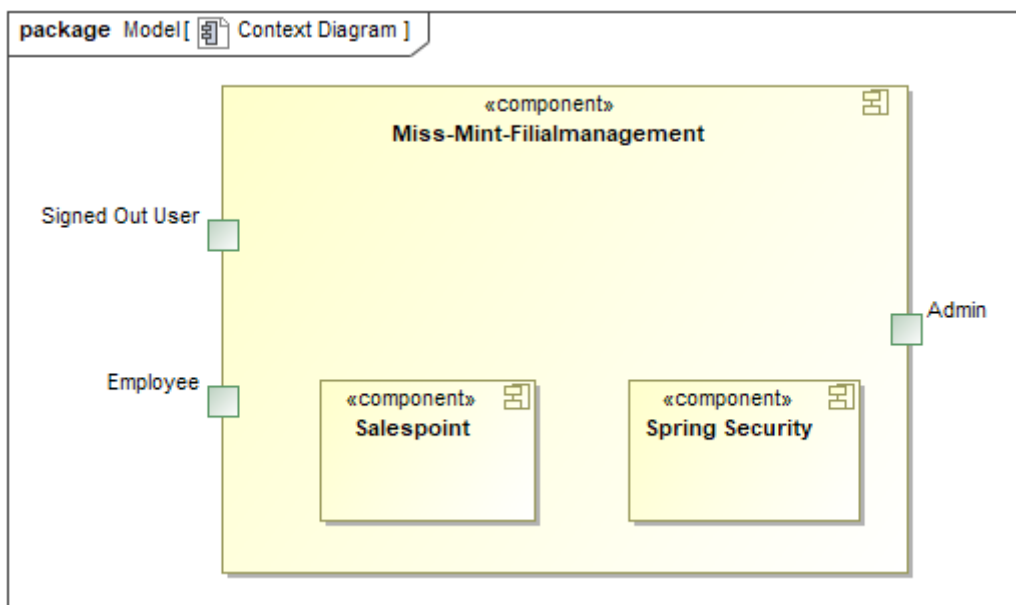
Die Software wird ausschließlich von *Miss Mint Mending Points* Mitarbeiter verwendet. Das System stellt Funktionen für die Annahme, Reparaturplanung, Ausgabe und Lagerung der Ware, sowie die Verwaltung der Ressourcen zur Verfügung. Zudem erhält der Filialleiter Einsicht auf die Finanzen.

Die Software soll rund um die Uhr in Betrieb sein und nur über das interne Firmennetz erreichbar sein.

Die Hauptnutzer der Software sind Mitarbeiter, die typische Website-Navigationsschemata kennen, sowie ein Filialleiter (Admin), der nicht unbedingt über einen technischen Hintergrund verfügt.

Das System muss technisch gewartet werden, da Sicherheitsupdates für Java aber auch die verwendeten Frameworks eingespielt werden müssen. Alle Daten müssen dauerhaft in einer Datenbank gespeichert sein und über die Applikation zugänglich sein (z.B. sollten für einen Filialleiter keine SQL-Kenntnisse erforderlich sein).

