System-Spezifikation

Hochschule Luzern
Technik & Architektur

Applikationsentwicklung – FS14

Gruppe 20 | Horw, 30.05.2014
Bontekoe Christian | Estermann Michael | Rohrer Felix

Autoren

Bontekoe Christian	
Studiengang	Informatik - Software Systems (Berufsbegleitend)
Adresse	
Telefon	
E-Mail	

Estermann Michael	
Studiengang	Informatik - Software Systems (Berufsbegleitend)
Adresse	
Telefon	
E-Mail	

Rohrer Felix	
Studiengang	Informatik - Software Systems (Berufsbegleitend)
Adresse	
Telefon	
E-Mail	

Änderungskontrolle

Version	Datum	Autor	Beschreibung
1.0	28.03.2014	Felix Rohrer	Vorlage erstellen, erste Texte schreiben
1.1	03.04.2014	Felix Rohrer	Dokument ergänzt für Review 1
1.2	24.04.2014	Felix Rohrer	Dokument ergänzt für Review 2
1.3	08.05.2014	Felix Rohrer	Dokument ergänzt für Review 3
1.4	22.05.2014	Felix Rohrer	Dokument ergänzt für Review 4
1.5	29.05.2014	Felix Rohrer	Diverse Ergänzungen
1.6	29.05.2014	Christian Bontekoe	Diverse Ergänzungen
1.7	30.05.2014	Felix Rohrer	Finalisieren für Abgabe

Inhalt

1 Systemübersicht 1

1.1 Use-Cases 2
1.2 Use-Case – Aktivitäten 4

2 Architektur und Designentscheide 13

2.1 Architekturübersicht 13
2.2 Technologieübersicht 14
2.3 Daten (Mengengerüst & Strukturen) 15

3 Schnittstellen 16

3.1 Externe Schnittstellen 16
3.2 Interne Schnittstellen in FBSData 17
3.3 Interne Schnittstellen in FBSService 18
3.4 Interne Schnittstellen in FBSGuiClient 20
3.5 Benutzerschnittstelle(n) 21

4 GUI 22

4.1 GUI - Mockups 22
4.2 GUI – Release 1.0.0 24

5 Environment-Anforderungen 26

5.1 Hardware 26
5.2 Software 26
5.3 Datenbank 26

6 Starten des Bestellsystems 27

6.1 FBSData, Backend / MySQL 27
6.2 FBSService 27
6.3 FBSGuiClient 27

7 ScrumDo 28

Abbildungsverzeichnis 30

Anhang 31

1 Systemübersicht

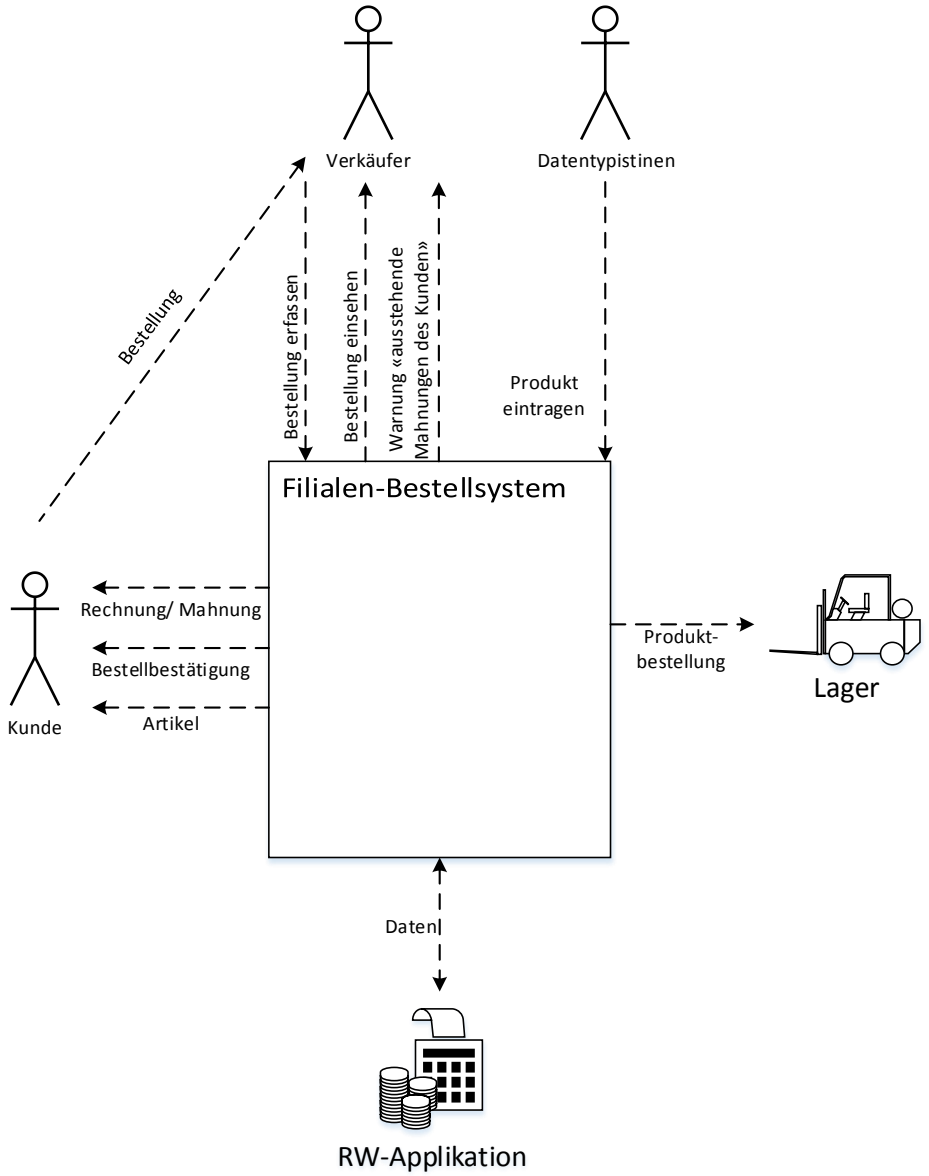


Abb. 1: Funktionale Sicht – Bestellsytem

Das Filialen-Bestellsystem hat verschiedene Schnittstellen zu den Anwendern, wie auch zu dem Zentrallager und der RW-Applikation. Für die RW-Applikation werden in der Datenbank die Felder bereitgestellt. Die RW-App ist nur virtuell und wird nicht implementiert.

Im Zentrallager können die Produkte bestellt werden, sollte im Filiallager die Mindestmenge unterschritten werden, wird automatisch eine Nachbestellung im Zentrallager ausgelöst. DatentypistInnen tragen ankommende Produkte im Filial-Bestellsystem ein.

Bestellungen vom Kunden werden durch die Verkäufer im System erfasst und verwaltet. Der Kunde kann auf verschiedene Arten (Telefon, Fax, Mail, etc.) bestellen.

1.1 Use-Cases

1.1.1 Accessmanagement

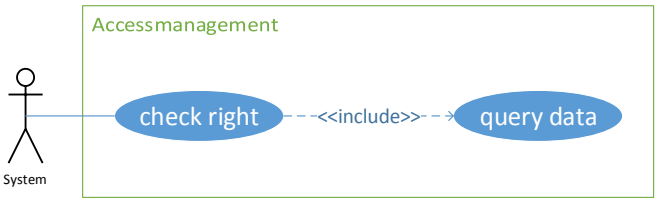


Abb. 2: Use-Case: Accessmanagement

Das System unterscheidet mehrere Benutzer-Rollen. Entsprechend können die Berechtigungen eines Users abgefragt werden.

1.1.2 Location

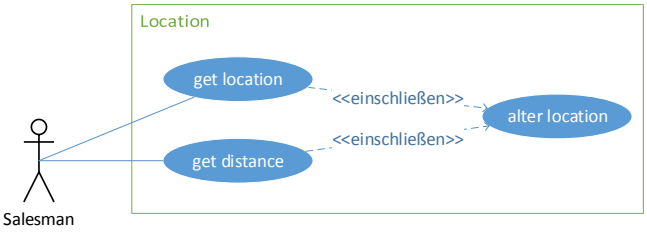


Abb. 3: Use-Case: Location

Für die Kundenadressen sowie Bestellungen werden die Kontakt, resp. Lieferadressen erfasst. Jede Ortschaft wird unabhängig und zentral verwaltet.

1.1.3 Inventory

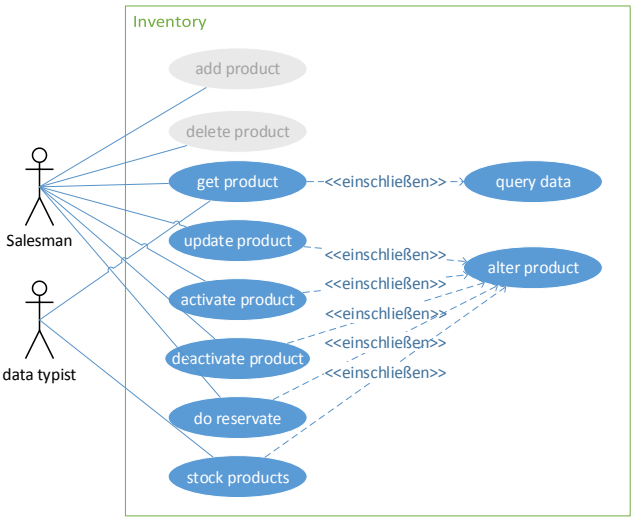


Abb. 4: Use-Case: Inventory

Die DatentypistInnen verwalten die Produkte welche vom Zentrallager angeliefert werden und tragen diese im Filialen-Bestellsystem ein. Die Verkäufer benutzen das Inventory um die einzelnen Produkte mit ihren Details, sowie deren Lagerstand, abzufragen. Jede Filiale bestimmt unabhängig von anderen Filialen, welche Produkte aus dem Gesamt-Sortiment in ihrer Filiale verkaufen wollen.

1.1.4 Order

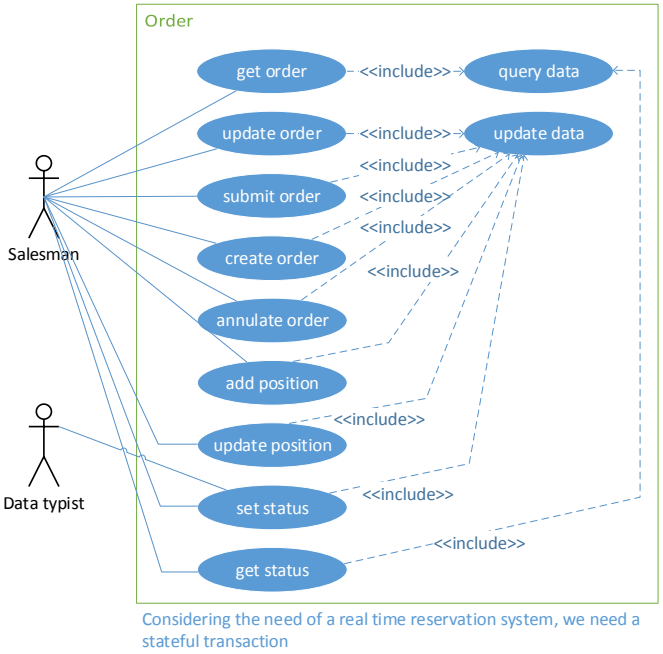


Abb. 5: Use-Case: Order

Die Applikation ist „stateful“ dadurch kann sichergestellt werden, dass es keine Überschneidungen gibt. Der Verkäufer erfasst die Bestellungen. Während des Erfassens können diese Bestellungen mutiert oder gelöscht (abgerochen) werden. Bestehende Bestellungen können betrachtet werden. Durch das Erfassen der Lieferungen aus dem Zentrallager durch die DatentypistInnen werden bei ausstehenden Produkte auch diese Bestellungen angepasst, resp. ausgeliefert sofern alles verfügbar ist.

