**Ausbaustufe 1 – SWK5 (Abgabe: 13. 11. 2022, 24:00 Uhr)**

Sämtliche Komponenten von *CaaS* sind mit den in der Lehrveranstaltung SWK5 behandelten Technologienzu realisieren. In dieser ersten Ausbaustufe benötigen Sie nur .NET und ein Datenbanksystem Ihrer Wahl.

(A1.1) *Datenbank/Datenmodell:* Alle auf der *CaaS*-Plattform erfassten Daten sind in einer relationalen Datenbank zu speichern. Entwerfen Sie ein strukturiertes (normalisiertes) Datenmodell für *CaaS*. Erstellen Sie eine Testdatenbank, welche für die Problemstellung repräsentative und ausreichendumfangreiche Daten enthält.

Stellen Sie auch sicher, dass sämtliche Operationen effizient ausgeführt werden können und IhreImplementierung auch imstande ist, Daten in einem realistischen Umfang zu speichern. Konkret sollte ihr System die Daten von zumindest zwei Shop-Betreibern speichern. Jeder Shop-Betreiberbietet zumindest 100 Produkte zum Kauf an und hat zumindest 100 Kund*innen. Jede Kund*in hatdurchschnittlich bereits 10 Bestellungen, die im Mittel 3 Produkte umfasst, getätigt. Bei mindes- tens 10 Kund*innen ist ein Warenkorb offen.

Verwenden Sie zum automatisierten Testen und für den Produktivbetrieb unterschiedliche Instan-zen Ihrer Datenbank.

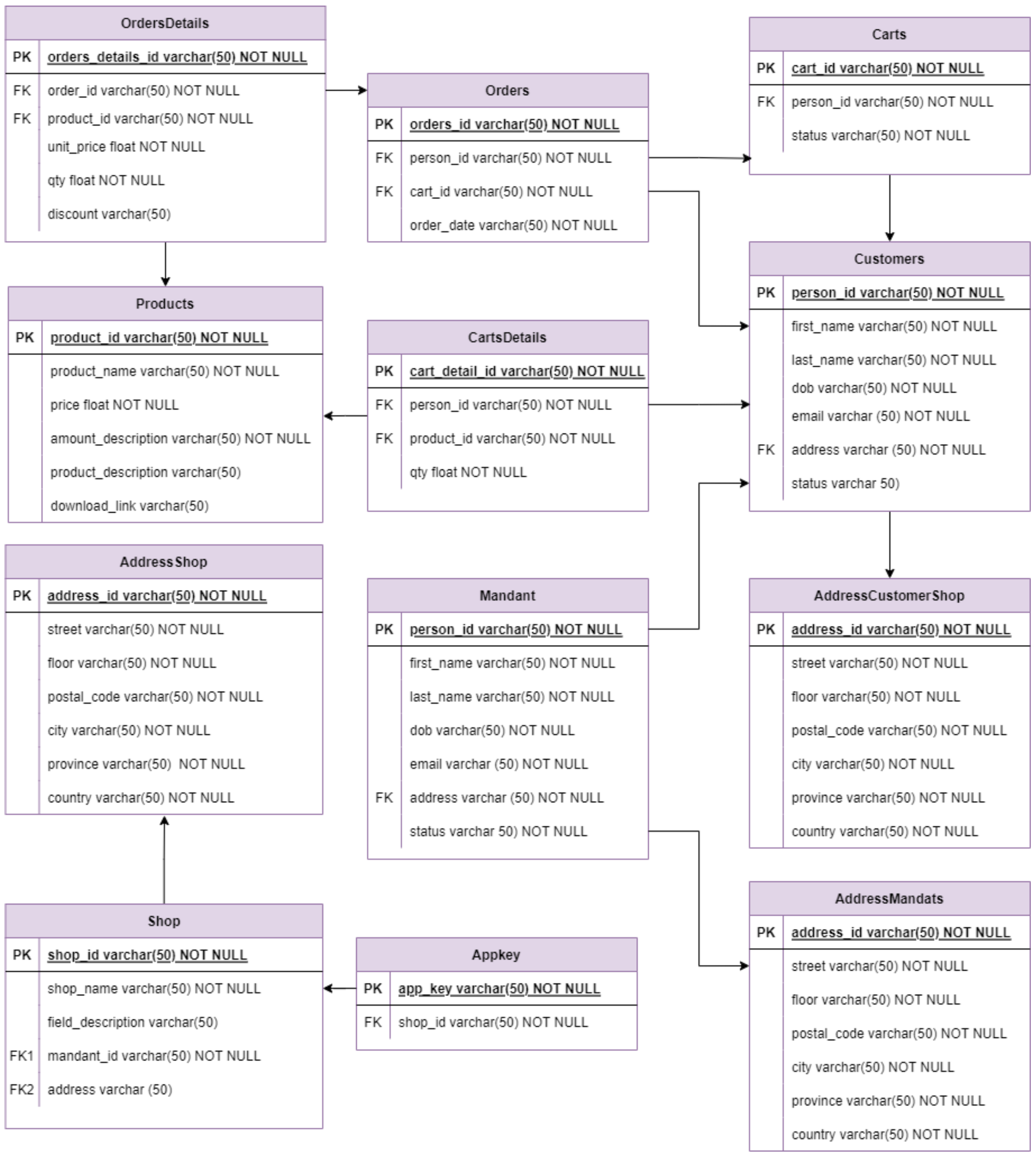
(A1.2) *Datenzugriffsschicht:* Überlegen Sie sich für die Entitäten Ihres Datenmodells (*Produkte, Waren- körbe, Kund*innen-Stammdaten,* etc.) die erforderlichen Zugriffsfunktionen und fassen Sie diese zu Interfaces zusammen. Die Geschäftslogik soll ausschließlich von diesen Interfaces abhängig sein.

Entwickeln Sie ein geeignetes Format für die Repräsentation der Daten. Definieren Sie dazu einfa-che Klassen (die primär aus Konstruktoren und Properties bestehen), welche zum Transport der Daten dienen. Diese Klassen werden häufig als *Domänenklassen* oder *Datentransferklassen* be- zeichnet. Im Wesentlichen werden Sie für jede Entität eine entsprechende Domänenklasse benö-tigen. Die Domänenklassen werden in den Interface-Methoden der Datenzugriffsschicht als Para-metertypen verwendet und müssen daher auch der Geschäftslogik bekannt gemacht werden.

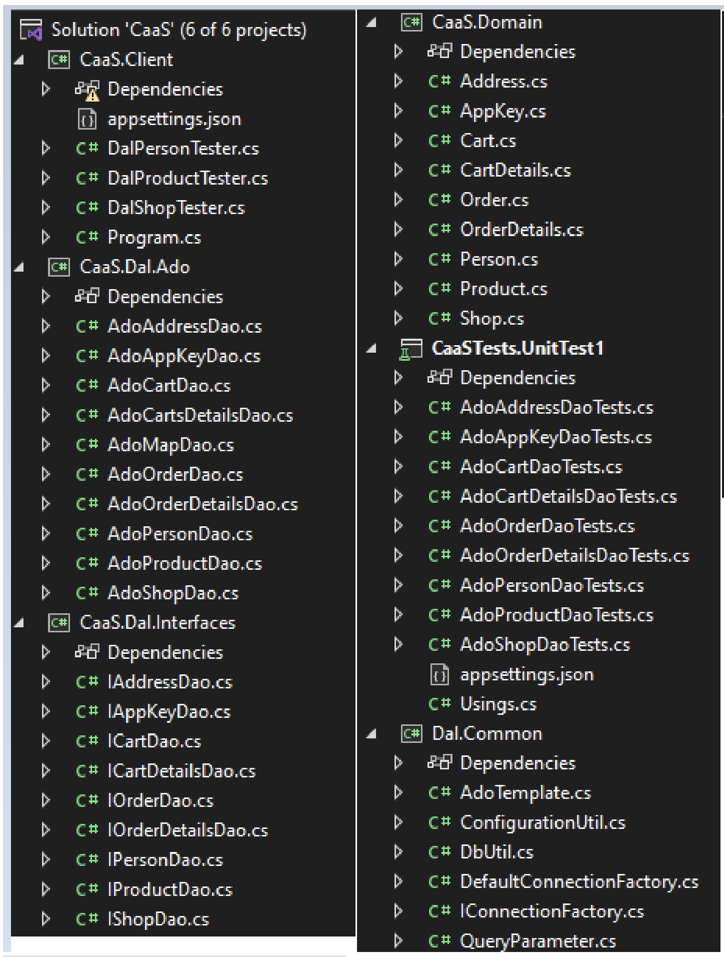
Implementieren Sie die Datenzugriffsschicht auf Basis von ADO.NET. Der Einsatz eines OR- Mapping-Werkzeugs (NHibernate, Entity Framework etc.) ist nicht gestattet.

(A1.3) *Unit-Tests:* Testen Sie die Datenzugriffsschicht ausführlich, indem Sie eine umfangreiche Unit- Test-Suite erstellen. Achten Sie auf eine möglichst große Testabdeckung. Erstellen Sie eher mehr,dafür aber kompakte Unit-Tests, die nach Möglichkeit nur einen Aspekt der Funktionalität über- prüfen. Die Wahl des Testframeworks steht Ihnen frei.

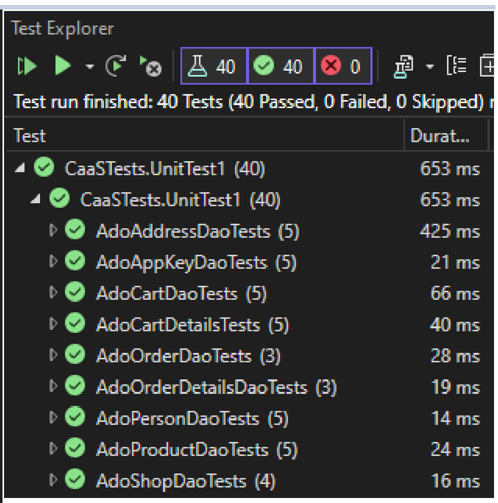
(A1.4) *Dokumentation:* Erstellen Sie eine übersichtliche, gut strukturierte Dokumentation. Ihre Doku- mentation sollte zumindest das Datenbankmodell und die Struktur der Datenzugriffsschicht ent- halten. Vermeiden Sie dabei die reine Aufzählung der verfügbaren Klassen, Methoden, etc. Doku-mentieren und begründen Sie Abwägungen und Entscheidungen, die Sie im Rahmen der Entwick-lung getroffen haben.


OrdersDetails
PK orders_details_id varchar(50) NOT NULL
FK order_id varchar(50) NOT NULL
FK product_id varchar(50) NOT NULL
unit_price float NOT NULL
qty float NOT NULL
discount varchar(50)
Orders
PK orders_id varchar(50) NOT NULL
FK person_id varchar(50) NOT NULL
FK cart_id varchar(50) NOT NULL
order_date varchar(50) NOT NULL
Carts
PK cart_id varchar(50) NOT NULL
FK person_id varchar(50) NOT NULL
status varchar(50) NOT NULL
Customers
PK person_id varchar(50) NOT NULL
first_name varchar(50) NOT NULL
last_name varchar(50) NOT NULL
dob varchar(50) NOT NULL
email varchar (50) NOT NULL
FK address varchar (50) NOT NULL
status varchar 50)
Products
PK product_id varchar(50) NOT NULL
product_name varchar(50) NOT NULL
price float NOT NULL
amount_description varchar(50) NOT NULL
product_description varchar(50)
download_link varchar(50)
CartsDetails
PK cart_detail_id varchar(50) NOT NULL
FK person_id varchar(50) NOT NULL
FK product_id varchar(50) NOT NULL
qty float NOT NULL
AddressShop
PK address_id varchar(50) NOT NULL
street varchar(50) NOT NULL
floor varchar(50) NOT NULL
postal_code varchar(50) NOT NULL
city varchar(50) NOT NULL
province varchar(50) NOT NULL
country varchar(50) NOT NULL
Mandant
PK person_id varchar(50) NOT NULL
first_name varchar(50) NOT NULL
last_name varchar(50) NOT NULL
dob varchar(50) NOT NULL
email varchar (50) NOT NULL
FK address varchar (50) NOT NULL
status varchar 50) NOT NULL
AddressCustomerShop
PK address_id varchar(50) NOT NULL
street varchar(50) NOT NULL
floor varchar(50) NOT NULL
postal_code varchar(50) NOT NULL
city varchar(50) NOT NULL
province varchar(50) NOT NULL
country varchar(50) NOT NULL
Shop
PK shop_id varchar(50) NOT NULL
shop_name varchar(50) NOT NULL
field_description varchar(50)
FK1 mandant_id varchar(50) NOT NULL
FK2 address varchar (50)
Appkey
PK app_key varchar(50) NOT NULL
FK shop_id varchar(50) NOT NULL
AddressMandats
PK address_id varchar(50) NOT NULL
street varchar(50) NOT NULL
floor varchar(50) NOT NULL
postal_code varchar(50) NOT NULL
city varchar(50) NOT NULL
province varchar(50) NOT NULL
country varchar(50) NOT NULL

