

**TECHNISCHE DOKUMENTATION**

**Einheitenumrechner**

***Kadrnoschka André***

***11.06.2025***

Inhalt

[**Einführung** ──────────────────────────────────────────────────────── 3](#_Toc200499988)

[**Projektstruktur** 3](#_Toc200499989)

[**Weitere Ordner-Informationen** 4](#_Toc200499990)

[**Klassen UML Diagramme [Allgemein]** 6](#_Toc200499991)

[Beschreibung 6](#_Toc200499992)

[**Datenstruktur** 7](#_Toc200499993)

[Categories 7](#_Toc200499994)

[Doxygen-Beschreibung: 7](#_Toc200499995)

[UnitDefinition 7](#_Toc200499996)

[Doxygen-Beschreibung: 7](#_Toc200499997)

[HistoryEntry 8](#_Toc200499998)

[Doxygen-Beschreibung: 8](#_Toc200499999)

[**SQL-Server-Datenbankdiagramme** 9](#_Toc200500000)

[**Klassen** 10](#_Toc200500001)

[ViewModel 10](#_Toc200500002)

[HistoryViewModel 10](#_Toc200500003)

[SettingViewModel 10](#_Toc200500004)

[UnitViewModel 10](#_Toc200500005)

[Data 10](#_Toc200500006)

[*FileJSONRepository* 10](#_Toc200500007)

[*UnitCalculatorContext* 10](#_Toc200500008)

[Views 11](#_Toc200500009)

[**Programmfunktion** 11](#_Toc200500010)

[Wesentliche Funktion: 11](#_Toc200500011)

[SaveConversion 12](#_Toc200500012)

[Beschreibung 12](#_Toc200500013)

[LoadHistory 13](#_Toc200500014)

[Beschreibung 13](#_Toc200500015)

[Abbildungsverzeichnis 14](#_Toc200500016)

# **Einführung** ────────────────────────────────────────────────────────

Dieses Dokument beschreibt die technische Umsetzung einer WPF-Anwendung zur Einheitenumrechnung mit C# und dem MVVM-Muster. Die Anwendung wurde in Visual Studio entwickelt und erlaubt das Umrechnen, Speichern und Verwalten von Einheiten über eine datei- oder datenbankbasierte Speicherung.

# **Projektstruktur**

────────────────────────────────────────────────────────

**/MVVM\_Einheitenumrechner**

**/Class**

Category.cs

HistoryEntry.cs

UnitDefinitions.cs

**/Data**

JSONFileCreate.cs

**/Images**

… (Images logischerweise)

**/NewFolder**

UnitCalculatorContext.cs

**/ViewModel**

HistoryViewModel.cs

SettingViewModel.cs

UnitViewModel.cs

**/Views**

HistoryView.xaml

SettingView.xaml

UnitView.xaml

**/App.xaml**

**/AssemblyInfo.cs**

**/MainWindow.xaml**

# **Weitere Ordner-Informationen**

────────────────────────────────────────────────────────

|  |  |
| --- | --- |
| **Ordner** | **Zusätzliche Informationen** |
| Class | Enthält Hilfsklassen und gemeinsam genutzte Datenstrukturen. |
| Data | Enthält den Code zum Erstellen/Laden der JSON-Files |
| Images | Speichert alle verwendeten Bilder und Icons der Anwendung. |
| NewFolder | Beinhaltet Datenbankkontext |
| ViewModel | Enthält die Logik zur Verbindung zwischen View und Model (MVVM). |
| Views | Beinhaltet die XAML-Oberflächen für die Benutzeransicht. |