1.2 Use-Case – Aktivitäten

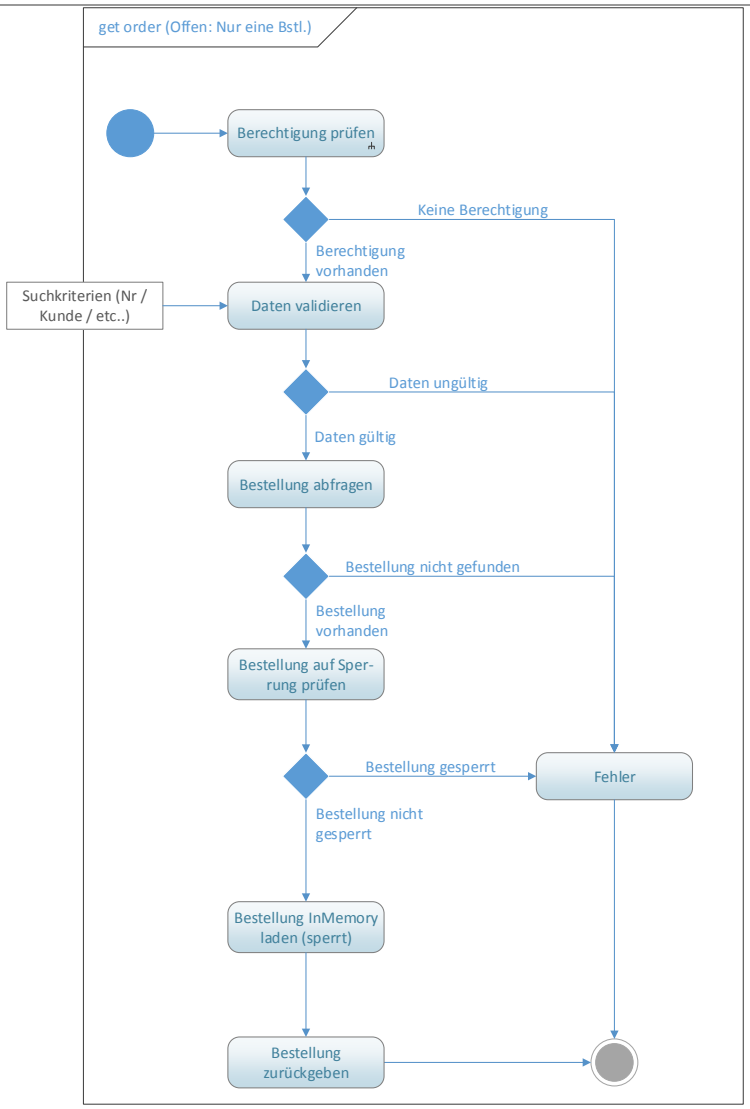
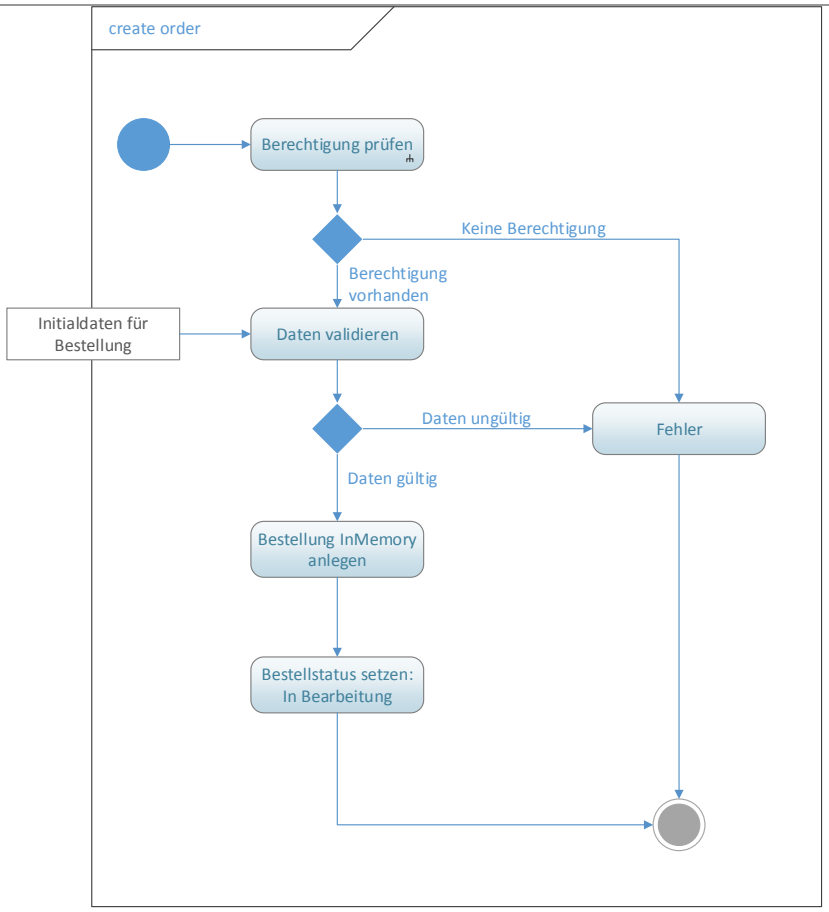
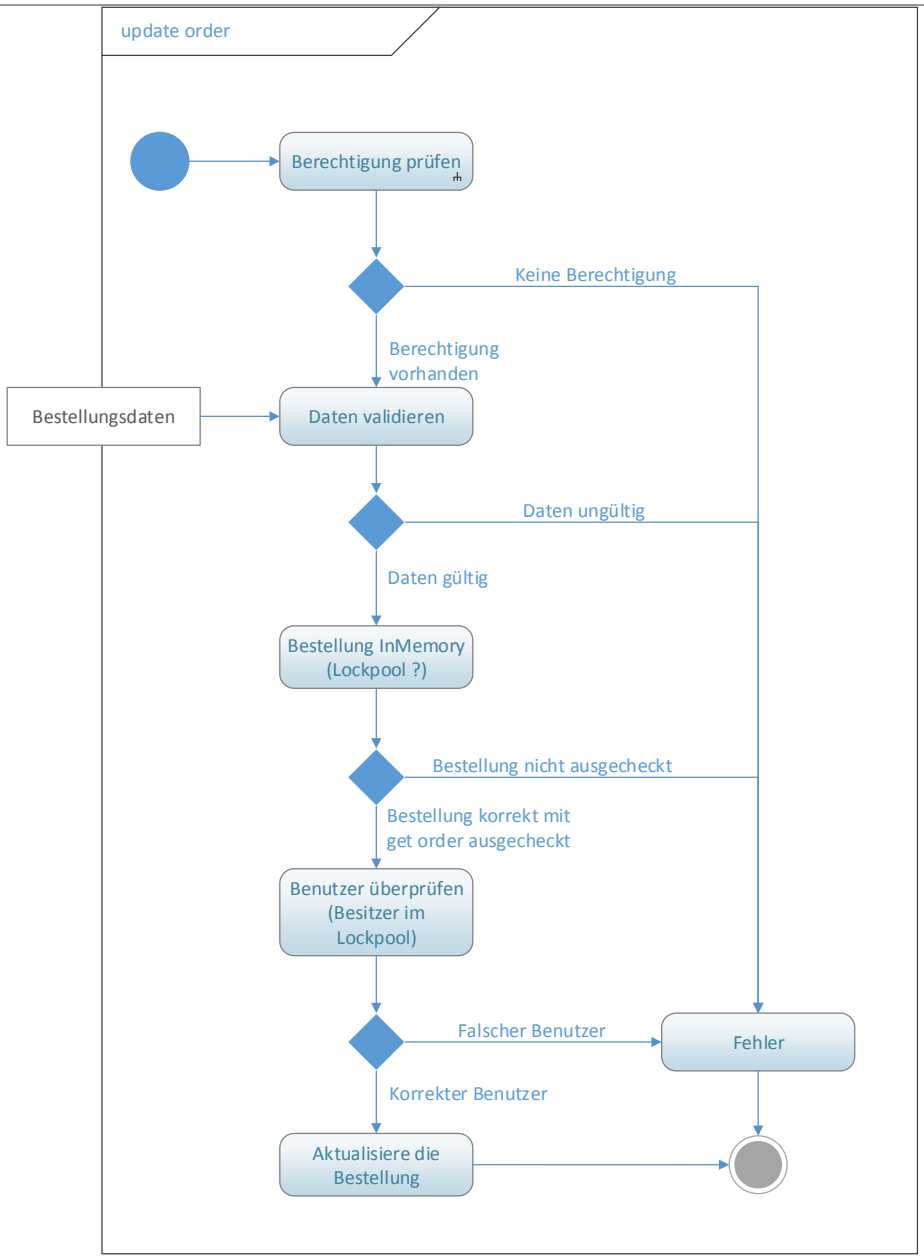
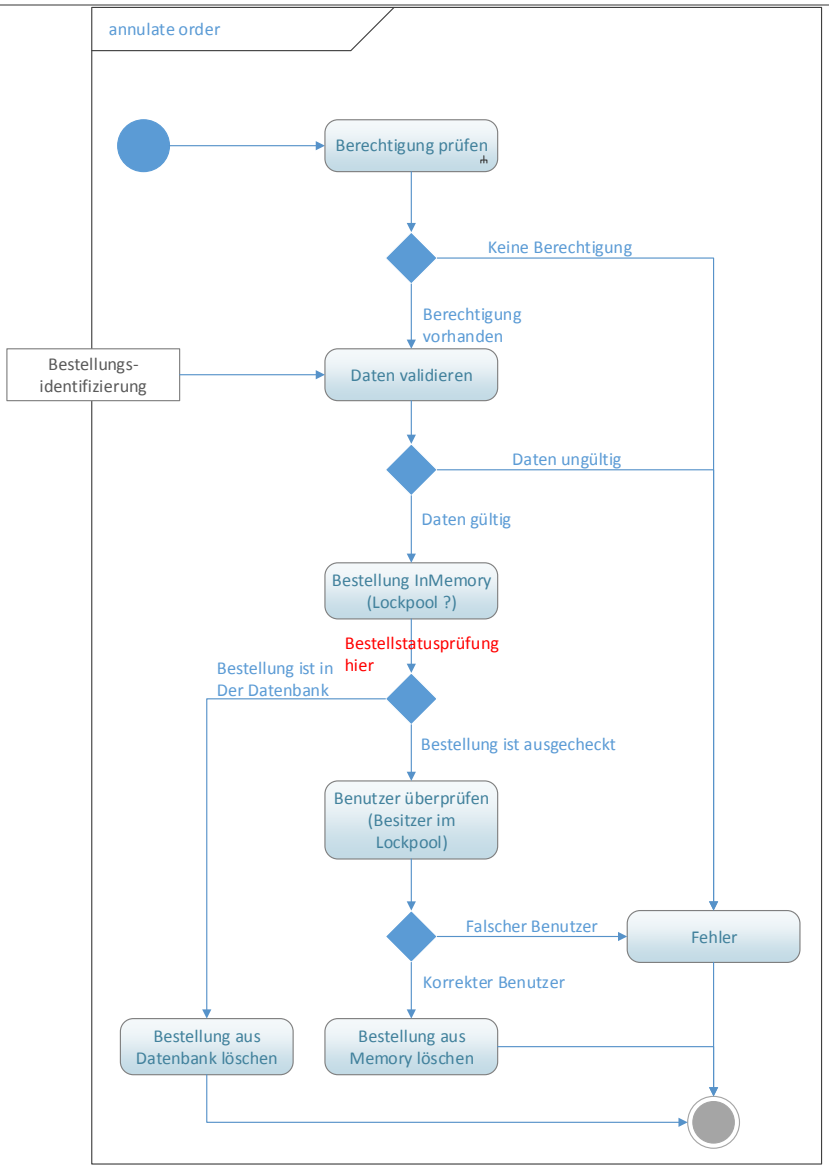
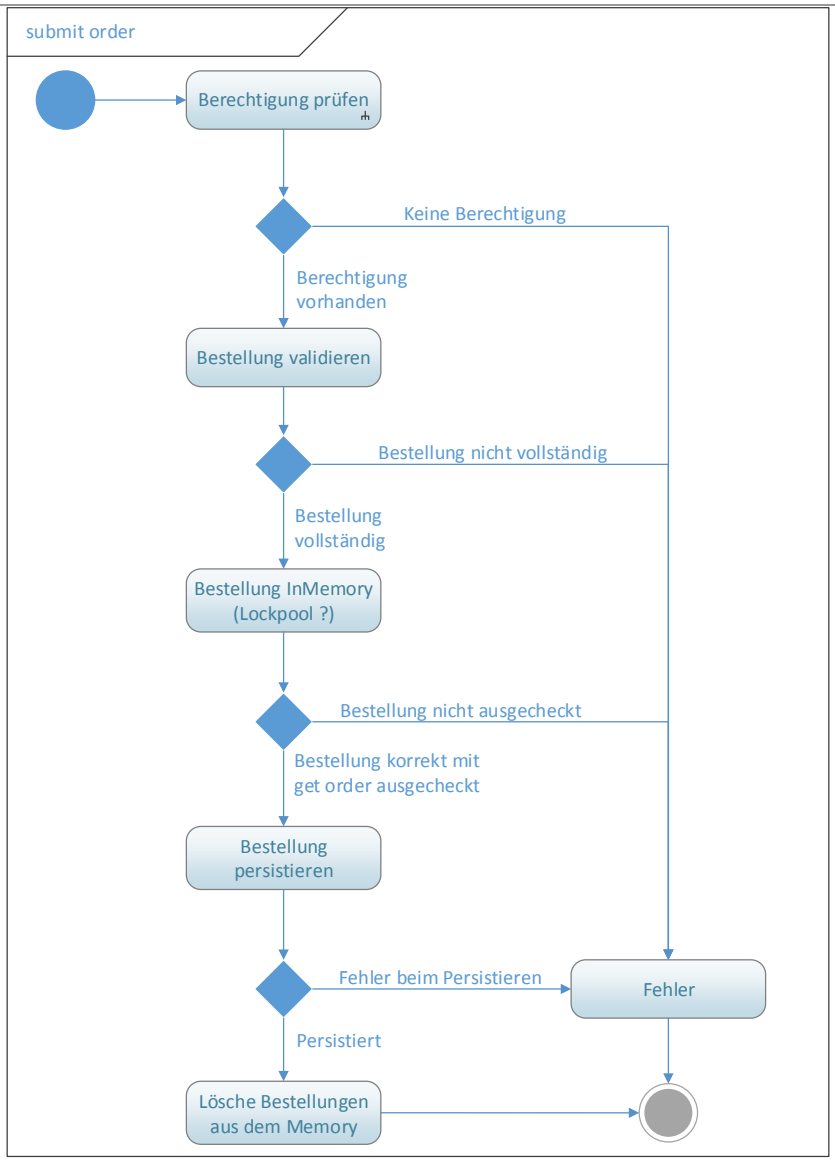
#1. get order	
Beschreibung	Eine Bestellung anzeigen
Akteure	Verkäufer
Auslöser	Der Verkäufer wählt die anzuzeigende Bestellung aus.
Vorbedingung	Es muss mindestens eine Bestellung vorhanden sein.
Ergebnis	Die Daten der Bestellung werden zurückgegeben.
Ergebnis im Fehlerfall	Fehlermeldung, es werden keine Daten zurückgegeben.
Ablauf	1. Berechtigung prüfen 2. Eingabe validieren 3. Bestellung abfragen 4. Lock der Bestellung überprüfen 5. Bestellung laden und zurückgeben
Diagramm	

