



MANAGEMENT- UND INFORMATIONSSYSTEME FÜR DEN HANDEL

LS

INFOWARE

WARENWIRTSCHAFT MIT SYSTEM

# WPF-Translate

Desktopanwendung zum Übersetzen von WPF-Ressourcen

Jan Wiesemann

---

21.06.2018

# Inhalt

---

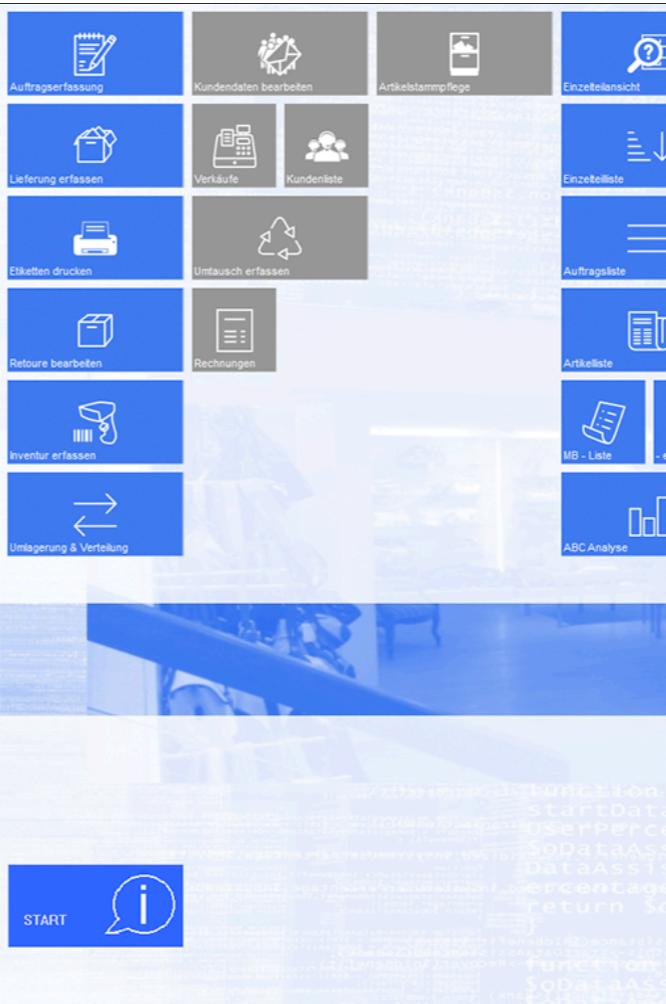
- Das Unternehmen
- Warum das Projekt?
- Planung
- Implementierung
- Ausblick
- Fazit

- Das Unternehmen
- Warum das Projekt?
  - Beispiel einer XAML-Datei
  - Handarbeit vs. WPF-Translate
  - Verkalkuliert Projektkosten
  - Amortisierung
- Planung
  - Oberflächenlayout
  - Bibliotheken
- Implementierung
  - XAML lesen
  - XAML schreiben
  - MVVM
    - Die View
    - Das ViewModel
    - Das Model
  - Google Translate
- Ausblick
- Fazit

# Das Unternehmen

## Landau Software GmbH

- 2002 Neugründung zur GmbH
- LS-INFOMAN
- LS-INFOCASH
- 5 Mitarbeiter
- Kunden z.B.
  - Sporthaus Kaps
  - Etuis-Mertl
  - Leder Meid



1984 Gründung -> 2002 Neugründung zur GmbH

80 Kunden in 8 Ländern

Eingehen auf Bild

# Inhalt

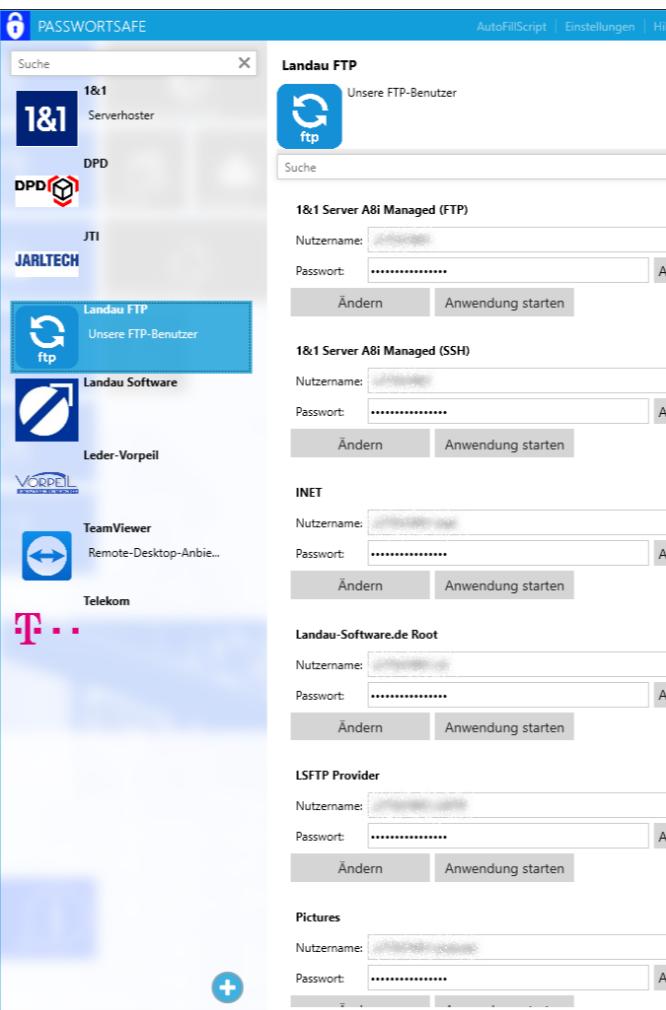
## Warum das Projekt?

