|  |
| --- |
| [Company name] |
| [Document title] |
| [Document subtitle] |

|  |
| --- |
| Yu Zhu  [Date] |

Inhaltsverzeichnis

[Erklärung 2](#_Toc518234530)

[Projektidee / Projektziel 3](#_Toc518234531)

[Anforderungsanalyse 4](#_Toc518234532)

[Projektverlauf 5](#_Toc518234533)

[Allgemeines 5](#_Toc518234534)

[Implementierung 5](#_Toc518234535)

[Test 5](#_Toc518234536)

[Klassen-Struktur / UML 6](#_Toc518234537)

[Bedienung 7](#_Toc518234538)

[Ziel 8](#_Toc518234539)

[Use01 9](#_Toc518234540)

[Use02 10](#_Toc518234541)

[Use03 11](#_Toc518234542)

[Use04 12](#_Toc518234543)

[Use05 13](#_Toc518234544)

[Use06 14](#_Toc518234545)

[Use07 15](#_Toc518234546)

[Problematiken 16](#_Toc518234547)

[Fazit 17](#_Toc518234548)

# Erklärung

Hiermit erkläre ich, dass ich die vorliegende Projektarbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe.

Die Stelle der Projektarbeit, die andere Quellen im Wortlaut oder dem Sinn nach entnommen wurden sind durch Angaben der Herkunft kenntlich gemacht. Dies gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet.

Ulm, den 27. Juni 2018



Yu Zhu



Quellen

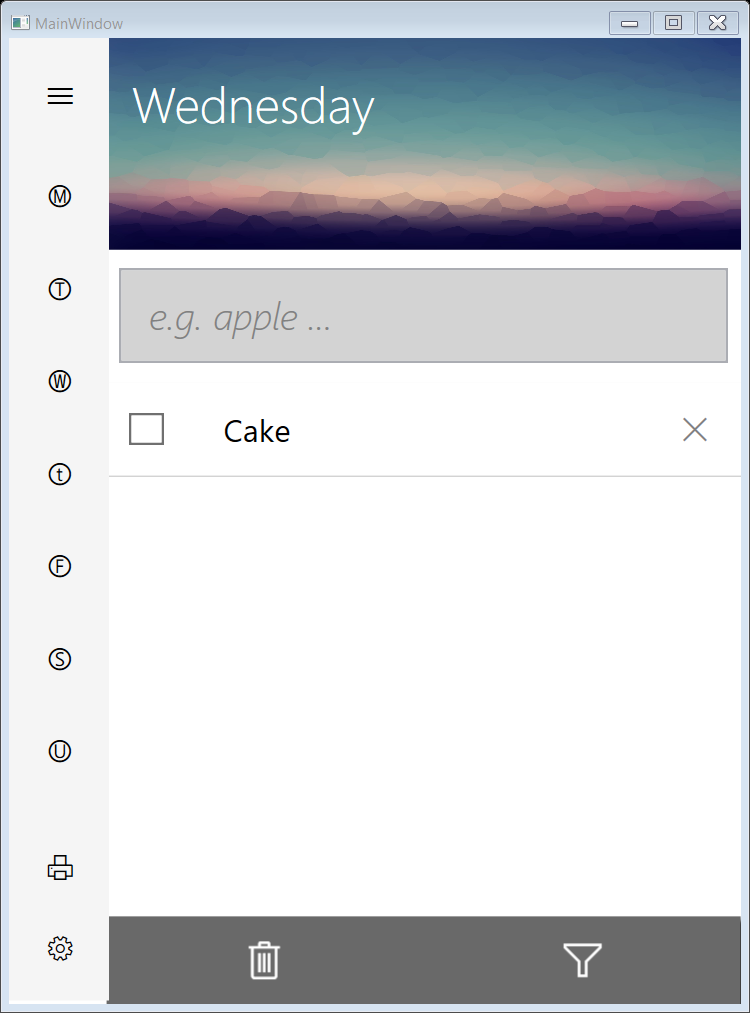


Nettonsoft.Json

o Visual Studio NuGet-Paket

# Projektidee / Projektziel

*ToBuy* ist eine “Todo List” Applikation. Man kann diese benutzen, wenn man einkaufen geht. Die Motivation das Applikation zu entwickeln ist, dass 2018 Neujahrswünsche (eine App für Mama schreiben) erfüllen. C# ist mein erste programmieren Sprache. Das ist auch einen gute Chance C# OOP (Object-oriented programming) programmieren wiederholen. Meine Ausbildungsfirma verendet meisten .NET Technik. Bei diesem Projekt kann ich WPF Desktop Anwendung Entwicklung lernen. Als Entwickler ist Design Fähigkeit nötig. Das kann ich auch in diesem Projekt üben.