## 3. Kontextabgrenzung

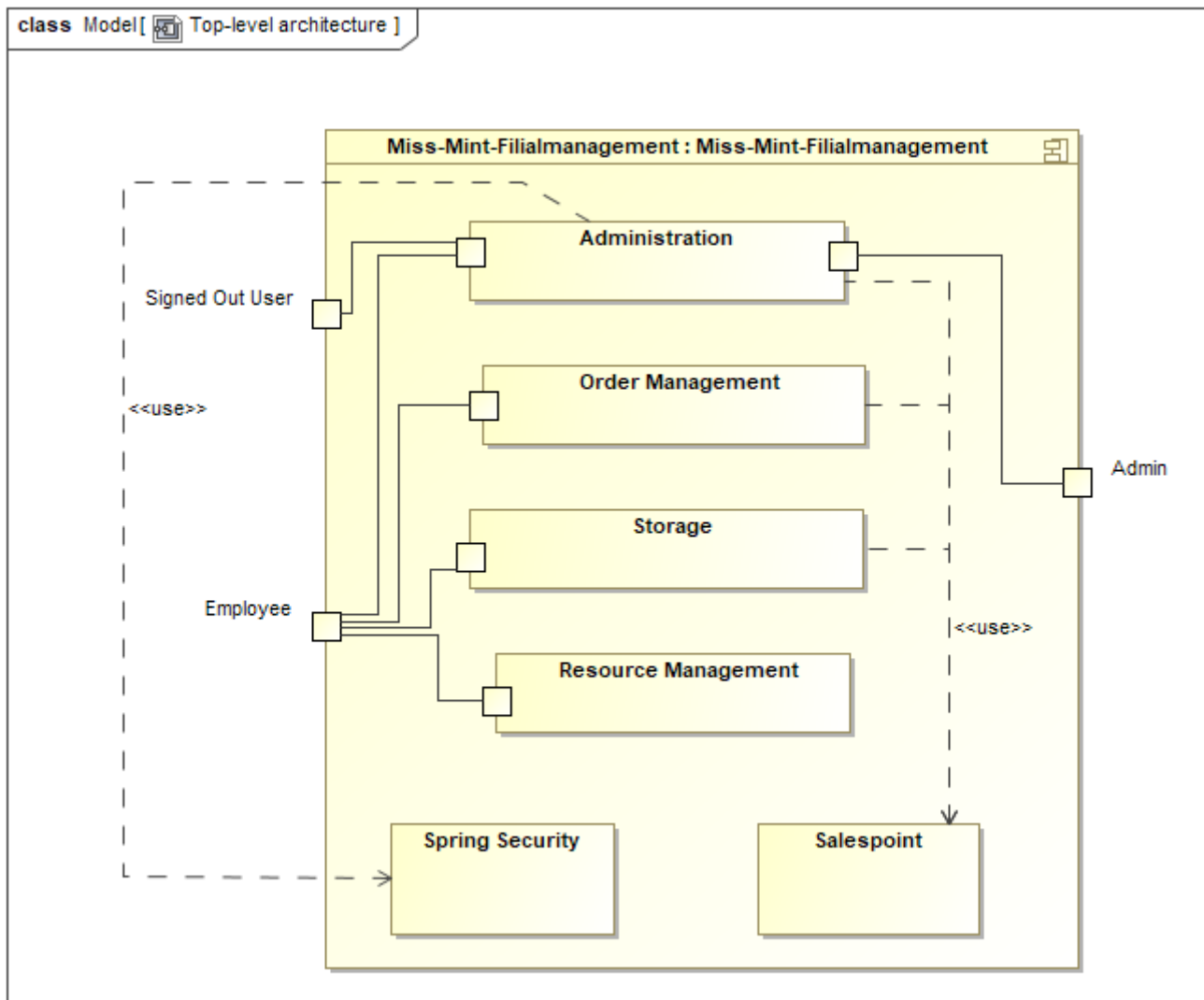


## 4. Lösungsstrategie

## 4.1. Erfüllung der Qualitätsziele

Qualitätsziel	Lösungsansatz
Wartbarkeit	<ul style="list-style-type: none"><li>• <b>Modularität:</b> Zusammensetzen der Anwendung aus möglichst eigenständigen Modulen um den Einfluss den die Änderung eines Moduls auf andere Module hat zu minimieren.</li><li>• <b>Wiederverwendbarkeit:</b> Sicherstellung der Wiederverwendbarkeit von Modulen durch andere Systemkomponenten</li><li>• <b>Modifizierbarkeit:</b> Die Anwendung sollte ohne Verschlechterung der Code-Qualität oder Auftreten von Fehlern veränderbar und/oder erweiterbar sein.</li></ul>
Nutzerfreundlichkeit	<ul style="list-style-type: none"><li>• <b>Erlernbarkeit:</b> Das System sollte verständlich und einfach bedienbar sein. Das lässt sich z.B. durch eindeutige Beschreibung von Eingaben mit Hilfe von Tooltips und/oder Labels realisieren.</li><li>• <b>Handhabung von Fehlern:</b> Nutzer sollten vor Fehlern geschützt werden. Eingaben dürfen unter keinen Umständen zu ungünstigen Systemzuständen führen.</li><li>• <b>Ästhetik der Nutzerschnittstelle:</b> Bereitstellung einer ansprechenden und zufriedenstellenden Interaktion für den Nutzer</li><li>• <b>Zugänglichkeit:</b> Es sollte, z.B. durch die Nutzung passender Schriftgrößen und Kontraste, sichergestellt werden dass Menschen mit einer möglichst großen Bandbreite von Eigenschaften die Möglichkeiten des Systems vollständig nutzen können.</li></ul>
Sicherheit	<ul style="list-style-type: none"><li>• <b>Vertraulichkeit:</b> Daten dürfen nur von Menschen mit den dafür nötigen Zugriffsrechten eingesehen werden. Dies kann mit Spring Security und dem Thymeleaf (<code>sec:authorize</code>-Tag) realisiert werden.</li><li>• <b>Integrität:</b> Nicht-autorisierte Modifikationen sollten verhindert werden. Dafür kann die Spring Security (<code>@PreAuthorize</code> - annotation) verwendet werden.</li><li>• <b>Verantwortung:</b> Nachverfolgbarkeit von Aktionen oder Ereignissen zu einer eindeutigen Entität oder Person.</li></ul>

## 4.2. Softwarearchitektur



## 4.3. Entwurfsentscheidungen

### 4.3.1. Verwendete Muster

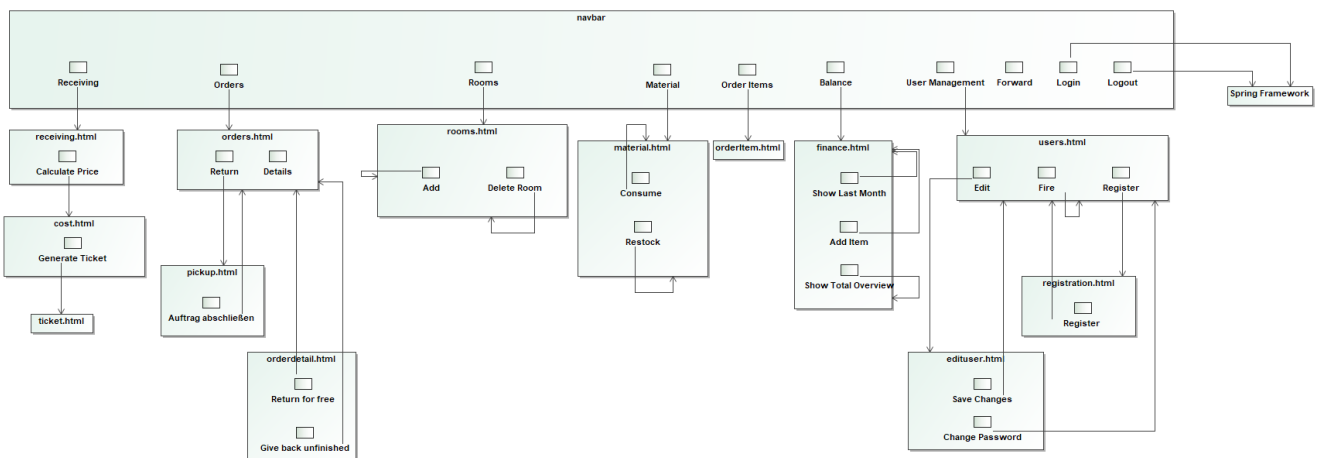
- **Model View Controller** mit Spring MVC
- **Singleton** mit Springs `@Component` etc.
- **Value Object** mit `MonetaryAmount` etc.
- **Data Transfer Object** mit den `Form`-Klassen, in denen Formulardaten aggregiert werden.
- **Dependency Injection** über die Konstruktoren in Spring-Komponenten
- **Repository** mit den Spring-Repositories

### 4.3.2. Persistenz

Die Anwendung verwendet Hibernate-Annotation-basiertes Mapping, um Java-Klassen Datenbanktabellen zuzuordnen. Als Datenbank wird H2 verwendet. Die Persistenz ist standardmäßig deaktiviert. Um den Persistenzspeicher zu aktivieren, müssen die folgenden zwei Zeilen in der Datei `application.properties` nicht kommentiert werden:

```
# spring.datasource.url=jdbc:h2:./db/missmint
# spring.jpa.hibernate.ddl-auto=update
```

### 4.3.3. Benutzeroberfläche



### 4.3.4. Verwendung externer Frameworks

Externe Klasse	Pfad der externen Klasse	Verwendet von (Klasse der eigenen Anwendung)
Accountancy	org.salepointframework.accountancy.Accountancy	FinanceController, FinanceService
Accountancy Entry	org.salepointframework.accountancy.AccountancyEntry	FinanceService
Assert	org.springframework.util.Assert	CatalogDataInitializer, FinanceController, FinanceService, InventoryInitializer, MissMintOrder, MissMintService, OrderOverviewController, OrderService, PickupController, PickupService, ReceivingController, ReceivingService, ServiceDataInitializer, ServiceManager, Staff, StaffController, StaffInitializer, StaffManagement, TimeController, TimeService, TimeTableService, Utils
BusinessTime	org.salepointframework.time.BusinessTime	FinanceService, IndexController, OrderService, ReceivingController, RoomService, SalaryService, TimeController, TimeService, TimeTableService

<b>Externe Klasse</b>	<b>Pfad der externen Klasse</b>	<b>Verwendet von (Klasse der eigenen Anwendung)</b>
Catalog	org.salespointframework.catalog.Catalog	CatalogDataInitializer, InventoryInitializer, MaterialManager, OrderItemController, OrderItemManager, OrderOverviewController, OrderService, ReceivingService, ServiceDataInitializer, ServiceManager
Component	org.springframework.stereotype.Component	CatalogDataInitializer, FinanceDataInitializer, InventoryInitializer, Messages, RoomDataInitializer, ServiceDataInitializer, StaffInitializer
Configuration	org.springframework.context.annotation.Configuration	Application
Controller	org.springframework.stereotype.Controller	FinanceController, IndexController, InventoryController, OrderItemController, OrderOverviewController, PickupController, ReceivingController, RoomController, StaffController, TimeController
CrudRepository	org.springframework.data.repository.CrudRepository	EntryRepository, RoomRepository, StaffRepository
DataInitializer	org.salespointframework.core.DataInitializer	CatalogDataInitializer, FinanceDataInitializer, InventoryInitializer, RoomDataInitializer, ServiceDataInitializer, StaffInitializer
Database	org.hibernate.dialect.Database	Room
EURO	org.salespointframework.core.Currencies.EURO	CatalogDataInitializer
EnableSalespoint	org.salespointframework.EnableSalespoint	Application
Errors	org.springframework.validation.Errors	FinanceController, ReceivingController, RoomController, StaffController
GetMapping	org.springframework.web.bind.annotation.GetMapping	FinanceController, IndexController, InventoryController, OrderOverviewController, PickupController, ReceivingController, StaffController
HttpSecurity	org.springframework.security.config.annotation.web.builders.HttpSecurity	Application