# **Klassen UML Diagramme [Allgemein]**

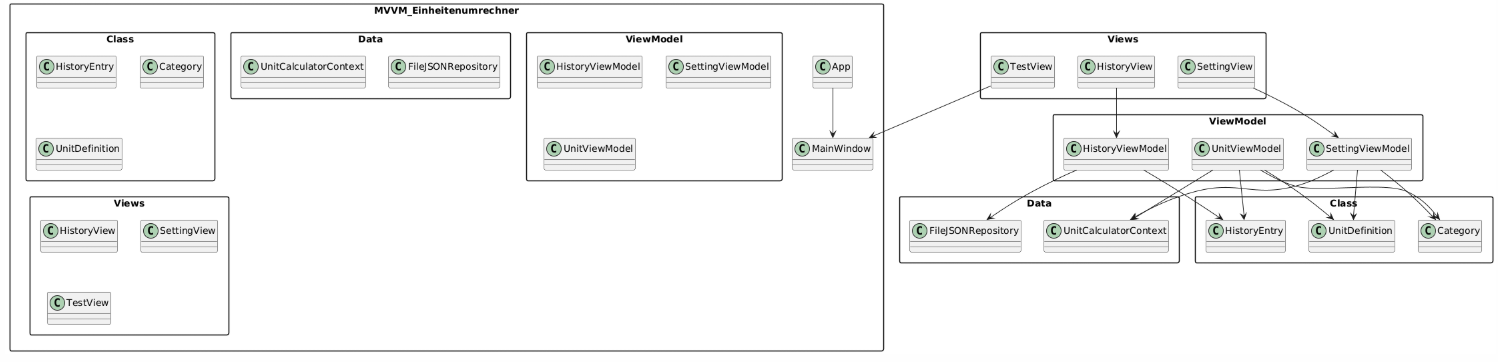
─────────────────────────────────────────────────────────────────────────────────────────

Abbildung 1: Diagrammstruktur (UML)

## Beschreibung

Das Diagramm zeigt die Struktur deines Projekts mit den wichtigsten Klassen, gruppiert nach ihren Ordnern. Außerdem sind die zentralen Abhängigkeiten und Verbindungen zwischen den ViewModels, Views und Datenklassen dargestellt.

# **Datenstruktur**

────────────────────────────────────────────────────────

Die Datenstruktur des Einheitenrechners besteht aus **drei** Klassen.

## Categories

* Repräsentiert eine Kategorie von Einheiten, z.B. "Länge" oder "Gewicht".
* Hat eine 1:n-Beziehung zu UnitDefinition (eine Kategorie hat viele Einheiten).
* Hat eine 1:n-Beziehung zu HistoryEntry (eine Kategorie kann viele Umrechnungseinträge haben).

### Doxygen-Beschreibung:

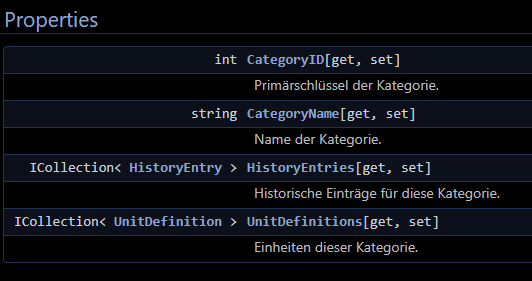


Abbildung 2: Categories-Klasse

## UnitDefinition

* Definiert eine einzelne Einheit innerhalb einer Kategorie, z.B. "Meter" oder "Kilogramm".
* Gehört zu genau einer Category (n:1-Beziehung).

## Doxygen-Beschreibung:



Abbildung 3: UnitDefinitions-Klasse

## HistoryEntry

* Speichert eine Umrechnungshistorie mit Angaben wie Eingabewert, Ergebnis, beteiligte Einheiten und Kategorie.
* Gehört zu genau einer Category (n:1-Beziehung).