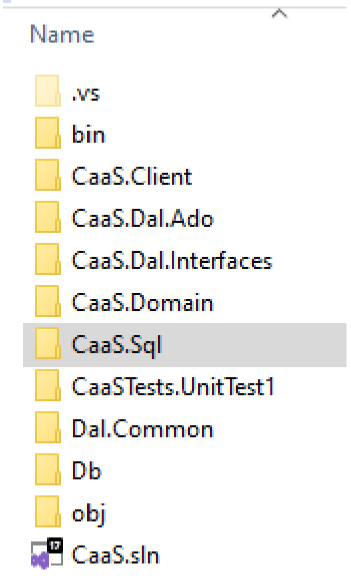
```
Der Aufbau meines Projekts lehnt sich stark an die Übung an
Die DomänenKlassen sind in CaaS.Domain zu finden. Dal.Common
beinhalten Files mit allgemeinen Hilfsfunktionen wie zum Settings,
Datenbankenverbindungen, ADO Template, usw
DAO Pattern wurde durch CaaS.Dal.Ado und CaaS.Dal.Interfaces
implementiert.
UnitTests für jeweilige DAO Pattern sind im CaaSTests.UnitTest1 zu
finden
```



```
Solution 'CaaS' (6 of 6 projects)
  CaaS.Client
    Dependencies
    appsettings.json
    DalPersonTester.cs
    DalProductTester.cs
    DalShopTester.cs
    Program.cs
  CaaS.Dal.Ado
    Dependencies
    AdoAddressDao.cs
    AdoAppKeyDao.cs
    AdoCartDao.cs
    AdoCartsDetailsDao.cs
    AdoMapDao.cs
    AdoOrderDao.cs
    AdoOrderDetailsDao.cs
    AdoPersonDao.cs
    AdoProductDao.cs
    AdoShopDao.cs
  CaaS.Dal.Interfaces
    Dependencies
    IAddressDao.cs
    IAppKeyDao.cs
    ICartDao.cs
    ICartDetailsDao.cs
    IOrderDao.cs
    IOrderDetailsDao.cs
    IPersonDao.cs
    IProductDao.cs
    IShopDao.cs
  CaaS.Domain
    Dependencies
    Address.cs
    AppKey.cs
    Cart.cs
    CartDetails.cs
    Order.cs
    OrderDetails.cs
    Person.cs
    Product.cs
    Shop.cs
  CaaSTests.UnitTest1
    Dependencies
    AdoAddressDaoTests.cs
    AdoAppKeyDaoTests.cs
    AdoCartDaoTests.cs
    AdoCartDetailsDaoTests.cs
    AdoOrderDaoTests.cs
    AdoOrderDetailsDaoTests.cs
    AdoPersonDaoTests.cs
    AdoProductDaoTests.cs
    AdoShopDaoTests.cs
    appsettings.json
    Usings.cs
  Dal.Common
    Dependencies
    AdoTemplate.cs
    ConfigurationUtil.cs
    DbUtil.cs
    DefaultConnectionFactory.cs
    IConnectionFactory.cs
    QueryParameter.cs
```



```
Test Explorer
40  40  0
Test run finished: 40 Tests (40 Passed, 0 Failed, 0 Skipped)
Test                                 Durat...
CaaSTests.UnitTest1 (40)             653 ms
  CaaSTests.UnitTest1 (40)           653 ms
    AdoAddressDaoTests (5)           425 ms
    AdoAppKeyDaoTests (5)             21 ms
    AdoCartDaoTests (5)               66 ms
    AdoCartDetailsTests (5)           40 ms
    AdoOrderDaoTests (3)              28 ms
    AdoOrderDetailsDaoTests (3)       19 ms
    AdoPersonDaoTests (5)             14 ms
    AdoProductDaoTests (5)            24 ms
    AdoShopDaoTests (4)               16 ms
```



```
Name
.vs
bin
CaaS.Client
CaaS.Dal.Ado
CaaS.Dal.Interfaces
CaaS.Domain
CaaS.Sql
CaaSTests.UnitTest1
Dal.Common
Db
obj
CaaS.sln
```

Der Ordner CaaS.Sql beinhaltet SQL Files zum Erstellen von Tabellen bei **Microsoft SQL Server**

# Ordnername /Projekname CaaS.Domain

## Address.cs

```
namespace CaaS.Domain;
public class Address
{
    public Address(string id, string street, string floor, double postalCode, string city,
string province,string country)
    {
        Id = id;
        Street = street;
        Floor = floor;
        PostalCode = postalCode;
        City = city;
        Province = province;
        Country = country;
    }

    public string Id { get; set; }
    public string Street { get; private set; }
    public string Floor { get; private set; }
    public double PostalCode { get; private set; }
    public string City { get; private set; }
    public string Province { get; private set; }
    public string Country { get; private set; }
    public override string ToString() =>
      $"Address(id:{Id}, Street:{Street}, Floor{Floor}, PostalCode:{PostalCode},
City:{City})";
}
```

## AppKey.cs

```
namespace CaaS.Domain;
public class AppKey
{
    public AppKey(string id, string shopId)
    {
        Id = id;
        ShopId = shopId;

    }
    public string Id { get; set; }
    public string ShopId { get; private set; }

    public override string ToString() =>
      $"AppKey(id:{Id}, CustId:{ShopId})";
}
```

## Cart.cs

```
namespace CaaS.Domain;
public class Cart
{
    public Cart(string id, string custId, string status)
    {
        Id = id;
        CustId = custId;
        Status = status;
    }
    public string Id { get; set; }
    public string CustId { get; private set; }
    public string Status { get; private set; }
    public override string ToString() =>
      $"Cart(id:{Id}, CustId:{CustId}, Status:{Status})";
}
CartDetails.cs
namespace CaaS.Domain;
public class CartDetails
{
    public CartDetails(string id, string cartId, string productId, double quantity)
    {
        Id = id;
        CartId = cartId;
        ProductId = productId;
        Quantity = quantity;
    }
    public string Id { get; set; }
    public string CartId { get; private set; }
    public string ProductId { get; private set; }
    public double Quantity { get; private set; }
    public override string ToString() =>
      $"Cart(id:{Id}, CartId:{CartId}, Status:{ProductId})";
}
Order.cs
namespace CaaS.Domain;
public class Order
{
    public Order(string id, string custId, string cartId, DateTime orderDate)
    {
        Id = id;
        CustId = custId;
        CartId = cartId;
        OrderDate = orderDate
    }
    public string Id { get; set; }
    public string CustId { get; private set; }
    public string CartId { get; private set; }
    public DateTime OrderDate { get; private set; }
    public override string ToString() =>
      $"Order(id:{Id}, CustId:{CustId}, CartId{CartId}, OrderDate:{OrderDate:yyyy-MM-dd})";
}
```

## OrderDetails.cs

```
namespace CaaS.Domain;
public class OrderDetails

{
    public OrderDetails(string id, string orderId, string productId, double unitPrice,
double quantity, double discount)
    {
        Id = id;
        OrderId = orderId;
        ProductId = productId;
        UnitPrice = unitPrice;
        Quantity = quantity;
        Discount = discount;
    }
    public string Id { get; set; }
    public string OrderId { get; private set; }
    public string ProductId { get; private set; }
    public double UnitPrice { get; private set; }
    public double Quantity { get; private set; }
    public double Discount { get; private set; }
    public override string ToString() =>
      $"OrdersDetails(id:{Id}, OrderId:{OrderId}, ProductId{ProductId},
UnitPrice:{UnitPrice}, Quantity:{Quantity})";
}
```

## Person.cs

```
namespace CaaS.Domain;
public class Person
{
    public Person(string id, string firstName, string lastName, DateTime dateOfBirth
,string email, string addressId, string status)
    {
        Id = id;
        FirstName = firstName;
        LastName = lastName;
        DateOfBirth = dateOfBirth;
        Email = email;
        AddressId = addressId;
        Status = status;
    }
    public string Id { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public DateTime DateOfBirth { get; set; }
    public string Email { get; }
    public string AddressId { get; }
    public string Status { get; }
    public override string ToString() =>
      $"Person(id:{Id}, FirstName:{FirstName}, LastName:{LastName},
DateOfBirth:{DateOfBirth:yyyy-MM-dd})";
}
```

## Product.cs

```
namespace CaaS.Domain;
public class Product
{
    public Product(string id, string name, double price, string amountDesc, string
productDesc, string downloadLink)
    {
        if (string.IsNullOrEmpty(id))
        {
            throw new ArgumentException($"'{nameof(id)}' cannot be null or empty.",
nameof(id));
        }
        if (string.IsNullOrEmpty(name))
        {
            throw new ArgumentException($"'{nameof(name)}' cannot be null or empty.",
nameof(name));
        }
        if (string.IsNullOrEmpty(amountDesc))
        {
            throw new ArgumentException($"'{nameof(amountDesc)}' cannot be null or empty.",
nameof(amountDesc));
        }
        Id = id;
        this.Name = name;
        this.Price = price;
        this.AmountDesc = amountDesc;
        this.ProductDesc = productDesc;
        this.DownloadLink = downloadLink;
    }
    public string Id { get; set; }
    public string Name { get; set; }
    public double Price{ get; set; }
    public string AmountDesc { get; set; }
    public string? ProductDesc { get; set; }
    public string? DownloadLink { get; set; }
    public override string ToString() =>
      $"Product(id:{Id}, name:{Name}, price:{Price}, amountDesc:{AmountDesc},
productDesc:{ProductDesc})";
}
```

## Shop.cs

```
namespace CaaS.Domain;
public class Shop
{
    public Shop(string id, string name,string fieldDesc, string mandantId, string address)
    {
        Id = id;
        this.Name = name;
        this.FieldDesc = fieldDesc;
        this.MandantId = mandantId;
        this.Address = address;
    }
    public string Id { get; set; }
    public string Name { get; set; }
    public string FieldDesc{ get; set; }
    public string MandantId { get; set; }
    public string Address { get;  set; }
    public override string ToString() =>
      $"Shop(id:{Id}, name:{Name}, mandant:{MandantId})";
}
```

# Ordnername /Projekname : CaaS.Dal.Interfaces

## IAddressDao.cs

```
using CaaS.Domain;
namespace CaaS.Dal.Interface
{
    public interface IAddressDao
    {
        Task<IEnumerable<Address>> FindAllAsync(string table);
        Task<Address?> FindByIdAsync(string id, string table);
        Task<bool> UpdateAsync(Address address, string table);
        Task<bool> DeleteByIdAsync(string id, string table);
        Task<bool> StoreAsync(Address address, string table);
    }
}
```

## IAppKeyDao.cs

```
using CaaS.Domain;
namespace CaaS.Dal.Interface
{
    public interface IAppKeyDao
    {
        Task<IEnumerable<AppKey>> FindAllAsync(string table);
        Task<AppKey?> FindByIdAsync(string id, string table);

        Task<bool> UpdateAsync(AppKey appkey, string table);
        Task<bool> DeleteByIdAsync(string id, string table);
        Task<bool> StoreAsync(AppKey appkey, string table);
    }
}
```

## ICartDao.cs

```
using CaaS.Domain;
namespace CaaS.Dal.Interface
{
    public interface ICartDao
    {
        Task<IEnumerable<Cart>> FindAllAsync(string table);
        Task<Cart?> FindByIdAsync(string id, string table);

        Task<bool> UpdateAsync(Cart cart, string table);
        Task<bool> DeleteByIdAsync(string id, string table);
        Task<bool> StoreAsync(Cart cart, string table);
    }
}
```