<b>Externe Klasse</b>	<b>Pfad der externen Klasse</b>	<b>Verwendet von (Klasse der eigenen Anwendung)</b>
HttpStatus	org.springframework.http.HttpStatus	PickUpController, Utils
Interval	org.salespointframework.time.Interval	FinanceService
InventoryItemIdentifier	org.salespointframework.inventory.InventoryItemIdentifier	InventoryController, MaterialForm, MaterialManager
LocaleContextHolder	org.springframework.context.i18n.LocaleContextHolder	Messages
LoggedIn	org.salespointframework.useraccount.web.LoggedIn	ReceivingController
MessageSource	org.springframework.context.MessageSource	Messages
MessageSourceAccessor	org.springframework.context.support.MessageSourceAccessor	Messages
Metric	org.salespointframework.quantity.Metric	CatalogDataInitializer, InventoryInitializer, Material, MaterialManager, ServiceConsumptionManager
Model	org.springframework.ui.Model	FinanceController, IndexController, InventoryController, OrderItemController, OrderOverviewController, PickUpController, ReceivingController, RoomController, StaffController
ModelAttribute	org.springframework.web.bind.annotation.ModelAttribute	InventoryController, ReceivingController, StaffController
Money	org.javamoney.moneta.Money	CatalogDataInitializer, FinanceDataInitializer, FinanceService, OrderItem, OrderService, SalaryService, ServiceDataInitializer
Order	org.springframework.core.annotation.Order	FinanceDataInitializer, ServiceDataInitializer
Order	org.salespointframework.order.Order	MissMintOrder
OrderManager	org.salespointframework.order.OrderManager	OrderItemController, OrderItemManager, OrderOverviewController, OrderService, PickUpService, ReceivingController, ReceivingService, TimeTableService
Page	org.springframework.data.domain.Page	OrderItemController, OrderItemManager, OrderOverviewController



<b>Externe Klasse</b>	<b>Pfad der externen Klasse</b>	<b>Verwendet von (Klasse der eigenen Anwendung)</b>
Pageable	org.springframework.data.domain. Pageable	OrderItemController, OrderItemManager, OrderOverviewController, OrderService, TimeTableService
Pair	org.springframework.data.util.Pair	CatalogDataInitializer, ServiceConsumptionManager, ServiceDataInitializer, TimeTableService
Password	org.salespointframework.useraccount. Password	StaffInitializer, StaffManagement
PathVariable	org.springframework.web.bind. annotation.PathVariable	OrderOverviewController, PickUpController, StaffController
PostMapping	org.springframework.web.bind. annotation.PostMapping	FinanceController, InventoryController, PickUpController, ReceivingController, StaffController, TimeController
PreAuthorize	org.springframework.security.access. prepost.PreAuthorize	FinanceController, InventoryController, OrderItemController, OrderOverviewController, PickUpController, ReceivingController, RoomController, StaffController, TimeController
Product	org.salespointframework.catalog.Product	Material, MissMintService, OrderItem
ProductIdentifier	org.salespointframework.catalog. ProductIdentifier	OrderItemController, OrderItemForm, OrderItemManager, OrderOverviewController, ReceivingForm, ServiceManager
PropertySource	org.springframework.context.annotation. PropertySource	Application
Quantity	org.salespointframework.quantity. Quantity	InventoryInitializer, MaterialManager, MissMintOrder, ReceivingService, ServiceConsumptionManager
RequestMapping	org.springframework.web.bind. annotation.RequestMapping	IndexController
ResponseStatusException	org.springframework.web.server. ResponseStatusException	PickUpController, Utils
Role	org.salespointframework.useraccount. Role	StaffInitializer, StaffManagement
SalespointSecurityConfiguration	org.salespointframework. SalespointSecurityConfiguration	Application

<b>Externe Klasse</b>	<b>Pfad der externen Klasse</b>	<b>Verwendet von (Klasse der eigenen Anwendung)</b>
Service	org.springframework.stereotype.Service	FinanceService, MaterialManager, OrderItemManager, OrderService, PickupService, ReceivingService, RoomService, SalaryService, ServiceManager, StaffManagement, TimeService, TimeTableService
Session	org.hibernate.Session	Utils
SessionAttribute	org.springframework.web.bind.annotation.SessionAttribute	ReceivingController
SpringApplication	org.springframework.boot.SpringApplication	Application
StreamUtils	org.springframework.data.util.StreamUtils	TimeTableService
Streamable	org.springframework.data.util.Streamable	EntryRepository, FinanceService, MaterialManager, ServiceManager, StaffRepository
Transactional	org.springframework.transaction.annotation.Transactional	OrderItemManager, StaffManagement
UniqueInventory	org.salespointframework.inventory.UniqueInventory	InventoryController, InventoryInitializer, MaterialManager, ReceivingService
UniqueInventoryItem	org.salespointframework.inventory.UniqueInventoryItem	InventoryController, InventoryInitializer, MaterialManager, ReceivingService
UserAccount	org.salespointframework.useraccount.UserAccount	MissMintOrder, ReceivingController, Staff, StaffManagement, StaffRepository
UserAccountManager	org.salespointframework.useraccount.UserAccountManager	StaffInitializer, StaffManagement
Value	org.springframework.beans.factory.annotation.Value	FinanceDataInitializer, FinanceService, OrderService, SalaryService, ServiceDataInitializer