- Das Unternehmen
- Warum das Projekt?
  - Was ist WPF?
  - Beispiel einer XAML-Datei
  - Handarbeit vs. WPF-Translate
  - Kalkulierte Projektkosten
  - Amortisierung
- Planung
- Implementierung
- Ausblick
- Fazit

# Warum das Projekt?

## Was ist WPF?

- Windows Presentation Foundation
- Fensterframework für .NET
- Schnelles rendern durch DirectX
- XAML
- Flexibel
- Zukunft von Windows



XAML -> Extensible Application Markup Language

Flexibel

Zukunft von Windows -> UWP Apps

# Warum das Projekt?

## Beispiel einer XAML-Datei

### Wörterbuch Definition

```
<ResourceDictionary
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:sys="clr-namespace:System;assembly=mscorlib">

    <ResourceDictionary.MergedDictionaries>
        <ResourceDictionary Source="pack://application:///LSDOTNETINFORMAN;component/Resources/Strings_de-de.xaml" />
        <ResourceDictionary Source="pack://application:///PermissionManager;component/Strings.xaml" />
    </ResourceDictionary.MergedDictionaries>

    <SolidColorBrush x:Key="LabelBackgroundColorBrush" Color="#FFC0FFFF" />

    <Style BasedOn="{StaticResource {x:Type Label}} TargetType="{x:Type Label}">
        <Setter Property="VerticalContentAlignment" Value="Center" />
        <!--<Setter Property="FontSize" Value="{DynamicResource FontSizeKey}" /-->
    </Style>

    <x:String x:Key="abbrechen">Abbrechen</x:String>

    <x:String x:Key="reparaturSpeichernFehler" xml:space="preserve">Die Änderungen können nicht in die Reparatur übertragen werden!
    Fehler: {0}</x:String>

    <x:Static x:Key="vText5Doppelpunkt" Member="sys:String.Empty" />
</ResourceDictionary>
```

Namespaces

Unter Wörterbücher

Nicht String-Werte

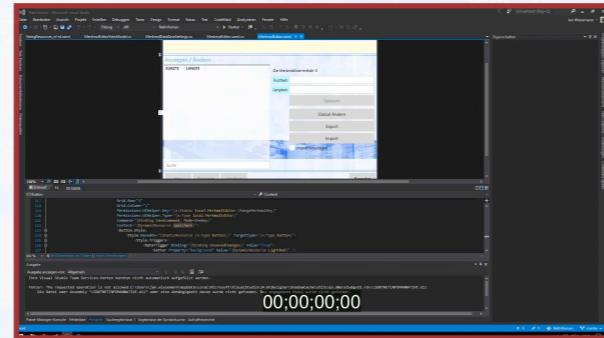
String-Werte

Leerer String-Wert

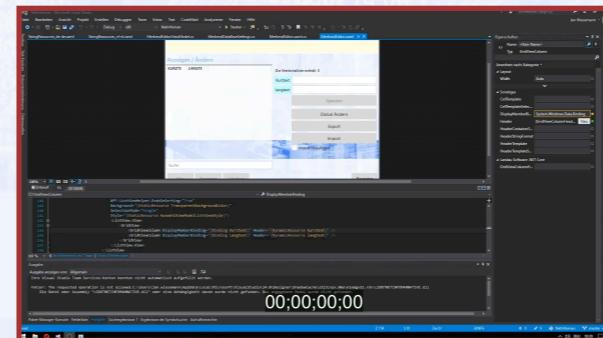
- Wörterbuch Definition
- Namespaces
- Unter Wörterbücher
- Nicht String-Werte
- String-Werte
- Leere String-Werte

# Warum das Projekt?

## Handarbeit vs. WPF-Translate



Manuelles Ändern



WPF-Translate

Ändern von zwei Dateien in NetInfoMan Resource Editor.

Manuell: 1:06

WPF-Translate: 0:35

Fast Doppelte Geschwindigkeit

# Warum das Projekt?

## Kalkulierte Projektkosten

Vorgang	Mitarbeiter	Zeit	Personal	Ressourcen	Gesamt
Entwicklungskosten	1x AZ	70h	700,00 €	1.050,00 €	1.750,00 €
Code Review	1x MA	2h 30m	175,00 €	37,50 €	212,50 €
Abnahme	1x MA	0h 30m	35,00 €	7,50 €	42,50 €
Projektkosten gesamt 2.005,00 €					



Kosten	Kosten/Stunde
Auszubildender	10 €
Mitarbeiter	70 €
Ressourcenkosten (z.B. Strom)	15 €

