# Ausbaustufe 2 – SWK5

(Abgabe: 22. 12. 2022, 24:00 Uhr)

(A2.1) *CaaS.Core:* Implementieren Sie auf Basis der in Ausbaustufe 1 entwickelten Datenzugriffsschicht die gesamte Funktionalität der Komponente *CaaS.Core*. Überlegen Sie sich, wie die Klienten auf die (Geschäfts-)Logik zugreifen sollen. Fassen Sie die erforderlichen Funktionen zu logischen Grup- pen zusammen und definieren Sie für jede Gruppe ein Interface. Alle REST-Services sollen (sta- tisch) ausschließlich von diesen Geschäftslogik-Interfaces abhängig sein.

Entwerfen Sie die Klassen der Geschäftslogik mit besonderer Sorgfalt und legen Sie besonderen Wert auf eine flexible Software-Architektur. Sorgen Sie insbesondere dafür, dass Komponenten des Rabatt- und Bezahlsystems einfach ausgetauscht werden können. Implementieren Sie alle Funktionen, die nicht unmittelbar für die Umsetzung der REST-Schnittstelle oder des Web-Clientsbenötigt werden, in der Geschäftslogikkomponente von *CaaS.Core*.

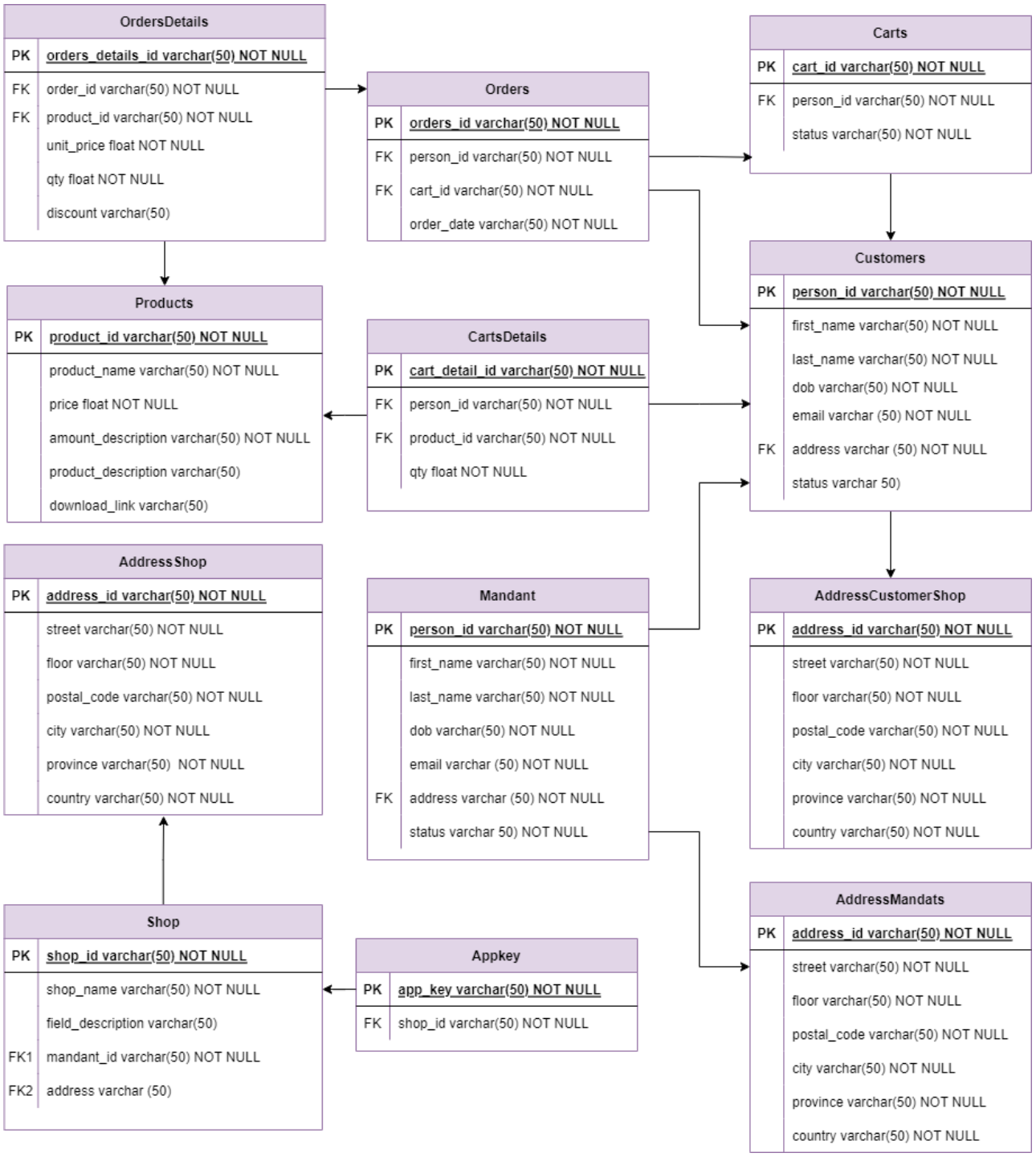
Die zentralen Bestandteile von *CaaS.Core* stellen die Rabattregeln und das Bezahlsystem dar. Überlegen Sie sich, wie Sie die korrekte Funktionsweise dieser Komponenten testen und demonst-rieren können.