## 5. Bausteinsicht

### 5.1. Aufträge

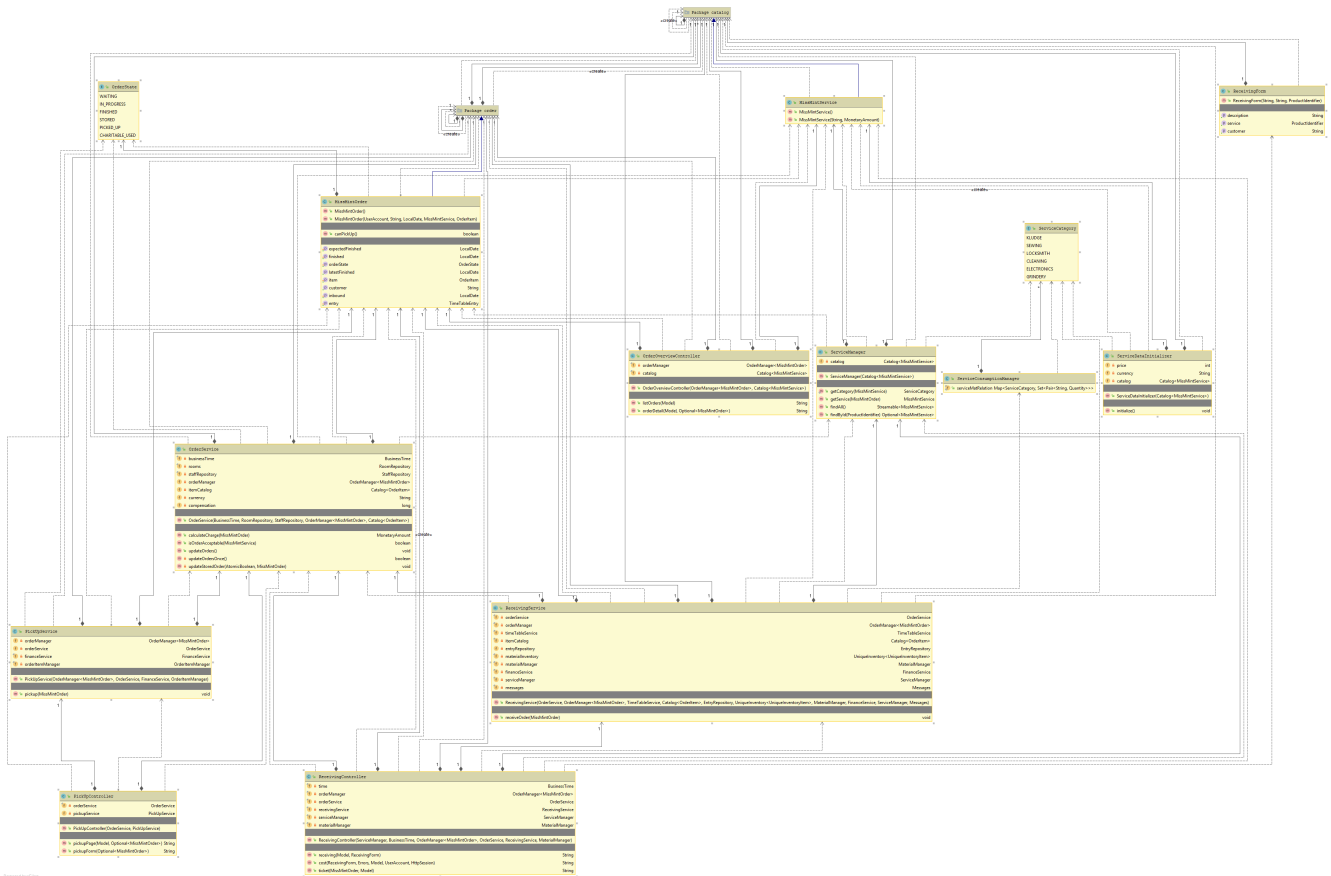


Figure 1. Klassendiagramm für das orders Paket

Klasse/Enumeration	Beschreibung
OrderOverviewController	Ein Spring MVC Controller, der Anfragen für die Anzeige der aktuellen Aufträge <b>MissMintOrder</b> beantwortet.
PickUpController	Ein Spring MVC Controller, der Anfragen für die Rückgabe von Gegenständen an den Kunden handhabt. Dazu berechnet die Klasse auch anfallende Kosten.
ReceivingController	Ein Spring MVC Controller, der Anfragen für die Aufnahme von Aufträgen <b>MissMintOrder</b> in das System beantwortet. Dazu berechnet die Klasse auch anfallende Kosten.
ReceivingForm	Eine Klasse, um die Mitarbeitereingaben für die Annahme zu validieren.
MissMintOrder	Diese Klasse ist eine Erweiterung der Salespoint <b>Order</b> . Sie enthält Kunden-, Dienstleistungs- und Zeit-Informationen. Zudem hat der Auftrag einen Zustand.
OrderService	Ein Dienst, der Hilfsfunktionen für die Auftragsverwaltung bereitstellt.
PickUpService	Hilfsdienst für den Abholprozess, um den Controller von der Business-Logik zu trennen.
ReceivingService	Unterstützt den ReceivingController in der Handhabung von neuen Aufträgen.
OrderState	Eine Enumeration für die Zustände der Aufträge.
MissMintService	Erweiterung des Salespoint <b>Products</b> , um es von anderen Produkten in der späteren Software abzugrenzen.





Klasse/Enumeration/Schnittstelle	Beschreibung
CatalogDataInitializier	Implementation des <b>DataInitializers</b> für den Produktkatalog
InventoryController	Ein Spring MVC Controller, der Anfragen bezüglich des Materialinventars annimmt.
InventoryInitializer	Implementation des <b>DataInitializers</b> für die Inventare der jeweiligen Produkte.
orderItem, Material	Erweiterung des Salespoint <b>Products</b> .
MaterialForm	Form für das manuelle Nachbestellen/Verbrauchen von Material
MaterialManager	Businesslogik für Nachbestellung/Verbrauch
OrderItemManager	Businesslogik für Entfernen und anzeigen von OrderItems

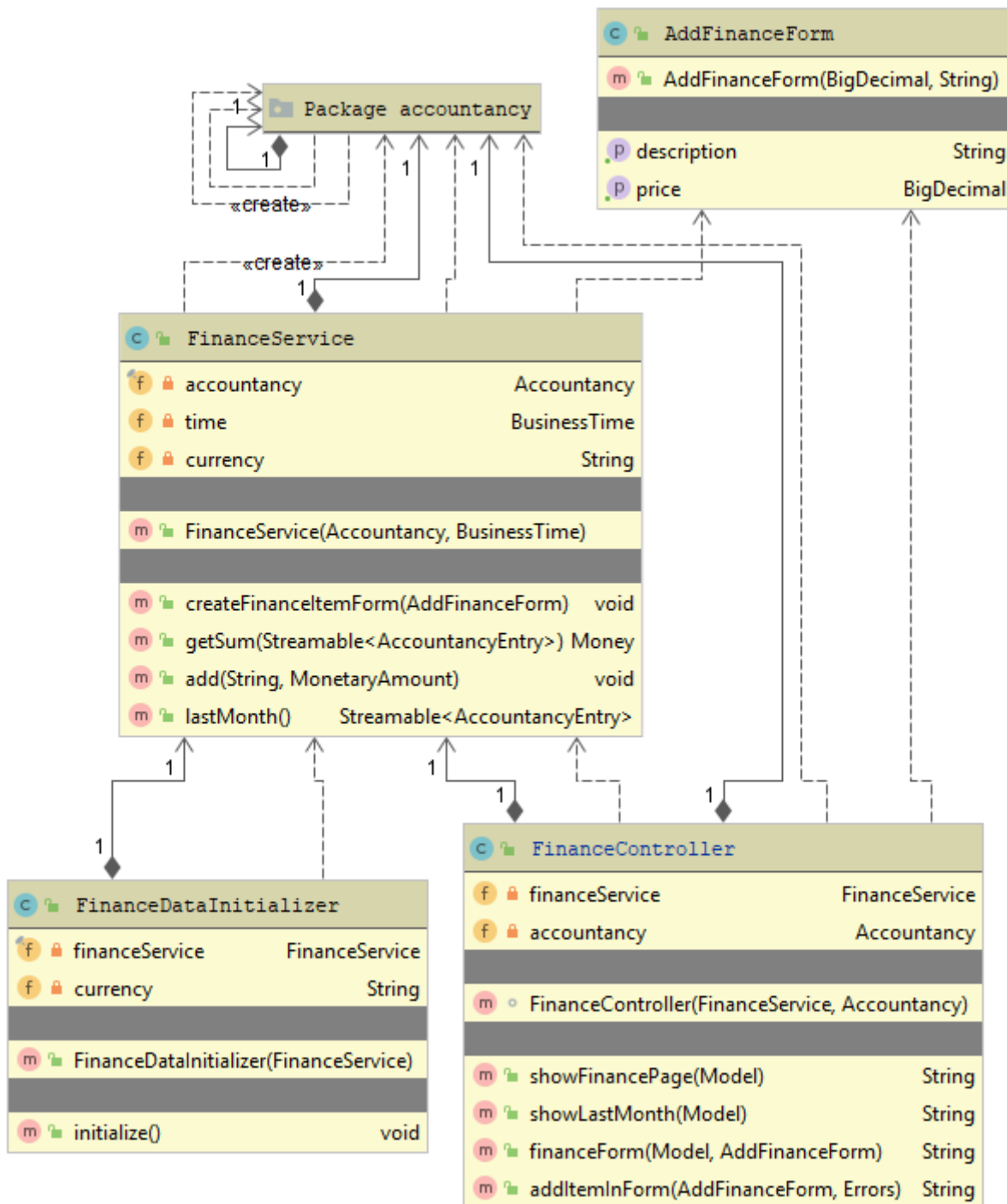
## 5.4. Räume



Klasse/Enumeration/Schnittstelle	Beschreibung
Room	Klasse die die Räume repräsentiert. Räume stehen in enger Verbindung mit den Timeslots, wobei jeder Raum seine eigenen TimeSlots besitzt.
RoomController	Ein Spring MVC Controller, der die Verwaltung von Räumen und das buchen von TimeSlots ermöglicht.
AddRoomForm	Diese Klasse prüft die Nutzereingabe auf fehler und erstellt ggf. einen neuen Raum.
RoomsRepository	Interface welches das Crud Repository von Spring erbt. Es speichert Raumeinträge.
RoomService	Dieser Service unterstützt die Aktualisierung von Räumen, sodass neue Räume hinzugefügt und alte gelöscht werden können.

