



PROJEKTDOKUMENTATION JETSTREAM SKISERVICE WPF

Alexander Ernst

Inhaltsverzeichnis

Ausgangslage	2
Anforderungen.....	2
Zeitplanung/PSP.....	3
Vorgehensweise.....	4
1. Informieren	4
1.1. Ausgangslage/Anforderungen	4
1.2. Informieren über WPF/Data, Command Binding.....	4
2. Planen	4
2.1. Features/Aufbau Planen	4
2.2. Zeitplanung und PSP	4
2.3. Wireframe Entwürfe	4
3. Entscheiden.....	6
3.1. Für Versionierungsplattform/Tools entscheiden.....	6
3.2. Für Wireframe/Design entscheiden.....	6
3.3. Features/Aufbau definieren.....	6
4. Realisieren.....	7
4.1. WPF Design erstellen	7
4.2. Data Binding/Command Binding integrieren.....	8
4.3. Logik erstellen	8
4.4. Zusatzfeatures erstellen.....	8
5. Kontrollieren	10
5.1. WPF Tool testen	10
6. Auswerten	10
6.1. Reflexion/Fazit zu Projekt	10
6.2. Dokumentation fertigstellen/Präsentation erstellen	10

Ausgangslage

Die Firma Jetstream-Service hat neue Touchscreen-fähige Hardware angeschafft (Tablet, Surface) und stellt diese den Mitarbeitern in der Werkstatt und Administration für die Datenpflege der Skiservice Aufträge zur Verfügung. Die Mitarbeiter können damit standortgebunden in der Werkstatt oder bei der An- oder Rückgabe der Skier die Stammdaten eines Auftrages über handliche mobile Tablets oder Handys mutieren. Grösstmögliches Augenmerk wird dabei auf eine einfache, intuitive und aufgaben angemessene Bedienung gelegt. Die jeweils erforderliche Anzahl Mausklicks oder Tipp-Eingaben zur Datenpflege der Aufträge sind dabei entscheidende Bewertungsfaktoren. In der Werkstatt wird aus Sicherheitsgründen mit Handschuhen gearbeitet, auch dieser Punkt gilt es folglich bei der Gestaltung der Benutzeroberfläche zu berücksichtigen.

Die bereits existierende Datenbasis zur Onlineanmeldung und der Service-App soll bestehen bleiben und mit der neuen GUI-Lösung für die Tablet- bzw. Handy erweitert werden.

In der Hauptsaison sind bis zu 10 Mitarbeiter mit der Durchführung der Serverarbeiten beschäftigt. Diese sollen einen autorisierten passwortgeschützten Zugang zu den anstehenden Aufträgen erhalten und diese zur Abarbeitung übernehmen und ändern können.

Anforderungen

Das Auftragsmanagement muss folgende Funktionen zur Verfügung stellen:

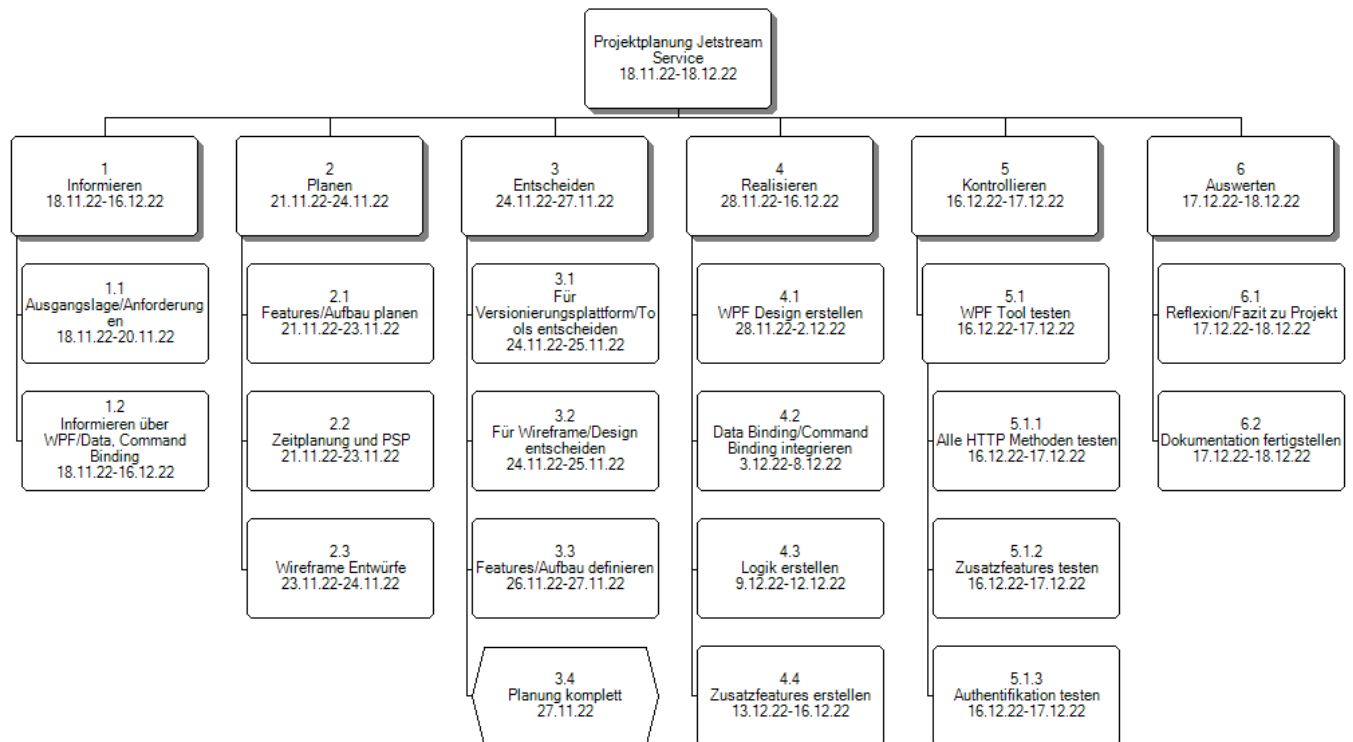
- Login mit Benutzername und Passwort
- Anstehende Serviceaufträge anzeigen (Listendarstellung)
- Filtern der Listeneinträge (Suche)
- Bestehende Serviceaufträge mutieren. Dazu stehen folgende Status zu Verfügung: «Offen», «In-Arbeit» und «Abgeschlossen».
- Aufträge löschen (ggf. bei Stornierung)

