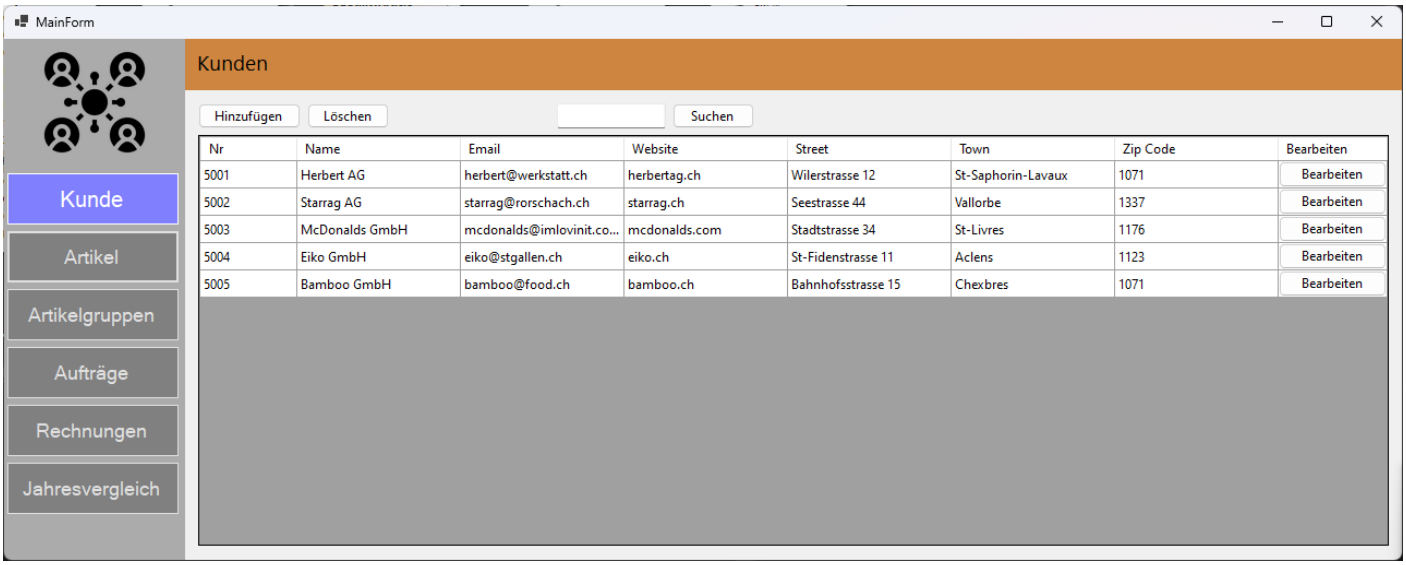


Projektarbeit – Auftragsverwaltung

Kim Devaux, Ousama Askri, Leandro Pajer
Kimdevaux@gmx.ch, ousama.askri@hotmail.com, Leandro.pajer@hotmail.com

Vorgehen und Ergebnisse

13. November 2022:

- Wir haben uns Gedanken darüber gemacht ob für die Datenbank-CRUD Methoden eine Abstrakte Klasse oder ein Interface verwenden sollen. Dadurch dass wir alle Methoden überschreiben müssen, haben wir uns für ein Interface entschieden.

- Für die Erstellung des Datenbankmodells, haben wir uns Gedanken gemacht ob das ganze mithilfe eines ERM-Modells besser visualisiert werden kann.
    - ERM erstellt, aufgrund dessen wurden Klassen sowie EntityConfiguration mit den dazugehörigen Notationen im Visual Studio programmiert.

- Aufgrund dessen, dass wir EFCore verwenden, haben wir uns dazu entschieden, die EntityConfugarations nicht im DBContext zu erstellen, sondern in separaten Klassen, was die Sauberkeit des Codes erhöht.

21. November 2022:

- Wir haben angefangen den Code First Ansatz im Visual Studo 2022 umzusetzen und entsprechende Migrations zu erstellen.