## ICartDetailsDao.cs

```
using CaaS.Domain;
namespace CaaS.Dal.Interface
{
    public interface IOrderDetailsDao
    {
        Task<IEnumerable<OrderDetails>> FindAllAsync(string table);
        Task<OrderDetails?> FindByIdAsync(string id, string table);
        Task<bool> UpdateAsync(OrderDetails Orderdetails, string table);
        Task<bool> DeleteByIdAsync(string id, string table);
        Task<bool> StoreAsync(OrderDetails Orderdetails, string table);
    }
}
```

## IOrderDetailsDao.cs

```
using CaaS.Domain;
namespace CaaS.Dal.Interface
{
    public interface ICartDetailsDao
    {
        Task<IEnumerable<CartDetails>> FindAllAsync(string table);
        Task<CartDetails?> FindByIdAsync(string id, string table);
        Task<bool> UpdateAsync(CartDetails cartdetails, string table);
        Task<bool> DeleteByIdAsync(string id, string table);
        Task<bool> StoreAsync(CartDetails cartdetails, string table);
    }
}
```

## IPersonDao.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using CaaS.Domain;
nameSpace CaaS.Dal.Interface;
public interface IPersonDao
{
    Task<IEnumerable<Person>> FindAllAsync(string table);
    Task<Person?> FindByIdAsync(string id,string table);
    Task<bool> UpdateAsync(Person person, string table);
    Task<bool> DeleteByIdAsync(string id, string table);
    Task<bool> StoreAsync(Person person, string table);
}
```

## IProductDao.cs

```
using CaaS.Domain;
namespace CaaS.Dal.Interface;
public interface IProductDao
{
    Task<IEnumerable<Product>> FindAllAsync(string table);
    Task<Product?> FindByIdAsync(string id, string table);
    Task<bool> UpdateAsync(Product product, string table);
    Task<bool> DeleteByIdAsync(string id, string table);
    Task<bool> StoreAsync(Product product, string table);
}
```

## IShopDao.cs

```
using CaaS.Domain;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace CaaS.Dal.Interface;
public interface IShopDao
{
    Task<IEnumerable<Shop>> FindAllAsync(string table);
    Task<Shop?> FindByIdAsync(string id, string table);

    Task<bool> UpdateAsync(Shop shop, string table);
    Task<bool> DeleteByIdAsync(string id, string table);
    Task<bool> StoreAsync(Shop shop, string table);
}
```

# Ordnername /Projekname : CaaS.Dal.Ado

## AdoAddressDao.cs

```
using Dal.Common;
using CaaS.Domain;
using CaaS.Dal.Interface;
using static CaaS.Dal.Ado.AdoMapDao;
using System;
using Org.BouncyCastle.Asn1.X509;
using System.Text.RegularExpressions;
namespace CaaS.Dal.Ado;
public class AdoAddressDao : IAddressDao
{
    private readonly AdoTemplate template;
    public AdoAddressDao(IConnectionFactory connectionFactory)
    {
        this.template = new AdoTemplate(connectionFactory);
    }
    public async Task<IEnumerable<Address>> FindAllAsync(string table)
    {
        return await template.QueryAsync($"select * from {table}",
MapRowToAddress);
    }
    public async Task<Address?> FindByIdAsync(string id, string table)
    {
        return await template.QuerySingleAsync($"select * from {table} where
address_id=@id",
            MapRowToAddress,
            new QueryParameter("@id", id));
    }
    public async Task<bool> UpdateAsync(Address address, string table)
    {
        Address? addr = await FindByIdAsync(address.Id, table);
        if (addr is not null)
        {
            string sqlcmd = $"update {table} set street=@street, floor=@floor,
postal_code=@postal_code, city=@city, province=@province,country=@country
where address_id=@id";
            // array für die Parameter erstellen
            return await template.ExecuteAsync(@sqlcmd,
                new QueryParameter("@id", address.Id),
                new QueryParameter("@street", address.Street),
                new QueryParameter("@floor", address.Floor),
                new QueryParameter("@postal_code", address.PostalCode),
                new QueryParameter("@city", address.City),
                new QueryParameter("@province", address.Province),
                new QueryParameter("@country", address.Country)) == 1;
        }
        return false;
    }
    public async Task<bool> DeleteByIdAsync(string id, string table)
    {
        Address? addr = await FindByIdAsync(id, table);
        if (addr is not null)
        {
            string sqlcmd = $"delete from {table} where address_id=@id";
            return await template.ExecuteAsync(@sqlcmd,
                  new QueryParameter("@id", id)) == 1;
        }
        return false;
    }
    public async Task<bool> StoreAsync(Address address, string table)
    {
        Address? addr = await FindByIdAsync(address.Id, table);
        if (addr is null)
        {
            string sqlcmd = $"insert into {table} ( address_id,street, floor,
postal_code, city, province, country ) " +
            "values (@id,@street,
@floor,@postal_code,@city,@province,@country) ";
            return await template.ExecuteAsync(@sqlcmd,
                new QueryParameter("@id", address.Id),
                new QueryParameter("@street", address.Street),
                new QueryParameter("@floor", address.Floor),
                new QueryParameter("@postal_code", address.PostalCode),
                new QueryParameter("@city", address.City),
                new QueryParameter("@province", address.Province),
                new QueryParameter("@country",address.Country)) == 1;
        }
        return await UpdateAsync(address, table);
    }
}
```

## AdoAppKeyDap.cs

```
using Dal.Common;
//using Microsoft.Data.SqlClient;
using CaaS.Dal.Interface;
using CaaS.Domain;
using static CaaS.Dal.Ado.AdoMapDao;
using Org.BouncyCastle.Asn1.X509;
using System;
namespace CaaS.Dal.Ado;
public class AdoAppKeyDao : IAppKeyDao
{
    private readonly AdoTemplate template;
    private readonly AdoShopDao shopDao;
    public AdoAppKeyDao(IConnectionFactory connectionFactory)
    {
        this.template = new AdoTemplate(connectionFactory);
        this.shopDao = new AdoShopDao(connectionFactory);
    }
    public async Task<IEnumerable<AppKey>> FindAllAsync(string table)
    {
        return await template.QueryAsync($"select * from {table}",
MapRowToAppKey);
    }
    public async Task<AppKey?> FindByIdAsync(string id, string table)
    {
        return await template.QuerySingleAsync($"select * from {table} where
app_key=@id",
            MapRowToAppKey,
            new QueryParameter("@id", id));
    }
    public async Task<bool> UpdateAsync(AppKey appKey, string table)
    {
        AppKey? apk = await FindByIdAsync(appKey.Id, table);
        if (apk is not null)
        {
            string sqlcmd = $"update {table} set shop_id=@shopId where
app_key=@id";
            // array für die Parameter erstellen
            return await template.ExecuteAsync(@sqlcmd,
                new QueryParameter("@id", appKey.Id),
                new QueryParameter("@shopId", appKey.ShopId)
                ) == 1;
        }
        return false;
    }
    public async Task<bool> DeleteByIdAsync(string id, string table)
    {
        AppKey? apk = await FindByIdAsync(id, table);
```

```
            if (apk is not null)
            {
                string sqlcmd = $"delete from {table} where app_key=@id";
                return await template.ExecuteAsync(@sqlcmd,
                        new QueryParameter("@id", id)) == 1;
            }
            return false;
        }
        public async Task<bool> StoreAsync(AppKey appKey, string table)
        {
            AppKey? apk = await FindByIdAsync(appKey.Id, table);
            if (apk is null)
            {
                Shop? sh = await shopDao.FindByIdAsync(appKey.ShopId, "Shops");
                if (sh is not null)
                {
                    string sqlcmd = $"insert into {table} (app_key,shop_id) values
    (@id,@shopId)";
                    return await template.ExecuteAsync(@sqlcmd,
                        new QueryParameter("@id", appKey.Id),
                        new QueryParameter("@shopId", appKey.ShopId)
                        ) == 1;
                }else
                    return false;
            }
            return await UpdateAsync(appKey, table);
        }
    }
```

# AdoCartDao.cs

```
  using Dal.Common;
//using Microsoft.Data.SqlClient;
using CaaS.Dal.Interface;
using CaaS.Domain;
using static CaaS.Dal.Ado.AdoMapDao;
using Org.BouncyCastle.Asn1.X509;
using System.Text.RegularExpressions;
namespace CaaS.Dal.Ado;
public class AdoCartDao : ICartDao
{
    private readonly AdoTemplate template;
    private readonly AdoPersonDao personDao;
    public AdoCartDao(IConnectionFactory connectionFactory)
    {
        this.template = new AdoTemplate(connectionFactory);
        this.personDao = new AdoPersonDao(connectionFactory);
    }

    public async Task<IEnumerable<Cart>> FindAllAsync(string table)
    {
        return await template.QueryAsync($"select * from {table}",
MapRowToCart);
    }

    public async Task<Cart?> FindByIdAsync(string id,string table)
    {
        return await template.QuerySingleAsync($"select * from {table} where
cart_id=@id",
            MapRowToCart,
            new QueryParameter("@id", id));
    }

    public async Task<bool> UpdateAsync(Cart cart,string table)
    {
        Cart? c = await FindByIdAsync(cart.Id, table);
        if (c is not null)
        {
            string sqlcmd = $"update {table} set cust_id=@custId,
status=@status where cart_id=@id";
            // array für die Parameter erstellen
            return await template.ExecuteAsync(@sqlcmd,
                new QueryParameter("@id", cart.Id),
                new QueryParameter("@custId", cart.CustId),
                new QueryParameter("@status", cart.Status)
                ) == 1;
        }
        return false;
    }
    public async Task<bool> DeleteByIdAsync(string id, string table)
    {
        Cart? c = await FindByIdAsync(id, table);
        if (c is not null)
        {
            string sqlcmd = $"delete from {table} where cart_id=@id";
            return await template.ExecuteAsync(@sqlcmd,
                   new QueryParameter("@id", id)) == 1;
        }
        return false;
    }
    public async Task<bool> StoreAsync(Cart cart, string table)
    {
        Cart? c = await FindByIdAsync(cart.Id, table);
        string[] substrings = Regex.Split(table, "Carts");
        string customershoptable = "Customers" + substrings[substrings.Length
- 1];
        Person? cust = await personDao.FindByIdAsync(cart.CustId,
customershoptable);
        await personDao.StoreAsync(cust, customershoptable);

        if (c is null )
        {
            if (cust is not null)
            {
                string sqlcmd = $"insert into {table} (cart_id,cust_id,
status) " +
               "values (@id,@custId, @status) ";
                return await template.ExecuteAsync(@sqlcmd,
                    new QueryParameter("@id", cart.Id),
                    new QueryParameter("@custId", cart.CustId),
                    new QueryParameter("@status", cart.Status)
                    ) == 1;
            }
            else
                return false;
        }
        return await UpdateAsync(cart, table);
    }
}
```

## AdoCartsDetailsDao.cs

```
  using Dal.Common;
//using Microsoft.Data.SqlClient;
using CaaS.Dal.Interface;
using CaaS.Domain;
using static CaaS.Dal.Ado.AdoMapDao;
using System.Text.RegularExpressions;
namespace CaaS.Dal.Ado;
public class AdoCartDetailsDao : ICartDetailsDao
{
    private readonly AdoTemplate template;
    private readonly AdoProductDao productDao;
    private readonly AdoCartDao cartDao;
    public AdoCartDetailsDao(IConnectionFactory connectionFactory)
    {
        this.template = new AdoTemplate(connectionFactory);
        this.cartDao = new AdoCartDao(connectionFactory);
        this.productDao = new AdoProductDao(connectionFactory);
    }
    public async Task<IEnumerable<CartDetails>> FindAllAsync(string table)
    {
        return await template.QueryAsync($"select * from {table}",
MapRowToCartDetails);
    }
    public async Task<CartDetails?> FindByIdAsync(string id,string table)
    {
        return await template.QuerySingleAsync($"select * from {table} where
cart_details_id=@id",
            MapRowToCartDetails,
            new QueryParameter("@id", id));
    }
    public async Task<bool> UpdateAsync(CartDetails cartDetails,string table)
    {
        CartDetails? cd = await FindByIdAsync(cartDetails.Id, table);
       if (cd is not null)
        {
            string sqlcmd = $"update {table} set cart_id=@cartId,
product_id=@productId, qty=@quantity where cart_details_id=@id";
            // array für die Parameter erstellen
            return await template.ExecuteAsync(@sqlcmd,
                new QueryParameter("@id", cartDetails.Id),
                new QueryParameter("@cartId", cartDetails.CartId),
                new QueryParameter("@productId", cartDetails.ProductId),
                new QueryParameter("@quantity", cartDetails.Quantity)) == 1;
        }
        return false;
    }
    public async Task<bool> DeleteByIdAsync(string id, string table)
    {
        CartDetails? cd = await FindByIdAsync(id, table);
        if (cd is not null)
        {
            string sqlcmd = $"delete from {table} where cart_details_id=@id";
            return await template.ExecuteAsync(@sqlcmd,
                   new QueryParameter("@id", id)) == 1;
        }
        return false;
    }
    public async Task<bool>StoreAsync(CartDetails cartDetails, string table)
    {
        CartDetails? cd = await FindByIdAsync(cartDetails.Id, table);
        if (cd is null)
        {
            string sqlcmd = $"insert into {table} ( cart_details_id,cart_id,
product_id, qty ) " +
            "values (@id,@cartId, @productId,@quantity)";

            return await template.ExecuteAsync(@sqlcmd,
                new QueryParameter("@id", cartDetails.Id),
                new QueryParameter("@cartId", cartDetails.CartId),
                new QueryParameter("@productId", cartDetails.ProductId),
                new QueryParameter("@quantity", cartDetails.Quantity)
                ) == 1;
        }
        return await UpdateAsync(cartDetails, table);
    }
}
```

# AdoMapDao.cs