Projektkosten 2005€

Bestehend aus

70h Azubi 10€/Stunde

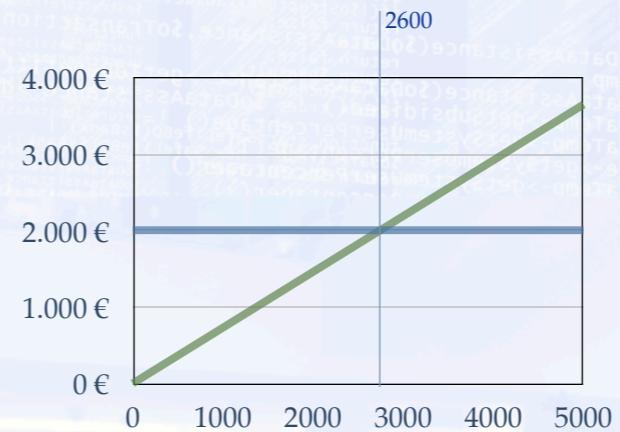
2h30m + 30m Mitarbeiter 70€/Stunde

70h + 2h30m + 30m Ressourcen 15€/Stunde

# Warum das Projekt?

## Amortisierung

Aktion	Kosten
WPF-Translate	0,85 €
Manuelles Bearbeiten	1,56 €
Kostenersparnis	0,71 €



Personalkosten		Bearbeitungsdauer		Projektkosten	
Kosten	Kosten/Stunde	Ändern	Dauer	Bezeichnung	Kosten
Auszubildender	10 €	WPF-Translate	0m 36s	Kosten	2.005,00 €
Mitarbeiter	70 €	Manuelles Bearbeiten	1m 6s		
Ressourcenkosten (z.B. Strom)	15 €				

~0.71 € Ersparnis pro Bearbeitung

~2600 Bearbeitungen bis Kostendeckung

Schneller bei mehrer Einträgen

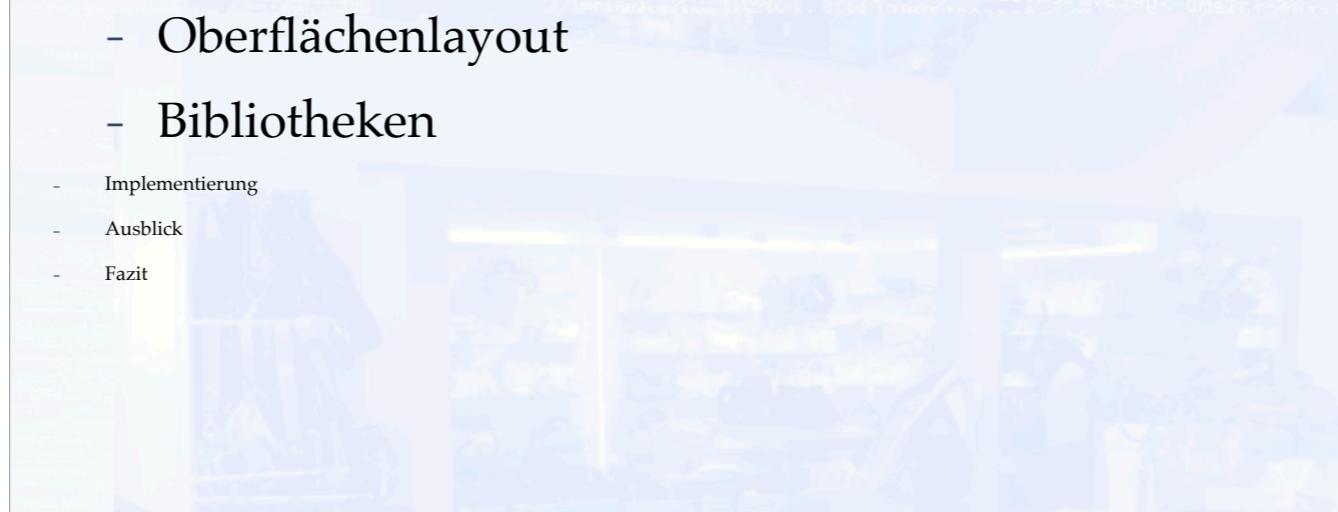
Ca 30.000 String in Infocash (pro Sprache; 8 Sprachen)

# Inhalt

## Planung

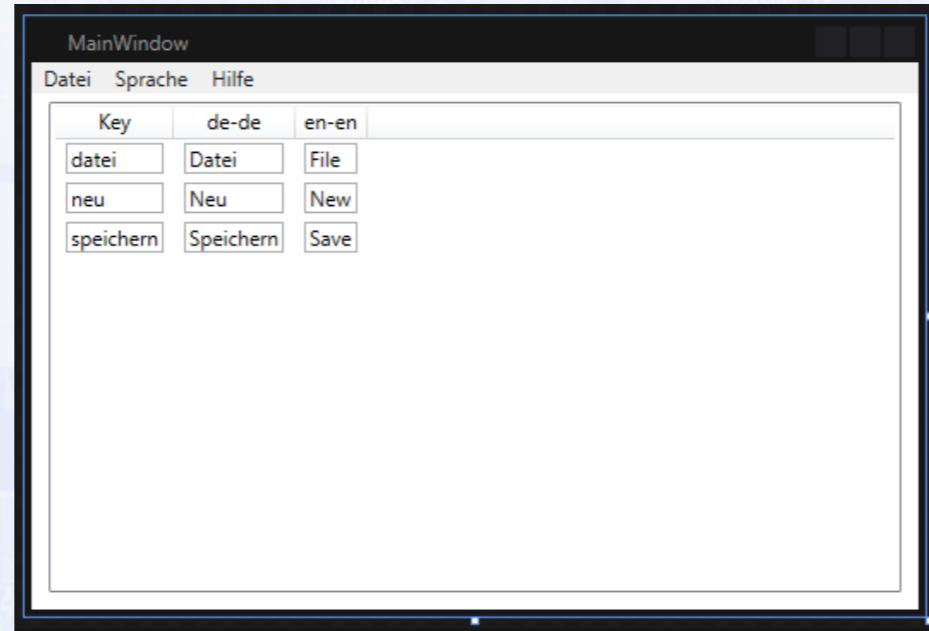
---

- Das Unternehmen
- Warum das Projekt?
- Planung
  - Oberflächenlayout
  - Bibliotheken
- Implementierung
- Ausblick
- Fazit