## 5.5. Finanzen



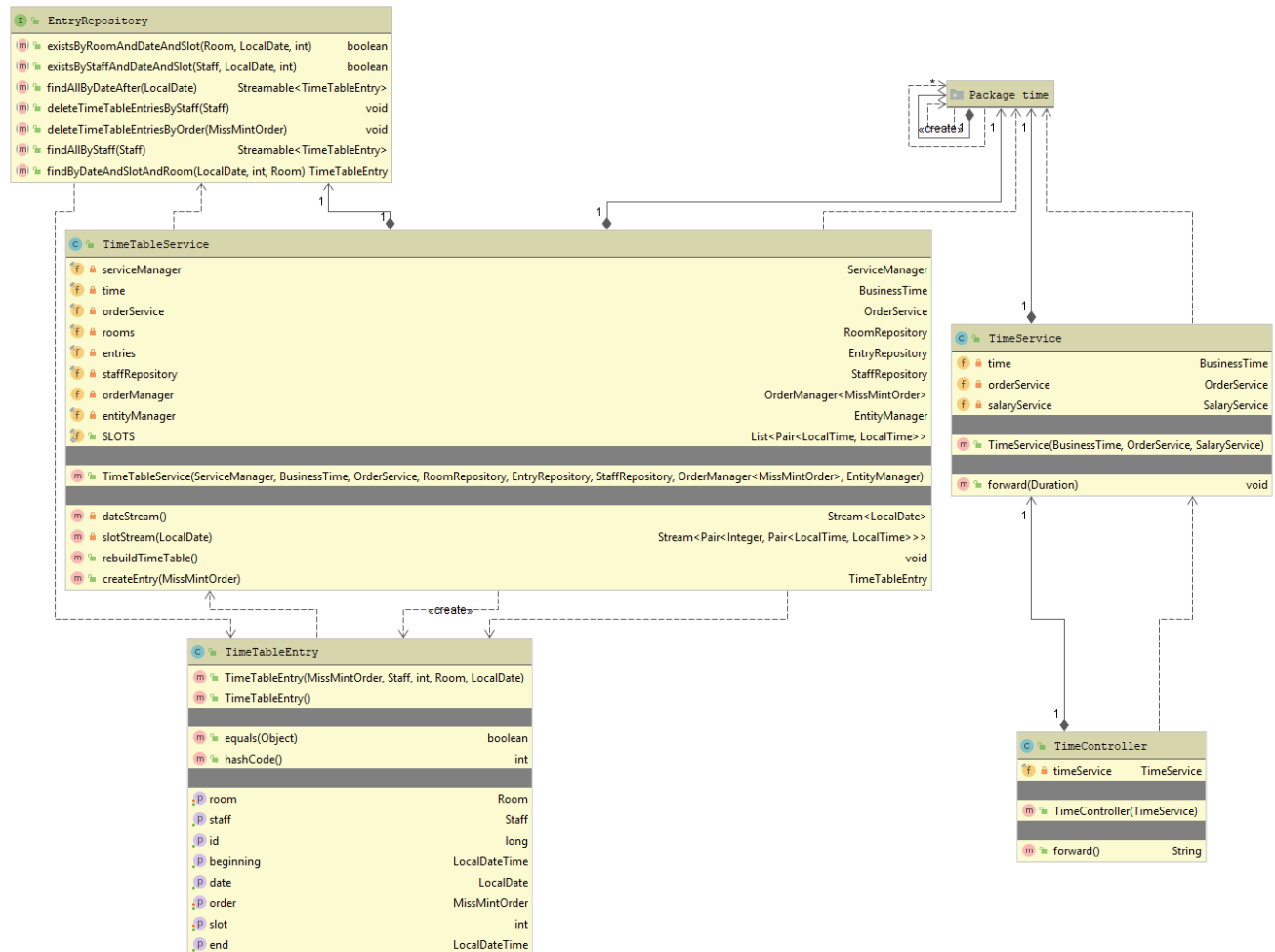


Powered by yFiles

Klasse/Enumeration/Schnittstelle	Beschreibung
FinanceController	Ein Spring Controller, der Anfragen bzgl. Financeübersicht für letztes Monat und für die ganze Zeit und zudem Erstellung von Finanzen Einträge verarbeitet.
FinaceService	Ein Dienst, der zuständig ist, die Accountancy im System zu verwalten. Man kann dort AccountancyEntry hinzufügen, die gesamte Summe von Finanzen Aufträge berechnen und letztes Monat anzeigen lassen.

Klasse/Enumeration/Schnittstelle	Beschreibung
FinanceDataInitializer	Ein Formular von <i>SalesPoint</i> für die Initialisierung von AccountancyEntry.
AddFinanceForm	Ein Formular um AccountancyEnty mit Kost und Beschreibung hinzufügen.

## 5.6. Time



Powered by yFiles

Klasse/Enumeration/Schnittstelle	Beschreibung
TimeTableEntry	Diese Klasse repräsentiert Zeitslots, die durch Buchung von Räumen in einer bestimmten Zeit erstellt werden.
EntryRepository	Interface welches das <i>CrudRepository</i> von Spring implementiert. Es speichert Objekte vom Typ TimeTableEntry.
TimeController	Schnittstelle, die den Nutzer ermöglicht, die Zeit vorzuspulen.
TimeService	Bildet das Backend für den TimeController.

Klasse/Enumeration/Schnittstelle	Beschreibung
TimeTableService	Dieser Service erledigt das Erstellen von Zeit-Slots und das Umsortieren dieser, um rechtzeitige Fertigstellung zu ermöglichen.

