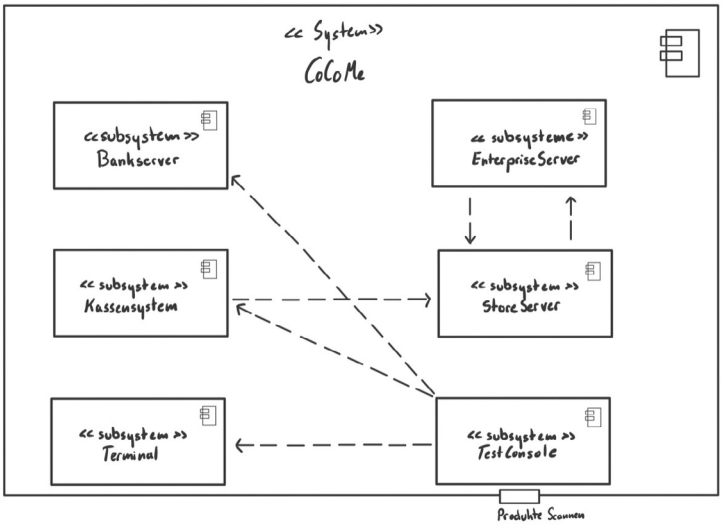
CoCoMe
Dokumentation

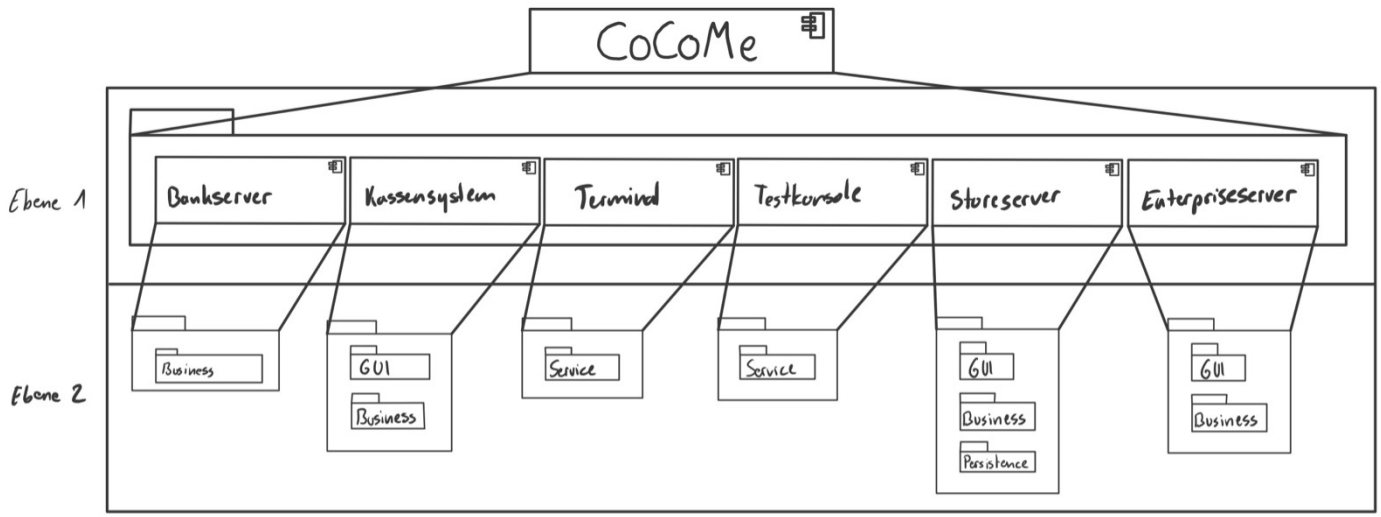
Inhalt

Bausteinsicht

Whitebox Gesamtsystem



Subsysteme	*Kurzbeschreibung*
Kassensystem	Das Kassensystem ist so zu sagen, der Bildschirm des Systems bei dem man die Produkte angezeigt bekommt.
Terminal	Das Terminal zeigt alles, was durch das Kassensystem entsteht und verarbeitet wird und gibt darüber eine Übersicht im Terminal an.
Testkonsole	Die Testkonsole ist der Verbindungspunkt des Kassensystems, dem Terminal und dem Bankserver. Hier sind Komponenten wie der Scanner oder die Cashbox enthalten. Sie ist sozusagen der Server hinter dem Kassensystem.
Enterpriseserver	Der Enterpriseserver ist dafür da, dass der Store Manager einen Delivery Report bekommt um sich eine Übersicht über die Bestellungen zu verschaffen.
Storeserver	Beim Storeserver kann der Manager neue Produkte bestellen und sein vorhandes Inventar sehen. Außerdem kann man einen Bericht über das Inventar erstellen.
Bankserver	Der Bankserver ist da um die Optionen zum Bezahlen mit Karte bereitzustellen und die Validierung der Karte durchzuführen.



Bausteinsicht

1.1.1 Ebene 1

In der ersten Ebene werden kurz die einzelnen Komponenten beschrieben die einen Teil des gesamten Projekts darstellen und welche Zusammenhänge diese besitzen.
Das ganze System basiert übrigens auf .Netcore 6 im Zusammenhang mit C#.

Bankserver
Der Bankserver existiert damit man in dem Kassensystem die Möglichkeit besitzt etwas mit Karte zu bezahlen. Hierbei wird die Validierung der Karte durchgeführt um zu überprüfen, dass alles stimmt und gültig ist. Der Bankserver arbeitet mit den Komponenten Kassensystem, Terminal und der Testkonsole zusammen, da diese immer auf die Validierung der Karte zurückgreifen können müssen. Eine Ebene weiter beim Bankserver würde die Businesslogik stehen, welche wiederum Komponenten wie Validierung, Contracts oder die Funktion enthält das Banking zu mocken.

Kassensystem