# Planung

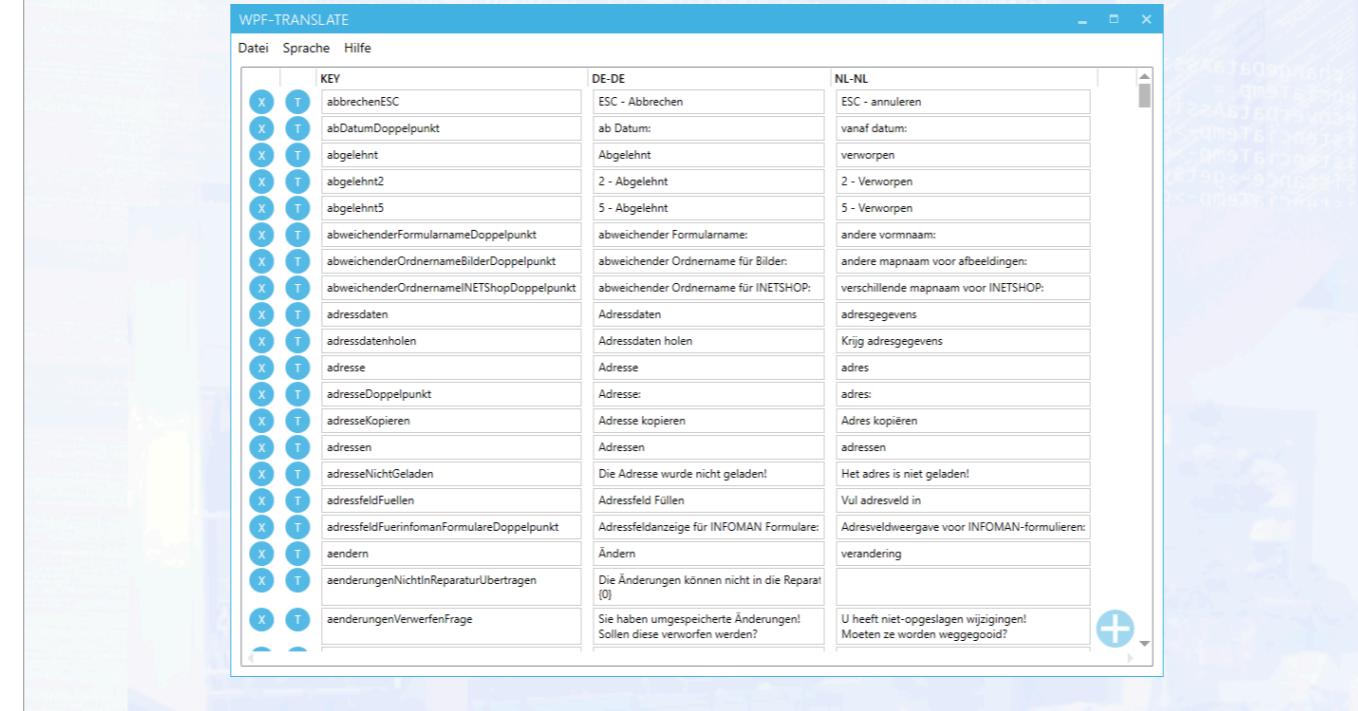
## Oberflächenlayout



**Halbzeit (7:30min)**

# Planung

## Oberflächenlayout



UI ansprechen

# Planung

## Bibliotheken

- LSDOTNETCORE
- JSON.NET
- DocFX
- MahApps.Metro
- MahApps.Metro.Resources
- ControlzEx

LSDOTNETCORE

RelayICommand, NotifyBase

JSON.NET

JSON für Google Translate antworten

ControlzEx

Benötigt für MahApps beinhaltet Anbindungen an WinApi und fixes für WPF fehler

DocFX

Dokumentations generator

MahApps.Metro

MetroWindow und UI Definitionen

MahApps.Metro.Resources

Ressourcen (Icons)

# Inhalt

## Implementierung

- Das Unternehmen
- Warum das Projekt?
- Planung
- **Implementierung**
  - XAML lesen
  - XAML schreiben
  - MVVM
    - Die View
    - Das ViewModel
    - Das Model
  - Anbindung an Google Translate
- Ausblick
- Fazit