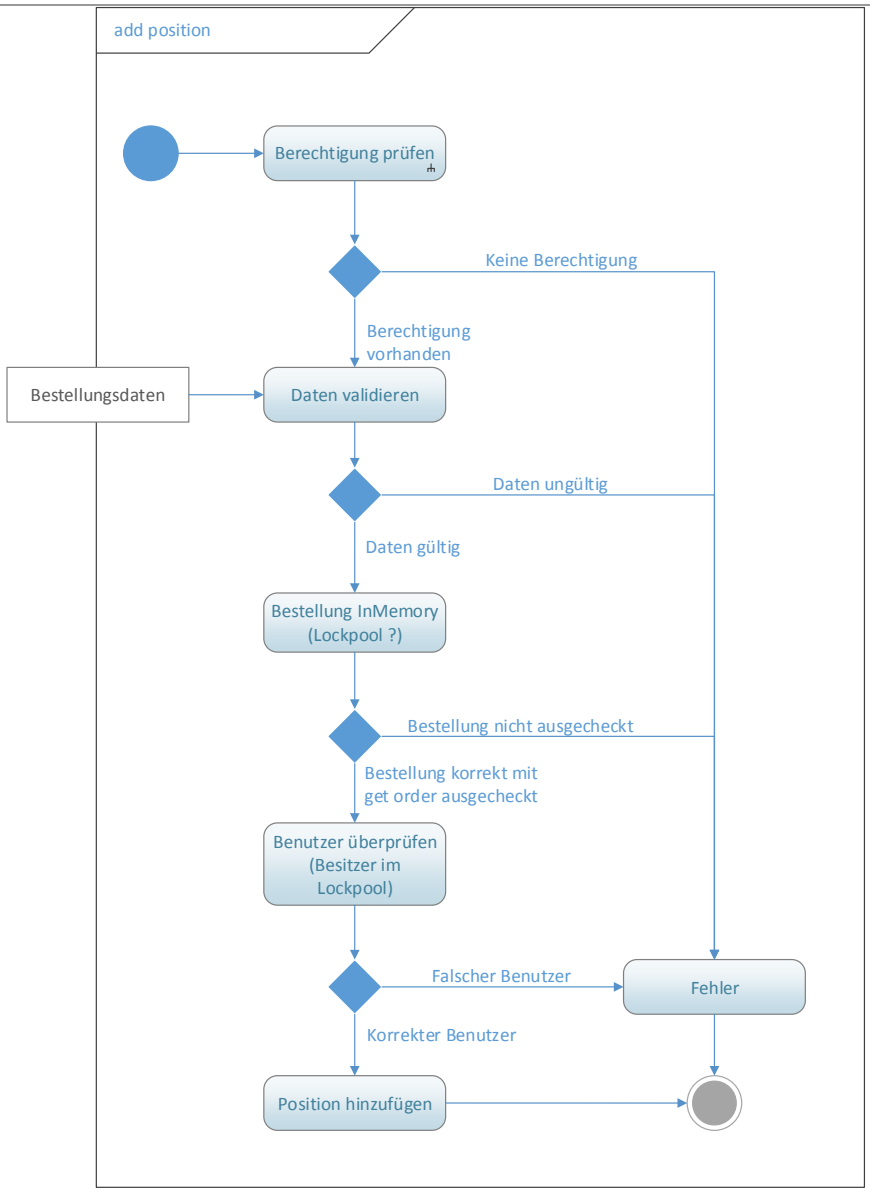
Abb. 6: Aktivität: get order

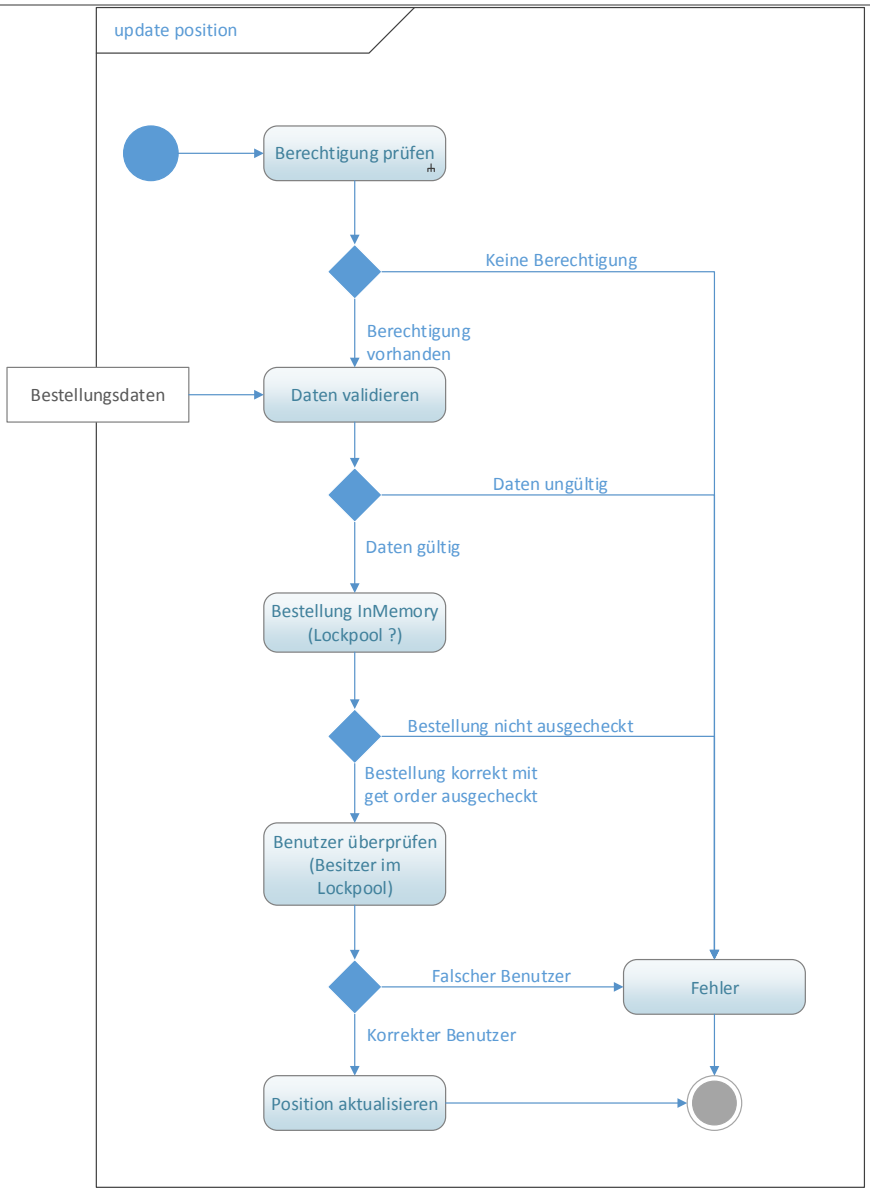
#2. create order	
Beschreibung	Neue Bestellung erfassen
Akteure	Verkäufer
Auslöser	Der Verkäufer erstellt eine neue Bestellung
Vorbedingung	Mindestens ein Kunde muss vorhanden sein.
Ergebnis	Es wird eine neue Bestellung erstellt.
Ergebnis im Fehlerfall	Fehlermeldung, es wird keine neue Bestellung erstellt.
Ablauf	1. Berechtigung prüfen 2. Eingabe validieren 3. Neue Bestellung erstellen
Diagramm	 Abb. 7: Aktivität: create order

#3. update order	
Beschreibung	Bestellung aktualisieren
Akteure	Verkäufer
Auslöser	Der Verkäufer aktualisiert eine neue Bestellung
Vorbedingung	Eine offene Bestellung muss vorhanden sein.
Ergebnis	Die Bestellung wird aktualisiert.
Ergebnis im Fehlerfall	Fehlermeldung, Bestellung wird aktualisiert.
Ablauf	1. Berechtigung prüfen 2. Eingabe validieren 3. Bestellung aktualisieren
Diagramm	 Abb. 8: Aktivität: update order

#4. annulate order	
Beschreibung	Bestellung abbrechen
Akteure	Verkäufer
Auslöser	Der Verkäufer bricht die Bestellung ab.
Vorbedingung	Eine offene Bestellung muss vorhanden sein.
Ergebnis	Die Bestellung wird gelöscht.
Ergebnis im Fehlerfall	Fehlermeldung, Bestellung wird nicht gelöscht.
Ablauf	1. Berechtigung prüfen 2. Eingabe validieren 3. Bestellung überprüfen 4. Bestellung löschen
Diagramm	 Abb. 9: Aktivität: annulate order

#5. submit order	
Beschreibung	Bestellung speichern
Akteure	Verkäufer
Auslöser	Der Verkäufer stellt die Bestellung fertig.
Vorbedingung	Eine offene Bestellung muss vorhanden sein.
Ergebnis	Die Bestellung wird gespeichert.
Ergebnis im Fehlerfall	Fehlermeldung, Bestellung wird nicht gespeichert.
Ablauf	1. Berechtigung prüfen 2. Bestellung validieren 3. Bestellung InMemory überprüfen 4. Bestellung speichern
Diagramm	 Abb. 10: Aktivität: submit order

#6. add position	
Beschreibung	Produkt in Bestellung einfügen
Akteure	Verkäufer
Auslöser	Der Verkäufer fügt ein Produkt zur Bestellung dazu.
Vorbedingung	Eine offene Bestellung muss vorhanden sein.
Ergebnis	Die Bestellung wird aktualisiert.
Ergebnis im Fehlerfall	Fehlermeldung, Bestellung wird nicht aktualisiert.
Ablauf	1. Berechtigung prüfen 2. Eingabe validieren 3. Bestellung überprüfen 4. Bestellung aktualisieren
Diagramm	 Abb. 11: Aktivität: add position

#7. update position	
Beschreibung	Produkt in Bestellung aktualisieren / löschen
Akteure	Verkäufer
Auslöser	Der Verkäufer aktualisiert/löscht ein Produkt in der Bestellung.
Vorbedingung	Eine offene Bestellung mit diesem Produkt muss vorhanden sein.
Ergebnis	Die Bestellung wird aktualisiert.
Ergebnis im Fehlerfall	Fehlermeldung, Bestellung wird nicht aktualisiert.
Ablauf	1. Berechtigung prüfen 2. Eingabe validieren 3. Bestellung überprüfen 4. Bestellung aktualisieren
Diagramm	 Abb. 12: Aktivität: update position

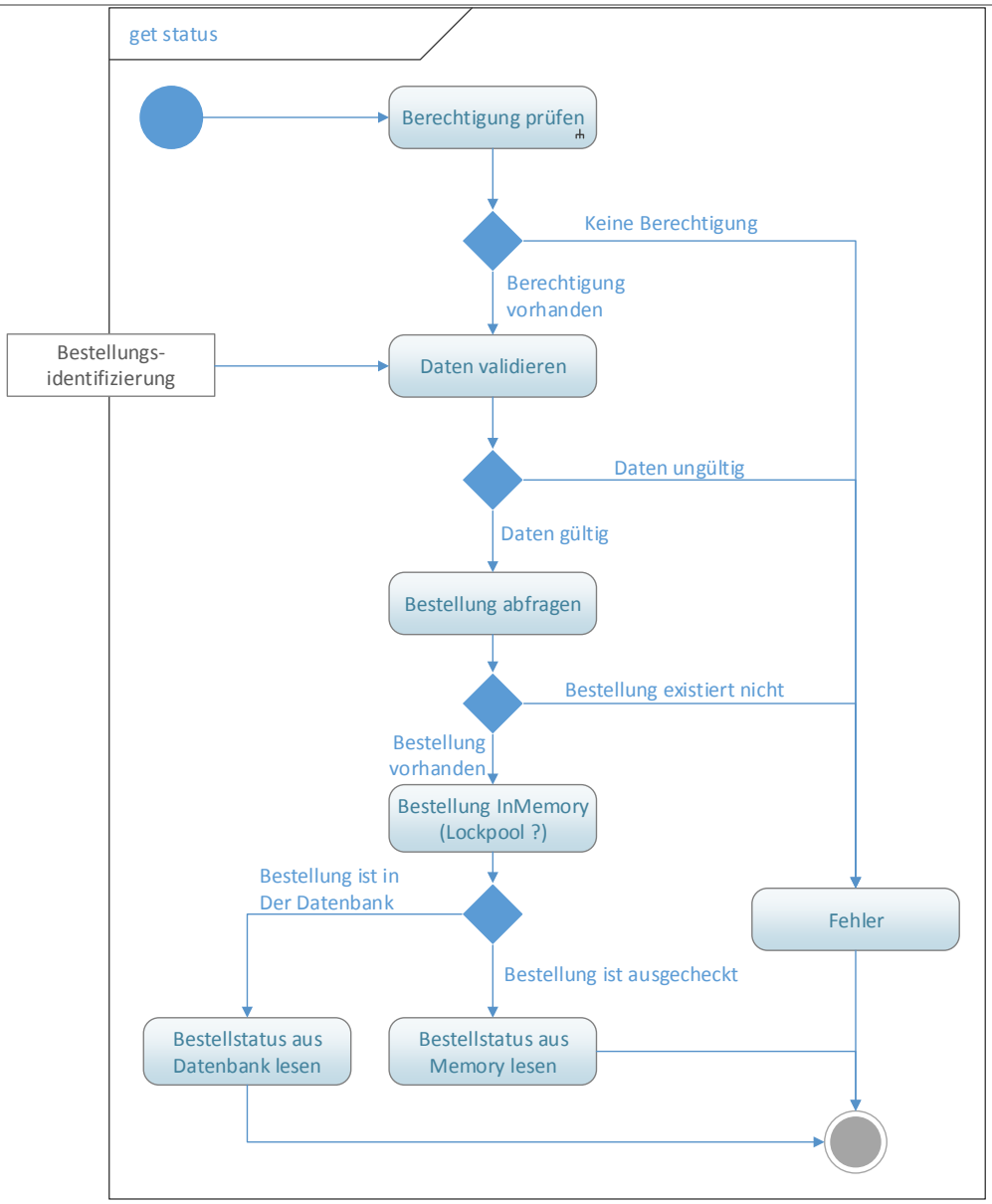
#8. get status	
Beschreibung	Status einer Bestellung abfragen.
Akteure	Verkäufer
Auslöser	Der Verkäufer fragt den Status einer Bestellung ab.
Vorbedingung	Eine Bestellung muss vorhanden sein.
Ergebnis	Der Status der Bestellung wird zurückgegeben.
Ergebnis im Fehlerfall	Fehlermeldung, Status der Bestellung wird nicht zurückgegeben.
Ablauf	1. Berechtigung prüfen 2. Eingabe validieren 3. Bestellung abfragen 4. Status zurück geben
Diagramm	

Abb. 13: Aktivität: get status

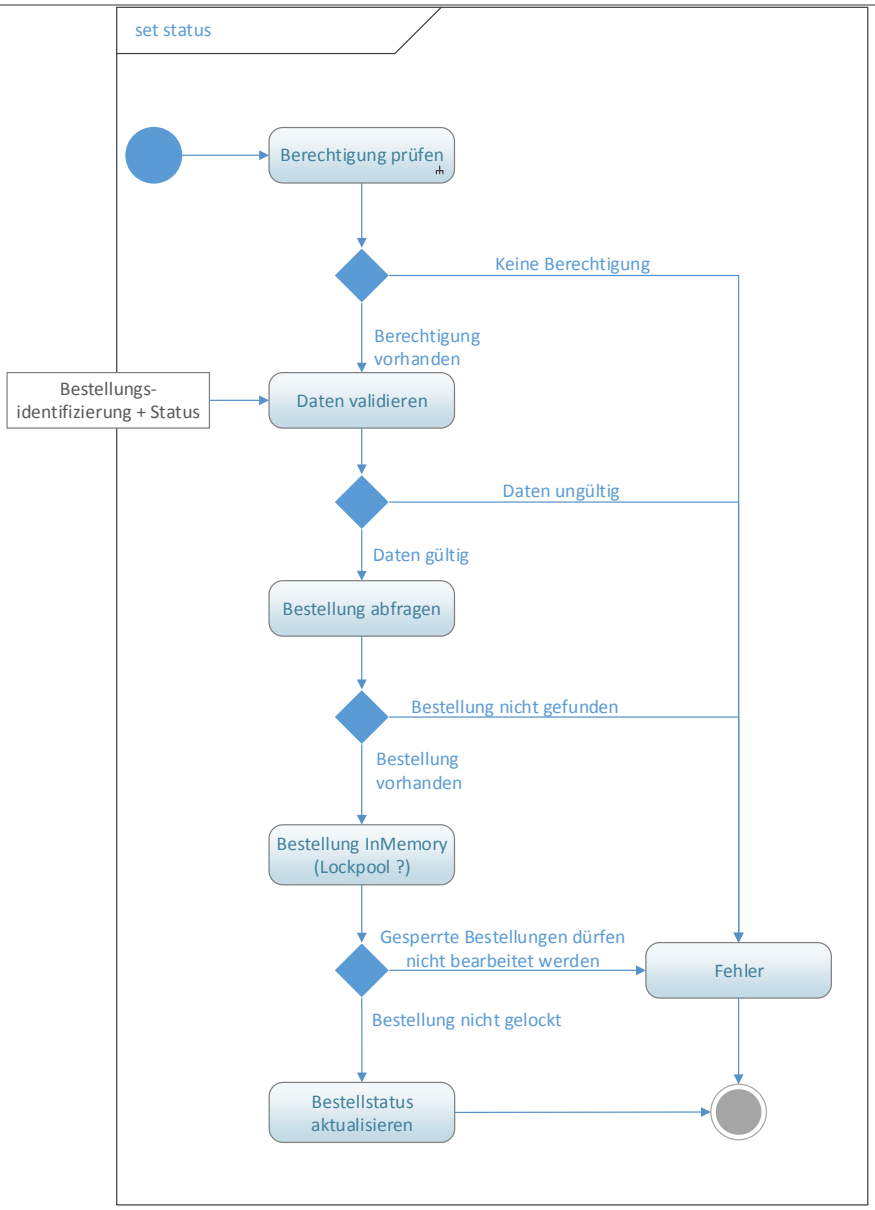
#9. set status	
Beschreibung	Status einer Bestellung setzen.
Akteure	Verkäufer
Auslöser	Der Verkäufer aktualisiert den Status der Bestellung.
Vorbedingung	Eine Bestellung muss vorhanden sein.
Ergebnis	Der Status der Bestellung wird aktualisiert.
Ergebnis im Fehlerfall	Fehlermeldung, Der Status der Bestellung wird nicht aktualisiert.
Ablauf	1. Berechtigung prüfen 2. Eingabe validieren 3. Bestellung überprüfen 4. Bestellung aktualisieren
Diagramm	

Abb. 14: Aktivität: set status

2 Architektur und Designentscheide

2.1 Architekturübersicht

Die Architektur wird durch eine drei Layer-Architektur realisiert, welche sich in die Layer Presentation (FBSGuiClient), Business (FBSService) und Data (FBSData) aufteilt.