## Doxygen-Beschreibung:

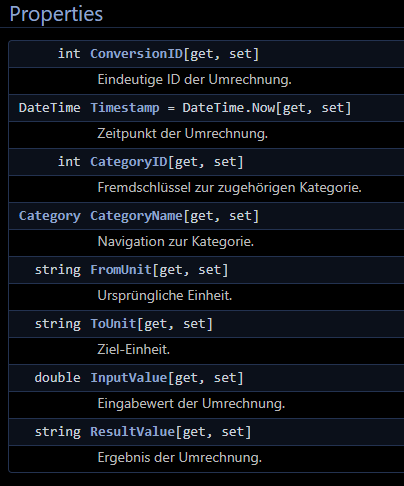


Abbildung 4: HistoryEntry-Klasse

# **SQL-Server-Datenbankdiagramme**

────────────────────────────────────────────────────────

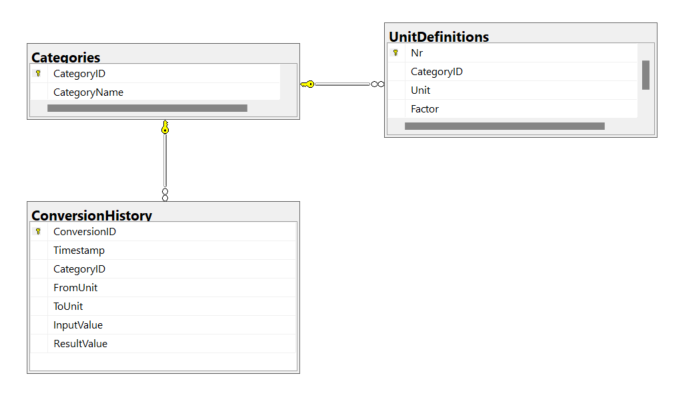


Abbildung 5: Datenbankdiagramme

Die Struktur wird durch diese Befehle erstellt:

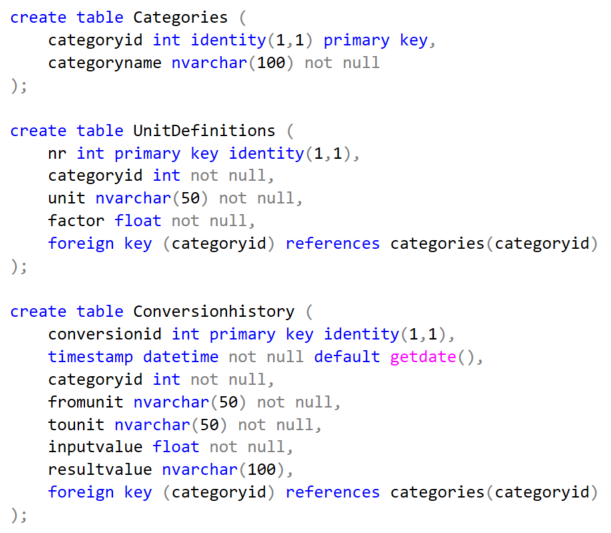


Abbildung 6: DatenbankCode

# **Klassen**

────────────────────────────────────────────────────────

## ViewModel

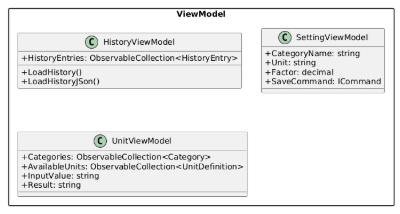


Abbildung 7: ViewModel

### HistoryViewModel

Bindet die Daten aus der Umrechnungshistorie an die zugehörige Oberfläche (View) und bereitet sie auf.

### SettingViewModel

Verwaltet Logik und Validierung zum Speichern neuer Kategorien und Einheiten.

### UnitViewModel

Führt Umrechnungen durch und speichert sie je nach Einstellung in Datenbank oder JSON.

## Data

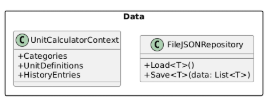


Abbildung 8: Data

### *FileJSONRepository*

Dient als generischer Helfer zum Speichern und Laden beliebiger Datenobjekte in JSON-Dateien.

### *UnitCalculatorContext*

Ist der Entity Framework-Datenkontext, der Datenbanktabellen wie Kategorien, Einheiten und Historie abbildet.

## Views

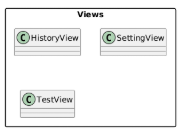


Abbildung 9: Views

**HistoryView**: WPF-Oberfläche zur Anzeige der gespeicherten Umrechnungen.

**SettingView**: WPF-Oberfläche zum Eingeben und Speichern neuer Einheiten und Kategorien.

**UnitView**: Stellt eine einfache Benutzeroberfläche zur Navigation zu den Hauptfunktionen dar.

# **Programmfunktion**

────────────────────────────────────────────────────────

## Wesentliche Funktion:

Die Anwendung ermöglicht die Umrechnung von Einheiten in verschiedenen Kategorien (z. B. Länge, Masse) unter Einsatz des MVVM-Musters. Nutzer können eine Kategorie sowie Ausgangs- und Zieleinheit auswählen und einen Wert eingeben. Die Logik im UnitViewModel berechnet auf Basis hinterlegter Umrechnungsfaktoren das Ergebnis und stellt es sofort dar.