```
using CaaS.Domain;
using System.Data;
namespace CaaS.Dal.Ado
{
    public class AdoMapDao {
        public static Person MapRowToPerson(IDataRecord row) =>
        new(
            id: (string)row["person_id"],
            firstName: (string)row["first_name"],
            lastName: (string)row["last_name"],
            dateOfBirth: (DateTime)row["dob"],
            email: (string)row["email"],
            addressId: (string)row["address"],
            status: (string)row["status"]
        );
        public static Address MapRowToAddress(IDataRecord row) =>
        new(
            id: (string)row["address_id"],
            street: (string)row["street"],
            floor: (string)row["floor"],
            postalCode: (double)row["postal_code"],
            city: (string)row["city"],
            province: (string)row["province"],
            country: (string)row["country"]
        );
        public static Shop MapRowToShop(IDataRecord row) =>
        new(
           id: (string)row["shop_id"],
           name: (string)row["shop_name"],
           fieldDesc: (string)row["field_descriptions"],
           mandantId: (string)row["mandant_id"],
           address: (string)row["address"]
        );
        public static Product MapRowToProduct(IDataRecord row) =>
        new(
            id: (string)row["product_id"],
            name: (string)row["product_name"],
            price: (double)row["price"],
            amountDesc: (string)row["amount_desc"],
            productDesc: (string)row["product_desc"],
            downloadLink: (string)row["download_link"]
        );
        public static OrderDetails MapRowToOrderDetails(IDataRecord row) =>
        new(
           id: (string)row["order_details_id"],
           orderId: (string)row["order_id"],
           productId: (string)row["product_id"],
           unitPrice: (double)row["unit_price"],
           quantity: (double)row["qty"],
           discount: (double)row["discount"]
       );
        public static Order MapRowToOrder(IDataRecord row) =>
        new(
            id: (string)row["order_id"],
            custId: (string)row["cust_id"],
            cartId: (string)row["cart_id"],
            orderDate: (DateTime)row["order_date"]
        );
        public static CartDetails MapRowToCartDetails(IDataRecord row) =>
        new(
            id: (string)row["cart_details_id"],
            cartId: (string)row["cart_id"],
            productId: (string)row["product_id"],
            quantity: (double)row["qty"]
        );
        public static Cart MapRowToCart(IDataRecord row) =>
        new(
            id: (string)row["cart_id"],
            custId: (string)row["cust_id"],
            status: (string)row["status"]
        );
        public static AppKey MapRowToAppKey(IDataRecord row) =>
        new(
           id: (string)row["app_key"],
           shopId: (string)row["shop_id"]
        );

    }
}
```

# AdoOrderDao.cs

```
    using Dal.Common;
//using Microsoft.Data.SqlClient;
using CaaS.Dal.Interface;
using CaaS.Domain;
using static CaaS.Dal.Ado.AdoMapDao;
using System;
using System.Text.RegularExpressions;
namespace CaaS.Dal.Ado;
public class AdoOrderDao : IOrderDao
{
    private readonly AdoTemplate template;
    private readonly AdoPersonDao personDao;
    private readonly AdoCartDao cartDao;
    public AdoOrderDao(IConnectionFactory connectionFactory)
    {
        this.template = new AdoTemplate(connectionFactory);
        this.personDao = new AdoPersonDao(connectionFactory);
        this.cartDao = new AdoCartDao(connectionFactory);
    }
    public async Task<IEnumerable<Order>> FindAllAsync(string table)
    {
        return await template.QueryAsync($"select * from {table}",
MapRowToOrder);
    }
    public async Task<Order?> FindByIdAsync(string id,string table)
    {
        return await template.QuerySingleAsync($"select * from {table} where
order_id=@id",
            MapRowToOrder,
            new QueryParameter("@id", id));
    }
    public async Task<bool> UpdateAsync(Order order,string table)
    {
        Order? o = await FindByIdAsync(order.Id, table);

        if (o is not null)
        {
            string sqlcmd = $"update {table} set cust_id=@custId,
cart_id=@cartId, order_date=@order_date where order_id=@id";
            // array für die Parameter erstellen
            return await template.ExecuteAsync(@sqlcmd,
                new QueryParameter("@id", order.Id),
                new QueryParameter("@custId", order.CustId),
                new QueryParameter("@cartId", order.CartId),
                new QueryParameter("@order_date", order.OrderDate)) == 1;
        }
        return false;
    }
    public async Task<bool> DeleteByIdAsync(string id, string table)
    {
        Order? o = await FindByIdAsync(id, table);
        if (o is not null)
        {
            string sqlcmd = $"delete from {table} where order_id=@id";
            return await template.ExecuteAsync(@sqlcmd,
                    new QueryParameter("@id", id)) == 1;
        }
        return false;
    }
    public async Task<bool>StoreAsync(Order order, string table)
    {
        Order? o = await FindByIdAsync(order.Id, table);
        if (o is null)
        {
            string sqlcmd = $"insert into {table} ( order_id,cust_id, cart_id,
order_date) " +
            "values (@id,@custId, @cartId,@order_date) ";
            //string[] substrings = Regex.Split(table, "Orders");
            //string customershoptable =
"Customers"+substrings[substrings.Length - 1];
            //string cartshoptable = "Carts"+substrings[substrings.Length -
1];
            //Person? cust = await personDao.FindByIdAsync(order.CustId,
customershoptable);
            //await personDao.StoreAsync(cust, customershoptable);
            //Cart? cart = await cartDao.FindByIdAsync(order.CustId,
cartshoptable);
            //await cartDao.StoreAsync(cart, cartshoptable);
            return await template.ExecuteAsync(@sqlcmd,
                new QueryParameter("@id", order.Id),
                new QueryParameter("@custId", order.CustId),
                new QueryParameter("@cartId", order.CartId),
                new QueryParameter("@order_date", order.OrderDate)
                ) == 1;
        }
        return await UpdateAsync(order, table);
    }
}
```

# AdoOrderDetailsDao.cs

```
  using Dal.Common;
//using Microsoft.Data.SqlClient;
using CaaS.Dal.Interface;
using CaaS.Domain;
using static CaaS.Dal.Ado.AdoMapDao;
using Org.BouncyCastle.Asn1.X509;
using System.Text.RegularExpressions;
namespace CaaS.Dal.Ado;
public class AdoOrderDetailsDao : IOrderDetailsDao
{
    private readonly AdoTemplate template;
    private readonly AdoProductDao productDao;
    private readonly AdoOrderDao orderDao;
    public AdoOrderDetailsDao(IConnectionFactory connectionFactory)
    {
        this.template = new AdoTemplate(connectionFactory);
        this.orderDao = new AdoOrderDao(connectionFactory);
        this.productDao =  new AdoProductDao(connectionFactory);
    }
    public async Task<IEnumerable<OrderDetails>> FindAllAsync(string table)
    {
        return await template.QueryAsync($"select * from {table}",
MapRowToOrderDetails);
    }
    public async Task<OrderDetails?> FindByIdAsync(string id,string table)
    {
        return await template.QuerySingleAsync($"select * from {table} where
order_details_id=@id",
            MapRowToOrderDetails,
            new QueryParameter("@id", id));
    }
    public async Task<bool> UpdateAsync(OrderDetails orderdetails,string
table)
    {
        OrderDetails? o = await FindByIdAsync(orderdetails.Id, table);
        if (o is not null)
        {
            string sqlcmd = $"update {table} set order_id=@orderId,
product_id=@productId,unit_price=@unitPrice," +
                $"qty=@quantity, discount=@discount where
order_details_id=@id";
            // array für die Parameter erstellen
            return await template.ExecuteAsync(@sqlcmd,
                new QueryParameter("@id", orderdetails.Id),
                new QueryParameter("@orderId", orderdetails.OrderId),
                new QueryParameter("@productId", orderdetails.ProductId),
                new QueryParameter("@unitPrice", orderdetails.UnitPrice),
                new QueryParameter("@quantity", orderdetails.Quantity),
                new QueryParameter("@discount", orderdetails.Discount)
                ) == 1;
        }
        return false;
    }
    public async Task<bool> DeleteByIdAsync(string id, string table)
    {
        OrderDetails? o = await FindByIdAsync(id ,table);
        if (o is not null)
        {
            string sqlcmd = $"delete from {table} where order_details_id=@id";
            return await template.ExecuteAsync(@sqlcmd,
                   new QueryParameter("@id", id)) == 1;
        }
        return false;
    }
    public async Task<bool> StoreAsync(OrderDetails orderdetails, string
table)
    {
        OrderDetails? od = await FindByIdAsync(orderdetails.Id, table);
        if (od is null)
        {
            string sqlcmd = $"insert into {table} (order_details_id,order_id,
product_id,unit_price,qty,discount) " +
           "values (@id,@orderId, @productId,@unitPrice, @quantity,@discount)
";
            //string[] substrings = Regex.Split(table, "OrdersDetails");
            //string ordersshoptable = "Orders" + substrings[substrings.Length
- 1];
            //string productshoptable = "Products" +
substrings[substrings.Length - 1];
            //Order? o = await orderDao.FindByIdAsync(orderdetails.OrderId,
ordersshoptable);
            //await orderDao.StoreAsync(o, ordersshoptable);
            //Product? prod = await
productDao.FindByIdAsync(orderdetails.ProductId, productshoptable);
            //await productDao.StoreAsync(prod, productshoptable);


            return await template.ExecuteAsync(@sqlcmd,
                new QueryParameter("@id", orderdetails.Id),
                new QueryParameter("@orderId", orderdetails.OrderId),
                new QueryParameter("@productId", orderdetails.ProductId),
                new QueryParameter("@unitPrice", orderdetails.UnitPrice),
                new QueryParameter("@quantity", orderdetails.Quantity),
                new QueryParameter("@discount", orderdetails.Discount)
                ) == 1;
        }
        return await UpdateAsync(orderdetails, table);
    }
}
```

# AdoPersonDao.cs