(A2.2) *CaaS.Api:* Diese Systemkomponente ist mit ASP.NET als REST-basiertes Web-Service zu realisieren. Die Anforderungen an die zu exportierende Funktionalität des Web-Service ergeben sich aus der Funktionalität von *CaaS.Web*.

(A2.3) *Automatisierte Tests:* Erweitern Sie Ihre Test-Suite um Tests der Geschäftslogik. Die Tests sollen so weit wie möglich die gesamte Funktionalität der Anwendung abdecken.

Isolieren Sie in Ihren Tests die Geschäftslogik von der Persistenzschicht. Ersetzen Sie dazu Ihre DAOs durch Mock-Objekte. Setzen Sie dafür ein Mocking-Framework ein.

(A2.4) *Dokumentation:* Führen Sie die notwendigen Ergänzungen in der Dokumentation durch. Aus IhrerDokumentation soll die Architektur des Gesamtsystems hervorgehen, insbesondere die Struktur der Klassen und Interfaces und die Kopplung der Systemkomponenten sollte übersichtlich dargestellt werden. Hierzu eignen sich besonders gut UML-Klassen- und Paketdiagramme.




Die Domänenklassen bleiben weiterhin unberührt. Von der ersten Ausbaustufe  weiterhin bis die zweite Ausbaustufe  wird sie verwendet. Im Folgenden werden die für die zweite Ausbaustufe relevante Projektfolder mit den darin enthaltenen Datei aufgelistet

# CaaS.Api

Der Aufbau von meiner Übung lehnt sich stark an die Übung auf. Program.cs beinhalten die Konfiguration fürs StartUp, Datenbanken, Routing, Controller.

# CaaS.Api.Controllers

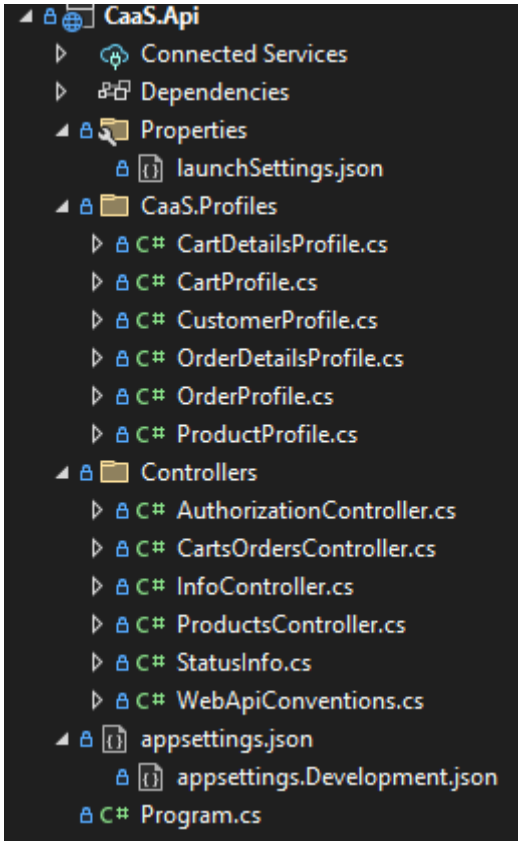
ProductController.cs
CartsOrderController.cs
StatusInfo.cs
WebApiConventions.cs

Beinhaltet die Kontrollstruktur zum Aufruf von Transferfunktionen anhand http Request/Response. Die Transferfunktionen sind z.B für den Transport von Daten über DTO zuständig.

# CaaS.Api.Profiles

CustomerProfile.cs
OrderProfile.cs
CartDetailsProfile.cs
OrderDetailsprofile.cs
Cartprofile.cs
Productprofile.cs