27. November 2022:

- Die Code First Basis für die Vorgehensweise wurde soweit fertiggestellt und die InitialCreate Migration wurde ebenfalls erstellt.

11.Dezember 2022:

Wir sind zusammen nochmals den Projektauftrag durchgegangen und haben gemerkt, dass sich unser Wissen im Bereich SWE erweitert hat und wir ein paar Schritte nachholen müssen. Wir haben gleich im Anschluss das Azure DevOps eingerichtet und uns für die ScrumBan Agile-Methode entschieden. Dann haben wir alle uns bekannten UserStories / Backlog aufgeschrieben. Auch die, welche wir schon erledigt haben.

Laut Projektauftrag haben wir das Planning Poker verwendet um den UserStories / Backlog einen Aufwandswert zuzuweisen. Um sie danach dementsprechen zu priorisieren.

Ausserdem haben wir uns ein wenig in das CI /CD Prinzip eingelesen um es in Zukunft in Azure DevOps um zusetzen.

Für das nächste Mal wollen wir ein Branching und Merging Konzept erstellen. Dadurch, dass dir Git einige Zeit später kennengelernt haben und die Code First Ansatz von EF schon gemacht haben, gibt es leider dafür keinen Branch und keinen Tag.

Ca. 30. Dezember

Erkenntnise für Branching Merging:

Uns ist noch aufgefallen, dass wir die ersten Teilaufgaben neu committen müssen, da wir nicht die passenden Tags dafür gewählt haben.

Ca. 4-8. Januar

Erkenntnisse für TestingDaten

Wir haben gesehen, dass es verschiedene Möglichkeiten gibt, Testdaten zu erstellen. Wir haben ns vorerst für die Variante Data Seeding migrationBuilder.InsertData() entschieden.Da diese auf Migrationen basiert.

Nach Fehlgeschlagene Versuchen machen wir es nun per manuellen SQL Inserts.

CTE für Artikelgruppen

Wir haben versucht die CTE als Stored Procedure auf der DB speichern, hatten aber probleme die daten aus dieser auszulesen, wir haben dann eine Function genommen und konnten alla ado.net die daten auslesen. Ausserdem mussten wir feststellen, dass die Function auf der Master DB gespeichert wurde und nicht in unser SemesterarbeitDBS.

TreeView aufbauen

Beim Aufbau der TreevView wussten wir nicht in welchem Propertie wir die ID und die ParentId im TreeNode speichern können. Wir haben dann einen Advanced Treenode installiert. Dieser stürtze jeweils ab, wenn wir versuchten radiobuttons als auswahl zu verwenden. Ich habe aus Not dann den ImageIndex und den Tag des TreeNode verwendet, da diese INT annehmen….