```
  using Dal.Common;
//using Microsoft.Data.SqlClient;
using CaaS.Dal.Interface;
using CaaS.Domain;
using static CaaS.Dal.Ado.AdoMapDao;
namespace CaaS.Dal.Ado;
public class AdoPersonDao : IPersonDao
{
    private readonly AdoTemplate template;
    private readonly AdoAddressDao addressDao;
    private bool IsAddressByIdSync(string addressId, string addresstable)
    {
        return template.QuerySingleSync($"select * from {addresstable} where
address_id=@id",
            MapRowToAddress,
            new QueryParameter("@id", addressId)) is not null;
    }
    public AdoPersonDao(IConnectionFactory connectionFactory)
    {
        this.template = new AdoTemplate(connectionFactory);
        this.addressDao = new AdoAddressDao(connectionFactory);
    }
    Public async Task<IEnumerable<Person>> FindAllAsync(string table)
    {
        return await template.QueryAsync($"select * from {table}",
MapRowToPerson);
    }
    public async Task<Person?> FindByIdAsync(string id,string table)
    {
        return await template.QuerySingleAsync($"select * from {table} where
person_id=@id",
            MapRowToPerson,
            new QueryParameter("@id", id));
    }
    public async Task<bool> UpdateAsync(Person person,string table)
    {
        Person? p = await FindByIdAsync(person.Id, table);
        if (p is not null)
        {
            string sqlcmd = $"update {table} set first_name=@fn,
last_name=@ln, dob=@dob where person_id=@id";
            // array für die Parameter erstellen
            return await template.ExecuteAsync(@sqlcmd,
                new QueryParameter("@id", person.Id),
                new QueryParameter("@fn", person.FirstName),
                new QueryParameter("@ln", person.LastName),
                new QueryParameter("@dob", person.DateOfBirth)) == 1;
        }
        return false;
    }

    public async Task<bool> DeleteByIdAsync(string id, string table)
    {
        Person? p = await FindByIdAsync(id,table);
        if (p is not null)
        {
            string sqlcmd = $"delete from {table} where person_id=@id";
            return await template.ExecuteAsync(@sqlcmd,
                   new QueryParameter("@id", id)) == 1;
        }
        return false;
    }

    public async Task<bool>StoreAsync(Person person, string table)
    {
        Person? p = await FindByIdAsync(person.Id, table);
        if (p is null)
        {
            string sqlcmd = $"insert into {table} ( person_id,first_name,
last_name, dob, email, address, status ) " +
            "values (@id,@fn, @ln,@dob,@email,@address,@status) ";
            //string addresstable = "Addresses" + table;
            //Address? addr = await addressDao.FindByIdAsync(person.AddressId,
addresstable);
            //await addressDao.StoreAsync(addr, addresstable);


            return await template.ExecuteAsync(@sqlcmd,
                new QueryParameter("@id", person.Id),
                new QueryParameter("@fn", person.FirstName),
                new QueryParameter("@ln", person.LastName),
                new QueryParameter("@dob", person.DateOfBirth),
                new QueryParameter("@email", person.Email),
                new QueryParameter("@address", person.AddressId),
                new QueryParameter("@status", person.Status)) == 1;
        }
        return await UpdateAsync(person, table);
    }

}
```

# AdoProductDao.cs

```
  using Dal.Common;
//using Microsoft.Data.SqlClient;
using CaaS.Dal.Interface;
using CaaS.Domain;
using System.Data;
using static CaaS.Dal.Ado.AdoMapDao;
using System
namespace CaaS.Dal.Ado;
public class AdoProductDao : IProductDao
{
    private readonly AdoTemplate template;

    private Product? FindByIdSync(string id, string table)
    {
        return template.QuerySingleSync($"select * from {table} where
product_id=@id",
            MapRowToProduct,
            new QueryParameter("@id", id));
    }
    public AdoProductDao(IConnectionFactory connectionFactory)
    {
        this.template = new AdoTemplate(connectionFactory);
    }
    public async Task<IEnumerable<Product>> FindAllAsync(string table)
    {
        return await template.QueryAsync($"select * from {table}",
MapRowToProduct);
    }
    public async Task<Product?> FindByIdAsync(string id,string table)
    {
        return await template.QuerySingleAsync($"select * from {table} where
product_id=@id",
            MapRowToProduct,
            new QueryParameter("@id", id));
    }
    public async Task<bool> UpdateAsync(Product product,string table)
    {
        Product? p = await FindByIdAsync(product.Id, table);
        if (p is not null)
        {
            string sqlcmd = $"update {table} set product_name=@name, price =
@price," +
            $"amount_desc=@amountDesc,
product_desc=@productDesc,download_link=@downloadLink  where product_id=@id";
            // array für die Parameter erstellen
            return await template.ExecuteAsync(@sqlcmd,
                new QueryParameter("@id", product.Id),
                new QueryParameter("@name", product.Name),
                new QueryParameter("@price", product.Price),
                new QueryParameter("@amountDesc", product.AmountDesc),
                new QueryParameter("@productDesc", product.ProductDesc),
                new QueryParameter("@downloadLink", product.DownloadLink)
                ) == 1;
        }
        return false;
    }
    public async Task<bool> DeleteByIdAsync(string id, string table)
    {
        Product? p = await FindByIdAsync(id, table);
        if (p is not null)
        {
            string sqlcmd = $"delete from {table} where product_id=@id";
            return await template.ExecuteAsync(@sqlcmd,
                   new QueryParameter("@id", id)) == 1;
        }
        return false;
    }
    public async Task<bool> StoreAsync(Product product, string table)
    {
        Product? p = await FindByIdAsync(product.Id, table);
        if (p is null)
        {
            string sqlcmd = $"insert into {table} (product_id,product_name,
price, amount_desc, product_desc, download_link ) " +
           "values (@id,@name, @price,@amountDesc,@productDesc,@downloadLink)
";
            return await template.ExecuteAsync(@sqlcmd,
                new QueryParameter("@id", product.Id),
                new QueryParameter("@name", product.Name),
                new QueryParameter("@price", product.Price),
                new QueryParameter("@amountDesc", product.AmountDesc),
                new QueryParameter("@productDesc", product.ProductDesc),
                new QueryParameter("@downloadLink", product.DownloadLink)
                ) == 1;
        }
        return await UpdateAsync(product, table);
    }
}
```

## AdoShopDao.cs

```
    using Dal.Common;
//using Microsoft.Data.SqlClient;
using CaaS.Dal.Interface;
using CaaS.Domain;
using static CaaS.Dal.Ado.AdoMapDao;
namespace CaaS.Dal.Ado;
public class AdoShopDao : IShopDao
{
    private readonly AdoTemplate template;
    private readonly AdoAppKeyDao appKeyDao;
    public AdoShopDao(IConnectionFactory connectionFactory)
    {
        this.template = new AdoTemplate(connectionFactory);
        this.appKeyDao = new AdoAppKeyDao(connectionFactory);
    }
    public async Task<IEnumerable<Shop>> FindAllAsync(string table)
    {
        return await template.QueryAsync($"select * from {table}",
MapRowToShop);
    }
    public async Task<Shop?> FindByIdAsync(string id, string table)
    {
        return await template.QuerySingleAsync($"select * from {table} where
shop_id=@id",
            MapRowToShop,
            new QueryParameter("@id", id));
    }
    public async Task<bool> UpdateAsync(Shop shop,string table)
    {
        Shop? s = await FindByIdAsync(shop.Id, table);
        if (s is not  null)
        {
            string sqlcmd = $"update {table} set shop_name=@name,
field_descriptions=@fieldDesc, mandant_id=@mandantId where shop_id=@id";
            // array für die Parameter erstellen
            return await template.ExecuteAsync(@sqlcmd,
                new QueryParameter("@id", shop.Id),
                new QueryParameter("@name", shop.Name),
                new QueryParameter("@fieldDesc", shop.FieldDesc),
                new QueryParameter("@mandantId", shop.MandantId)
                ) == 1;
        }
        return false;
    }
    public async Task<bool> DeleteByIdAsync(string id, string table)
    {
        Shop? s = await FindByIdAsync(id, table);
        if (s is not null)
        {
            string sqlcmd = $"delete from {table} where shop_id=@id";
            return await template.ExecuteAsync(@sqlcmd,
                   new QueryParameter("@id", id)) == 1;
        }
        return false;
    }
    public async Task<bool> StoreAsync(Shop shop, string table)
    {
        Shop? s = await FindByIdAsync(shop.Id, table);
        if (s is null)
        {
             string sqlcmd = $"insert into {table} ( shop_id,shop_name,
field_descriptions, mandant_id, address) " +
            "values (@id,@name, @fieldDesc,@mandantId,@address) ";
            AppKey? appkey = new AppKey(("ak-" +
shop.Id).Replace("shop","s"),shop.Id);
            await appKeyDao.StoreAsync(appkey, "AppKey");
            return await template.ExecuteAsync(@sqlcmd,
            new QueryParameter("@id", shop.Id),
            new QueryParameter("@name", shop.Name),
            new QueryParameter("@fieldDesc", shop.FieldDesc),
            new QueryParameter("@mandantId", shop.MandantId),
            new QueryParameter("@address", shop.Address)
            ) == 1
        }
        return await UpdateAsync(shop, table);
    }
}
```

# Ordnername /Projekname Dal.Common

## Ado.Template.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Common;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Dal.Common;
// Delegates in eigenes File geben
public delegate T RowMapper<T>(IDataRecord row);
public class AdoTemplate
{
    private readonly IConnectionFactory connectionFactory;

    public AdoTemplate(IConnectionFactory connectionFactory)
    {
        this.connectionFactory = connectionFactory;
    }
    private void AddParameters(DbCommand command, QueryParameter[] parameters)
    {
        foreach (var p in parameters)
        {
            DbParameter dbParam = command.CreateParameter();
            dbParam.ParameterName = p.Name;
            dbParam.Value = p.Value;
            command.Parameters.Add(dbParam);
        }
    }
    public async Task<IEnumerable<T>> QueryAsync<T>(string sql, RowMapper<T> rowMapper,
        params QueryParameter[] parameters)
    {
        await using DbConnection connection = await
connectionFactory.CreateConnectionAsync();
        await using DbCommand command = connection.CreateCommand();
        command.CommandText = sql;
        AddParameters(command, parameters);

        var items = new List<T>();
        await using (DbDataReader reader = await command.ExecuteReaderAsync())
        {
            while (await reader.ReadAsync())
            {
                items.Add(rowMapper(reader));
            }
        }
        return items;
    }
    public IEnumerable<T> QuerySync<T>(string sql, RowMapper<T> rowMapper,
        params QueryParameter[] parameters)
    {
        using DbConnection connection = connectionFactory.CreateConnectionSync();
        using DbCommand command = connection.CreateCommand();
        command.CommandText = sql;
        AddParameters(command, parameters);
        var items = new List<T>();
        using (DbDataReader reader = command.ExecuteReader())
        {
            while (reader.Read())
            {
                items.Add(rowMapper(reader));
            }
        }
        return items;
    }
    public async Task<T?> QuerySingleAsync<T>(string sql, RowMapper<T> rowMapper,
        params QueryParameter[] parameters)
    {
        return (await QueryAsync(sql, rowMapper, parameters)).SingleOrDefault();
    }


    public T? QuerySingleSync<T>(string sql, RowMapper<T> rowMapper,
        params QueryParameter[] parameters)
    {
        return QuerySync(sql, rowMapper, parameters).SingleOrDefault();
    }

    public async Task<int> ExecuteAsync(string sql, params QueryParameter[] parameters)
    {
        await using DbConnection connection = await
connectionFactory.CreateConnectionAsync();
        await using DbCommand command = connection.CreateCommand();
        command.CommandText = sql;
        AddParameters(command, parameters);

        return await command.ExecuteNonQueryAsync();
    }
}
```

## ConfigurationUtil.cs

```
namespace Dal.Common;

using Microsoft.Extensions.Configuration;

public static class ConfigurationUtil
{
  private static IConfiguration? configuration = null;

  public static IConfiguration GetConfiguration() =>
    configuration ??= new ConfigurationBuilder()
      .AddJsonFile("appsettings.json", optional: false)
      .Build();

  public static (string connectionString, string providerName)
GetConnectionParameters(string configName)
  {
    var connectionConfig =
GetConfiguration().GetSection("ConnectionStrings").GetSection(configName);
    return (connectionConfig["ConnectionString"], connectionConfig["ProviderName"]);
  }
}
```

## DbUtil.cs

```
namespace Dal.Common;
using System.Data.Common;
public static class DbUtil
{
  public static void RegisterAdoProviders()
  {
    // Use new Implementation of MS SQL Provider
    DbProviderFactories.RegisterFactory("Microsoft.Data.SqlClient",
Microsoft.Data.SqlClient.SqlClientFactory.Instance);
    DbProviderFactories.RegisterFactory("MySql.Data.MySqlClient",
MySql.Data.MySqlClient.MySqlClientFactory.Instance);
  }
}
```

