# **Projekt „CASE“**

„CASE“ Software ist eine Microsoft ASP.NET Web-Anwendung (<https://msdn.microsoft.com/en-us/library/4w3ex9c2.aspx>).

Das Kernsystem wurde von dem Unternehmen „NewDirection“ entwickelt. Information zu den Kernsystemfunktionalitäten siehe in dem CASE Handbuch im Git repository (/docs/handbuch\_CASE.doc)

Von mir (Spiridonov Yury) wurden nur notwendige fachliche Erweiterungen, sowie technische Verbesserungen (wie z.B Microsoft Unity für DependencyInjections) durchgeführt. Alle Source-Code Änderungen sind im Git mit den Kommentaren zu finden, deswegen werden in diesem Dokument nur die wesentlichen Code-Änderungen beschrieben.

# **Technische Dokumentation**

Server: Software braucht eine Microsoft .Net Umgebung für die Installation (.Net Framework 4.5.2, Microsoft SQL Server 2008/2012).

Client: standard Browser (z.B Internet Explorer ab Version 9.0)

In dem Git Repository befindet sich ein Solution „KVS.sln“. Solution besteht aus folgenden Projekten:

1. KVSWebApplication: ASP.NET Web Projekt besteht aus
   1. CSS Styling und Images
   2. JavaScript Bibliotheken
   3. ASP.NET Web Seiten
2. KVSDataAccess: Datenzugriffsschicht
3. KVSContracts: Schnittstellendefinitionen
4. KVSConfiguration: Dependency Injections werden mittels Microsoft Unity umgesetzt (<https://msdn.microsoft.com/en-us/library/ff647202.aspx> )
5. KVSCommon: Datentypen und allgemein genutzten Objekte

Für den Aufbau von Web-Seiten wurden folgende Technologien und Frameworks benutzt:

1. Jquery 1.9.1 (<https://jquery.com/>): DOM-Operationen
2. Telerik für ASP.NET (<http://www.telerik.com/products/aspnet-ajax.aspx> ): UI-Library mit verschiedensten Steuerelementen wie z.B DataGrid.

Für den Feinstaubplaketten- und Etikettendruck wird Komponente „WebClientPrint for ASP.NET“ verwendet (<https://www.neodynamic.com/products/printing/raw-data/aspnet-mvc/download/> ). Die entsprechende Lizenz wurde von bios-tec gekauft.

Für die Barcodegenerierung wird eine Open-Source Komponente „Spire.Barcode for ASP.NET“ verwendet (<https://www.e-iceblue.com/Introduce/barcode-for-net-introduce.html#.WVyNLLtPqUk> ).

# **Code-Änderungen Dokumentation**

* **DependencyInjection wurde eingebaut: siehe das Projekt KVSConfiguration. Die Klasse UnityConfiguration.cs enthält die Registrierung von Typendefinitionen für Unity Resolving.**

namespace KVSConfiguration

{

public static partial class UnityConfiguration

{

public static void ConfigureContainer(IUnityContainer container)

{

…..

}

}

}

* **Es wurde die Datenzugriffsschicht zugefügt: public partial interface IKVSEntities : IEntities**

**Die Schnittstelle enthält alle CRUD Datenbank Operationen:**

namespace KVSCommon.Entities

{

/// <summary>

/// Entities interface

/// </summary>

public interface IEntities

{

int? LogUserId { get; set; }

/// <summary>

/// Fügt dem Datenbankkontext einen Logeintrag hinzu.

/// </summary>

/// <param name="logText">Text fьr den Logeintrag.</param>

/// <param name="type">Typ des Logeintrages.</param>

/// <param name="referenceId">Id des betroffenen Datensatzes.</param>

/// <param name="tableName">Tabellenname des betroffenen Datensatzes.</param>

void WriteLogItem(string logText, LogTypes type, int referenceId, string tableName);

/// <summary>

/// Fügt dem Datenbankkontext einen Logeintrag hinzu.

/// </summary>

/// <param name="logText">Text fьr den Logeintrag.</param>

/// <param name="type">Typ des Logeintrages.</param>

/// <param name="referenceId">Id des betroffenen Datensatzes.</param>

/// <param name="tableName">Tabellenname des betroffenen Datensatzes.</param>

/// <param name="childReferenceId">Id des untergeordneten Datensatzes.</param>

void WriteLogItem(string logText, LogTypes type, int referenceId, string tableName, int childReferenceId);

/// <summary>

/// Fügt dem Datenbankkontext einen Logeintrag hinzu.