# Implementierung

## XAML lesen

1. Öffnen
2. Root lesen
3. Namespaces lesen
4. Durchlaufen der Unterelemente
  1. Merged
  2. String
  3. Leerer String
  4. Sonstige
5. Schließen

```
<ResourceDictionary
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/pre
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:sys="clr-namespace:System;assembly=mscorlib">
    <ResourceDictionary.MergedDictionaries>
        <ResourceDictionary Source="pack://application:,,,/
            <ResourceDictionary Source="pack://application:,,,/
                </ResourceDictionary.MergedDictionaries>
                <sys:String x:Key="abbrechen">Abbrechen</sys:String>
                <sys:String x:Key="abbrechenESC">ESC – Abbrechen</sys:S
                <sys:String x:Key="abDatumDoppelpunkt">ab Datum:</sys:S
                <sys:String x:Key="abgelehnt">Abgelehnt</sys:String>
                <sys:String x:Key="abgelehnt2">2 – Abgelehnt</sys:Strin
                <sys:String x:Key="abgelehnt5">5 – Abgelehnt</sys:Strin
                <sys:String x:Key="abweichenderFormularnameDoppelpunkt"
                    <sys:String x:Key="abweichenderOrdnernameBilderDoppel
                    <sys:String x:Key="abweichenderOrdnernameINETShopDoppel
                    <sys:String x:Key="adressdaten">Adressdaten</sys:Strin
                    <sys:String x:Key="adressdatenholen">Adressdaten holen<
                    <sys:String x:Key="adresse">Adresse</sys:String>
                    <sys:String x:Key="adresseDoppelpunkt">Adresse:</sys:St
                    <sys:String x:Key="adresseKopieren">Adresse kopieren</s
                    <sys:String x:Key="adressen">Adressen</sys:String>
                    <sys:String x:Key="adresseNichtGeladen">Die Adresse wur
                    <sys:String x:Key="adressfeldFuellen">Adressfeld Füllen<
                    <sys:String x:Key="adressfeldFuerInfomanFormulareDoppel
                    <sys:String x:Key="aendern">Ändern</sys:String>
                    <sys:String x:Key="aenderungenNichtInReparaturÜbertrage
{0}</sys:String>
                    <sys:String x:Key="aenderungenVerwerfenFrage" xml:space
Sollen diese verworfen werden?</sys:String>
                    <sys:String x:Key="aenderungenVerwerfenFragezeichen">Än
                    <sys:String x:Key="aenderungstextUndAdressen nicht Ausgef
                    <sys:String x:Key="aktuellerKundeDoppelpunkt">Aktueller
                    <sys:String x:Key="aktuellerReklamationsstatus">Aktuell
                    <sys:String x:Key="alle">Alle</sys:String>
```

## Erklären und markieren

- 1.Öffnen
- 2.Root Lesen
- 3.Namespaces lesen
- 4.Durchlaufen Unterelemente
  - 1.Merged
  - 2.String
  - 3.Leerer String
  - 4.Sonstige
- 5.Schließen
- 6.Anzeigen

# Implementierung

## XAML schreiben

1. Memory Stream erstellen
2. Root schreiben
3. Namespaces schreiben
4. Merged schreiben
5. Einträge schreiben
  1. String
  2. Leerer String
  3. Sonstige
6. Datei öffnen
7. MemoryStream formatiert in Datei schreiben
8. Datei und MemoryStream schließen

```
<ResourceDictionary
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/pre
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:sys="clr-namespace:System;assembly=mscorlib">
    <ResourceDictionary.MergedDictionaries>
        <ResourceDictionary Source="pack://application:,,,/
            <ResourceDictionary Source="pack://application:,,,/
                </ResourceDictionary.MergedDictionaries>
                <sys:String x:Key="abbrechen">Abbrechen</sys:String>
                <sys:String x:Key="abbrechenESC">ESC – Abbrechen</sys:S
                <sys:String x:Key="abDatumDoppelpunkt">ab Datum:</sys:S
                <sys:String x:Key="abgelehnt">Abgelehnt</sys:String>
                <sys:String x:Key="abgelehnt2">2 – Abgelehnt</sys:Strin
                <sys:String x:Key="abgelehnt5">5 – Abgelehnt</sys:Strin
                <sys:String x:Key="abweichenderFormularnameDoppelpunkt"
                    <sys:String x:Key="abweichenderOrdnernameBilderDoppel
                    <sys:String x:Key="abweichenderOrdnernameINETShopDoppel
                    <sys:String x:Key="adressdaten">Adressdaten</sys:Strin
                    <sys:String x:Key="adressdatenholen">Adressdaten holen<
                    <sys:String x:Key="adresse">Adresse</sys:String>
                    <sys:String x:Key="adresseDoppelpunkt">Adresse:</sys:St
                    <sys:String x:Key="adresseKopieren">Adresse kopieren</s
                    <sys:String x:Key="adressen">Adressen</sys:String>
                    <sys:String x:Key="adresseNichtGeladen">Die Adresse wur
                    <sys:String x:Key="adressfeldFuellen">Adressfeld Füllen<
                    <sys:String x:Key="adressfeldFuerInfomanFormulareDoppel
                    <sys:String x:Key="aendern">Ändern</sys:String>
                    <sys:String x:Key="aenderungenNichtInReparaturÜbertrage
{0}</sys:String>
                    <sys:String x:Key="aenderungenVerwerfenFrage" xml:space
Sollen diese verworfen werden?</sys:String>
                    <sys:String x:Key="aenderungenVerwerfenFragezeichen">Än
                    <sys:String x:Key="aenderungstextUndAdressen nicht Ausgef
                    <sys:String x:Key="aktuellerKundeDoppelpunkt">Aktueller
                    <sys:String x:Key="aktuellerReklamationsstatus">Aktuell
                    <sys:String x:Key="alle">Alle</sys:String>
```