## DefaultConnectionFactory.cs

```
using System.Data.SqlClient;
namespace Dal.Common;
using System.Data.Common;
using System.Threading.Tasks;
using Dal.Common;
using Microsoft.Extensions.Configuration;
public class DefaultConnectionFactory : IConnectionFactory
{
    private readonly DbProviderFactory dbProviderFactory;
    public static IConnectionFactory FromConfiguration(IConfiguration config, string
connectionStringConfigName)
    {
        var connectionConfig =
config.GetSection("ConnectionStrings").GetSection(connectionStringConfigName);
        string connectionString = connectionConfig["ConnectionString"];
        string providerName = connectionConfig["ProviderName"];
        return new DefaultConnectionFactory(connectionString, providerName);
    }
    public DefaultConnectionFactory(string connectionString, string providerName)
    {
        this.ConnectionString = connectionString;
        this.ProviderName = providerName;
        DbUtil.RegisterAdoProviders();
        this.dbProviderFactory = DbProviderFactories.GetFactory(providerName);
    }
    public string ConnectionString { get; }
    public string ProviderName { get; }
    public async Task<DbConnection> CreateConnectionAsync()
    {
        var connection = dbProviderFactory.CreateConnection();
        if (connection is null)
        {
            throw new InvalidOperationException("DbProviderFactory.CreateConnection()
returned null");
        }
        connection.ConnectionString = this.ConnectionString;
        await connection.OpenAsync();
        return connection;
    }
    public DbConnection? CreateConnectionSync()
    {
        var connection = dbProviderFactory.CreateConnection();
        if (connection is null)
        {
            throw new InvalidOperationException("DbProviderFactory.CreateConnection()
returned null");
```

```
        }

        connection.ConnectionString = this.ConnectionString;
        connection.Open();
        return connection;
    }
}
```

## IConnectionFactory.cs

```
using System.Data.Common;
namespace Dal.Common;
public interface IConnectionFactory
{
    string ConnectionString { get; }
    string ProviderName { get; }
    Task<DbConnection> CreateConnectionAsync();
    DbConnection? CreateConnectionSync();
}
```

## QueryParameter.cs

```
namespace Dal.Common;
public class QueryParameter
{
  public QueryParameter(string name, object? value)
  {
    this.Name = name;
    this.Value = value;
  }
  public string  Name { get; }
  public object? Value { get; }
}
```

# Ordnername /Projekname : CaaSTests.UnitTest1

## AdoAddressDaoTests.cs

```
using CaaS.Dal.Ado;
using CaaS.Dal.Interface;
using CaaS.Domain;
using Dal.Common;
using Microsoft.Extensions.Configuration;
namespace CaaSTests.UnitTest1
{
    [TestFixture]
    public class AdoAddressDaoTests
    {
        private IAddressDao _AddressDao;
        private string _table = "AddressesShops";
        [SetUp]
        public void Setup()
        {
            IConfiguration configuration = 
ConfigurationUtil.GetConfiguration();
            IConnectionFactory? connectionFactory = 
DefaultConnectionFactory.FromConfiguration(configuration, "CaaSDbConnection");
            _AddressDao = new AdoAddressDao(connectionFactory);
        }
        [Test]
        public async Task TestFindAllAsync()
        {
            List<Address> AddressList = (await 
_AddressDao.FindAllAsync(_table)).ToList();
            Assert.True(4 >= AddressList.Count);
        }
        [Test]
        public async Task TestFindByIdAsync()
        {
            Address? address1 = await _AddressDao.FindByIdAsync("addr-sh1", 
_table);
            Assert.IsNotNull(address1);
            Assert.True(address1.Province.Trim() == "Wien");
        [Test]
        public async Task TestUpdateAsync()
        {
            Address? address = await _AddressDao.FindByIdAsync("addr-sh3", 
_table);
```

```
            address.Province = "Lower Austria";
            await _AddressDao.UpdateAsync(address, _table);
            Assert.That(address.Province == "Lower Austria");
            Address? address2 = new Address("addr-sh3", "Meilissenweg 3", "",
4020, "Linz", "Upper Austria", "Austria");
            await _AddressDao.UpdateAsync(address2, _table);
            Assert.That(address2.Province == "Upper Austria");
        }
        [Test]
        public async Task TestDeleteByIdAsync()
        {
            Address? address = await _AddressDao.FindByIdAsync("addr-sh12",
_table);
            Assert.IsNotNull(address);
            await _AddressDao.DeleteByIdAsync("addr-sh12", _table);
            address = await _AddressDao.FindByIdAsync("addr-sh12", _table);
            Assert.IsNull(address);

        }
        [Test]
        public async Task TestStoreAsync()
        {
            Address? address = new Address("addr-sh12", "Gleinkernweg 6", "",
4420, "Wels", "Upper Austria", "Austria");
            await _AddressDao.StoreAsync(address, _table);
            Address? address2 = await _AddressDao.FindByIdAsync("addr-sh12",
_table);
            Assert.True(address.Province == address2.Province);
        }
    }

}
```

## AdoAppKeyDaoTests.cs

```
using CaaS.Dal.Ado;
using CaaS.Dal.Interface;
using CaaS.Domain;
using Dal.Common;
using Microsoft.Extensions.Configuration;
namespace CaaSTests.UnitTest1
{
    public class AdoAppKeyDaoTests
    {
        private IAppKeyDao _appKeyDao;
        private string _table = "AppKey";
        [SetUp]
        public void Setup()
        {
            IConfiguration configuration =
ConfigurationUtil.GetConfiguration();
            IConnectionFactory? connectionFactory =
DefaultConnectionFactory.FromConfiguration(configuration, "CaaSDbConnection");
            _appKeyDao = new AdoAppKeyDao(connectionFactory);
        }
        [Test]
        public async Task TestFindAllAsync()
        {
            List<AppKey> appKeyList = (await
_appKeyDao.FindAllAsync(_table)).ToList();
            Assert.True(1 == appKeyList.Count);
        }
        [Test]
        public async Task TestFindByIdAsync()
        {
            AppKey? appKey1 = await _appKeyDao.FindByIdAsync("ak-s1", _table);
            Assert.IsNotNull(appKey1);
        [Test]
        public async Task TestUpdateAsync()
        {
            AppKey? appKey = new AppKey("ak-s2", "shop-3");
            await _appKeyDao.UpdateAsync(appKey, _table);
            AppKey? appKey2 = await _appKeyDao.FindByIdAsync("ak-s2", _table);
            Assert.That(appKey2.ShopId == "shop-3");
            appKey2.ShopId = "shop-2";
            await _appKeyDao.UpdateAsync(appKey, _table);
        }
        [Test]
        public async Task TestDeleteByIdAsync()
        {
            await _appKeyDao.DeleteByIdAsync("ak-s2", _table);
            AppKey? appKey2 = await _appKeyDao.FindByIdAsync("ak-s2", _table);
            Assert.IsNull(appKey2);
        }
        [Test]
        public async Task TestStoreAsync()
        {
            AppKey? appKey = await _appKeyDao.FindByIdAsync("ak-s2", _table);
            Assert.IsNull(appKey);
            appKey = new AppKey("ak-s2", "shop-3");
            await _appKeyDao.StoreAsync(appKey, _table);
            AppKey? appKey2 = await _appKeyDao.FindByIdAsync("ak-s2", _table);
            Assert.True(appKey2.ShopId == "shop-3");
        }
    }
}
```

# AdoCartDaoTests.cs

```
using CaaS.Dal.Ado;
using CaaS.Dal.Interface;
using CaaS.Domain;
using Dal.Common;
using Microsoft.Extensions.Configuration;
using MySqlX.XDevAPI.Relational;
using System;
using System.Configuration;
namespace CaaS.UnitTest1
{
    [TestFixture]
    public class AdoCartDaoTests    {
        private ICartDao _CartDao;
        private string _table = "CartsShop1";
        [SetUp]
        public void Setup()
        {
            IConfiguration configuration = 
ConfigurationUtil.GetConfiguration();
            IConnectionFactory? connectionFactory = 
DefaultConnectionFactory.FromConfiguration(configuration, "CaaSDbConnection");
            _CartDao = new AdoCartDao(connectionFactory);
        }
        [Test]
        public async Task TestFindAllAsync()
        {
            List<Cart> CartList = (await 
_CartDao.FindAllAsync(_table)).ToList();
            Assert.True(999 == CartList.Count);
        }
        [Test]
        public async Task TestFindByIdAsync()
        {
            Cart? cart1 = await _CartDao.FindByIdAsync("cart-3-c-7", _table);
            Assert.IsNotNull(cart1);
            Assert.True(cart1.Status== ("open")&&cart1.CustId=="cust-7");
        }
        [Test]
        public async Task TestUpdateAsync()
        {
            Cart? cart = await _CartDao.FindByIdAsync("cart-10-c-17", _table);
            Assert.That(cart.Status == "open");
            cart.Status = "closed";
            await _CartDao.UpdateAsync(cart, _table);
            Assert.That(cart.Status == "closed");
            cart.Status = "open";
            await _CartDao.UpdateAsync(cart, _table);
            Assert.That(cart.Status == "open");
        }
        [Test]
        public async Task TestDeleteByIdAsync()
        {
            await _CartDao.DeleteByIdAsync("cart-4-c-6", _table);
            Cart? cart2 = await _CartDao.FindByIdAsync("cart-4-c-6", _table);
            Assert.IsNull(cart2);
        }
        [Test]
        public async Task TestStoreAsync()
        {
            Cart? cart = await _CartDao.FindByIdAsync("cart-4-c-6", _table);
            Assert.IsNull(cart);
            cart= new Cart("cart-4-c-6", "cust-6", "open");
            await _CartDao.StoreAsync(cart, _table);
            Cart? cart2 = await _CartDao.FindByIdAsync("cart-4-c-6", _table);
            Assert.True(cart2.CustId=="cust-6");
        }
    }
}
```

# AdoCartDetailsDaoTests.cs

```
using CaaS.Dal.Ado;
using CaaS.Dal.Interface;
using CaaS.Domain;
using Dal.Common;
using Microsoft.Extensions.Configuration;
using MySqlX.XDevAPI.Relational;
using System;
using System.Configuration;
namespace CaaS.UnitTest1
{
    [TestFixture]
    public class AdoCartDetailsTests    {
        private ICartDetailsDao _cartdetailsDao;
        private string _table = "CartsDetailsShop1";
        [SetUp]
        public void Setup()
        {
            IConfiguration configuration = 
ConfigurationUtil.GetConfiguration();
            IConnectionFactory? connectionFactory = 
DefaultConnectionFactory.FromConfiguration(configuration, "CaaSDbConnection");
            _cartdetailsDao = new AdoCartDetailsDao(connectionFactory);
        }
        [Test]
        public async Task TestFindAllAsync()
        {
            List<CartDetails> cartdetailsList = (await 
_cartdetailsDao.FindAllAsync(_table)).ToList();
            Console.WriteLine(cartdetailsList.Count);
            Assert.True(cartdetailsList.Count>0);
        }
        [Test]
        public async Task TestFindByIdAsync()
        {
            CartDetails? cartdetails1 = await 
_cartdetailsDao.FindByIdAsync("cd-9", _table);
            Assert.IsNotNull(cartdetails1);
            Assert.True(cartdetails1.ProductId == ("arz-18"));
        }
        [Test]
        public async Task TestUpdateAsync()
        {
            CartDetails? cartdetails = await 
_cartdetailsDao.FindByIdAsync("cd-8", _table);
            cartdetails.Quantity = 0;
            await _cartdetailsDao.UpdateAsync(cartdetails, _table);
            Assert.That(cartdetails.Quantity == 0);
            cartdetails.Quantity = 18;
            await _cartdetailsDao.UpdateAsync(cartdetails, _table);
            Assert.That(cartdetails.Quantity == 18);
        }
        [Test]
        public async Task TestDeleteByIdAsync()
        {
            CartDetails? cartdetails = await
_cartdetailsDao.FindByIdAsync("cd-4", _table);
            Assert.IsNotNull(cartdetails);
            await _cartdetailsDao.DeleteByIdAsync("cd-4", _table);
            cartdetails = await _cartdetailsDao.FindByIdAsync("cd-4", _table);
            Assert.IsNull(cartdetails);
        }
        [Test]
        public async Task TestStoreAsync()
        {
            CartDetails? cartdetails = await
_cartdetailsDao.FindByIdAsync("cd-4", _table);
            Assert.IsNull(cartdetails);
            cartdetails= new CartDetails("cd-4", "cart-6-c-4", "arz-94", 7);
            await _cartdetailsDao.StoreAsync(cartdetails, _table);
            CartDetails? cartdetails1 = new CartDetails("cd-1100", "cart-10-c-
1", "arz-23", 20);
            await _cartdetailsDao.StoreAsync(cartdetails1, _table);
            CartDetails? cartdetails2 = await
_cartdetailsDao.FindByIdAsync("cd-1100", _table);
            Assert.That(cartdetails2.CartId,
Is.EqualTo((cartdetails1.CartId)));
        }
    }
}
```