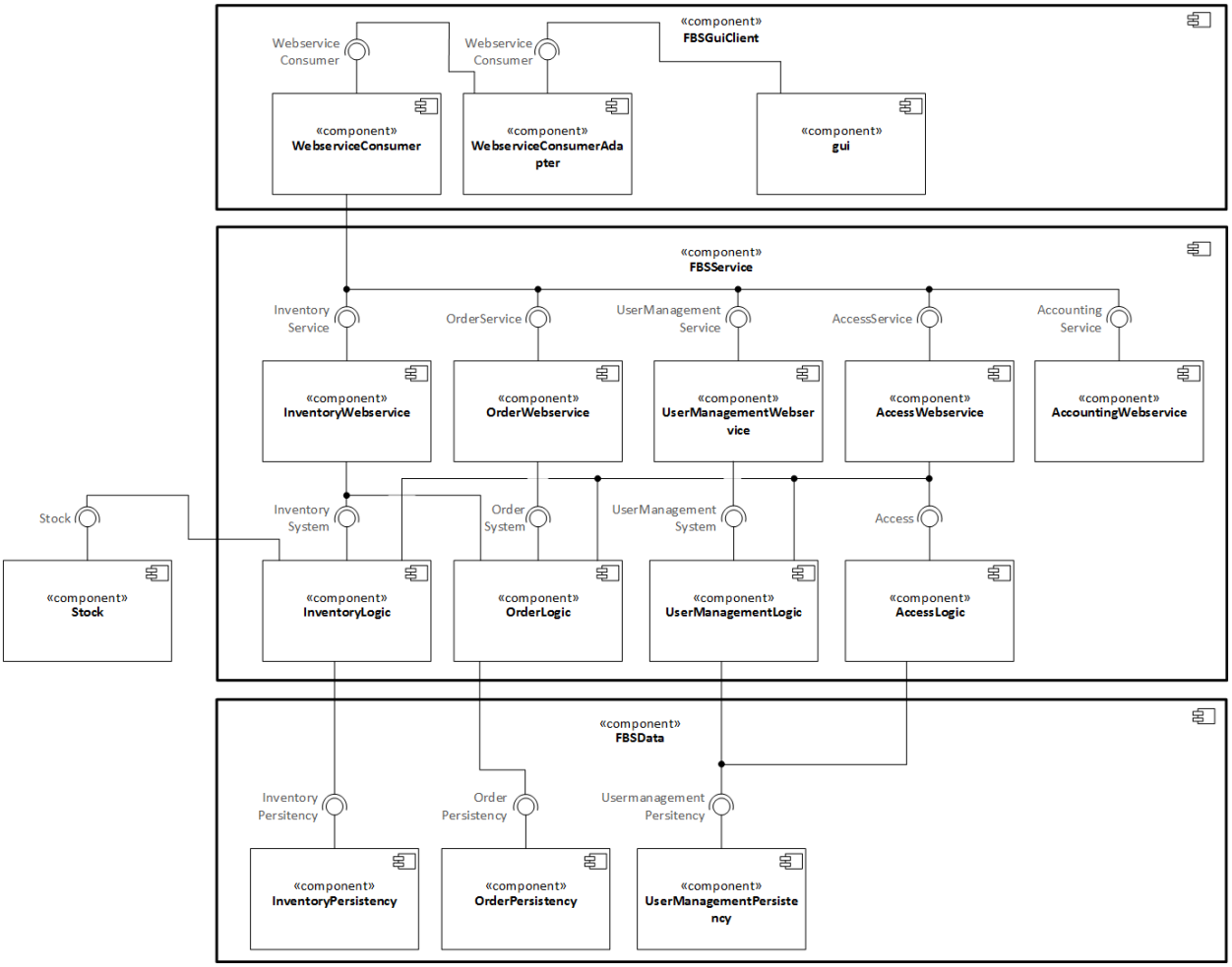


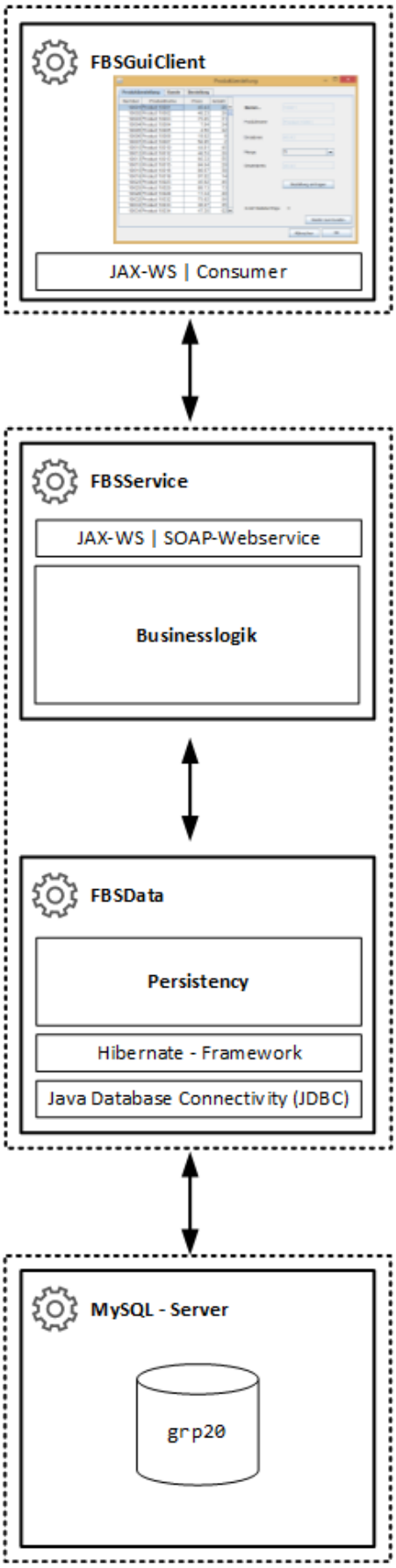
Abb. 15: Komponentendiagramm

Die Authentifikation des Users wird beim Starten der Applikation durch Abfrage von Benutzername und Passwort vorgenommen. Das GUI kommuniziert via WebService mit der BusinessLogik. Jede Anfrage muss mit einem gültigen SessionKey autorisiert werden.

Die Transaktionssicherheit wird innerhalb von FBSService sichergestellt. Dort werden einzelnen Bestellung gelockt und Produkte Reservationen vorgenommen. FBSData arbeitet mit Transaktionen auf der Datenbanken. Dadurch ist die Konsistenz der Daten zusammen mit der Logik in FBSService gewährleistet.

Für eine Skalierung kann das Backend (MySQL-Server) jederzeit durch einen leistungsstärkeren DB-Server getauscht werden. Der zentrale WebService könnte auf mehrere Webservices aufgeteilt werden. Das GUI kann bereits im aktuellen Release dediziert auf verschiedenen Clients ausgeführt werden.

2.2 Technologieübersicht



2.2.1 FBSGuiClient

FBSGuiClient ist der Rich-Client welcher für die Bestellsoftware zum Einsatz kommt.

Nach dem Login mit Username und Passwort wird die Bestellmaske angezeigt. Zusätzlich zur Möglichkeit eine Bestellung zu tätigen, können die Kunden mit ihren Adressen verwaltet und erweitert werden.

2.2.2 FBSService

In FBSService ist die eigentliche Logik für die Bestellsoftware. Neben Authentication und Autorisation wird hier auch der Transaktionsschutz sichergestellt. D.h. Produktreservationen werden hier zentral verwaltet, bis eine Bestellung bestätigt wird. Für die Umsysteme wird ein SOAP-Webservice eingesetzt. Dieser wird vom Rich-Client wie auch z.B. von der RW-Applikation verwendet.

2.2.3 FBSData

In FBSData wird die Anbindung an die zentrale Datenbank mittels JDBC und dem Hibernate-Framework realisiert.

Es werden diverse Schnittstellen für FBSService zur Verfügung gestellt.

2.2.4 MySQL – Server

Der MySQL – Server mit der verwendeten Datenbank wird von der HSLU zur Verfügung gestellt.

In der Datenbank werden sämtliche Daten persistent gespeichert.

Alternativ kann die Datenbank auch durch einen MS-SQL Server ersetzt werden. Mittels JDBC und Hibernate ist dies abstrahiert und kann ausgetauscht werden.

Abb. 16: Technologieübersicht

2.3 Daten (Mengengerüst & Strukturen)

2.3.1 Logisches ER-Model

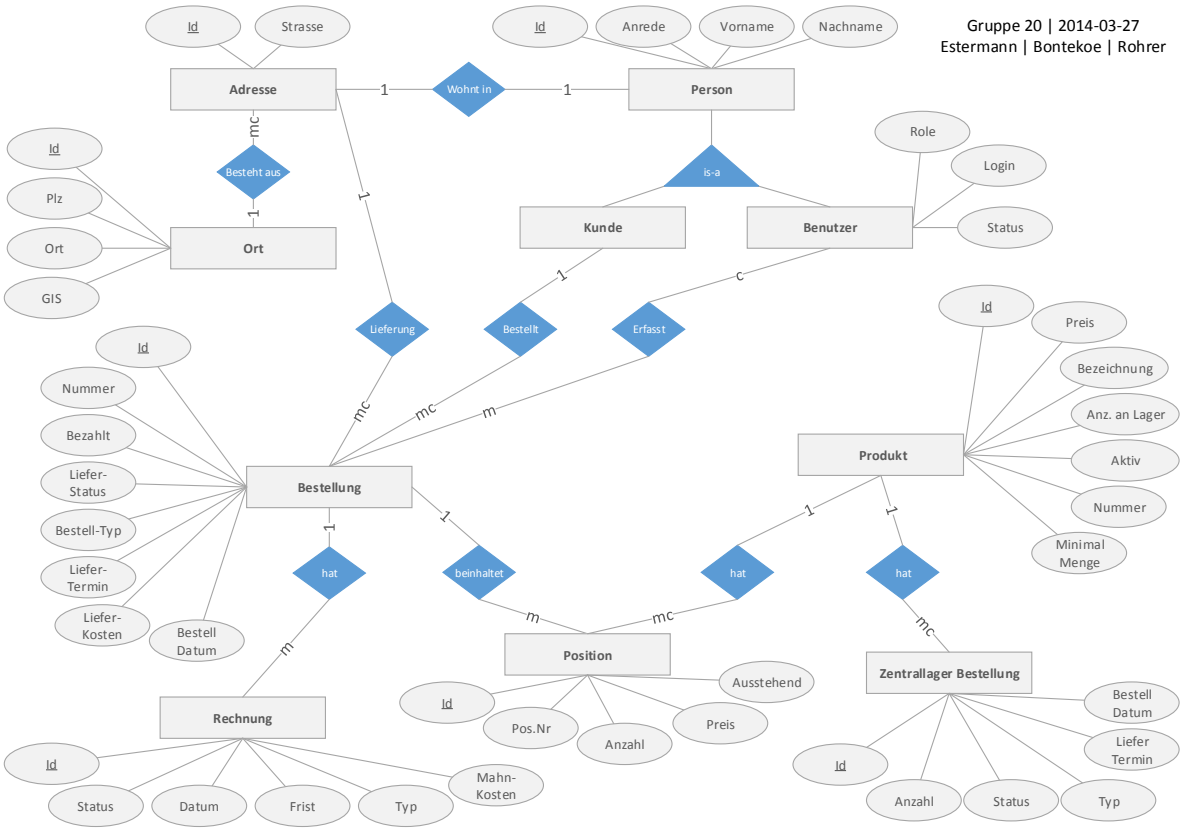


Abb. 17: Datenbank ER-Model

2.3.2 Physisches ER-Model

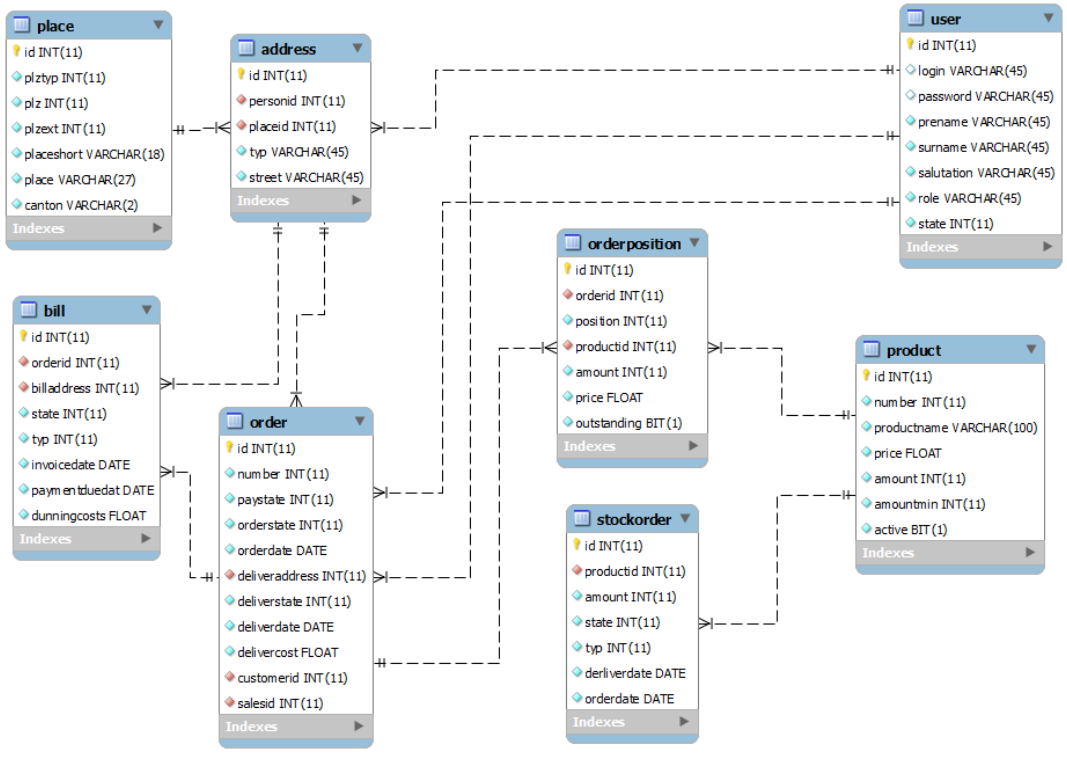


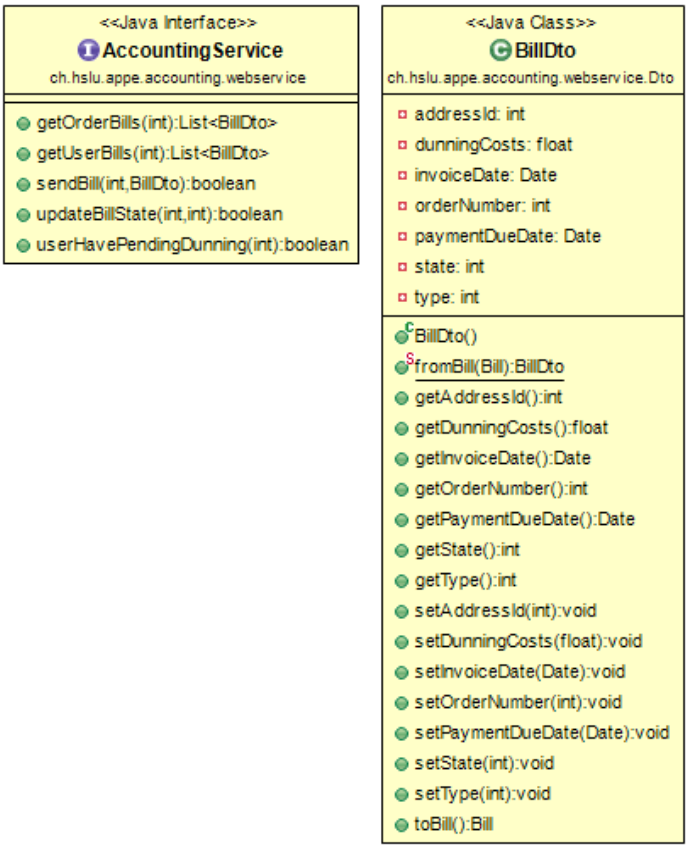
Abb. 18: Physisches ER-Model

3 Schnittstellen

3.1 Externe Schnittstellen

3.1.1 AccountingService

Für die Anbindung der RW-Applikation wird ein zusätzliches Interface angeboten welches ebenfalls via SOAP Webservice angesprochen wird. Zusammen mit den bestehenden Interfaces von FBSService kann das Rechnungswesen vollumfänglich abgedeckt werden.



3.1.2 Zentrallager (Stock-Library)

Im Zentrallager besteht ein Legacy-System welches mittels der Stock-Library angesprochen werden kann.

Es wurde die Version 4.0.2 von Appe-Stock verwendet. Diese liefert ein wohldefiniertes Interface, worüber im Wesentlichen Artikel im Zentrallager reserviert und bestellt werden können.

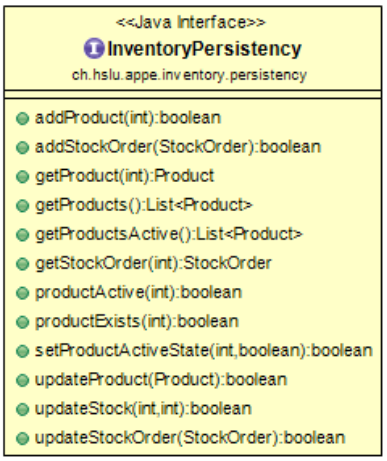
3.2 Interne Schnittstellen in FBSData

3.2.1 Inventory

Verantwortlich für das aktivieren/ deaktivieren von Produkten. Des Weiteren kann ein bestehendes Produkt bearbeitet oder aktualisiert werden. Via „getProduct“ wird das aktuelle Produkt geholt.