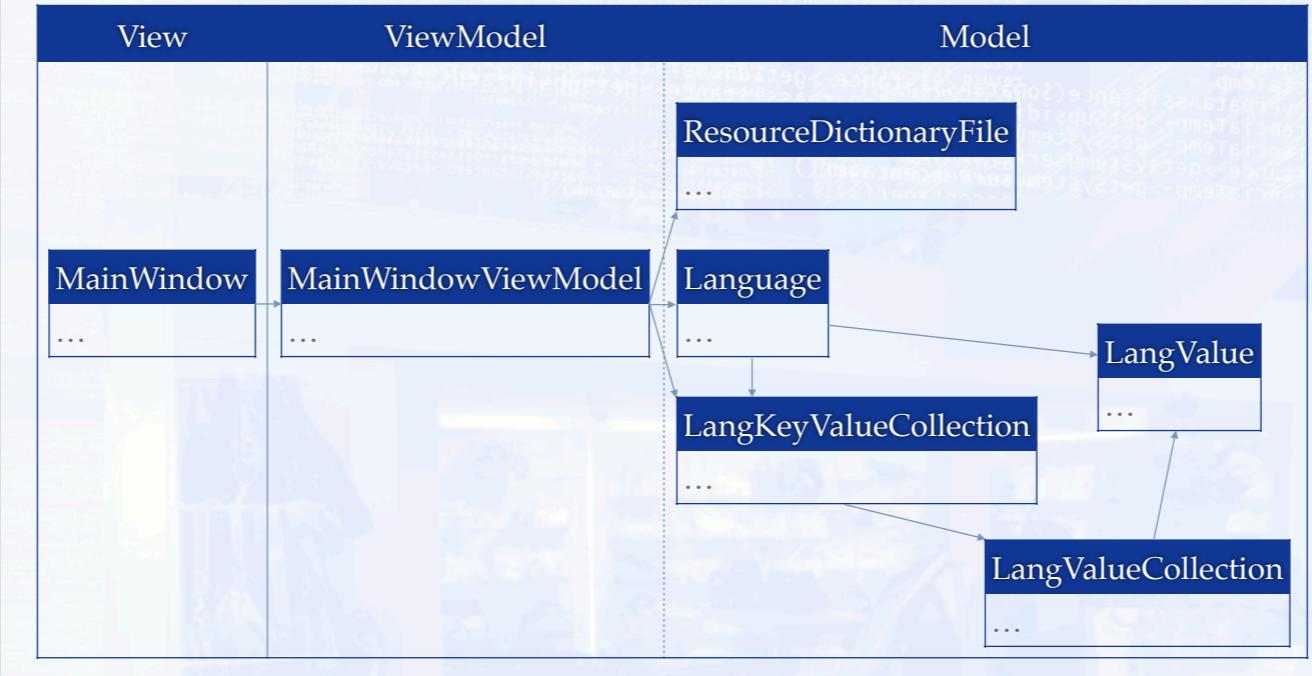
## Erklären und markieren

- 1.Memory Stream erstellen
- 2.Root Schreiben
- 3.Namespaces schreiben
- 4.Merged schreiben
- 5.Einträge schreiben
  - 1.String
  - 2.Leerer String
  - 3.Sonstige
- 6.Datei öffnen
- 7.MemoryStream formatiert in Datei Schreiben
- 8.Datei und MemoryStream Schließen

Warum nicht direkt in Datei geschrieben?

# Implementierung

## MVVM



View ist UI

ViewModel beinhaltet alle Informationen für die View und kapselt Model

Model Daten für UI

ViewModel und Model brauchen INotifyPropertyChanged (hier NotifyBase bzw ObservableCollection<T>)