# AdoOrderDaoTests.cs

```
using CaaS.Dal.Ado;
using CaaS.Dal.Interface;
using CaaS.Domain;
using Dal.Common;
using Microsoft.Extensions.Configuration;
namespace CaaS.UnitTest1
{
    [TestFixture]
    public class AdoOrderDaoTests    {
        private IOrderDao _orderDao;
        private string _table = "OrdersShop1";
        [SetUp]
        public void Setup()
        {
            IConfiguration configuration =
ConfigurationUtil.GetConfiguration();
            IConnectionFactory? connectionFactory =
DefaultConnectionFactory.FromConfiguration(configuration, "CaaSDbConnection");
            _orderDao = new AdoOrderDao(connectionFactory);
        }
        [Test]
        public async Task TestFindAllAsync()
        {
            List<Order> orderList = (await
_orderDao.FindAllAsync(_table)).ToList();
            Assert.IsTrue(990==orderList.Count);
        }
        [Test]
        public async Task TestFindByIdAsync()
        {
            Order? order1 = await _orderDao.FindByIdAsync("or-9-c-9", _table);
            Assert.IsNotNull(order1);
            Assert.True(order1.CustId == ("cust-9"));
        }
        [Test]
        public async Task TestUpdateAsync()
        {
            Order? order = await _orderDao.FindByIdAsync("or-9-c-9", _table);
            order.OrderDate = new DateTime(2018,3,30);
            await _orderDao.UpdateAsync(order, _table);
            Order? order2 = await _orderDao.FindByIdAsync("or-9-c-9", _table);
            Assert.That(order2.OrderDate== new DateTime(2018, 3, 30));
        }
        [Test]
        public async Task TestDeleteByIdAsync()
        {
            Order? order = await _orderDao.FindByIdAsync("or-9-c-99", _table);
            Assert.IsNotNull(order);
            await _orderDao.DeleteByIdAsync(order.Id, _table);
            order = await _orderDao.FindByIdAsync("or-9-c-99", _table);
            Assert.IsNull(order);
        }
        [Test]
        public async Task TestStoreAsync()
        {
            Order? order = await _orderDao.FindByIdAsync("or-9-c-99", _table);
            Assert.IsNull(order);
            order= new Order("or-9-c-99", "cust-99", "cart-9-c-99", new
DateTime(2017,09,15));
            await _orderDao.StoreAsync(order, _table);
            Order? order2 = await _orderDao.FindByIdAsync("or-9-c-99",
_table);
            Assert.True(order.CartId== order2.CartId);


        }
    }
}
```

# AdoOrderDetailsDaoTests.cs

```
using CaaS.Dal.Ado;
using CaaS.Dal.Interface;
using CaaS.Domain;
using Dal.Common;
using Microsoft.Extensions.Configuration;
namespace CaaSTests.UnitTest1
{
    [TestFixture]
    public class AdoOrderDetailsDaoTests    {

        private IOrderDetailsDao _orderDetailsDao;
        private string _table = "OrdersDetailsShop1";
        [SetUp]
        public void Setup()
        {
            IConfiguration configuration =
ConfigurationUtil.GetConfiguration();
            IConnectionFactory? connectionFactory =
DefaultConnectionFactory.FromConfiguration(configuration, "CaaSDbConnection");
            _orderDetailsDao = new AdoOrderDetailsDao(connectionFactory);
        }

        [Test]
        public async Task TestFindAllAsync()
        {
            List<OrderDetails> orderdetailsList = (await
_orderDetailsDao.FindAllAsync(_table)).ToList();
            Assert.IsTrue(2933== orderdetailsList.Count);
        }
       [Test]
        public async Task TestFindByIdAsync()
        {
            OrderDetails? orderdetails1 = await
_orderDetailsDao.FindByIdAsync("od-10", _table);
            Assert.IsNotNull(orderdetails1);
            Assert.True(orderdetails1.ProductId == ("arz-16"));
        }
        [Test]
        public async Task TestUpdateAsync()
        {
            OrderDetails? orderdetails = await
_orderDetailsDao.FindByIdAsync("od-10", _table);
            orderdetails.UnitPrice = 6.75;
            await _orderDetailsDao.UpdateAsync(orderdetails, _table);
            OrderDetails? orderdetails2 = await
_orderDetailsDao.FindByIdAsync("od-10", _table);
            Assert.True(orderdetails2.UnitPrice== 6.75);
            orderdetails2.UnitPrice = 6.75;
            await _orderDetailsDao.UpdateAsync(orderdetails2, _table);
        }
    }
}
```

## AdoPersonDaoTests.cs

```
using CaaS.Dal.Ado;
using CaaS.Dal.Interface;
using CaaS.Domain;
using Dal.Common;
using Microsoft.Extensions.Configuration;
using MySqlX.XDevAPI.Relational;
using System;
using System.Configuration;
namespace CaaS.UnitTest1
{
    [TestFixture]
    public class AdoPersonDaoTests    {
        private IPersonDao _personDao;
        private string _table = "Mandants";
        [SetUp]
        public void Setup()
        {
            IConfiguration configuration =
ConfigurationUtil.GetConfiguration();
            IConnectionFactory? connectionFactory =
DefaultConnectionFactory.FromConfiguration(configuration, "CaaSDbConnection");
            _personDao = new AdoPersonDao(connectionFactory);
        }
        [Test]
        public async Task TestFindAllAsync()
        {
            List<Person> personList = (await
_personDao.FindAllAsync(_table)).ToList();
            Assert.True(2 == personList.Count);
        }
        [Test]
        public async Task TestFindByIdAsync()
        {
            Person? person1 = await _personDao.FindByIdAsync("mandant-1",
_table);
            Assert.IsNotNull(person1);
            Assert.True(person1.FirstName == ("andrew"));
        }
        [Test]
        public async Task TestUpdateAsync()
        {
            Person? person = await _personDao.FindByIdAsync("mandant-2",
_table);
            person.FirstName = "ryoshi";
            await _personDao.UpdateAsync(person, _table);
            Assert.That(person.FirstName == "ryoshi");
        }
        [Test]
        public async Task TestDeleteByIdAsync()
        {
            Person? person = await _personDao.FindByIdAsync("mandant-2",
_table);
            await _personDao.DeleteByIdAsync(person.Id, _table);
            person = await _personDao.FindByIdAsync("mandant-2", _table);
            Assert.IsNull(person);
        }
        [Test]
        public async Task TestStoreAsync()
        {
            Person? person = await _personDao.FindByIdAsync("mandant-4",
_table);
            Assert.IsNull(person);
            person= new Person("mandant-2", "ryo", "kimura", new
DateTime(1971, 11, 7), "ryokimura@example.com", "addr-m2", "mandant");
            await _personDao.StoreAsync(person, _table);
            person = await _personDao.FindByIdAsync("mandant-2", _table);
            Assert.True(person.FirstName=="ryo");
            Person? person1 = new Person("mandant-3", "robert", "kimura", new
DateTime(1974, 12, 7), "robertkimura@example.com", "addr-m3", "mandant");
            await _personDao.StoreAsync(person1, _table);
            Person? person2 = await _personDao.FindByIdAsync("mandant-3",
_table);
            Assert.That(person2.FirstName, Is.EqualTo("robert"));
        }
    }
}
```

# AdoProductDaoTests.cs

```
using CaaS.Dal.Ado;
using CaaS.Dal.Interface;
using CaaS.Domain;
using Dal.Common;
using Microsoft.Extensions.Configuration;
namespace CaaS.UnitTest1
{
    [TestFixture]
    public class AdoProductDaoTests    {
        private IProductDao _ProductDao;
        private string _table = "ProductShop2";
        [SetUp]
        public void Setup()
        {
            IConfiguration configuration =
ConfigurationUtil.GetConfiguration();
            IConnectionFactory? connectionFactory =
DefaultConnectionFactory.FromConfiguration(configuration, "CaaSDbConnection");
            _ProductDao = new AdoProductDao(connectionFactory);
        }
        [Test]
        public async Task TestFindAllAsync()
        {
            List<Product> ProductList = (await
_ProductDao.FindAllAsync(_table)).ToList();
          Assert.IsTrue(109==ProductList.Count);
        }
      [Test]
        public async Task TestFindByIdAsync()
        {
            Product? Product1 = await _ProductDao.FindByIdAsync("978-0-7503-
1047-5", _table);
            Assert.IsNotNull(Product1);
            Assert.True(Product1.Name == ("Introduction to Networks of
Networks"));
       }
        [Test]
        public async Task TestUpdateAsync()
        {
            Product? product = await _ProductDao.FindByIdAsync("978-0-7503-
1645-3", _table);
            product.AmountDesc = "100 pc";
            await _ProductDao.UpdateAsync(product, _table);
            Product? product2 = await _ProductDao.FindByIdAsync("978-0-7503-
1645-3", _table);
            Assert.That(product2.AmountDesc == product.AmountDesc);
        }
        [Test]
        public async Task TestDeleteByIdAsync()
        {
            Product? product = await _ProductDao.FindByIdAsync("978-0-7503-
1645-3", _table);
            await _ProductDao.DeleteByIdAsync(product.Id, _table);
            product = await _ProductDao.FindByIdAsync("978-0-7503-1645-3",
_table);
            Assert.IsNull(product);
        }
        [Test]
        public async Task TestStoreAsync()
        {
            Product? product = await _ProductDao.FindByIdAsync("mandant-4",
_table);
            Assert.IsNull(product);
            product= new Product("978-0-7503-1645-3", "Functional Carbon
Materials", 52.3,"1 pc",  "not yet", "not yet");
            await _ProductDao.StoreAsync(product, _table);
            Product? product2 = await _ProductDao.FindByIdAsync("978-0-7503-
1645-3", _table);
            Assert.True(product.Name== product2.Name);
            Product? product3 = new Product("978-0-7503-1047-5", "Introduction
to Networks of Networks",44.1, "1pc", "not yet", "not yet");
            await _ProductDao.StoreAsync(product3, _table);
            Product? product4 = await _ProductDao.FindByIdAsync("978-0-7503-
1047-5", _table);
            Assert.That(product4.Id ,Is.EqualTo(product4.Id));

        }
    }

}
```

## AdoShopDaoTests.cs

```
using CaaS.Dal.Ado;
using CaaS.Dal.Interface;
using CaaS.Domain;
using Dal.Common;
using Microsoft.Extensions.Configuration;
using MySqlX.XDevAPI.Relational;
using System;
using System.Configuration;
namespace CaaS.UnitTest1
{
    [TestFixture]
    public class AdoShopDaoTests    {
        private IShopDao _shopDao;
        private IAppKeyDao _appKeyDao;
        private string _table = "Shops";
        [SetUp]
        public void Setup()
        {
            IConfiguration configuration =
ConfigurationUtil.GetConfiguration();
            IConnectionFactory? connectionFactory =
DefaultConnectionFactory.FromConfiguration(configuration, "CaaSDbConnection");
            _shopDao = new AdoShopDao(connectionFactory);
            _appKeyDao = new AdoAppKeyDao(connectionFactory);
        }
        [Test]
        public async Task TestFindAllAsync()
        {
            List<Shop> ShopList = (await
_shopDao.FindAllAsync(_table)).ToList();
            Assert.True( ShopList.Count<=2 && ShopList.Count>0);
        }
        [Test]
        public async Task TestFindByIdAsync()
        {
            Shop? shop1 = await _shopDao.FindByIdAsync("shop-1", _table);
            Assert.IsNotNull(shop1);
        }
        Test]
        public async Task TestUpdateAsync()
        {
            Shop? shop = await _shopDao.FindByIdAsync("shop-1", _table);
            Assert.That(shop.Name== "LoveRead");
            shop.Name = "MedStorDrug";
            Assert.That(shop.Name== "MedStorDrug");
            await _shopDao.UpdateAsync(shop, _table);
            shop.Name = "LoveRead";
            await _shopDao.UpdateAsync(shop, _table);
            Assert.That(shop.Name == "LoveRead");
        }
    }
}
```