/// </summary>

/// <param name="logText">Text fьr den Logeintrag.</param>

/// <param name="type">Typ des Logeintrages.</param>

/// <param name="referenceId">Id des betroffenen Datensatzes.</param>

/// <param name="tableName">Tabellenname des betroffenen Datensatzes.</param>

/// <param name="propertyName">Name des betroffenen Feldes des Datensatzes.</param>

void WriteLogItem(string logText, LogTypes type, int referenceId, string tableName, string propertyName);

/// <summary>

/// Gets set of entities

/// </summary>

/// <typeparam name="TEntity">type of entities</typeparam>

/// <returns></returns>

IQueryable<TEntity> GetSet<TEntity>() where TEntity : class;

/// <summary>

/// Add entity to context

/// </summary>

/// <typeparam name="TEntity">Type of adding entity</typeparam>

/// <param name="entity">Adding entity</param>

void AddObject<TEntity>(TEntity entity) where TEntity : class;

/// <summary>

/// Add entity to context

/// </summary>

/// <typeparam name="TEntity">Type of adding entity</typeparam>

/// <param name="entity">Adding entity</param>

void AddRange<TEntity>(IEnumerable<TEntity> entity) where TEntity : class;

/// <summary>

/// Update entity in context

/// </summary>

/// <typeparam name="TEntity">Type of updating entity</typeparam>

/// <param name="entity">Updating entity</param>

void UpdateObject<TEntity>(TEntity entity) where TEntity : class;

/// <summary>

/// Create entity without adding it to context. Use <see cref="AddObject{TEntity}" /> to add object to context

/// </summary>

/// <typeparam name="TEntity">Type of creating entity</typeparam>

/// <returns>Entity</returns>

TEntity CreateObject<TEntity>() where TEntity : class;

/// <summary>

/// Delete entity in context

/// </summary>

/// <typeparam name="TEntity">Type of deleting entity</typeparam>

/// <param name="entity">Deleting entity</param>

void DeleteObject<TEntity>(TEntity entity) where TEntity : class;

/// <summary>

/// Save all changes from underlying context to database

/// </summary>

void SaveChanges();

}

}

Entsprechende Implementierungen befinden sich in der Klasse: public abstract class EntitiesBase : DataContext

* **Es wurden Datenbank „Repository“ Klassen zugefügt, die Zugriff auf die einzelne Datenbanktabellen ermöglichen:**

***AdressManager.cs***

***BankAccountManager.cs***

***BicManager.cs***

***CarOwnerManager.cs***

***ContactManager.cs***

***CostManager.cs***

***CustomerManager.cs***

***DeregistrationOrderManager.cs***

***DockerListManager.cs***

***DocumentManager.cs***

***InvoiceItemAccountItemManager.cs***

***InvoiceItemManager.cs***

***InvoiceManager.cs***

***InvoiceTypesManager.cs***

***LargeCustomerRequiredFieldManager.cs***

***LocationManager.cs***

***OrderManager.cs***

***OrderStatusManager.cs***

***OrderTypeManager.cs***

***PackagingListManager.cs***

***PriceManager.cs***

***ProductCategoryManager.cs***

***ProductManager.cs***

***RegistrationLocationManager.cs***

***RegistrationManager.cs***

***RegistrationOrderManager.cs***

***RegistrationOrderTypeManager.cs***

***UserManager.cs***

***VehicleManager.cs***

* **Es wurde ein Http-Handler für den Feinstaubplaketten- und Etikettendruck zugefügt:**

public class Print : IHttpHandler

{

public void ProcessRequest(HttpContext context)

{

…….

}

}

}

* **Die Klasse für den DATEV Export wurde komplett überarbeitet: public static class DateVExport**
* **Es wurden Basisklassen für die Zulassungs und - Abmeldungsbearbeitung Seiten zugefügt, um die mehrmals vorhandenen duplizierten Methoden zu löschen. Das sind folgende Klassen:**

/// <summary>

/// BasePage class

/// </summary>

public abstract class BasePage : Page

/// <summary>

/// Base UserControl class

/// </summary>

public abstract class BaseUserControl : UserControl

/// <summary>

/// Edit Incoming Orders base class

/// </summary>

public abstract class EditOrdersBase : BaseUserControl

/// <summary>

/// Incoming Orders base class

/// </summary>

public abstract class IncomingOrdersBase : BaseUserControl

/// <summary>

/// View model for orders

/// </summary>

public class OrderViewModel

/// <summary>

/// Admission Orders base class

/// </summary>

public abstract class AdmissionOrderBase : IncomingOrdersBase

/// <summary>

/// Cancellation Orders base class

/// </summary>

public abstract class CancellationOrderBase : IncomingOrdersBase