MVVM

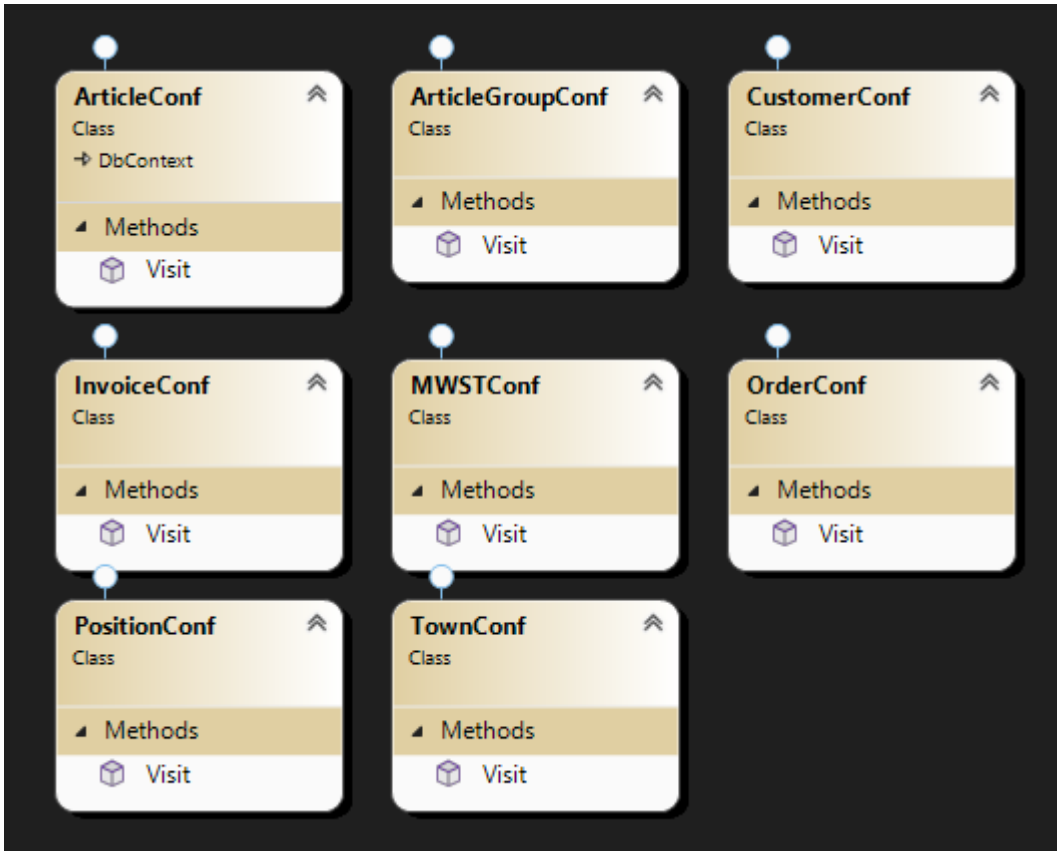
Wir hatten Mühe mit dem MVVM Pattern, da wir am Anfang nicht genau wussten was für was genau zuständig ist und was genau in die einzelnen Klassen reingehört oder eben nicht. Beim Switch von WPF zu WinForms haben wir aber gemerkt, dass wir ausser der View nicht viel anpassen müssen, was uns positiv überrascht hat, obwohl wir wohl nicht alles richtig gemacht haben.

WPF

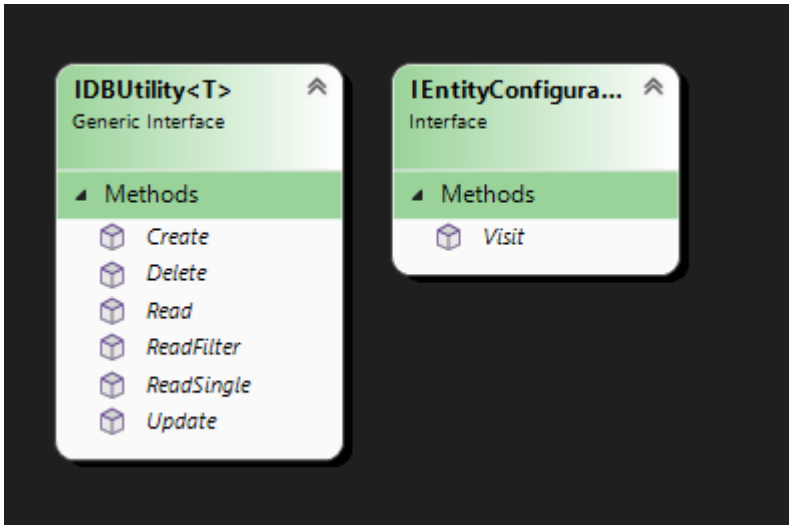
Probleme mit dem Databinding und dem OnPropertyChanged Interface haben uns gezwungen zu WinForms zu wechseln um nicht noch weitere Stunden zu versaumen. Es hat schon vieles Funktioniert, aber es gab zu viele unbekannten im Hintergrund.