Das Kassensystem umfasst, wenn man eine Ebene runter geht, die nötigen Elemente um die GUI für die Kasse, sowohl für den Kassierer als auch für den Kunden, bereitzustellen. Zudem noch die Logik für den Barcodescanner Service. Das Kassensystem arbeitet eng mit dem Storeserver zusammen und holt sich die wichtigen Informationen aus dem Storeserver.

Terminal
Das Terminal ist nur dafür da um die Vorgänge die im Kassensystem ausgelöst werden zusammenzufassen und in einer Übersicht im Terminal zu zeigen, damit man nachverfolgen kann welche Schritte gerade betätigt wurden.

Testkonsole
Die Testkonsole ist hier sozusagen das Backend des Kassensystems. Hierbei enthält die Testkonsole die Businesslogik der Kasse und beinhaltet somit die verschiedenen Services die angeboten werden.

Storeserver
Der Storeserver besteht in der zweiten Ebene aus dem Storeclienten(GUI), der Businesslogik und den Daten, eingespeichert in eine Datenbank. Der Storeserver kommuniziert sowohl mit dem Kassensystem um Daten zu liefern und Änderungen zu bekommen als auch mit dem Enterpriseserver um einen Delivery Report zu erhalten.

Enterpriseserver
Der Enterpriseserver enthält die Businesslogik für die Kommunikation der Storeserver untereinander. Außerdem besitzt er noch eine Benutzeroberfläche, bei der mna bestimmte Sachen triggern/anfordern kann.

1.1.2 Ebene 2

In der zweiten Ebene werden die einzelnen Projekte beschrieben und welche weiteren Ebenen, Architekturen oder Interfaces benutzt werden.

Bankserver
Im Bankserver sitzt die Logik für den Karten-Mock. Hierbei wird die Bezahlung mit einer Karte nachgestellt um diese bei dem Kassensystem anbieten zu können. Somit sind die Packages BankingMock, Bankerver und Contracts enthalten.

Kassensystem
Das Kassensystem enthält die Businesslogik, Schnittstellen und die GUI. Die GUI wird mit der MVC Architektur und Angular erstellt und gibt die Ansicht für den Kunden als auch für den Kassierer/in zurück.

Bausteinsicht

Terminal
Im Terminal enthalten ist die einfach Logik für das Ausgeben der Informationen.