Wenn das Umbenennen abgeschlossen wurde, soll dies folgendermaßen aussehen:  
  
Benutzer kann neue Einkaufens Liste als Wochentage erstellen. Bei jeder Liste kann man neue Artikel hinzufügen. Wenn Benutzer einen Artikel gekauft hat. Er kann sich der Artikel maskieren. Man kann jeden Artikel in einer Liste löschen oder alle Artikel in einer Liste auslassen. Wenn es viele Artikel gibt, man kann die Artikel sortieren. Das ist auch möglich, dass Benutzer die aktuelle Liste ausdrucken.

# Anforderungsanalyse

LIST BEARBEITEN

* Liste als Hauptelementedarstellung.
* Der User kann einen neuen „ToBuy“ Artikel hinzufügen.
* Der User kann einzelne Artikel von der Liste entfernen.
* Der User kann alle Artikel löschen.
* Der User kann die Liste nach Bedingung sortieren.

DATEN SPEICHERN

* Der User kann die Daten als JSON Datei in Lokal speichern.
* Wenn die App wieder gestartet wird, können die Daten wieder geladen werden.
* Vor dem Herunterfahren der App, werden die Daten automatisch gespeichert.

BNUTZERFREUNDLICH

* Zielbenutzer kommt aus China, dieser Umstand muss berücksichtigt werden.
* Bei der PC Applikation, ist das Bearbeiten per Tastatur möglich.

# Projektverlauf

Allgemeines

Mi, 12.06

* Planung des UMLs/Klassen-Struktur
* Vorstellung der Projektidee (genehmigt)

Mi, 13.06

* GUI designt
* Klassen erstellen

Mi, 15.06

* Icon erstellt und eingebaut
* UI Elemente implementieren

Mi, 16.06

* Button und Textbox mit View Model verbinden
* Funktion „Add“ und „Delete“ Item von Liste

Mi, 17.06

* Funktion „Sort“ Items von Liste implementiert
* UI optimieren

Mi, 18.06

* Daten in Json Datei speichern
* Dokumentation erstellt

## Implementierung

Bei der Implementierung habe ich als erstes die GUI designt. Ich habe MVVM Pattern verwendet. “ShoppingList” und “ToBuy” sind Modal Klassen. Klasse “DataAccess” benötigt für Daten speichern und laden. Klassen „MainWindowViewModal“ ist mit der grafischen Oberfläche (XAML) verknüpft werden. Jeden Feature zu Beispiel „Add“, „Arrange“, „Remove“, „Clear“, die die Liste Bearbeiten kann. Sie haben eigene Klasse. Damit kann man den Code besser organisieren.

## Test

Während der Entwicklung wurden bereits einige Tests durchgeführt um die Funktion der Methoden sicherzustellen. Die Methoden „ToLoad()“ und „ToSave()“ benötigen für die Daten als Json Datei zu speichern. Dafür gibt es Unit Test durchführen um die Methoden sicherzustellen.

# Klassen-Struktur / UML

Das Programm setzt sich aus zwei Klassen zusammen. Einmal die Klasse „NameSetter“, welche die Funktion hat den kompletten Umbenennungsprozess zu koordinieren. Die Klasse „Filter“ unterstützt dabei die „NameSetter“ Klasse beim Filtern der erhaltenen Objekte.

**Implementation**(Implementation;  
ListManager)  
DataAccess.cs  
ShoppingListManager.cs

**UnitTest**  
ToLoadTests.cs  
ToSaveTests.cs

**Interface**(OperationContract)  
IDataAccess.cs  
IShoppingListManager.cs

ToBuy

+Name {set; get}: string  
+IsDone {set; get}: bool

+ToBuy()

ShoppingList

+ToBuys {set; get}: List<ToBuy>  
+WeekDay {set; get}: DayOfWeek

+ShoppingList()

1

1…\*

**Model**(DataContract)  
ToBuy.cs  
ShoppingList.cs

**View**  
App.xaml  
MainWindow.xaml

MainWindowViewModel.cs

**ViewModel**

**(ListWindow)**  
AddCommand.cs  
ArrangeCommand.cs  
ClearCommand.cs  
RemoveSelectedCommand.cs

**(SideDrawer)**  
MondayCommand.cs  
TuesdayCommand.cs  
WednesdayCommand.cs  
ThursdayCommand.cs  
FridayCommand.cs  
SaturdayCommand.cs  
SundayCommand.cs  
PrintCommand.cs

Notification

Data binding

Dabei reflektieren die kompletten Funktionen der zwei Klassen auf die grafische Oberfläche.

# Bedienung

## Ziel

## Use01

## Use02

## Use03

## Use04

## Use05

## Use06

## Use07

# Problematiken

# Fazit