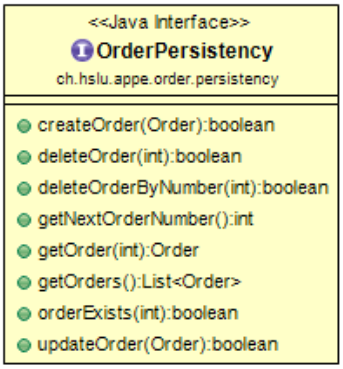
3.2.2 InventoryPersistency

Verantwortlich für jegliche Produkt- und Lagerinformationen. Des Weiteren können neue Produkte hinzugefügt werden und bestehende Produkte editiert werden.



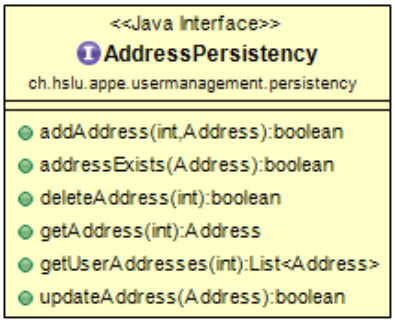
3.2.3 Order Persistency

Verantwortlich für das Erstellen einer Bestellung, der Löschung einer Bestellung sowie das überprüfen ob eine Bestellung bereits besteht.



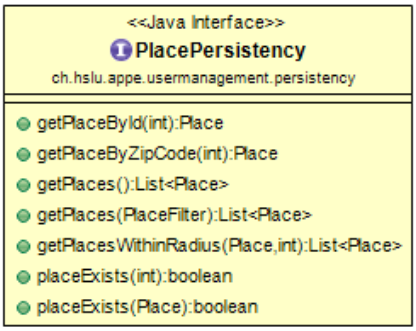
3.2.4 AddreessPersistency

Verantwortlich für das Hinzufügen, Löschen und Editieren einer Adresse. Des Weiteren kann eine spezifische Adresse anhand der „KundenId“ ausgelesen werden oder es kann überprüft werden ob eine Adresse existiert.



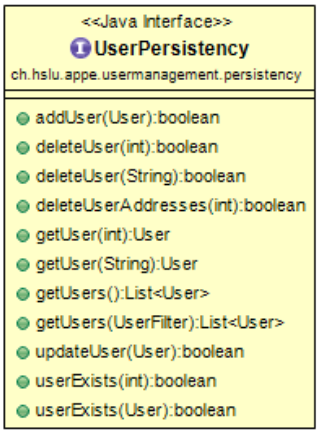
3.2.5 PlacePersistency

Verantwortlich für das Auslesen eines Ortes sowie der passenden Postleitzahl. Des Weiteren kann anhand der ID oder anhand des Objetes „Place“ überprüft werden ob ein Ort existiert.



3.2.6 UserPersistency

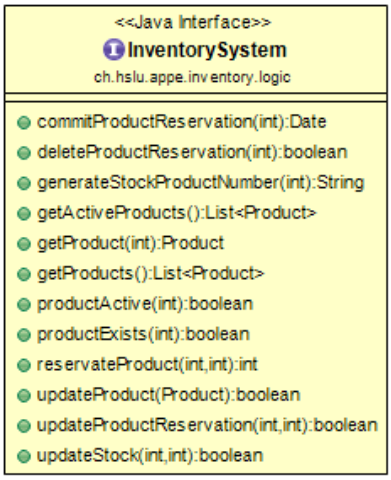
Verantwortlich für das Hinzufügen, Löschen und Editieren eines Benutzers. Des Weiteren kann überprüft werden ob ein Benutzer bereits existiert.



3.3 Interne Schnittstellen in FBSService

3.3.1 Inventory System

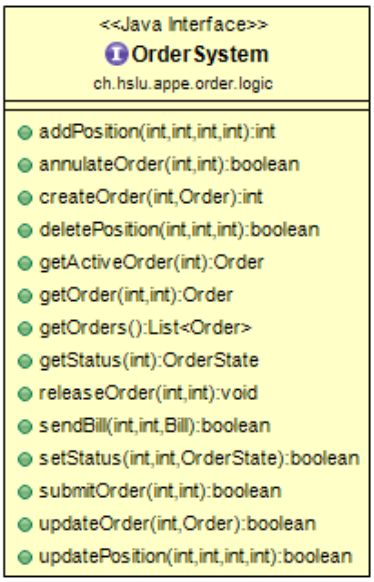
Verantwortlich für das Produkt. Produkte können reserviert werden. Eine Reservierung kann storniert werden. Produkte können überprüft werden ob Sie existieren und ob Sie aktiv sind.



3.3.2 Order System

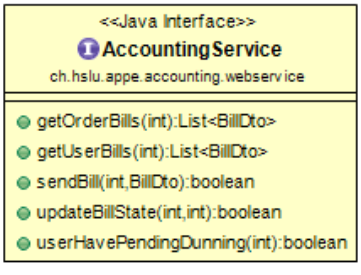
Verantwortlich für das Bestellsystem. Bestellungen können erstellt werden. Bestellungen können annulliert werden und Bestellungen können bestätigt werden.

Des Weiteren kann der aktuelle Status abgefragt werden und eine Rechnung verschickt werden.



3.3.3 AccountingService

Verantwortlich für die Rechnung. Rechnung können versendet werden, der aktuelle Status kann abgefragt werden und es kann überprüft werden ob der Kunde noch offene Rechnungen hat.



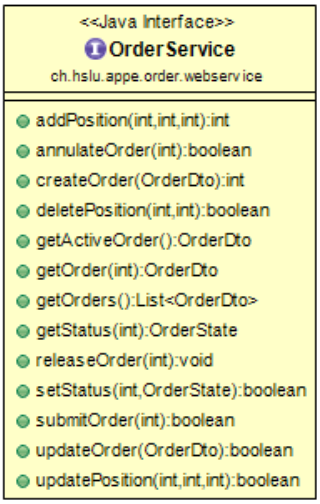
3.3.4 InventoryService

Verantwortlich für das Auslesen einer oder mehreren Produkten. Produkte können überprüft werden ob Sie existieren.

3.3.5 OrderService

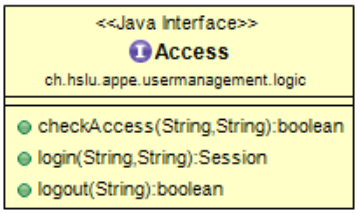
Verantwortlich für den Bestellservice. Bestellungen können erstellt werden. Bestellungen können annulliert werden und Bestellungen können bestätigt werden.

Des Weiteren kann der aktuelle Status abgefragt werden und eine Rechnung verschickt werden



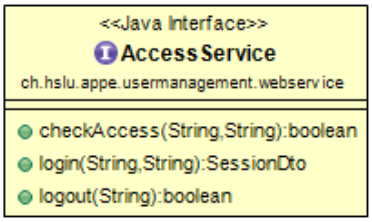
3.3.6 Access

Verantwortlich für den Zugriff. Login, Logout sowie Überprüfung der Zugriffsberechtigung.



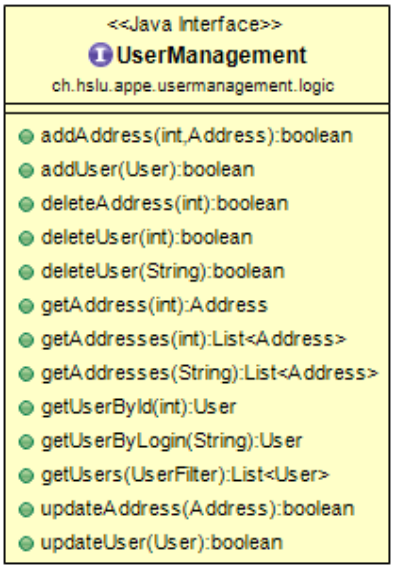
3.3.7 AccessService

Verantwortlich für den Zugriff. Login, Logout sowie Überprüfung der Zugriffsberechtigung.



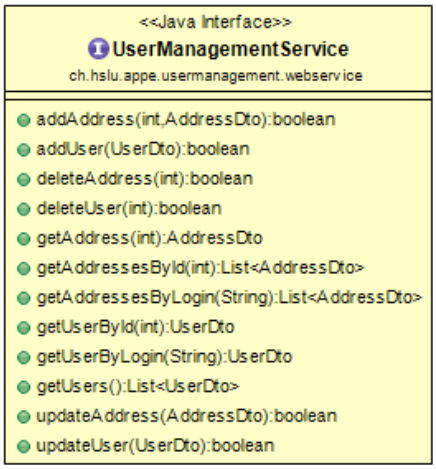
3.3.8 UserManagement

Verantwortlich für das Hinzufügen, Löschen und Editieren eines Benutzers. Benutzer können überprüft werden, ob Sie bereits existieren. Des Weiteren können neue Adressen hinzugefügt werden. Bestehende Adressen können gelöscht oder editiert werden.



3.3.9 UserManagementService

Verantwortlich für das Hinzufügen, Löschen und Editieren eines Benutzers. Benutzer können überprüft werden, ob Sie bereits existieren. Des Weiteren können neue Adressen hinzugefügt werden, bestehende Adressen können gelöscht oder editiert werden.



3.4 Interne Schnittstellen in FBSGuiClient

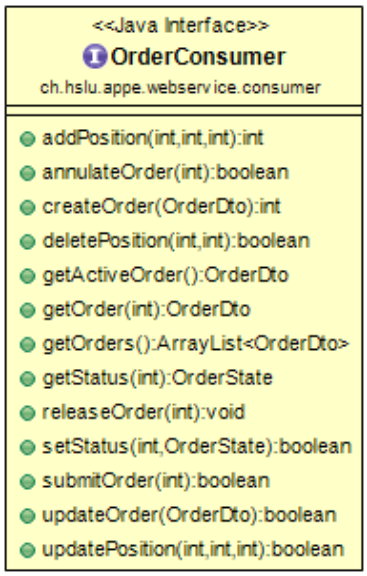
3.4.1 InventoryConsumer

Verantwortlich für das Auslesen der aktuellen Produkte. Des Weiteren kann überprüft werden ob ein Produkt aktiv ist oder ob es überhaupt existiert.

3.4.2 OrderConsumer

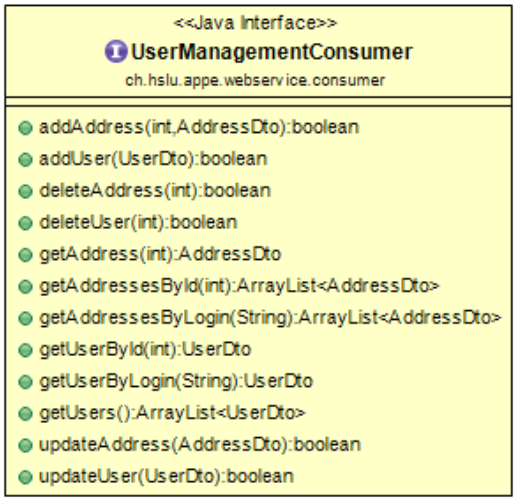
Verantwortlich für die Bestellung. Bestellungen können erstellt werden. Bestellungen können annulliert werden und Bestellungen können bestätigt werden.

Des Weiteren kann der aktuelle Status abgefragt werden und eine Rechnung verschickt werden



3.4.3 UserManagementConsumer

Verantwortlich für das Hinzufügen, Löschen und Editieren eines Benutzers. Benutzer können überprüft werden, ob Sie bereits existieren. Des Weiteren können neue Adressen hinzugefügt werden, bestehende Adressen können gelöscht oder editiert werden.



3.5 Benutzerschnittstelle(n)

Der Benutzer kann über das Rich-GUI auf die Bestellapplikation zugreifen.

Infolge der sauberen Trennung zwischen Logik und Präsentation kann z.B. die Applikation erweitert werden und ein Web-GUI angeboten werden, welche die gleichen Interfaces von FBSService verwendet.

4 GUI

Bevor mit der Implementierung der Benutzeroberfläche begonnen wurde, wurde für jeden Teilbereich ein Prototyp erstellt. Anschliessend wurde das GUI mit Hilfe von SWING in Java erstellt.