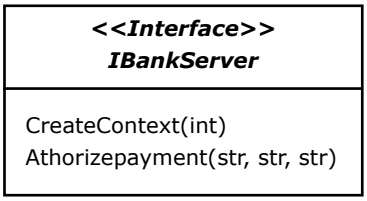
Testkonsole
In der Testkonsole besteht die wichtigste und meiste Logik des gesamten Kassensystems. Dort enthalten sind die Packages, BankServer, verschiedene Services und ein Package für Controller.

Storeserver
Der Storeserver enthält sowohl die komplette Businesslogik des Stores als auch den Storeclient(GUI), der mit Razor Pages oder der mVC Architektur umgesetzt wird. Er besitzt Packages wie Data, in dem die Datenbanken erstellt werden, Migrations, Models und Pages, die für den Clienten vorhanden sind, falls sich für Razor Pages entschieden wird.

Enterpriseserver
Der Enterpriseserver kann auch sowohl mit Razor Pages oder der MVC Architektur umgesetzt werden, wenn man sich entschließt ein Frontend mit einzubauen. Die Logik des Servers wird sonst in Controller-Klassen oder einfachem C#-Klassen umgesetzt.

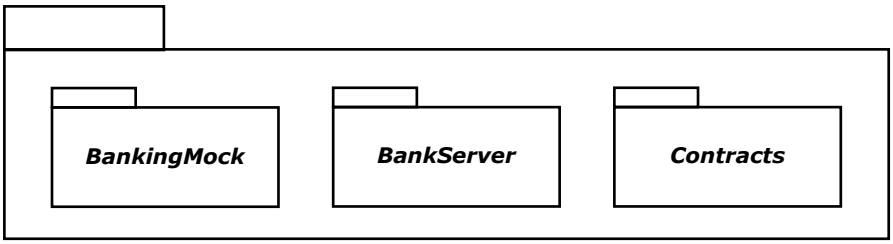
1.2 Bankserver

Bankserver-Interfaces



Methode	***Kurzbeschreibung***
CreateContext	Erstellt den Transaktions-Context bei dem die Summe des Einkaufs mitgegeben wird.
Authorizepayment	Autorisiert die Bezahlung indem der Token, Account und die passende ID mitüber-geben werden.

Bankserver-Packages



Package	***Kurzbeschreibung***
BankingMock	Ist für den Mock der Bank da und verantwortlich. Die Klassen TransactionContext.cs und BankServer.cs sind hier enthalten.
BankServer	Enthäkt Klassen Dtos.cs und Provider.cs
Contracts	Hier ist das Interface und verschiedene Exception-Klassen enthalten.

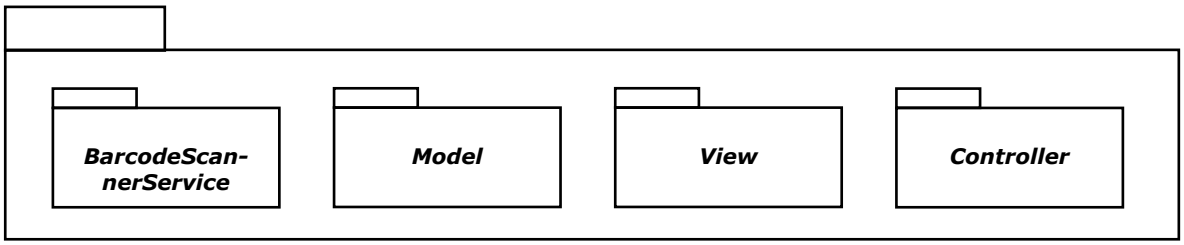
1.3 Kassensystem

Kassensystem-Interfaces

<<Interface>>   ***IBarcodeScanner***
ListenToBarcodes()

Methode	***Kurzbeschreibung***
ListenToBarcodes	Methode um den BarCodeScanner zu mocken.