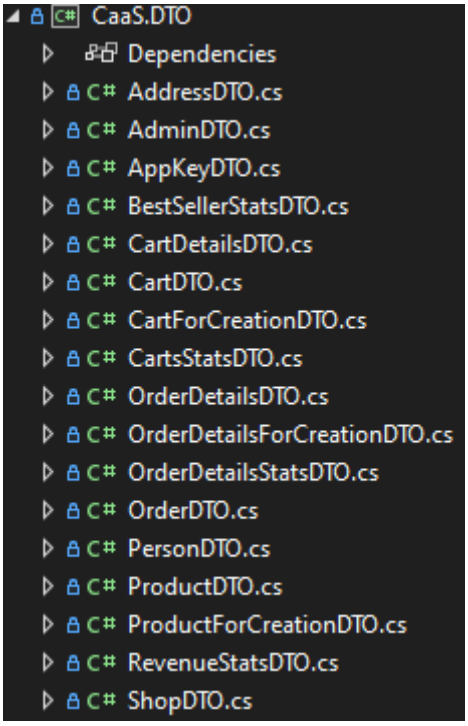
Profiles stellt das Mapping zwichen Domänenklassen und DTO Object



# CaaS.DTO

AddressDTO.cs
AppKeyDTO.cs
CartDetailsDTO.cs
CartDTO.cs
OrderDetailsDTO.cs
OrderDTO.cs
PersonDTO.cs
ProductDTO.cs
ShopDTO.cs

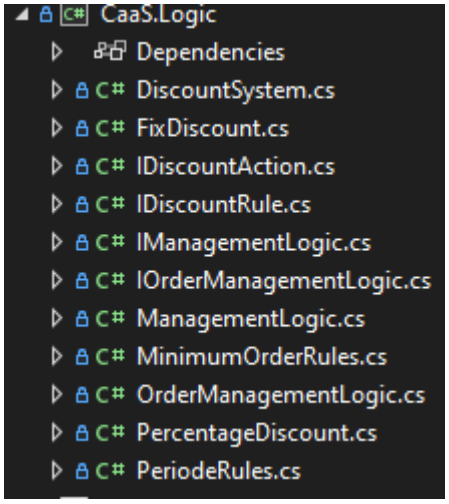
DTO spiegelt die Domänenklassen wieder und dient als Transportobjekt (Encapsulation Concept)



# CaaS.Logic

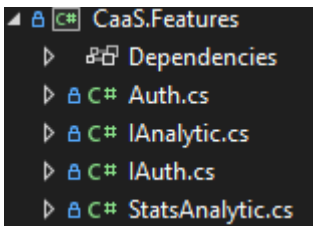
IManagementLogic.cs
IOrderManagementLogic .cs
ManagementLogic.cs
OrderManagementLogic.cs

Die Logik werden dem Controller im Konstruktor (Dependency Injection) übergeben. Es gilt genauso für das Rabattsystem. Das Projekt beherrbergt die Interfaces und Implementierungen der Logik, die fürController zuständig sind. Zudem wird das Rabattsystem in CaaS.Logic implementiert. Das Rabattsystem besteht aus Rabattregel(IDiscountRule) und Rabattaktion(IDiscountAction)



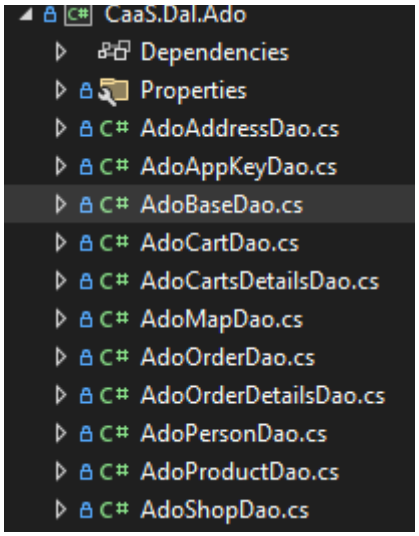

# CaaS.Features

Die Statistik werden im Order CaaS.Feature implementiert, diese dann dem InfoController.cs im Konstruktor(Dependency Injection) übergeben.
Ebenfalls gilt es für die Authorisierung über die Datei Auth.cs in CaaS.Features.



# Anpassung von erster Ausbaustufe

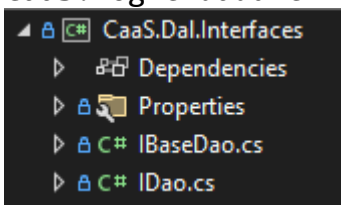
## CaaS.Dal.Ado




Hier wird eine neue Klasse AdoBaseDao, die das neue Interfaces IBaseDao implementiert entworfen.

## CaaS.Dal.Interfaces

Hier werden 2 neue Interfaces IBaseDao und IDao für das bessere Mappen AdoTDao entworfen. Es wird der Aufruf der grundlegenden Funktionen für die Datenbanken wie(FindById,Find, Get, Delete, Update) von anderem Projekt CaaS.Logic dadurch ermöglicht.




>Info: Die Domänenklassen für die Datenbanken sind im Folder EntityAlsExcelfürSSMS gespeichert. Verwendet wird SQL Server. Man kann die Exceldatei ( enthält die Daten der Domänenklassen ) über Microsoft SQL Server Management Studio  ins SQL Server bringen.