4.1 GUI - Mockups

4.1.1 Kundenverwaltung

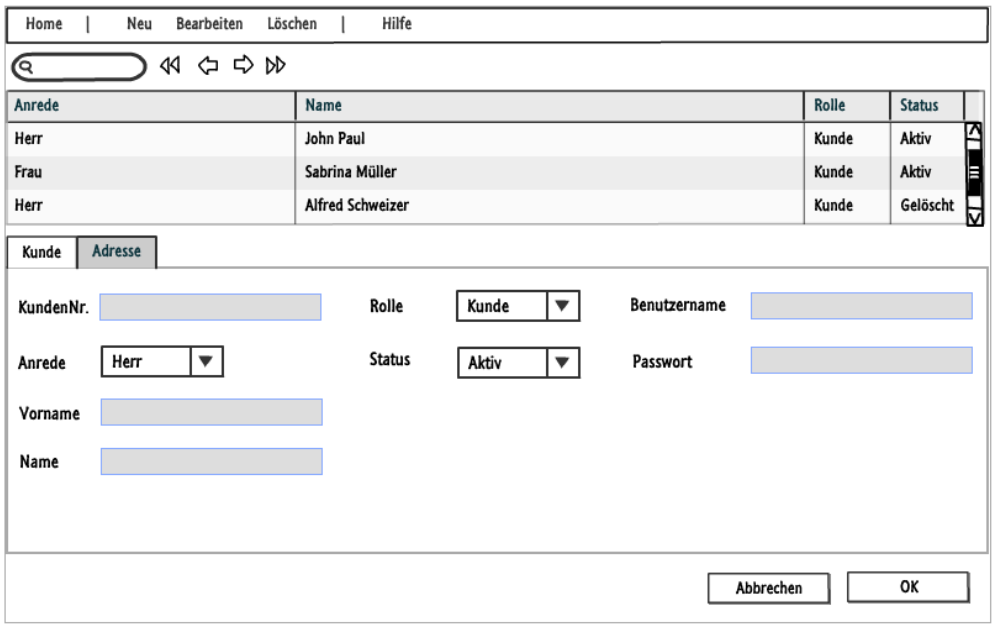


Abb. 19: Mockup: Kundenverwaltung

4.1.2 Adressverwaltung

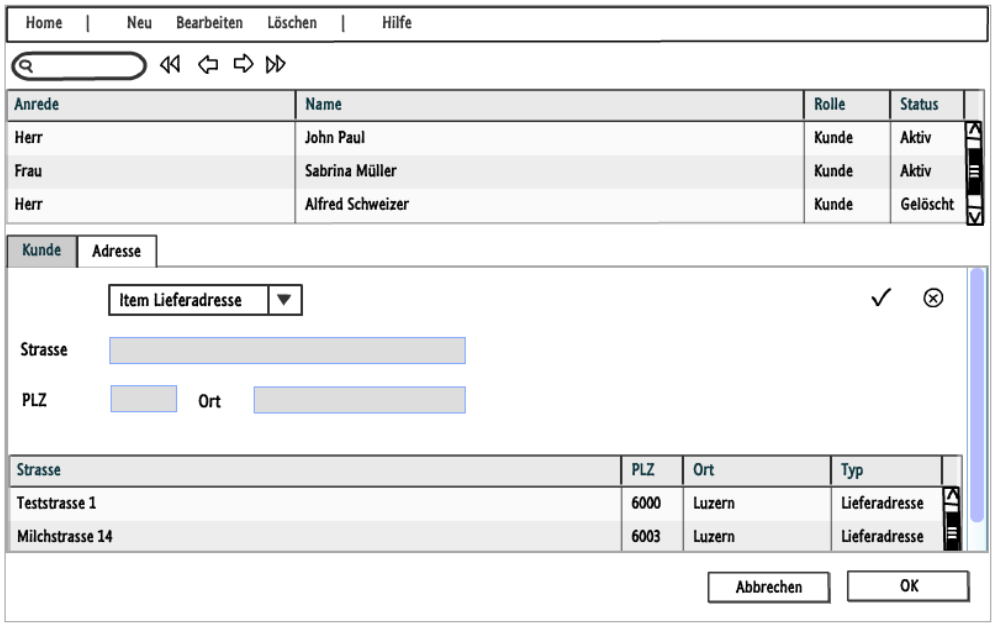


Abb. 20: Mockup: Adressverwaltung

4.1.3 Produktauswahl

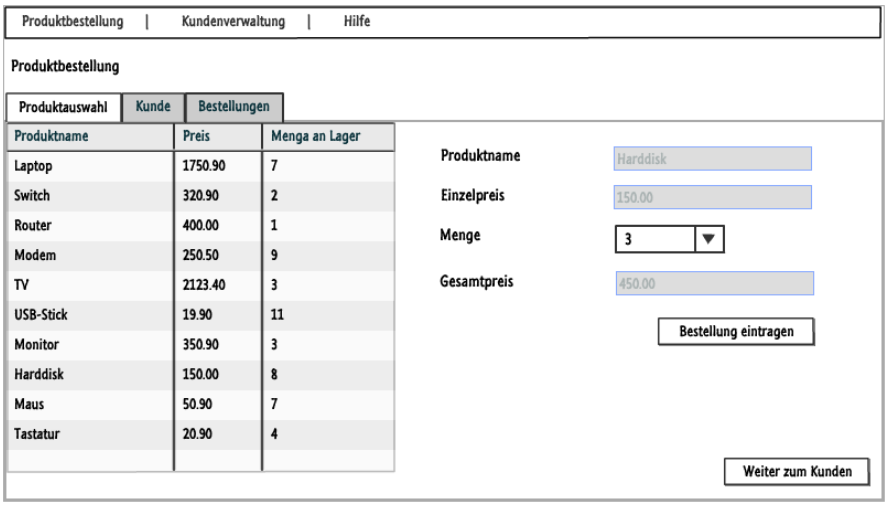


Abb. 21: Mockup: Produktauswahl

4.1.4 Kundenauswahl

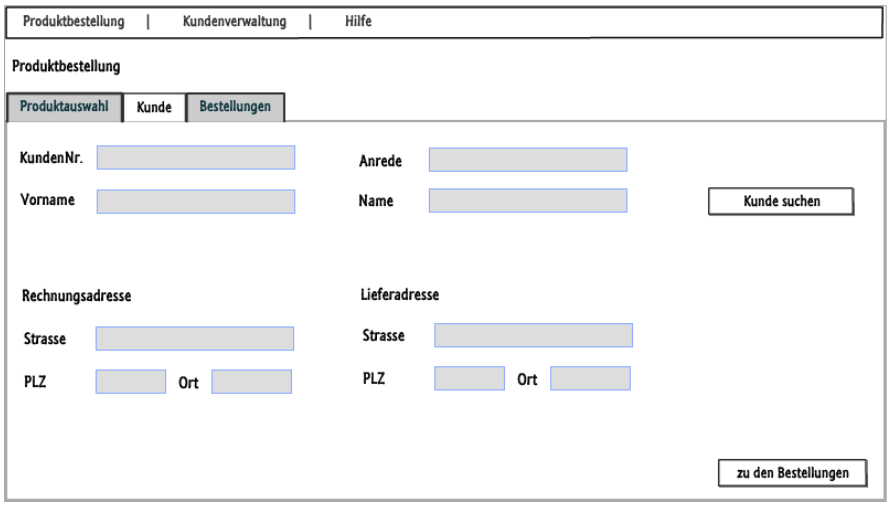


Abb. 22: Mockup: Kundenauswahl

4.1.5 Bestellbestätigung

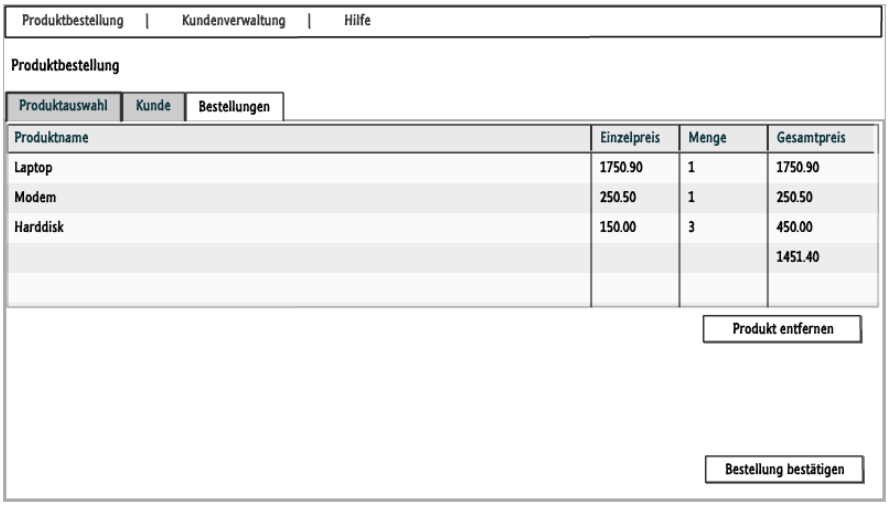


Abb. 23: Mockup: Bestellbestätigung

4.2 GUI – Release 1.0.0

4.2.1 Login

Abb. 24: GUI Release 1.0.0: Login

4.2.2 Kundenverwaltung

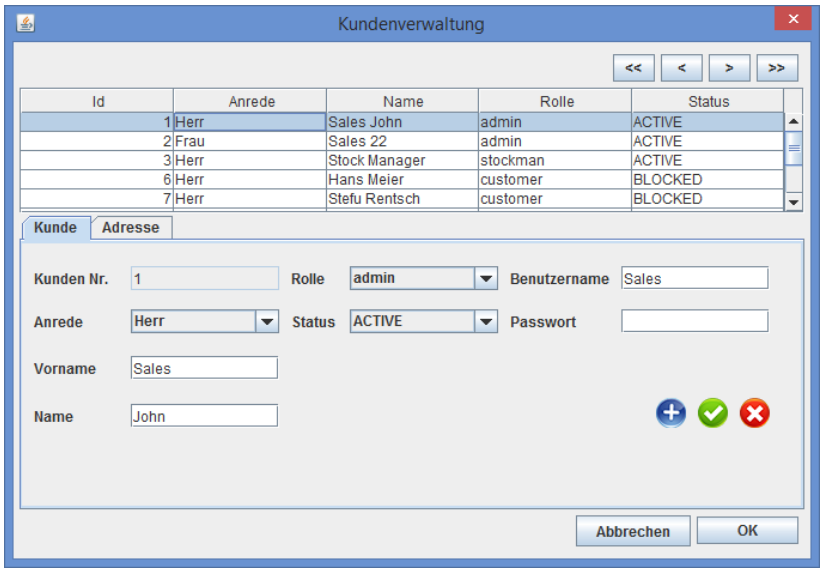


Abb. 25: GUI Release 1.0.0: Kundenverwaltung

4.2.3 Adressverwaltung

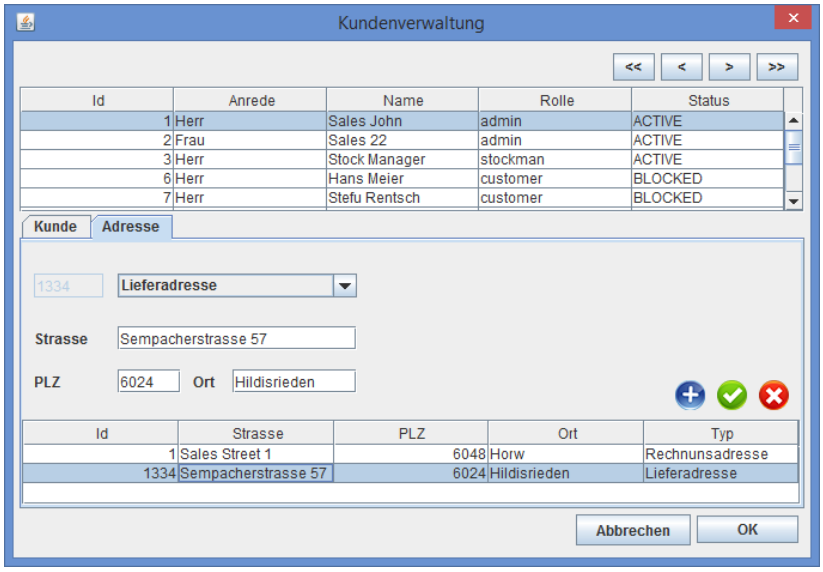


Abb. 26: GUI Release 1.0.0: Adressverwaltung

4.2.4 Produktauswahl

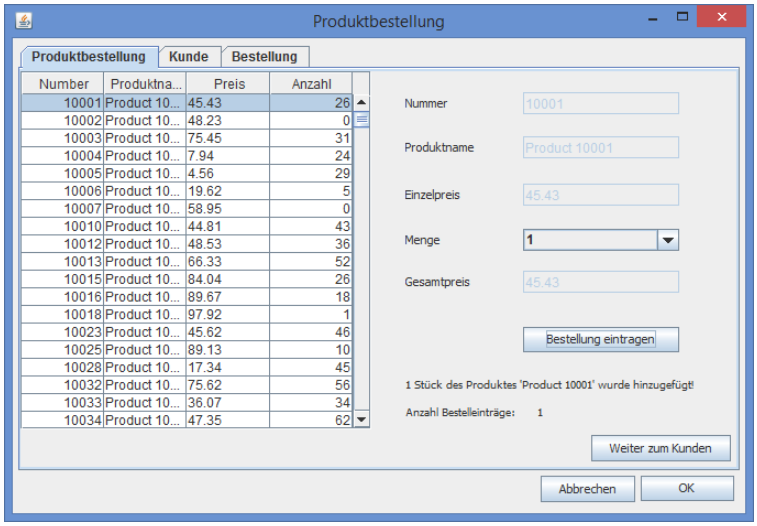


Abb. 27: GUI Release 1.0.0: Produktauswahl

4.2.5 Kundenauswahl

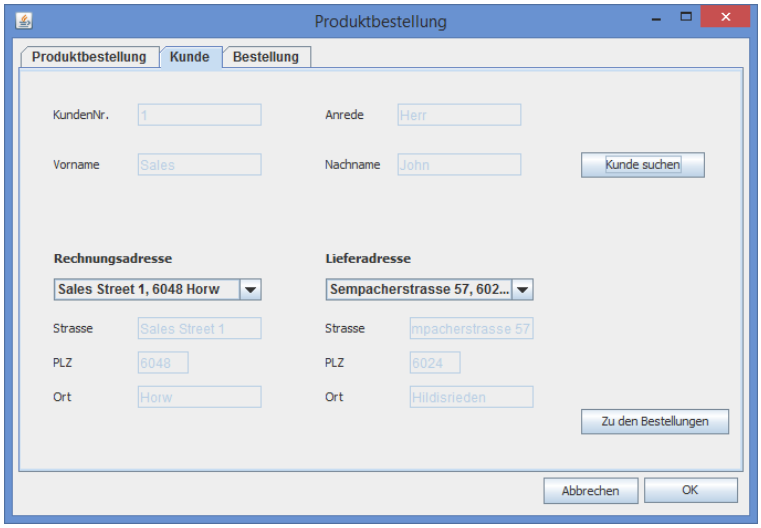


Abb. 28: GUI Release 1.0.0: Kundenauswahl

4.2.6 Bestellbestätigung

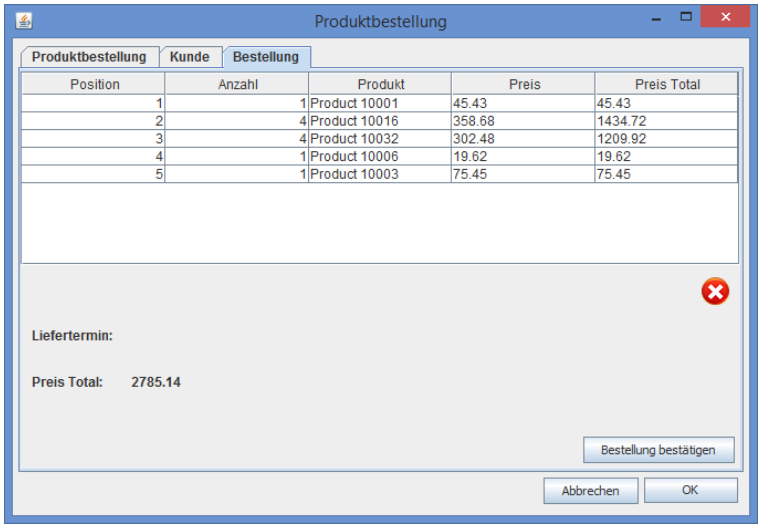


Abb. 29: GUI Release 1.0.0: Bestellbestätigung

5 Environment-Anforderungen

5.1 Hardware

Der Webservice sowie die Benutzeroberfläche laufen auf fast jeder Hardware.

5.2 Software

Betriebssystem, welches Java 7 unterstützt muss vorhanden sein. Die Business-Logik sowie das GUI setzen mindestens Java Version 1.7 voraus.

Für die Verwendung vom GUI wird eine Auflösung von 1024x768 empfohlen.

5.3 Datenbank

Für das Backend muss eine Datenbank vorhanden sein, aktuell wird dafür MySQL eingesetzt.

6 Starten des Bestellsystems

6.1 FBSData, Backend / MySQL

Für den Betrieb wird ein zentraler MySQL Server benötigt. Die Verbindungsparameter sind in FBSData in der Datei hibernate.cfg.xml definiert.

6.2 FBSService

Die Business-Logik ist in FBSService und kann mittels ch.hslu.program gestartet werden. Per Default läuft der Service auf dem Localhost auf Port 8080.

Benötigte JARs: fbsdata-1.0.0.jar und fbsservice-1.0.0.jar

6.3 FBSGuiClient

Das GUI ist in FBSGuiClient und kann mittels ch.hslu.appe.gui.login gestartet werden.