Kassensystem-Packages



Package	***Kurzbeschreibung***
BarcodeScanner-Service	Enthält den Client und das Interface für den BarcodeScanner.
Model	Enthält alle Models für die MVC Architektur.
View	Enthält alle View für die MVC Architektur. Die Views enthalten kaum Logik.
Controller	Enthält alle Controller für die MVC Architektur. Hier ist die meiste Logik zu finden.

1.4 Terminal

Terminal-Interfaces

<<Interface>>   ***ICardReaderService***
Authorize(int, byte [], token)   Confirm()   Abort(string)

<<Interface>>   ***IPrintingService***
PrintLine(string)   StartNext()

<<Interface>>   ***IBarcodeScannerService***
ListenToBarcodes()

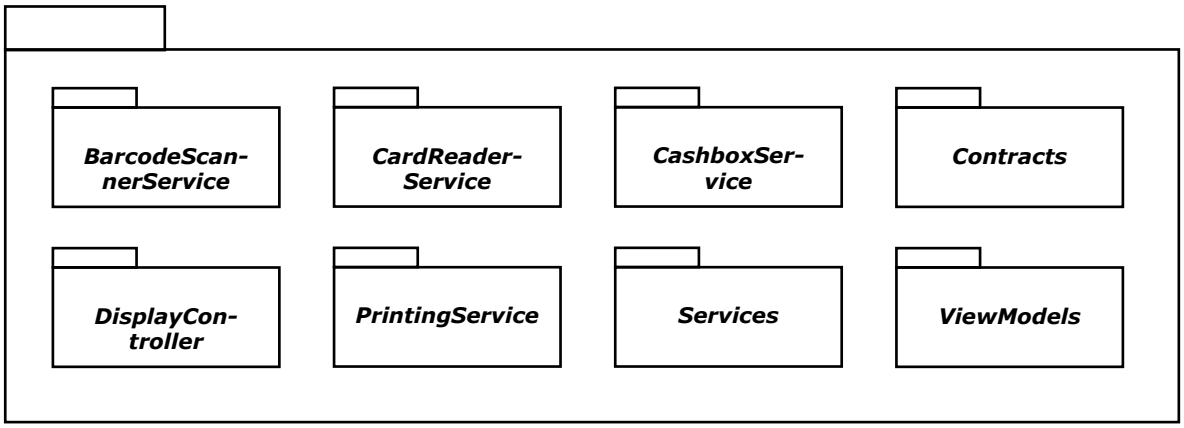
<<Interface>>   ***ICashboxService***
ListenToCashdeskButtons()

<<Interface>>   ***IDisplayController***
SetDisplayText(string)

Bausteinsicht

Interface	***Kurzbeschreibung***
ICardReaderService	Interface enthält Methoden um die „Karte" zu autorisieren und zu bestätigen oder den Vorgang abzubrechen.
IPrintingService	Interface mit Methoden die zum printen von Zeilen da sind.
IBarcodeScanner-Service	Interface, was die Listener-Methode enthält um den Scanner zu benutzen.
ICashboxService	Diese Listener-Methode in dem Interface ist dafür da um zu schauen wann die Kassen Knöpfe gedrückt wurden und das weiterzuleiten.
IDisplayController	Methode um den Text zu setzen der auf dem Display anezeigt wird.