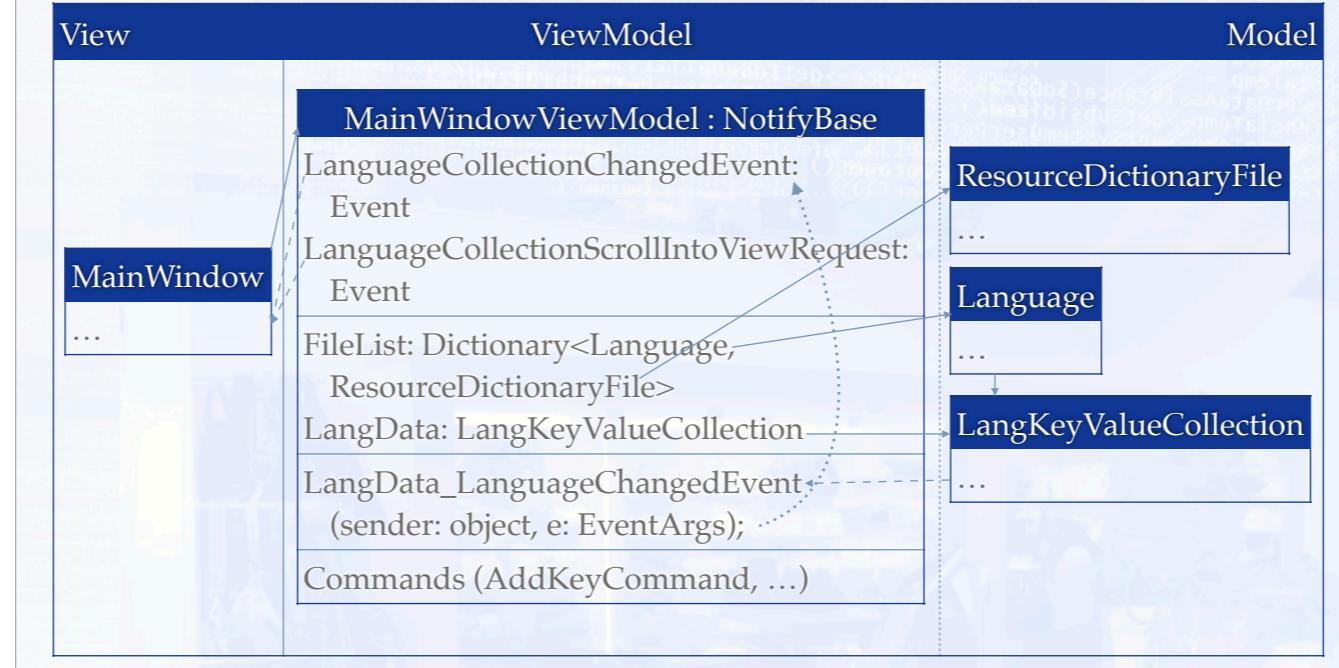
# Implementierung

## MVVM die View



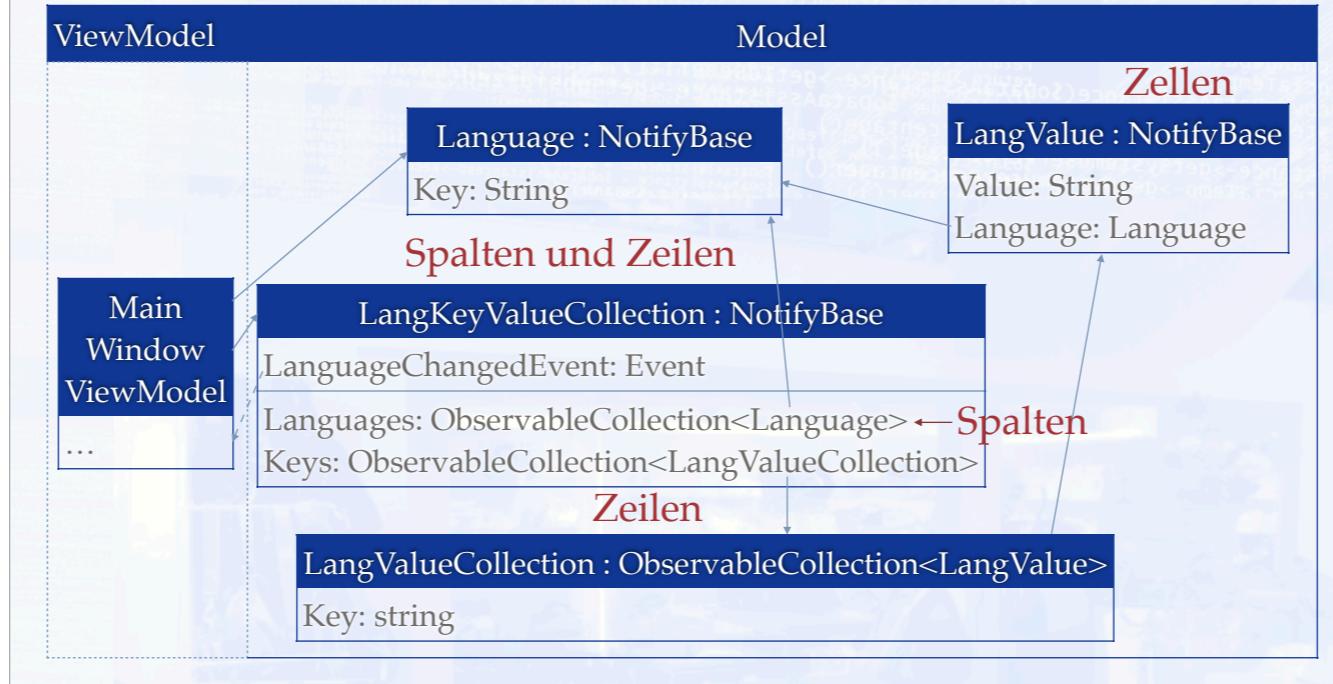
# Implementierung

## MVVM das ViewModel



# Implementierung

## MVVM das Model

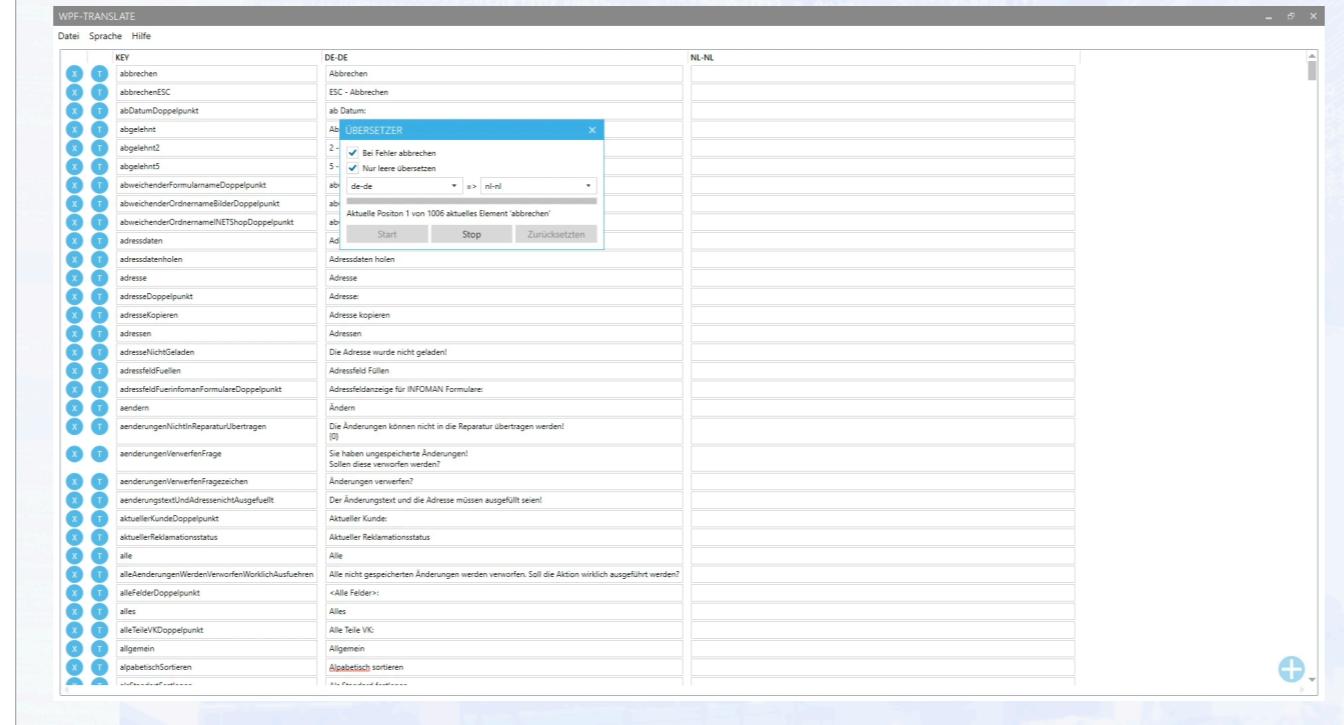


Language

Key -> LCID -> Windows Language Code Identifier

# Implementierung

## Anbindung an Google Translate



Nicht offiziell daher Überarbeitung nötig

# Ausblick

---

- Import und Export für CSV-Dateien
- Überarbeitung Google Translate
- Öffnen und Speichern von Win32 Ressourcen





# Fazit

Erfahrungen in Planung und Durchführung gesammelt

Planung ist wichtig

Ein blick in externe Bibliotheken

Kosten verringert von 2005€ auf 1962,50€ durch -30min code Review (-42,50€)

Leicht Abweichungen im Zeitplan:

Startphase 3h -> 3,5h

Projektplanung 13h -> 12,5h

Implementierung 34h -> 35h

Abschluss 20h -> 19h

Keine Auswirkungen auf Projekt



MANAGEMENT- UND INFORMATIONSSYSTEME FÜR DEN HANDEL

**LS** **INFOWARE**

WARENWIRTSCHAFT MIT SYSTEM

Vielen Dank für Ihre  
Aufmerksamkeit

## WPF-Translate

Desktopanwendung zum Übersetzen von WPF-Ressourcen

Jan Wiesemann

---

---

# Abweichungen Zeitplan

Projektphase	Soll	Ist	Differenz
Startphase	3h	3h 30m	0h 30m
Projektplanung	13h	12h 30m	-0h 30m
Implementierungsphase	34h	35h	1h 0m
Abschlussphase	20h	19h	-1h 0m
Gesamt 70 Stunden		70 Stunden	0 Stunden

# DocFX

## Class Language

Stellt eine Sprache dar

### Inheritance

↳ System.Object

↳ Language

Namespace: de.LandauSoftware.WPFTranslate

Assembly: WPF-Translate.dll