Benötigte JARs: fbsguiclient-1.0.0.jar

7 ScrumDo

Für das Projekt wurden folgende Epics erstellt:

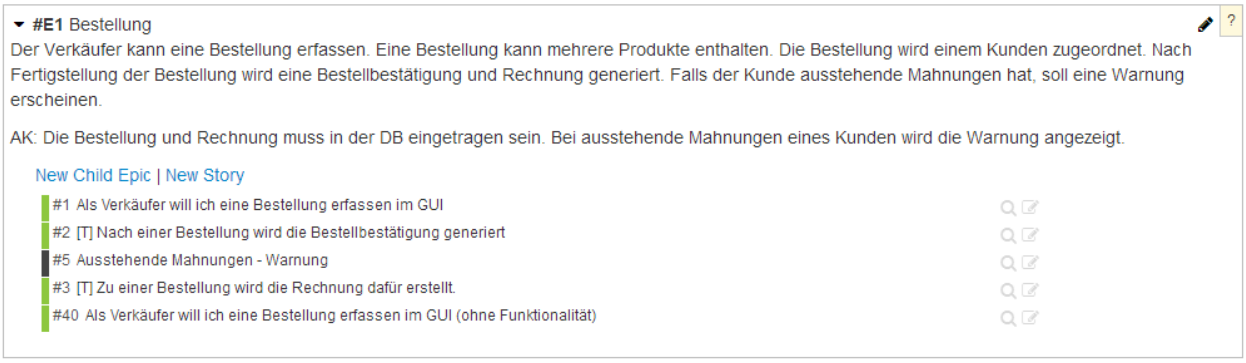


Abb. 30: ScrumDo: Bestellung

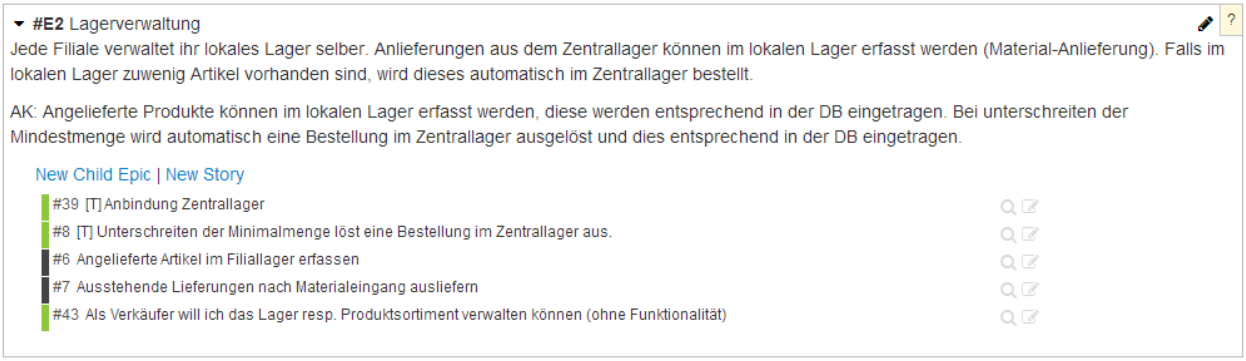


Abb. 31: ScrumDo: Lagerverwaltung

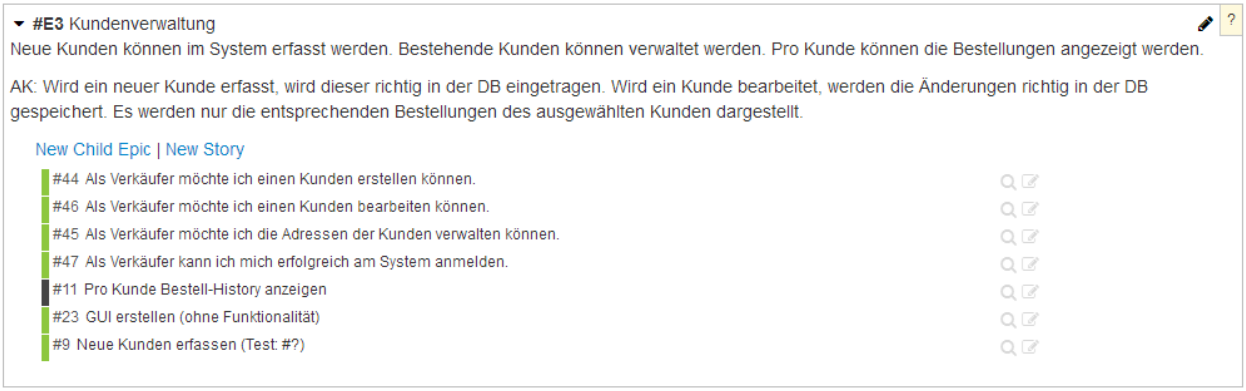


Abb. 32: ScrumDo: Kundenverwaltung

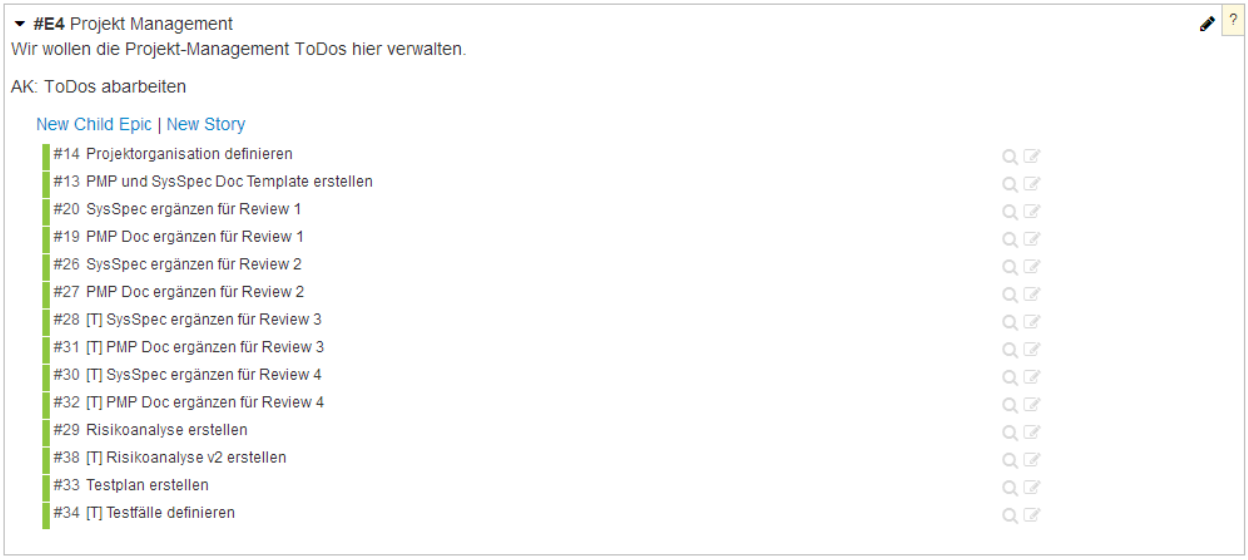


Abb. 33: ScrumDo: Projekt Management

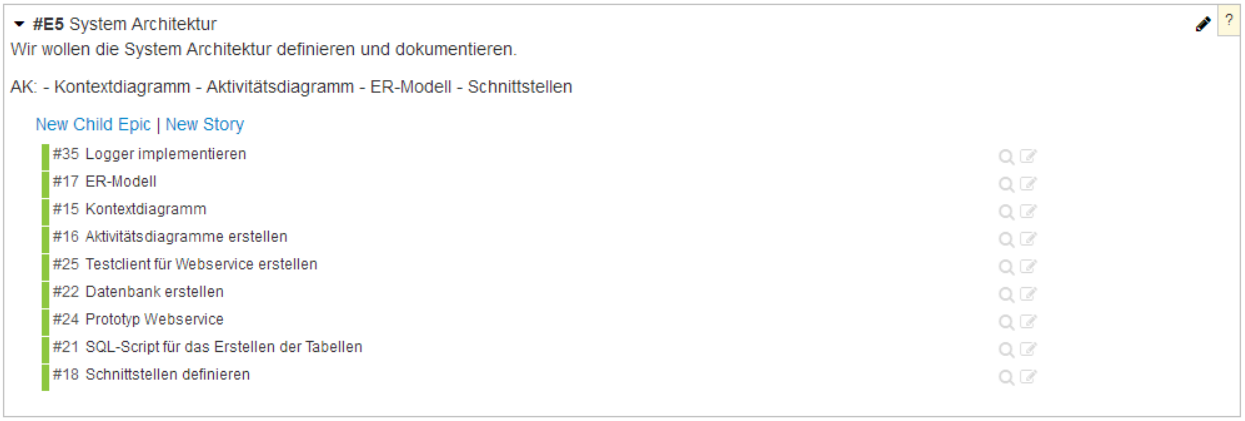


Abb. 34: ScrumDo: System Architektur

Die detaillierte Beschreibung der einzelnen Stories befindet sich im Anhang.

Abbildungsverzeichnis

Abb. 1: Funktionale Sicht – Bestellsytem 1
Abb. 2: Use-Case: Accessmanagement 2
Abb. 3: Use-Case: Location 2
Abb. 4: Use-Case: Inventory 2
Abb. 5: Use-Case: Order 3
Abb. 6: Aktivität: get order 4
Abb. 7: Aktivität: create order 5
Abb. 8: Aktivität: update order 6
Abb. 9: Aktivität: annulate order 7
Abb. 10: Aktivität: submit order 8
Abb. 11: Aktivität: add position 9
Abb. 12: Aktivität: update position 10
Abb. 13: Aktivität: get status 11
Abb. 14: Aktivität: set status 12
Abb. 15: Komponentendiagramm 13
Abb. 16: Technologieübersicht 14
Abb. 17: Datenbank ER-Model 15
Abb. 18: Physisches ER-Model 15
Abb. 19: Mockup: Kundenverwaltung 22
Abb. 20: Mockup: Adressverwaltung 22
Abb. 21: Mockup: Produktauswahl 23
Abb. 22: Mockup: Kundenauswahl 23
Abb. 23: Mockup: Bestellbestätigung 23
Abb. 24: GUI Release 1.0.0: Login 24
Abb. 25: GUI Release 1.0.0: Kundenverwaltung 24
Abb. 26: GUI Release 1.0.0: Adressverwaltung 24
Abb. 27: GUI Release 1.0.0: Produktauswahl 25
Abb. 28: GUI Release 1.0.0: Kundenauswahl 25
Abb. 29: GUI Release 1.0.0: Bestellbestätigung 25
Abb. 30: ScrumDo: Bestellung 28
Abb. 31: ScrumDo: Lagerverwaltung 28
Abb. 32: ScrumDo: Kundenverwaltung 28
Abb. 33: ScrumDo: Projekt Management 29
Abb. 34: ScrumDo: System Architektur 29

Anhang

ScrumDo Stories

Story ID	Summary	Detail	Status	Assignee	Epics
40	Als Verkäufer will ich eine Bestellung erfassen im GUI (ohne Funktionalität)	GUI / Mokup für die Oberfläche einer Bestellung ist vorhanden.	Done	tabontek	#E1
1	Als Verkäufer will ich eine Bestellung erfassen im GUI	Eine Bestellung kann via Telefon, Schalter, Fax, Email oder Brief erfasst werden. Die Bestellung wird einem Kunden zugeordnet. Es können nur "aktive" Produkte bestellt werden. AK: Die Bestellung wird korrekt in der DB eingetragen und Lagerbestand nachgeführt. Transaktionssicherheit gewährleistet ist Bei der Bestellung können nur definierte (aktive) Produkte bestellt werden.	Done	taesterm	#E1
2	[T] Nach einer Bestellung wird die Bestellbestätigung generiert	Nach Abschluss der Bestellung wird eine Bestellbestätigung erstellt. Diese beinhaltet das Bestelldatum, voraussichtliche Lieferdatum, Lieferkosten, Gesamtpreis. AK: Das Bestelldatum, Lieferdatum, Lieferkosten und Gesamtpreis wird korrekt berechnet resp. dargestellt. In der Bestellbestätigung werden alle bestellten Produkte dargestellt.	Done	tabontek	#E1
3	[T] Zu einer Bestellung wird die Rechnung dafür erstellt.	Zur Bestellung wird die Rechnung erstellt und die notwendigen Daten für das RW-System generiert. AK: Es werden alle benötigten Daten in der DB für das RW-System eingetragen.	Done	tabontek	#E1

43	Als Verkäufer will ich das Lager resp. Produktsortiment verwalten können (ohne Funktionalität)	GUI / Mockup für die Produkt- / Lagerverwaltung ist vorhanden.	Done	tabontek	#E2
39	[T] Anbindung Zentrallager	Anbindung des Zentrallagers. AK: Produkte welche sich im Zentrallager befinden können über eine externe Schnittstelle (Komponente) ausgelesen werden	Done	tdrohrer	#E2
8	[T] Unterschreiten der Minimalmenge löst eine Bestellung im Zentrallager aus.	Wird die Minimalmenge eines Artikels unterschritten, wird automatisch eine Bestellung im Zentrallager ausgelöst. AK: Bei unterschreiten der Minimalmenge des Artikels wird dieser im Zentrallager bestellt, die Daten werden korrekt in der DB eingetragen.	Done	tdrohrer	#E2
23	GUI erstellen (ohne Funktionalität)	GUI, ohne Funktionalität, für die Kundenverwaltung erstellen. AK: - Alle benötigten Felder sind vorhanden.	Done	tabontek	#E3
9	Neue Kunden erfassen (Test: #?)	Neue Kunden können im System erfasst werden. AK: - Verbindung auf DB funktioniert - ER-Modell resp. Tabellen sind vorhanden - Die Informationen vom Kunden werden mittels JPA in der richtigen DB, in den richtigen Tabellen gespeichert. - Dieser Fall wird mittels dem Test #??? abgedeckt.	Done	taesterm	#E3
44	Als Verkäufer möchte ich einen Kunden erstellen können.	- Kunde kann im GUI eingetragen werden - Kunde wird mittels Business-Logic übertragen - Kunde wird im Data-Layer persistiert	Done	taesterm	#E3

46	Als Verkäufer möchte ich einen Kunden bearbeiten können.	- Im GUI werden die bestehenden Kunden dargestellt - Im GUI kann ein Kunden ausgewählt und bearbeitet werden - Änderungen werden via Business-Logik übertragen und im Data-Layer persistiert	Done	taesterm	#E3
45	Als Verkäufer möchte ich die Adressen der Kunden verwalten können.	- Die Adressen können im GUI dargestellt werden - Mutationen der Adressen werden via Business-Logic übertragen - Die angepasste Adresse wird im Data-Layer persistiert	Done	tabontek	#E3
47	Als Verkäufer kann ich mich erfolgreich am System anmelden.	Der Verkäufer kann sich mit einem gültigen und aktiven Account am System anmelden. AK: - Es können sich nur aktive Accounts am System anmelden - Es dürfen sich nur User anmelden mit den benötigten Rechte (Customer kann sie nicht anmelden) - Zwischen GUI und BL findet die Authentifizierung und Autorisierung statt, für den User wird eine Session inkl. SessionID geniert. Die SessionID wird beim jedem Webservice Request (SOAP) überprüft. - Die Story wird zusätzlich mit einem Testfall abgedeckt.	Done	tdrohrer	#E3
14	Projektorganisation definieren	Projektorganisation definiert und dokumentiert.	Done	tdrohrer	#E4
13	PMP und SysSpec Doc Template erstellen	Draft Version der Dokumente sind vorhanden.	Done	tdrohrer	#E4
29	Risikoanalyse erstellen	PMP erweitern: Erste Risikoanalyse erstellen AK: - Risiken sind im PMP dokumentiert	Done	tabontek	#E4

33	Testplan erstellen	PMP erweitern: Erste Testplan erstellen AK: - Testplan ist im Dokument PMP eingetragen	Done	tabontek	#E4
20	SysSpec ergänzen für Review 1	Das SysSpec Doc ist ergänzt und ready für "Review 1" AK: - Kontextdiagramm muss vorhanden sein - Aktivitätsdiagramme müssen vorhanden sein	Done	tdrohrer	#E4
19	PMP Doc ergänzen für Review 1	Das PMP ist bereit für "Review 1" AK: - #14 Projektorganisation muss vorhanden sein - Termine für die Sprints müssen definiert sein	Done	tdrohrer	#E4
26	SysSpec ergänzen für Review 2	SysSpec für Review 2 vervollständigen.	Done	tdrohrer	#E4
27	PMP Doc ergänzen für Review 2	Das PMP Doc ist bereit für "Review 2"	Done	tdrohrer	#E4
38	[T] Risikoanalyse v2 erstellen	Risikoanalyse überprüfen und anschliessend neu erstellen (Version 2) AK: Risikoanalyse ist im Dokument PMP hinterlegt	Done	tabontek	#E4
34	[T] Testfälle definieren	Testfälle welche nicht automatisiert ausgeführt werden können werden explizit im PMP erfasst. Dies betrifft insbesondere GUI Aktivitäten. AK: - Testfälle sind im Dokument PMP eingetragen	Done	tabontek	#E4
28	[T] SysSpec ergänzen für Review 3	Das SysSpec Doc ist ergänzt und ready für "Review 3"	Done	tdrohrer	#E4
31	[T] PMP Doc ergänzen für Review 3	Das PMP Doc ist bereit für "Review 3"	Done	tdrohrer	#E4