Terminal-Packages



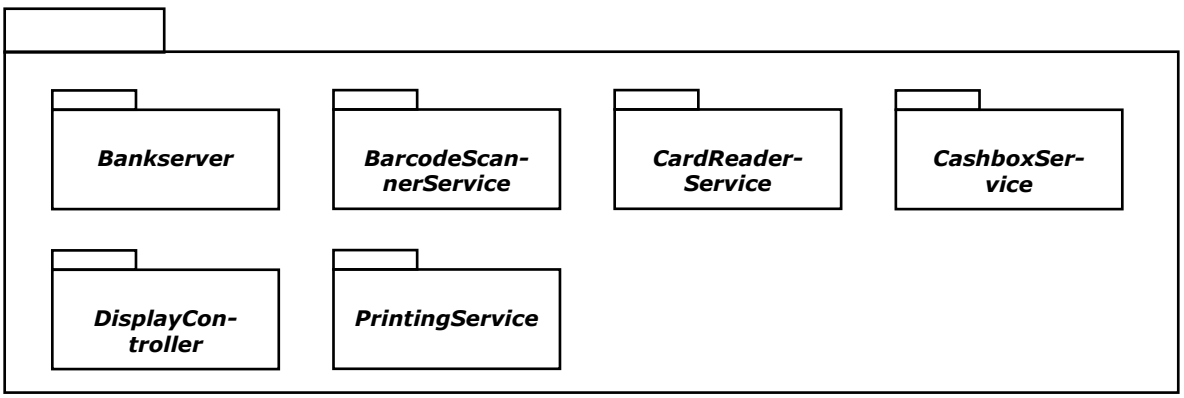
Package	***Kurzbeschreibung***
Service-Ordner	Die verschiedenen Service Ordner bieten einfach unterschiedliche Methoden an um die Informationenn im Terminal darzustellen.
DisplayController	Enthält die Controller für den Display.
ViewModels	In diesem Package sind die Models drin, die von den anderen Klassen genutzt werden können.

1.5 Testkonsole

Testkonsole-Interfaces

ICardReaderService, IPrintingService, IBarcodeScannerService, IBankServer, ICashboxService, IDisplayController sind alle Interfaces die in der Testkonsole enthalten und verwendet werden. Für genauere Beschreibungen bitte in den vorherigen Beschreibungen der Interafces schauen, da Sie die gleichen Methoden beinhalten.

Testkonsole-Packages



Bausteinsicht

Alle Packages die in der Testkonsole enthalten und verwendet sind auch in anderen Projekten vorhanden. Für genauere Beschreibungen bitte in den vorherigen Beschreibungen der Packeges schauen.

1.6 StoreServer

StoreServer-Interfaces

<<Interface>> ***IKassensystem***
getItemById(int) completeSale([{product,int}])

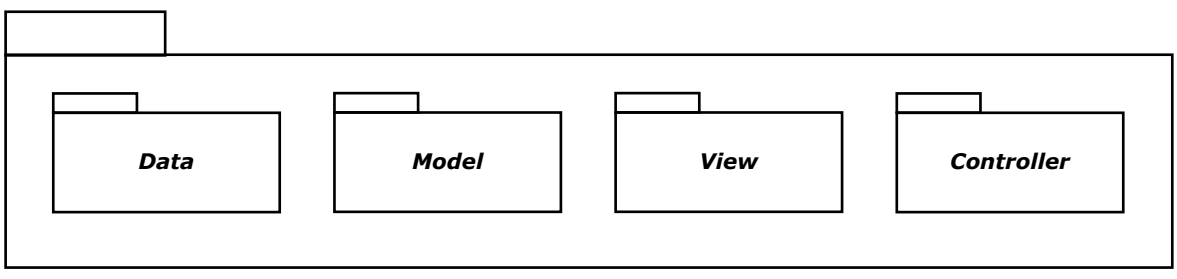
<<Interface>> ***IEnterpriseServer***
generateDeliveryReport(int)

<<Interface>> ***IStoreClient***
triggerDataPushToEnterprise() incomingDelivery(list<product>, int) deliveryRequest(list<product>, int)

<<Interface>> ***IStoreServer***
getAvailableProducts() changePrice(product) generateStockReport(int) rollInReceived(int) showOutOfStock()