Diagramme

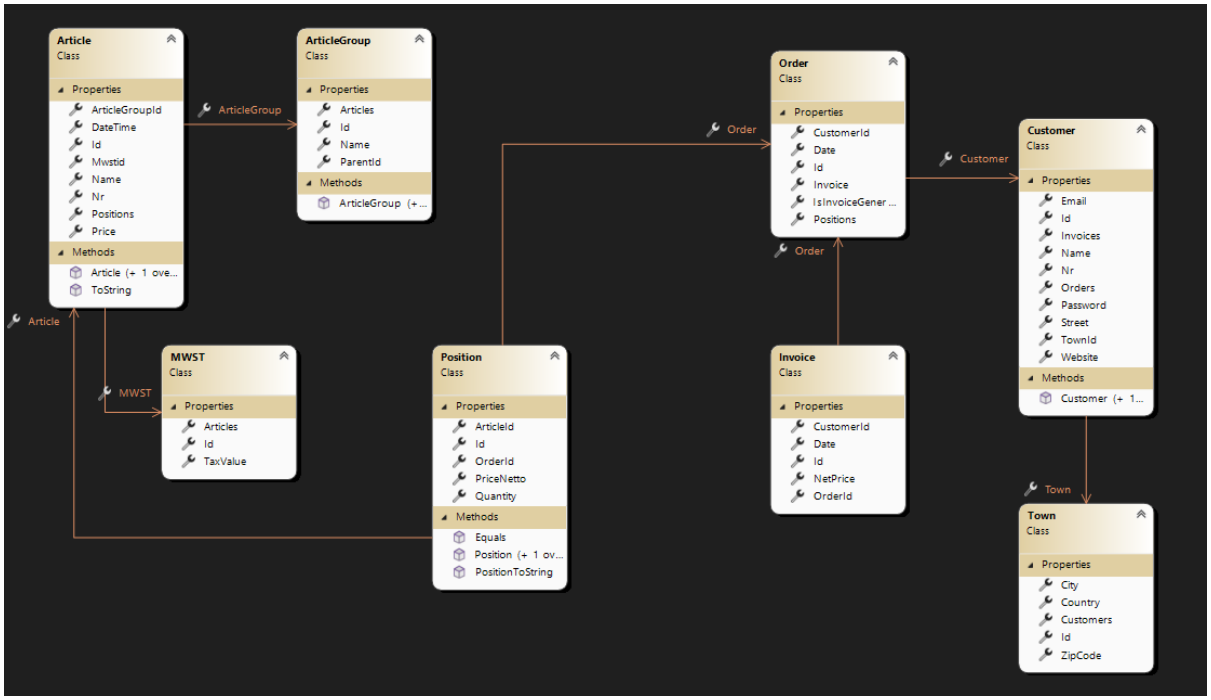
ArticleGroup_Classdiagram:



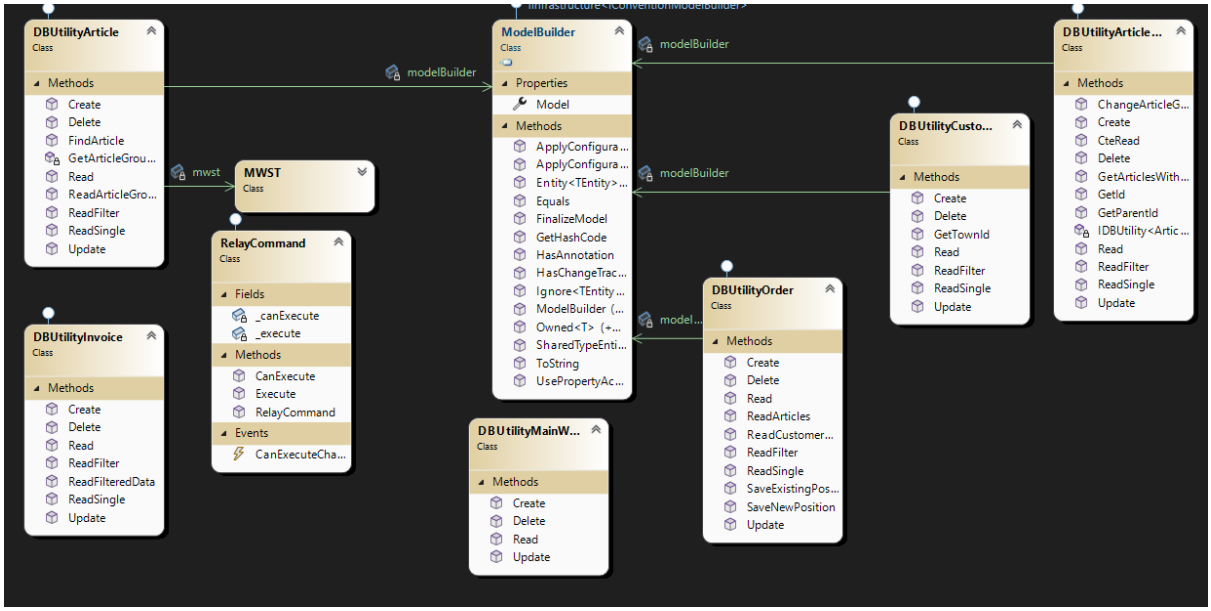
Interfaces_Classdiagram:



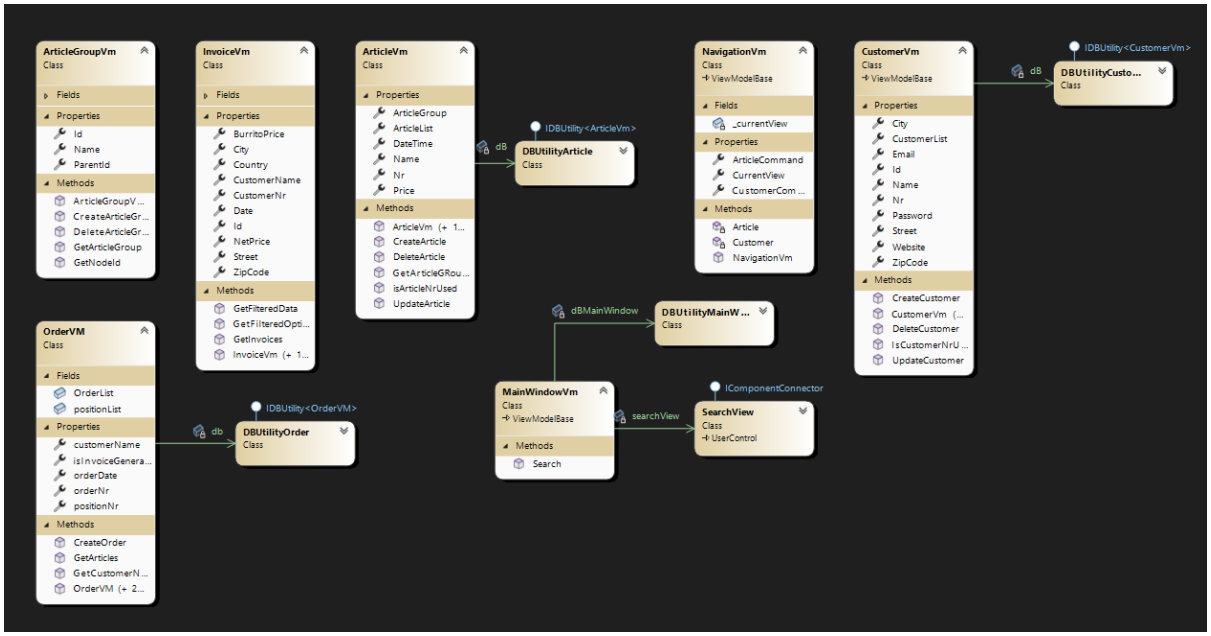
Model_ClassDiagram:



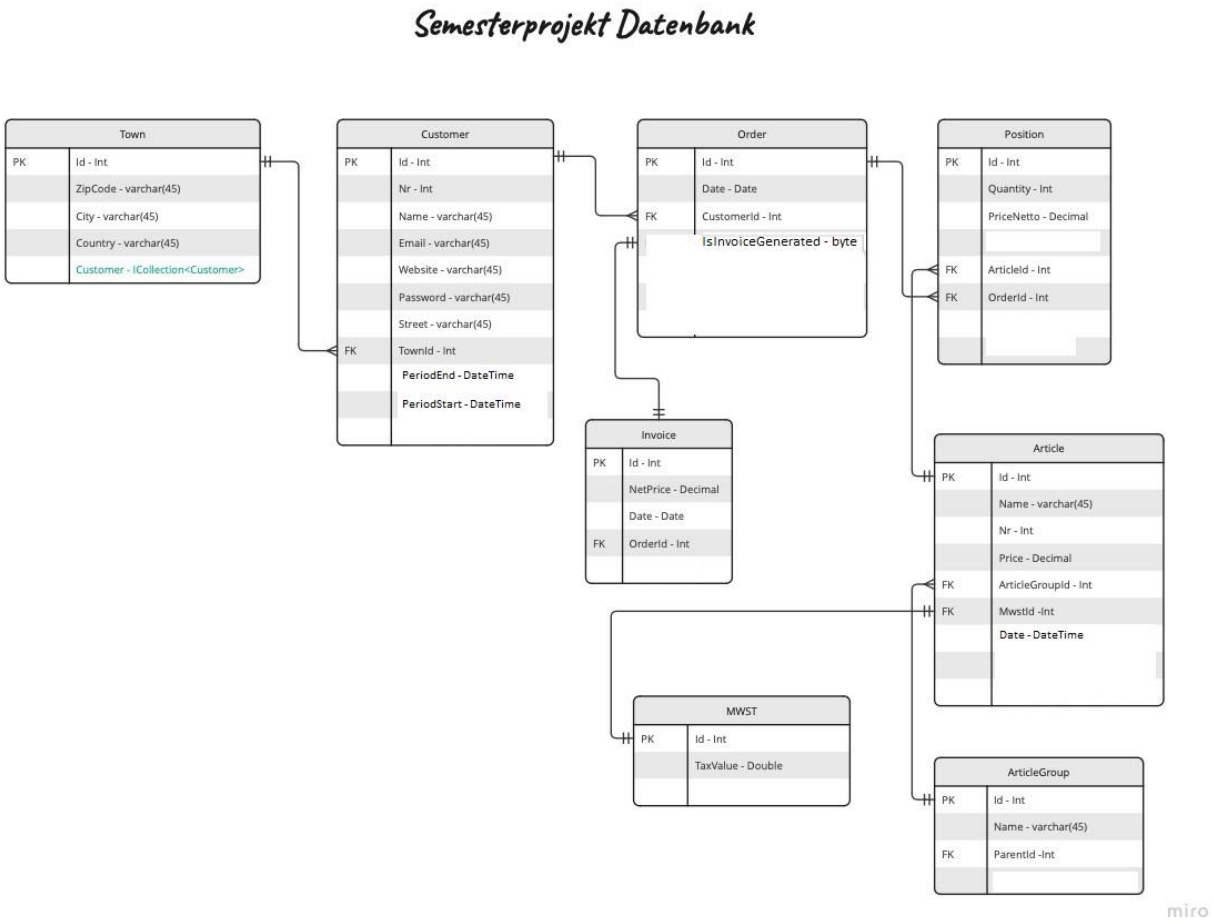
Utilities_ClassDiagram:



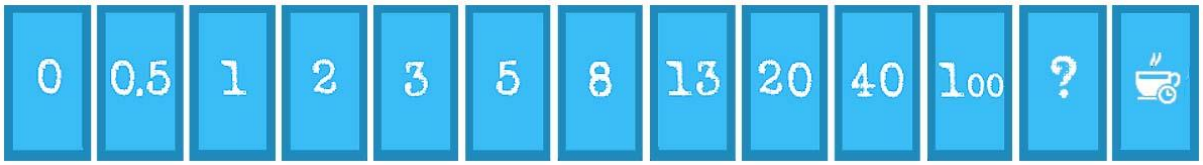
ViewModel_ClassDiagram:



ERM-Diagram



Priorisierungsvorgang – User Stories



Quelle: https://t2informatik.de/wissen-kompakt/planning-poker/

Wir haben uns für die Planning Poker Methode entschieden, da sie uns gerade auf dem Silbertablett präsentiert wurde (Projektauftrag) und unser Erachtens auch am meisten Sinn gemacht hat, da wir für jede UserStory den Durchschnitt unserer Schätzung nehmen konnten. Jeder Teilnehmer des Projekts erhält eine Anzahl von Karten mit einer Zahl drauf. Diese Zahlen sollen die Dauer und Ressourcenverwendung der UserStroy symbolisieren. Je höher die Schätzung desto länger dauert es um die User Story umzusetzen. Unsere Einschätzungen haben wir in dem von uns erstellten Discord-Server durchgeführt.

Unser Ablauf sah dabei wie folgt aus:

- Unser Moderator nennt die UserStory die geschätzt werden muss.
- Alle in unserer Gruppe schätzen selbst die UserStory und weist der User Story den passenden Wert zu.
- Alle posten gleichzeitig Ihre Schätzung im Chat des Discord-Servers.
- Die Gruppenmitglier mit der höchsten und tiefsten Schätzung haben sich dann auf einen Wert geeinigt.

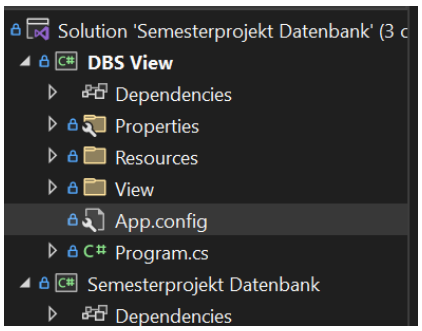
Die Priorisierungswerte sind alle im Azure DevOps ersichtlich.

Inbetriebnahme

https://github.com/ousama681/Semesterprojekt_Datenbank

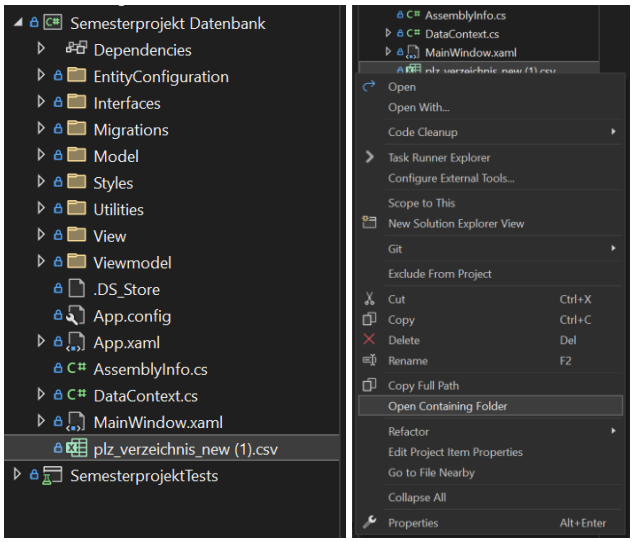
ConnectionString zur DB in app.config im Projekt DBS View anpassen.

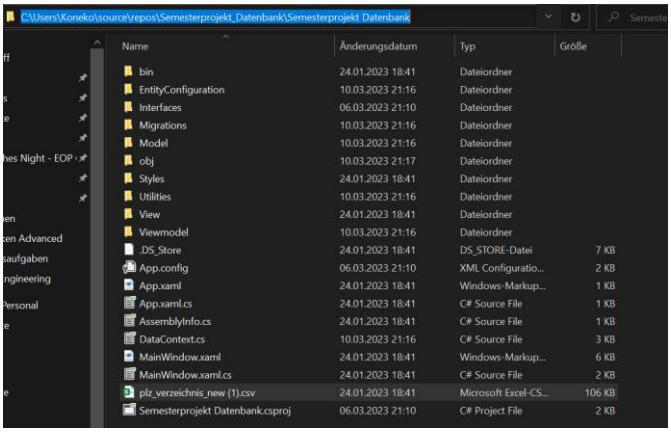
Pfad zur Exceldatei(plz_verzeichnis_new(1).csv) in „value“ in AppSettings einfügen.