30	[T] SysSpec ergänzen für Review 4	Das SysSpec Doc ist ergänzt und ready für "Review 4"	Done	tdrohrer	#E4
32	[T] PMP Doc ergänzen für Review 4	Das PMP Doc ist bereit für "Review 4"	Done	tdrohrer	#E4
15	Kontextdiagramm	Erstellen des Kontextdiagramm	Done	tdrohrer	#E5
16	Aktivitätsdiagramme erstellen	- get order - update order - submit order - create order - annulate order - add position - update position - set status - get status	Done	taesterm	#E5
17	ER-Modell	ER-Modell erstellt und Review mit Hr. Olnhoff durchgeführt.	Done	tdrohrer	#E5
35	Logger implementieren	Log Einträge werden von den einzelnen Komponenten generiert und in den jeweiligen Logfiles festgehalten. AK: Log Einträge werden in die Logfiles geschrieben	Done	taesterm	#E5
25	Testclient für Webservice erstellen	Testclient für Webservice erstellen: AK: - Client kann erfolgreich mit dem Webservice kommunizieren	Done	taesterm	#E5
22	Datenbank erstellen	Die Datenbank wird mittels dem SQL-Script korrekt erstellt. AK: Die benötigten Tabellen sind in der DB vorhanden.	Done	tdrohrer	#E5
24	Prototyp Webservice	Prototyp für Webservice erstellen AK: - Webservice funktioniert	Done	taesterm	#E5
21	SQL-Script für das Erstellen der Tabellen	SQL Script um die benötigten Tabellen zu erstellen. AK: Die benötigten Tabellen werden gemäss ER-Modell korrekt erstellt.	Done	tdrohrer	#E5

18	Schnittstellen definieren	Schnittstellen zwischen den Layers / Tiers definiert und dokumentiert.	Done	taesterm	#E5
5	Ausstehende Mahnungen - Warnung	Sollte ein Kunde noch ausstehende Mahnungen haben, wird beim Erfassen einer Bestellung eine Warnung angezeigt. AK: Bei ausstehende Mahnungen wird die Warnung angezeigt.	Todo		#E1
6	Angelieferte Artikel im Filiallager erfassen	Datentypisten/innen können angelieferte Artikel aus dem Zentrallager im Filiallager erfassen. AK: Angelieferte Artikel werden korrekt in der DB eingetragen.	Todo		#E2
7	Ausstehende Lieferungen nach Materialeingang ausliefern	Nachdem ausstehende Artikel wieder im Filiallager verfügbar sind, müssen ausstehende Lieferungen ausgelöst werden. AK: Ausstehende Lieferungen werden korrekt ausgeliefert. Die Daten werden korrekt in der DB eingetragen.	Todo		#E2
11	Pro Kunde Bestell-Historie anzeigen	Pro Kunde können seine Bestellungen angezeigt werden. AK: Es werden die korrekten Bestellungen dieses Kunden dargestellt.	Todo		#E3

Produktanforderungen:

Lucerne University of
Applied Sciences and Arts

HOCHSCHULE
LUZERN

Technik & Architektur

Version	Datum	Wer	Was	Status
2.0	11.2.11	Jörg Hofstetter	Angepasst für Durchführung 11	freigegeben
2.1	23.2.11	Jörg Hofstetter	Kleine Anpassungen	freigegeben
2.2	9.1.13	Jörg Hofstetter	Multi-User Betrieb	freigegeben
2.3	17.2.14	Jörg Hofstetter	Kleine Anpassungen für FS14	Entwurf

Produktanforderungen
Filialen-Bestellsystem

Unser Kunde hat ein Informatik-System für ein elektronisches Bestellsystem in seinen dezentralen Filialen bestellt. Dieses wird durch Mitarbeiter/innen in mehreren Filialen bedient. Jede Filiale hat eine lokale Datenbank. Wichtige Anforderungen:

- Mit dem System sollen Bestellungen für ein oder mehrere Produkte, welche per Telefon, Schalterkontakt, Fax, Email oder Brief eingehen, durch das Verkaufspersonal einfach erfasst und gespeichert werden.
- Für auszuführende Bestellungen wird eine Rechnung erzeugt. Die Buchhaltung inkl. Mahnwesen für die Rechnungserstellung/-behandlung erfolgt in einer separaten RW-Applikation (Rechnungswesen RW) auf der lokalen DB. Eine Schnittstelle zu dieser Applikation ist zu spezifizieren und es ist aufzuzeigen, wie das realisierte System sie in einer Erweiterung nutzen kann. Die RW-Applikation ist nur bezüglich Schnittstelle anzudenken. Das DB-Schema soll auch die RW-Daten beinhalten.
- Kunden erhalten eine Bestellbestätigung, welche insbesondere den Liefertermin und den Preis beinhaltet. Für diese Bestätigung müssen nur entsprechende DB-Daten aktualisiert werden (kein mailing!).
- In der lokalen Datenbank wird insbesondere der Lagerbestand der Filiale verwaltet. Ist ein Artikel im Filiallager nicht mehr vorhanden, kann auf das zentrale Lager zurückgegriffen werden. Dort ist ein bestehendes System (Legacy System) vorhanden, welches über eine zur Verfügung gestellte Komponente einzubinden ist (erhalten Sie zu gegebener Zeit). Im Zentralen Lager sind die Artikel entweder sofort lieferbar oder müssen nachbestellt werden.
- Die Filiale kann nur Produkte verkaufen, welche zumindest im zentralen Lager vorhanden sind (Sortiment). Sie kann aber selber entscheiden, welche sie tatsächlich verkaufen will.
- Angelieferte Artikel im Filiallager werden durch Datentypisten/innen im System eingetragen.
- Die Bestellungen und ihr Zustand müssen jederzeit einsehbar sein. Bestellungen können während der Eingabe annulliert werden.

- Vorhandene Produkte können in Listen dargestellt werden. Es sind insbesondere folgende Attribute darstellbar:
  - Bezeichnung
  - Artikel-Nummer
  - Preis
  - Anzahl (lokal) vorhandener Exemplare
- Unterschreitet ein Produkt im Filiallager eine einstellbare Minimalmenge, muss aus dem Zentrallager automatisch nachbestellt werden.
- Für jede Bestellung wird im Minimum festgehalten:
  - Produkte
  - Preis
  - Verkäufer/in
  - Datum/Uhrzeit
  - Kunde
- Das System unterscheidet unterschiedliche Benutzergruppen mit entsprechenden Rechten: SysAdmin, Filialleiter/in, Verkäufer/in, Datentypist/in.
- Gleichzeitig können mehrere Benutzer/innen einer Filiale auf das System zugreifen. Dabei dürfen keine inkonsistenten Zustände entstehen (Bsp.: gleiches Exemplar für mehrere Personen „gebucht“).
- Die erfolgten Bestellungen, Nachbestellungen und Lieferungen aus der Zentrale sind für den/die Filialleiter/in jederzeit einsehbar.
- Gibt ein Kunde eine Bestellung auf und dieser hat ausstehende Mahnungen, erfolgt für das Verkaufspersonal eine Warnung.

APPE Projektabwicklung:

APPE Projektabwicklung FS 14

Version	Datum	Wer	Was	Status
V1.0	9.2.2013	Jörg Hofstetter	Test und Integration	freigegeben
V1.1	10.2.14	Jörg Hofstetter	Anpassungen betr. SoDa	freigegeben
V1.2	12.2.14	Jörg Hofstetter	Anpassungen in SWE Deliverables, Use Case „Bestellung“ explizit verlangt in Umsetzung.	freigegeben

1. Allgemeines

Termine: Siehe separat abgegebener Semesterplan für das APPE-Modul.

Für alle Gruppenmitglieder obligatorisch (Testatbedingung):

- Teilnahme an Sprint-Reviews gemäss sep. Semesterplan.
- Test und Integration: Am Ende der Sprints ist eingecheckter/getaggter integrierter Code (mit Testfällen auf dem Buildserver, ohne Fehler baubar und ausführbar) vorhanden Mit den Fachcoaches PRG wird insb. Sprint-Review 4 durchgeführt.
- Erstellen Rahmenplan mit Meilensteinen, Sprint-Planung, Risikoanalyse.
- Use Case Beschreibungen (sollten im ersten Iterations-Review vorliegen)
- Teilnahme am Architektur Review
    - Architektur-Skizze(n) auf Papier mitnehmen.
        - Klare Zuordnung der eigenen Klassen zu einer Schicht.
        - Datenfluss über Schichtgrenzen hinweg aufzeigen.
        - Verantwortlichkeiten der Klassen sind klar
- DB-Sitzung: Besprechen des konzeptionellen Datenmodelles (ER-Diagramm) mit dem Fachcoach DB. Notation: siehe technische Vorgaben DB.
- Schluss-Präsentation: Details werden noch bekannt gegeben. Es werden basierend auf den Review Themen zur Vorstellung festgelegt.
- Abgabe der geforderten Dokumente (pdf). Allfällige, verlangte Nachbesserungen der Schlussdokumentation und/oder der Software werden bis spätestens 2 Wochen nach Ende Semester abgeliefert.

2. Anforderungen

Siehe sep. Dokument „APPE Produktanforderungen“

3. Technische Vorgaben

Einzusetzende Programmiersprache: **Java**
Mindestens die Schnittstellen sind mit JavaDoc zu dokumentieren.

Grundarchitektur - Minimalanforderungen:

- Erstellen eines Rich-GUI-Clients in einer mehrschichtigen Architektur (Multi-User fähig).
- Für die Realisierung des GUI's kann z.B. SWING eingesetzt werden.
- Anbindung an die Datenbank via EclipseLink (Java Persistence Interface).
  Schwerpunkt: Es sind klar getrennte Schichten (Layer gemäss Architektur-Input) zu realisieren. Sollte es sich im Laufe des Projektes zeigen, dass diese Schichten an bestimmten Stellen verletzt werden müssen, ist dies zu begründen und zu dokumentieren.

Überlegungen zu folgenden Fragen sind explizit zu dokumentieren:
Wie könnte die umgesetzte Anwendung:

- robuster gemacht werden?
- die Performance und Skalierung verbessert werden?
- Datenschutz (Berechtigungs-Kontrolle) gewährleistet werden?

Zur Robustheit gehört insbesondere **Transaktionsschutz**.

Datenbank:
Alle langlebigen Daten müssen in einer DB gespeichert werden. Sie werden dazu pro Gruppe eine im EnterpriseLab zentral installierte MySQL-Server Instanz erhalten. Diese Datenbank verwenden Sie auch für sämtliche Tests, was bei lokalen Entwicklertests eine Koordination unter den Teammitgliedern erfordert.

Externe Schnittstelle zum zentralen Lager:
Für den Zugriff auf das Lagersystem wird Ihnen zu gegebener Zeit eine Komponente mit einem wohldefinierten Interface zur Verfügung gestellt. Im Wesentlichen wird die Komponente folgende Funktionen aufweisen:

- Abfrage des Lagerbestandes eines Artikels
- Artikeln für Bestellung reservieren (optional)
- [Reservierte] Artikeln bestellen (Garantiert wenn Reserviert)
- Reservation freigeben

Damit Sie unabhängig arbeiten können, beinhaltet diese Komponente eine interne Simulation und ist unabhängig von einer zentralen Serverinfrastruktur.

4. Deliverables

Im Folgenden werden die Deliverables für die Bereiche Software Engineering (SWE), Datenbanken (DB) und Programmierung (PRG) aufgeführt:

SWE:
Die vorgegebenen Produktanforderungen sind einzuhalten, allerdings sind darin nicht alle Details festgehalten, die Teams haben eine gewisse Freiheit.
Es müssen insbesondere folgende Artefakte gemäss SoDa-Vorgaben als pdf-Dokumente abgegeben werden:

- SysSpec insb. mit:
  - Kontext-Diagramm
  - Übersichtsdiagramm aller Use Cases, textuelle Beschreibung der wichtigsten Use Cases, mit Schwergewicht auf dem Use Case „Bestellung ausführen“.
  - Backlog:
    Siehe dazu Folie „Stories in APPE“
    Vorgegebene Epic: „Als Verkäufer/in kann ich mehrere Produkte aus einem Produktkatalog auswählen um damit eine Bestellung auszulösen.„
  - Dokumentation der Backlog-Items (Epic, Story, Akzeptanz-Kriterien Ergänzungen)
    Hier ist z.B. ein Link auf das eingesetzte Unterstützungs-Tool (welches die Daten beinhaltet) und ein integrierter EXCEL-Datenexport denkbar.
  - Konzeptionelles Datenmodell (siehe DB)
  - Architektur-Modell(e) und Architekturentscheide (nachvollziehbar)

- PMP (Project Management):
  - Rahmenplan mit wichtigen Meilensteinen und Sprints
  - Testplan: Testfalldesign und Testfälle (mit Referenzen auf Epics/Stories)
  - Risiken

- Testprotokoll (minimal: Referenz auf Testfall, Version der Software unter Test, Tester, Datum, Zusammenfassung der Testresultate / Reports / Statistiken, Test Log)

PRG:

- Ein lauffähiges Programm welches:
  - In den vorgegebenen Modulen in Subversion verwaltet wird
  - Auf dem Buildserver laufend integriert wurde
  - automatisierte Tests ohne Fehler durchläuft
  - eine begründete minimale Codeabdeckung erfüllt
- Testdokumente gemäss oben
- Eine Deploy- bzw. Installationsanleitung

Logger: Die Überwachung der Anwendung während des produktiven Betriebes gewährleisten Sie durch den Einsatz eines Logging-Frameworks. Sämtliche Exceptions müssen geloggt werden und im Normalbetrieb sollte pro wesentlichen Business-Vorgang (Bestellung, Abfrage etc.) ein Logger-Eintrag produziert werden. Zusätzlich können Sie Ihre Anwendung mit ein-/ausschaltbaren Debug-Einträgen ergänzen, um eine allfällige Fehlersuche zu vereinfachen.

DB:

Die DB muss vollständig sein. Sie muss auch Informationen für das Rechnungswesen und zur Bestellbestätigung enthalten (Zusatzinfo).

Mit einem *logischen* ER-Diagramm muss die vollständige DB dargestellt werden. Auch für die Zusatzinfo sind die entsprechenden Entitäten/Relationships zu realisieren (sie bleiben leer). Die Beziehungstypen (Kardinalitäten) sind mit 1, c, m, mc darzustellen.

5. Projektabwicklung

5.1. Projektgruppen

Alle Gruppenmitglieder müssen in die Softwareerstellung involviert sein, die Verantwortlichkeiten sind in der Projekt-Dokumentation aufzuzeigen. Die Gruppeneinteilung und Organisation erfolgt autonom durch die Studierenden.

Es ist ein iteratives, inkrementelles Vorgehen gemäss SoDa umzusetzen.

5.2. Planung

Umfang der Umsetzung:
Es ist Aufgabe der Gruppen, in Absprache mit den Coaches einen vernünftigen Projekt-Umfang festzulegen, d.h. Schwerpunkte zu definieren.
Minimalanforderungen ist, dass der vorgegebene Use Case „Bestellung ausführen“ über alle Schichten hinweg umgesetzt wird („Durchstich“), d.h. alle wesentlichen technischen Aspekte (inkl. Transaktionsmanagement) müssen berücksichtig werden.
Eine einfache Erweiterung des „Durchstiches“ soll stets Ziel sein. Kann bei der Umsetzung weiterer Use Cases die Software nicht auf allen Schichten gleich weit vorangetrieben werden, ist dies sauber zu dokumentieren und darauf zu achten, dass die einzelnen Teile trotzdem testbar sind.
Aus der Projektdokumentation muss klar ersichtlich sein, was ursprünglich geplant und schlussendlich umgesetzt wurde.

> **Schnittstellen nach Aussen**: Allfälliges Drucken, Versenden von Emails, Versenden von Material, Schnittstelle zum Buchhaltungssystem etc.: Diese Aspekte sind zwar anzudenken, zu spezifizieren und entsprechende Software-Rümpfe zu erstellen, diese müssen aber nicht ausprogrammiert werden (siehe dazu Mock-Objekte in Wikipedia).