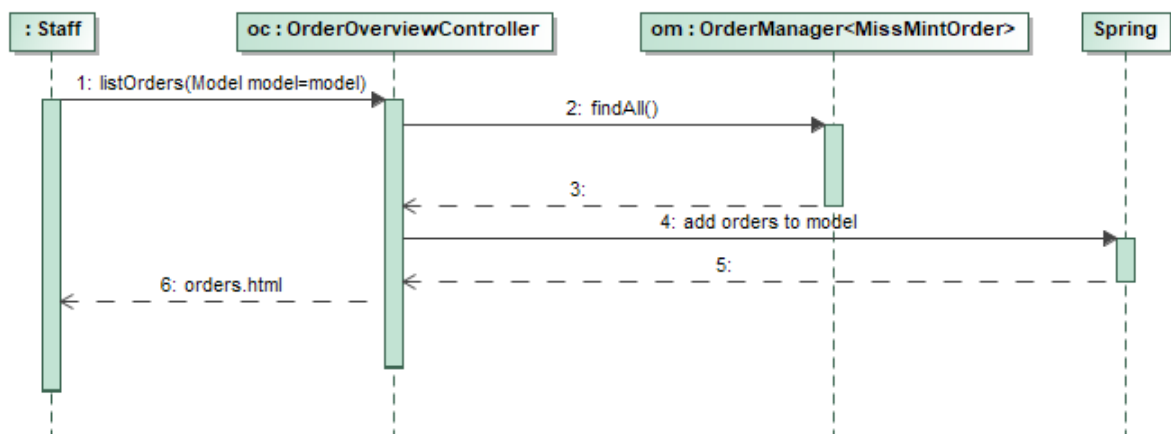
## 5.7. Rückverfolgbarkeit zwischen Analyse- und Entwurfsmodell

Klasse/Enumeration (Analysemodell)	Klasse/Enumeration (Entwurfsmodell)
Order	MissMintOrder
Service	MissMintService
State	OrderState
Order Management	OrderService
Service Category	ServiceCategory
OrderTicket	-
User	User
Staff	Staff
-	UserManagement
Storage	Inventory
Material	Inventory
ResourceManagement	-
Resource	orderItem, Material
Room	Room
-	RoomService
FinancialManagement	FinanceService
TimeTableEntry	TimeTableEntry
TimeTable	EntryRepository