Zur Speicherung der Umrechnungsverläufe kann zwischen **zwei** Modi gewählt werden:

* Datenbankbasiert (SQL) mittels Entity Framework,
* oder dateibasiert (JSON) über FileJSONRepository.

Ein integrierter Timer sorgt dafür, dass Eingaben verzögert gespeichert werden, um unnötige Schreibvorgänge zu vermeiden.

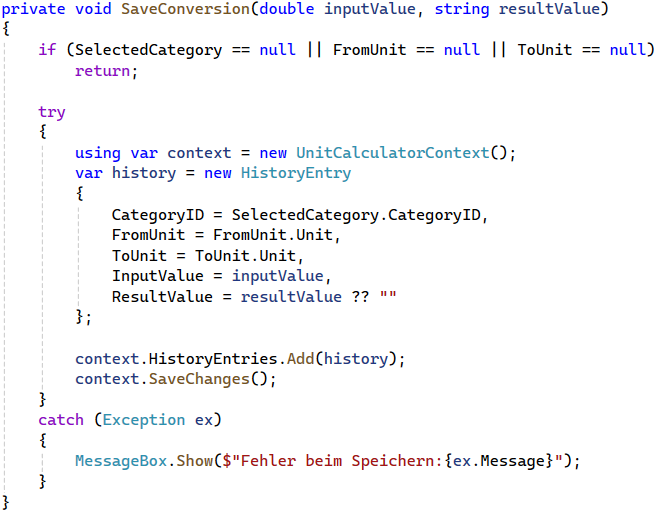
SaveConversion

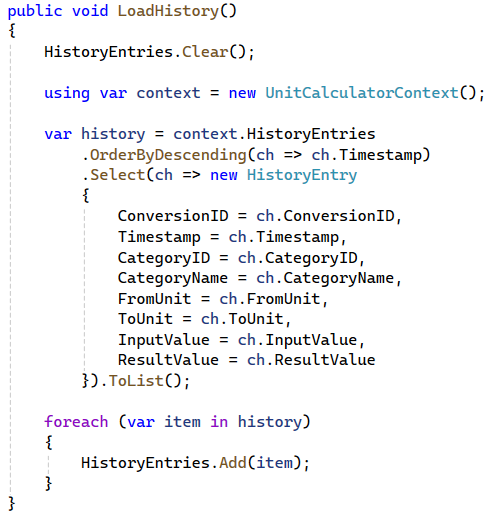
Abbildung 10: SaveConversion

### Beschreibung

Diese Methode speichert eine Umrechnung in die SQL-Datenbank.

Sie erstellt ein neues HistoryEntry-Objekt mit Informationen zur Kategorie, Ausgangs- und Zieleinheit, Eingabewert und Ergebnis. Anschließend wird dieses Objekt mit Entity Framework in die Tabelle ConversionHistory eingefügt und dauerhaft gespeichert.

## LoadHistory



### Beschreibung

Diese Methode lädt gespeicherte Umrechnungen aus der SQL-Datenbank (bzw. alternativ aus der JSON-Datei, wenn im JSON-Modus).

Sie liest alle vorhandenen HistoryEntry-Objekte, wandelt sie ggf. in eine ObservableCollection um und stellt sie der Benutzeroberfläche zur Anzeige bereit – etwa in der HistoryView.

# Abbildungsverzeichnis

[Abbildung 1: Diagrammstruktur (UML) 5](file:///C:\Users\Andre.DESKTOP-OIR8S4A\Downloads\Technische%20Dokumentation.docx#_Toc200499754)

[Abbildung 2: Categories-Klasse 6](file:///C:\Users\Andre.DESKTOP-OIR8S4A\Downloads\Technische%20Dokumentation.docx#_Toc200499755)

[Abbildung 3: UnitDefinitions-Klasse 6](#_Toc200499756)

[Abbildung 4: HistoryEntry-Klasse 7](#_Toc200499757)

[Abbildung 5: Datenbankdiagramme 8](#_Toc200499758)

[Abbildung 6: DatenbankCode 8](#_Toc200499759)

[Abbildung 7: ViewModel 9](#_Toc200499760)

[Abbildung 8: Data 9](#_Toc200499761)

[Abbildung 9: SaveConversion 10](#_Toc200499762)