Exceldatei(plz_verzeichnis_new(1).csv) befindet sich im Projekt Semesterprojekt_Datenbank.

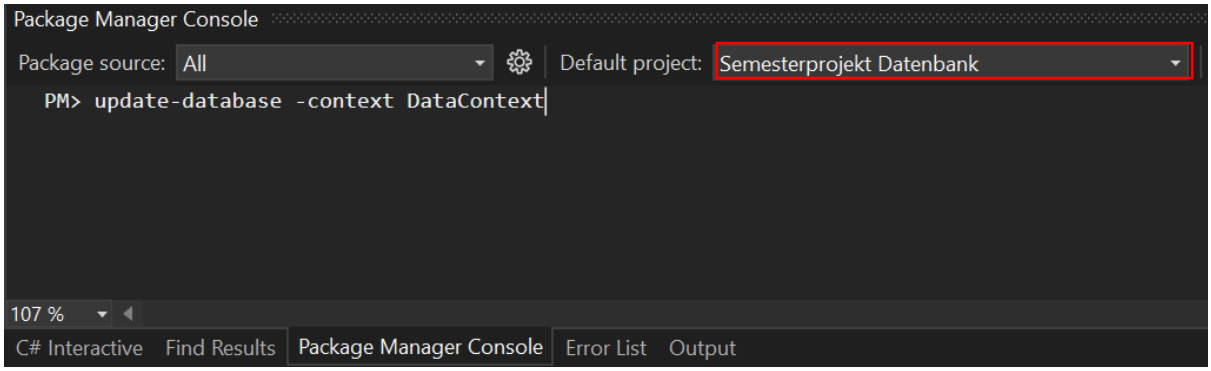
Mit Rechtsklick auf die Datei und „Open Containing Folder“ kann der Explorer die Datei direkt anzeigen.






```
<configuration>
    <connectionStrings>
        <add name="connection"
             connectionString="Server=Koneko\KONEKO; Database=SemesterarbeitDBS; Trusted_Connection=true; Encrypt=false;"
             providerName="System.Data.SqlClient"/>
    </connectionStrings>
    <appSettings>
        <add key="PathZipCodes"
             value="C:\Users\Koneko\source\repos\Semesterprojekt_Datenbank\Semesterprojekt Datenbank\plz_verzeichnis_new (1).csv"/>
    </appSettings>
</configuration>
```

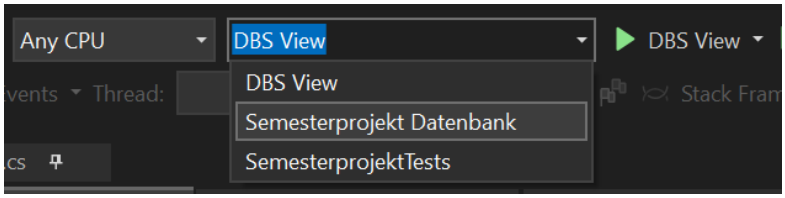
In der Package Manager Console „SemesterProjekt_datenbank“ als default Projekt wählen und folgendes eingeben:



Falls Fehler Configuration System not initialised → Build Solution und DBS View Projekt starten, damit Projekt mal ohne DB starten kann, dann nochmals probieren.

Datenbank sollte nun erstellt sein.

Da wir zuerst mit einem WPF Projekt gestartet haben und später dann auf WinForms geswitched sind, muss das Projekt DBS View gestartet werden.


Any CPU
DBS View
DBS View
vents
Thread:
Stack Fran
DBS View
Semesterprojekt Datenbank
SemesterprojektTests
cs