Methode	***Kurzbeschreibung***
getItemById	Methode um das Produkt zu bekommen, dessen ID du mitgegeben hast. Sucht nach der ID. Das Interface mit dieser Methode wird von dem StoreServer bereitgestellt.
completeSale	Methode um den Verkauf zu erfassen und die wichtigen Informationen zubekommen. Wird von StoreServer bereitgestellt.
generateDeliveryReport	Dieses Interface dient dazu um einen Bericht zu bekommen über alle Bestellungen des Servers. Dieser wird sich vom Enterprise Server generiert und bekommen.
triggerDataPushToEnterprise	Bei dem Interface IStoreClient mit dieser Methode wird, durch einen trigger ausgelöst, die Daten von einem StoreServer erwartet, welcher diese dann an den Enterprise Server weiterleitet.
incomingDelivery	Dieses Interface wird genutzt um sich die eintreffenden Bestellugen anzeigen zu lassen im Client.
deliveryRequest	Die Methode ist für Bestellungen da, die betätigt werden müssen.
getAvailableProducts	Hierbei werden sich alle Produkte geholt die zur Zeit vorhanden sind.
changePrice	Dabei wird die Methode bereitgestellt, welche dem Store ermöglicht die Preise der einzelnen Produkte zu verändern.
generateStockReport	Bei dieser Methode wird im StoreClient etwas getriggered und gibt dann einen Bericht zurück über das vorhandene Inventar.
rollInReceived	Bei dieser Methode wird gezeigt, dass die Bestellung eingegangen ist im Store.
showOutOfStock	Dabei werden alle Produkte angezeigt, die nicht mehr vorhanden sind im Store.

StoreServer-Packages

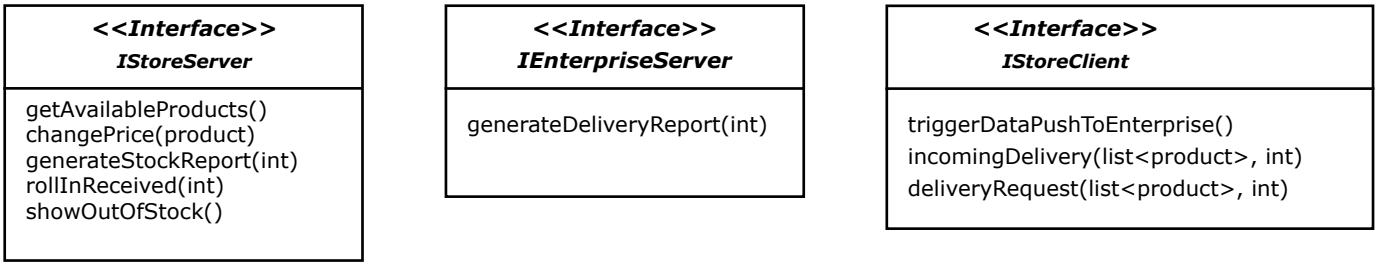


Bausteinsicht

Package	***Kurzbeschreibung***
MVC	Normale MVC Packages mit den verschiedenen Models, Views und Controllern in den passenden Packages.
Data	In diesem Package werden die Daten für die Datenbank und die Datenbank selbst erstellt.

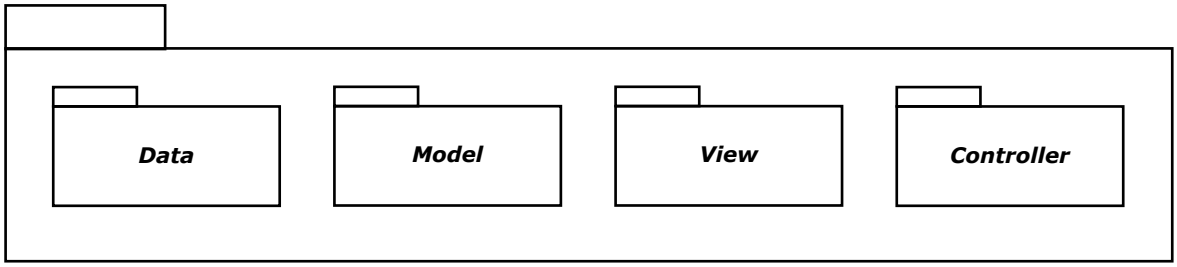
1.7 EnterprsieServer

EnterpriseServer-Interfaces



Alle Interfaces die in dem EnterprsieServer enthalten und verwendet werden wurden schon zuvor beim StoreServer erklärt. Für genauere Beschreibungen bitte in den vorherigen Beschreibungen der Interafces schauen.

EnterpriseServer-Packages



Alle Packages die in dem EnterpriseServer enthalten und verwendet werden sind auch in anderen Projekten vorhanden. Für genauere Beschreibungen bitte in den vorherigen Beschreibungen der Packages schauen, wie zum Beispiel dem StoreServer.