# Ordnername /Projekname : CAASSQL

## AddressesCustomersShop.Table.sql

```
USE [DbCaaS00Testing]
GO
/****** Object:  Table [dbo].[AddressesCustomersShop1]    Script Date: 
20.11.2022 21:23:49 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[AddressesCustomersShop1](
  [address_id] [varchar](40) NOT NULL,
  [street] [varchar](40) NOT NULL,
  [floor] [varchar](40) NOT NULL,
  [postal_code] [float] NOT NULL,
  [city] [varchar](40) NOT NULL,
  [province] [varchar](40) NOT NULL,
  [country] [varchar](40) NOT NULL,
 CONSTRAINT [PK_AddressesCustomersShop1] PRIMARY KEY CLUSTERED 
(
  [address_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, 
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = 
OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

## AddressesMandants.Table.sql

```
USE [DbCaaS00Testing]
GO
/****** Object:  Table [dbo].[AddressesMandants]    Script Date: 20.11.2022 
21:23:49 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[AddressesMandants](
  [address_id] [varchar](40) NOT NULL,
  [street] [varchar](40) NOT NULL,
  [floor] [varchar](40) NOT NULL,
  [postal_code] [float] NOT NULL,
  [city] [varchar](40) NOT NULL,
  [province] [varchar](40) NOT NULL,
  [country] [varchar](40) NOT NULL,
 CONSTRAINT [PK_AddressesMandants] PRIMARY KEY CLUSTERED
(
  [address_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

## AddressesShops.Table.sql

```
USE [DbCaaS00Testing]
GO
/****** Object:  Table [dbo].[AddressesShops]    Script Date: 20.11.2022
21:23:50 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[AddressesShops](
  [address_id] [varchar](40) NOT NULL,
  [street] [varchar](40) NULL,
  [floor] [varchar](40) NULL,
  [postal_code] [float] NULL,
  [city] [varchar](40) NULL,
  [province] [varchar](40) NULL,
  [country] [varchar](40) NULL,
 CONSTRAINT [PK_AddressesShops] PRIMARY KEY CLUSTERED
(
  [address_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

## AppKey.Table.sql

```
USE [DbCaaS00Testing]
GO
/****** Object:  Table [dbo].[AppKey]    Script Date: 20.11.2022 21:23:50
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[AppKey](
  [app_key] [varchar](40) NOT NULL,
  [shop_id] [varchar](40) NOT NULL,
 CONSTRAINT [PK_AppKey] PRIMARY KEY CLUSTERED
(
  [app_key] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[AppKey]  WITH CHECK ADD  CONSTRAINT [FK_AppKey_Shops]
FOREIGN KEY([shop_id])
REFERENCES [dbo].[Shops] ([shop_id])
ON UPDATE CASCADE
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[AppKey] CHECK CONSTRAINT [FK_AppKey_Shops]
GO
```

## CartsDetailsShop.Table.sql

```
USE [DbCaaS00Testing]
GO
/****** Object:  Table [dbo].[CartsDetailsShop1]    Script Date: 20.11.2022
21:23:50 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[CartsDetailsShop1](
  [cart_details_id] [varchar](40) NOT NULL,
  [cart_id] [varchar](40) NOT NULL,
  [product_id] [varchar](40) NOT NULL,
  [qty] [float] NULL,
 CONSTRAINT [PK_CartsDetailsShop1] PRIMARY KEY CLUSTERED
(
  [cart_details_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[CartsDetailsShop1]  WITH CHECK ADD  CONSTRAINT
[FK_CartsDetailsShop1_CartsShop1] FOREIGN KEY([cart_id])
REFERENCES [dbo].[CartsShop1] ([cart_id])
GO
ALTER TABLE [dbo].[CartsDetailsShop1] CHECK CONSTRAINT
[FK_CartsDetailsShop1_CartsShop1]
GO
```

```sql
ALTER TABLE [dbo].[CartsDetailsShop1]  WITH CHECK ADD  CONSTRAINT
[FK_CartsDetailsShop1_ProductShop1] FOREIGN KEY([product_id])
REFERENCES [dbo].[ProductShop1] ([product_id])
GO
ALTER TABLE [dbo].[CartsDetailsShop1] CHECK CONSTRAINT
[FK_CartsDetailsShop1_ProductShop1]
GO
```

## CartsShop.Table.sql

```sql
USE [DbCaaS00Testing]
GO
/****** Object:  Table [dbo].[CartsShop1]    Script Date: 20.11.2022 21:23:50
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[CartsShop1](
  [cart_id] [varchar](40) NOT NULL,
  [cust_id] [varchar](40) NOT NULL,
  [status] [varchar](16) NULL,
 CONSTRAINT [PK_CartsShop1] PRIMARY KEY CLUSTERED
(
  [cart_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[CartsShop1]  WITH CHECK ADD  CONSTRAINT
[FK_CartsShop1_CustomersShop1] FOREIGN KEY([cust_id])
REFERENCES [dbo].[CustomersShop1] ([person_id])
GO
ALTER TABLE [dbo].[CartsShop1] CHECK CONSTRAINT [FK_CartsShop1_CustomersShop1]
GO
```

## CustomersShop.Table.sql

```sql
USE [DbCaaS00Testing]
GO
/****** Object:  Table [dbo].[CustomersShop1]    Script Date: 20.11.2022
21:23:50 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[CustomersShop1](
  [person_id] [varchar](40) NOT NULL,
  [first_name] [varchar](40) NOT NULL,
  [last_name] [varchar](40) NOT NULL,
  [dob] [datetime] NOT NULL,
  [email] [varchar](60) NOT NULL,
  [address] [varchar](40) NOT NULL,
  [status] [varchar](16) NULL,
 CONSTRAINT [PK_CustomersShop1] PRIMARY KEY CLUSTERED
(
  [person_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[CustomersShop1]  WITH CHECK ADD  CONSTRAINT
[FK_CustomersShop1_AddressesCustomersShop1] FOREIGN KEY([address])
REFERENCES [dbo].[AddressesCustomersShop1] ([address_id])
GO
ALTER TABLE [dbo].[CustomersShop1] CHECK CONSTRAINT
[FK_CustomersShop1_AddressesCustomersShop1]
GO
```

## Mandants.Table.sql

```
USE [DbCaaS00Testing]
GO
/****** Object:  Table [dbo].[Mandants]    Script Date: 20.11.2022 21:23:50
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Mandants](
  [person_id] [varchar](40) NOT NULL,
  [first_name] [varchar](40) NOT NULL,
  [last_name] [varchar](40) NOT NULL,
  [dob] [datetime] NULL,
  [email] [varchar](60) NOT NULL,
  [address] [varchar](40) NOT NULL,
  [status] [varchar](16) NOT NULL,
 CONSTRAINT [PK_Mandants] PRIMARY KEY CLUSTERED
(
  [person_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Mandants]  WITH CHECK ADD  CONSTRAINT
```

```
[FK_Mandants_AddresesMandants] FOREIGN KEY([address])
REFERENCES [dbo].[AddressesMandants] ([address_id])
GO
ALTER TABLE [dbo].[Mandants] CHECK CONSTRAINT [FK_Mandants_AddresesMandants]
GO
```

## OrdersDetailsShop.Table.sql

```
USE [DbCaaS00Testing]
GO
/****** Object:  Table [dbo].[OrdersDetailsShop1]    Script Date: 20.11.2022
21:23:50 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[OrdersDetailsShop1](
  [order_details_id] [varchar](40) NOT NULL,
  [order_id] [varchar](40) NOT NULL,
  [product_id] [varchar](40) NOT NULL,
  [unit_price] [float] NOT NULL,
  [qty] [float] NOT NULL,
  [discount] [float] NULL,
 CONSTRAINT [PK_OrdersDetailsShop1] PRIMARY KEY CLUSTERED
(
  [order_details_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[OrdersDetailsShop1] ADD  DEFAULT ('x') FOR [order_id]
GO
ALTER TABLE [dbo].[OrdersDetailsShop1] ADD  DEFAULT ((0)) FOR [product_id]
GO
ALTER TABLE [dbo].[OrdersDetailsShop1]  WITH CHECK ADD  CONSTRAINT
[FK_OrdersDetailsShop1_OrdersShop1] FOREIGN KEY([order_id])
REFERENCES [dbo].[OrdersShop1] ([order_id])
ON DELETE SET DEFAULT
GO
ALTER TABLE [dbo].[OrdersDetailsShop1] CHECK CONSTRAINT
[FK_OrdersDetailsShop1_OrdersShop1]
GO
ALTER TABLE [dbo].[OrdersDetailsShop1]  WITH CHECK ADD  CONSTRAINT
[FK_OrdersDetailsShop1_ProductShop1] FOREIGN KEY([product_id])
REFERENCES [dbo].[ProductShop1] ([product_id])
ON DELETE SET DEFAULT
GO
ALTER TABLE [dbo].[OrdersDetailsShop1] CHECK CONSTRAINT
```

```
[FK_OrdersDetailsShop1_ProductShop1]
GO
```

## OrdersShop.Table.sql

```
USE [DbCaaS00Testing]
GO
/****** Object:  Table [dbo].[OrdersShop1]    Script Date: 20.11.2022 21:23:50
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[OrdersShop1](
  [order_id] [varchar](40) NOT NULL,
  [cust_id] [varchar](40) NOT NULL,
  [cart_id] [varchar](40) NOT NULL,
  [order_date] [datetime] NOT NULL,
 CONSTRAINT [PK_OrdersShop1] PRIMARY KEY CLUSTERED
(
  [order_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[OrdersShop1]  WITH CHECK ADD  CONSTRAINT
[FK_OrdersShop1_CartsShop1] FOREIGN KEY([cart_id])
REFERENCES [dbo].[CartsShop1] ([cart_id])
GO
ALTER TABLE [dbo].[OrdersShop1] CHECK CONSTRAINT [FK_OrdersShop1_CartsShop1]
GO
ALTER TABLE [dbo].[OrdersShop1]  WITH CHECK ADD  CONSTRAINT
[FK_OrdersShop1_CustomersShop1] FOREIGN KEY([cust_id])
REFERENCES [dbo].[CustomersShop1] ([person_id])
GO
ALTER TABLE [dbo].[OrdersShop1] CHECK CONSTRAINT
[FK_OrdersShop1_CustomersShop1]
GO
```

## person.Table.sql

```
USE [DbCaaS00Testing]
GO
/****** Object:  Table [dbo].[person]    Script Date: 20.11.2022 21:23:50
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[person](
  [Id] [int] NOT NULL,
  [first_name] [varchar](20) NOT NULL,
  [last_name] [varchar](50) NOT NULL,
  [date_of_birth] [date] NOT NULL,
PRIMARY KEY CLUSTERED
(
  [Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

## ProductShop.Table.sql

```
USE [DbCaaS00Testing]
GO
/****** Object:  Table [dbo].[ProductShop1]    Script Date: 20.11.2022
21:23:50 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[ProductShop1](
  [product_id] [varchar](40) NOT NULL,
  [product_name] [nvarchar](255) NOT NULL,
  [price] [float] NOT NULL,
  [amount_desc] [nvarchar](255) NOT NULL,
  [product_desc] [nvarchar](255) NULL,
  [download_link] [nvarchar](255) NULL,
 CONSTRAINT [PK_ProductShop1] PRIMARY KEY CLUSTERED
(
  [product_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

## Shops.Table.sql

```
USE [DbCaaS00Testing]
GO
/****** Object:  Table [dbo].[Shops]    Script Date: 20.11.2022 21:23:50
******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Shops](
  [shop_id] [varchar](40) NOT NULL,
  [shop_name] [varchar](60) NOT NULL,
  [field_descriptions] [nvarchar](255) NULL,
  [mandant_id] [varchar](40) NOT NULL,
  [address] [varchar](40) NOT NULL,
PRIMARY KEY CLUSTERED
(
  [shop_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Shops] ADD  DEFAULT ('x') FOR [mandant_id]
GO
ALTER TABLE [dbo].[Shops]  WITH CHECK ADD  CONSTRAINT
[FK_Shops_AddressesShops] FOREIGN KEY([address])
REFERENCES [dbo].[AddressesShops] ([address_id])
ON DELETE CASCADE
GO
ALTER TABLE [dbo].[Shops] CHECK CONSTRAINT [FK_Shops_AddressesShops]
GO
ALTER TABLE [dbo].[Shops]  WITH CHECK ADD  CONSTRAINT [FK_Shops_Mandants]
FOREIGN KEY([mandant_id])
REFERENCES [dbo].[Mandants] ([person_id])
ON DELETE SET DEFAULT
GO
ALTER TABLE [dbo].[Shops] CHECK CONSTRAINT [FK_Shops_Mandants]
GO
```