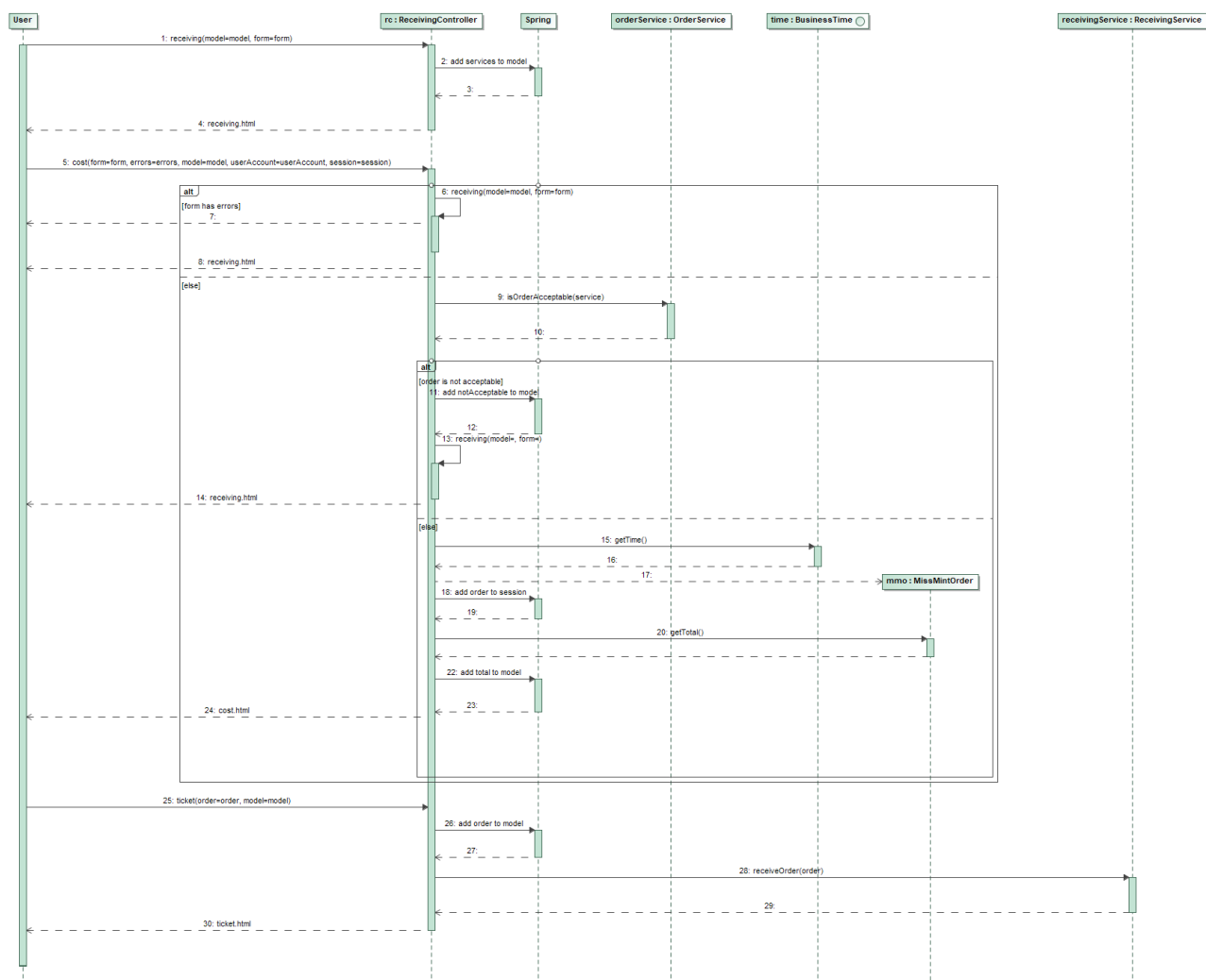
## 6. Laufzeitsicht

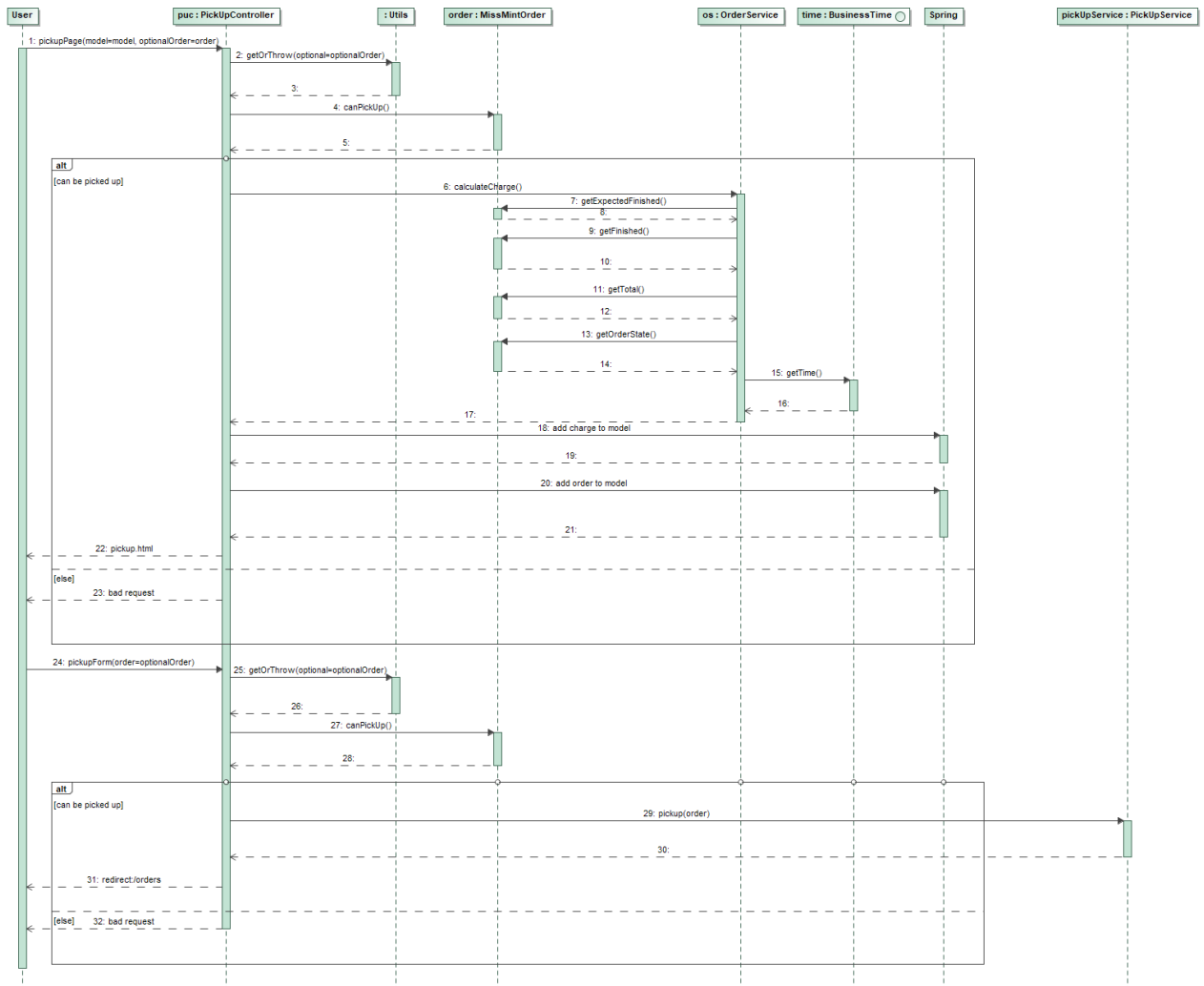
### 6.1. Aufträge

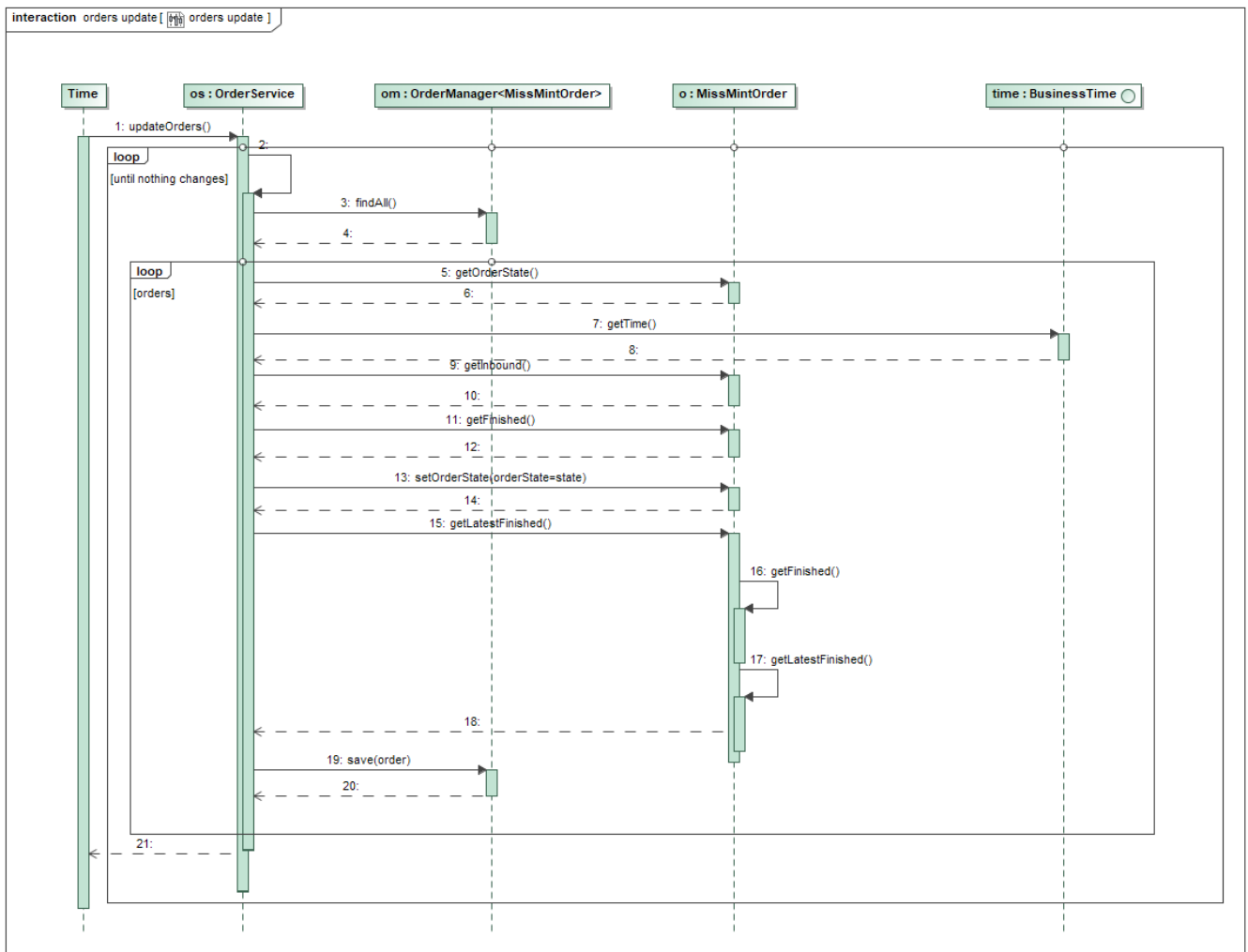
# interaction orders list [ 0-10 orders list ]



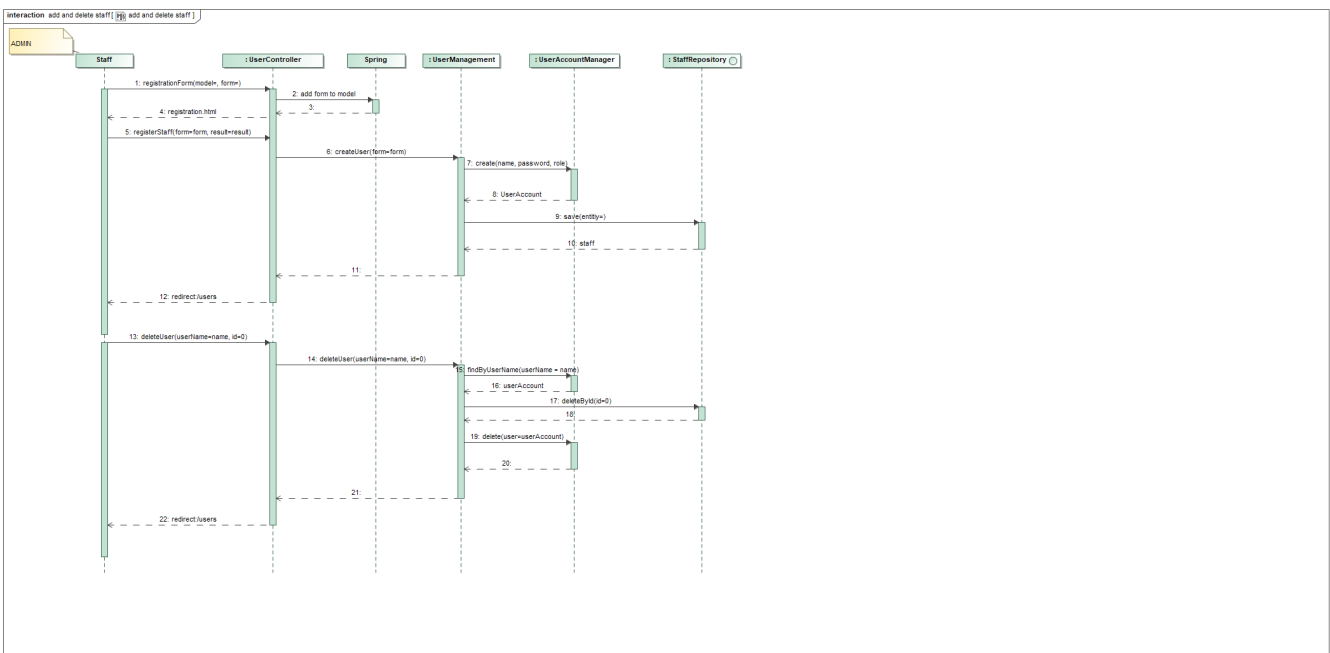
# interaction orders receiving [ 0-10 orders receiving ]