### Syntax

```
public class Language : NotifyBase
```

### Constructors

#### Language()

Initialisiert eine neue Instanz der Sprache

##### Declaration

```
public Language()
```

#### Language(String)

Initialisiert eine neue Instanz der Sprache

##### Declaration

```
public Language(string key)
```

### Parameters

Type	Name	Description
System.String	key	Sprachkey z.B. en-us

### Properties

#### LangKey

Ruft den Sprachkey ab oder setzt diesen

##### Declaration

```
public string LangKey { get; set; }
```

##### Property Value

Type	Description
System.String	

## Class LangValueCollection

Stellt eine Samlung an Werten dar

### Inheritance

↳ System.Object

↳ LangValueCollection

Namespace: de.LandauSoftware.WPFTranslate

Assembly: WPF-Translate.dll

### Syntax

```
public class LangValueCollection : ObservableCollection<LangValue>
```

### Constructors

#### LangValueCollection(String, IEnumerable<Language>)

Erstellt eine neue Samlung

##### Declaration

```
public LangValueCollection(string key, IEnumerable<Language> langs)
```

### Parameters

Type	Name	Description
System.String	key	
IEnumerable<Language>	langs	

### Properties

#### BackgroundIsHighlighted

Legt fest, ob der Hintergrund dieser Spalte rot markiert ist

##### Declaration

```
public bool BackgroundIsHighlighted { get; set; }
```

##### Property Value

Type	Description
System.Boolean	

#### Key

##### Key

##### Declaration

```
public string Key { get; set; }
```

##### Property Value

Type	Description
System.String	