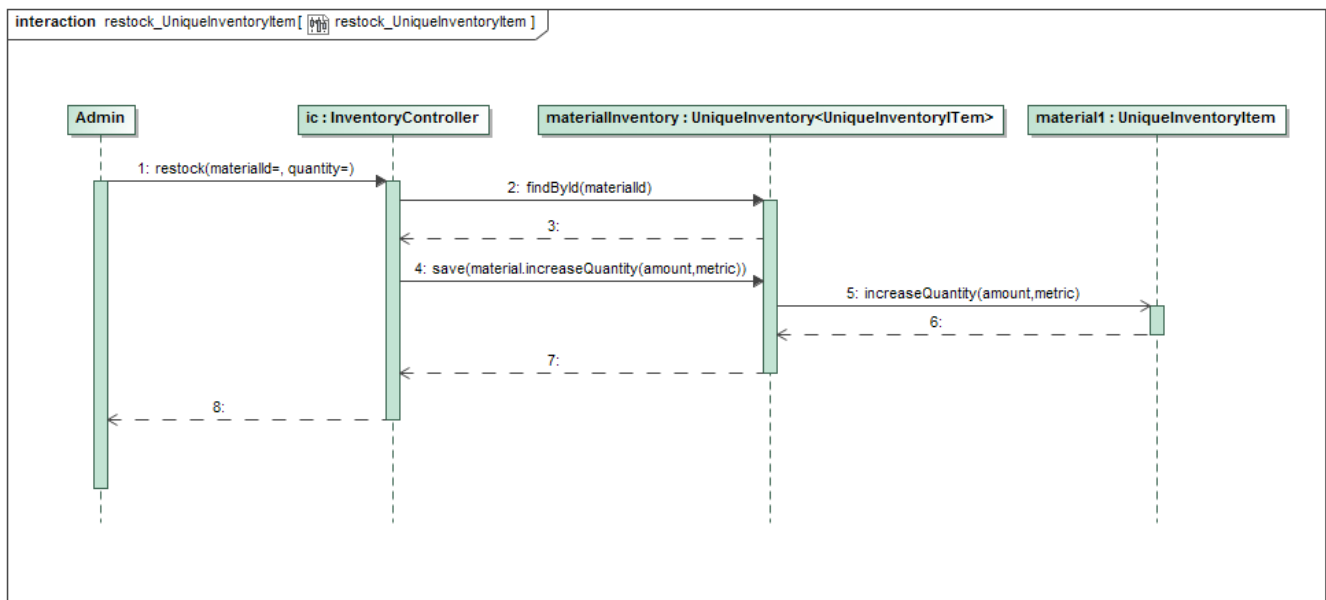
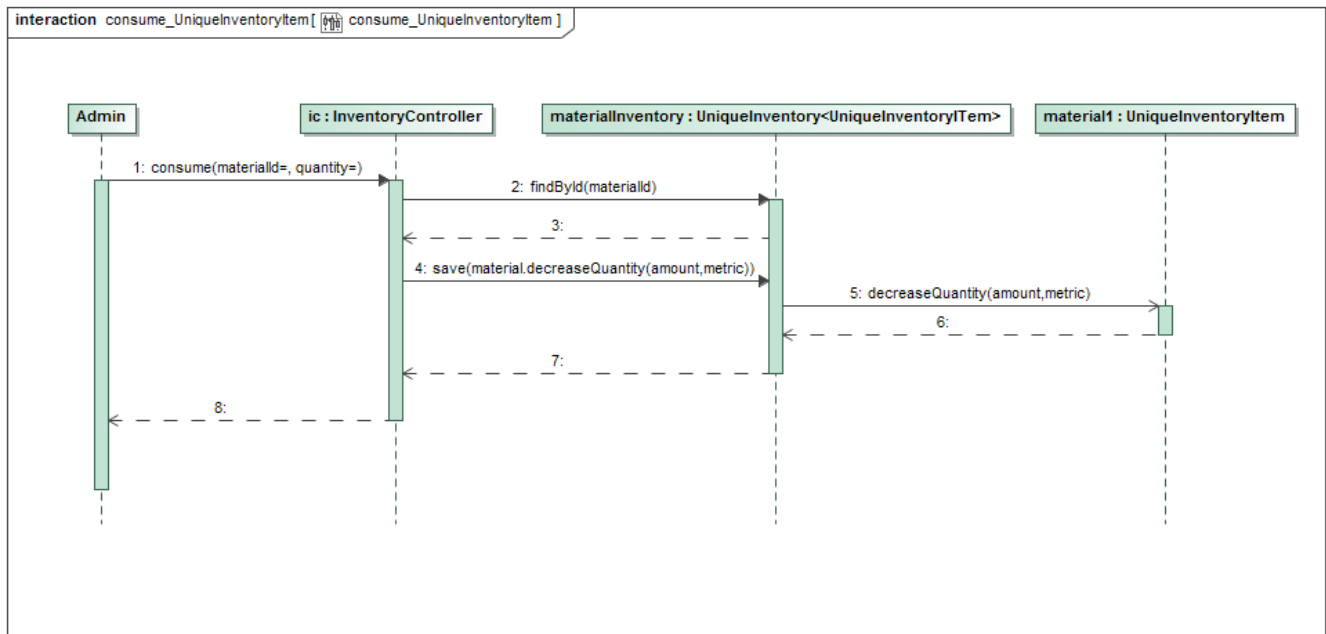





## 6.2. Mitarbeiter

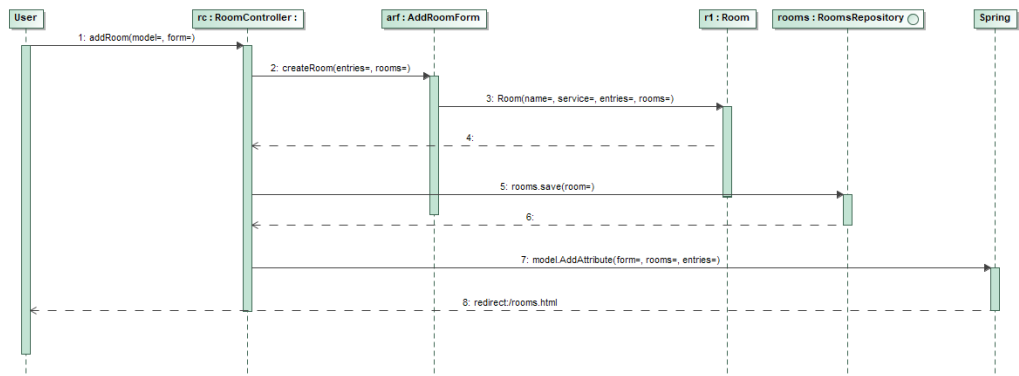



## 6.3. Inventar



## 6.4. Räume

Interaction rooms addRoom [  rooms addRoom ]



Interaction rooms deleteRoom [  rooms deleteRoom ]