Nr.	Beschreibung
A1	Login Dialog mit Passwort für den autorisierten Zugang der Mitarbeiter
A2	Erfasste Serviceaufträge müssen übersichtlich angezeigt sein (Listendarstellung)
A3	Die Listenanzeige der Serviceaufträge ist nach Priorität zu sortieren
A4	Listeneinträge können gefiltert werden (Suche von Einträgen)
A5	Mitarbeiter können eine Statusänderung eines Auftrages vornehmen
A6	Die .NET Anwendung muss nach dem MVVM (Model, View, ViewModel) Design Pattern strukturiert sein.
A7	Datenaustausch per Web-API, REST-Service (PHP/ASP.NET)
A8	Intuitive u. Aufgabenangemessene Bedienung
A9	Grundsätze der Dialoggestaltung aus der Norm EN ISO 9241-110 berücksichtigt
A10	Ganzes Projektmanagement nach IPERKA dokumentiert

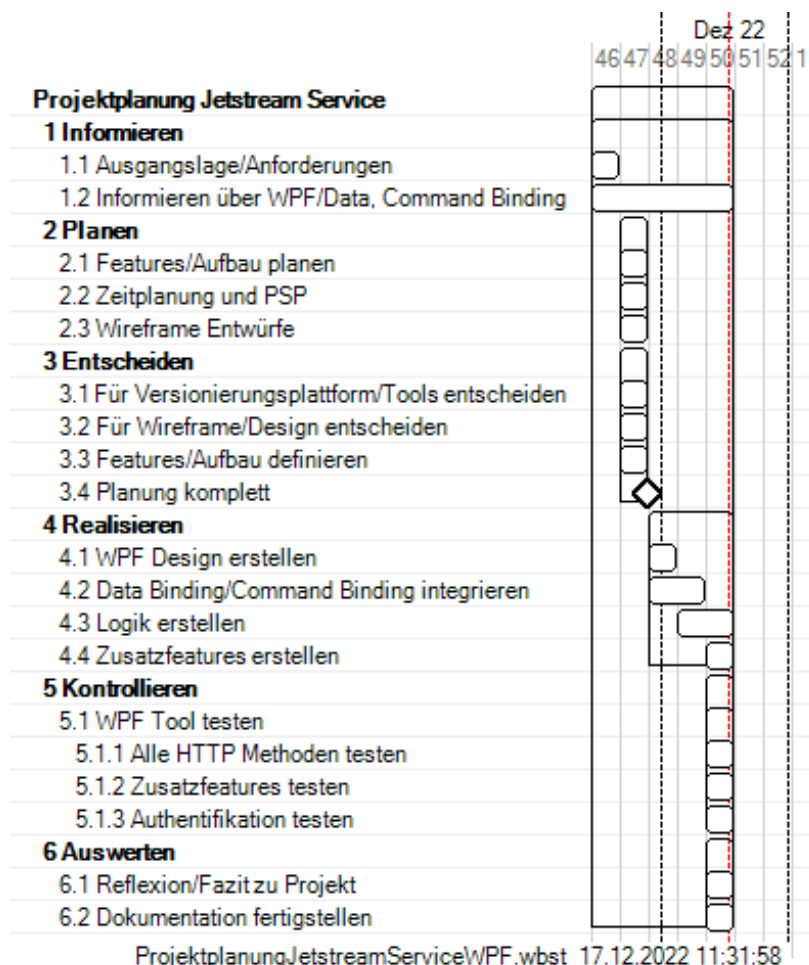
Nr.	Beschreibung
AO1	Dashboard mit Statistikdaten usw.
AO2	Die Mitarbeiter können zu einem Auftrag einen Freitext bzw. Kommentar hinterlegen
AO3	Ein Auftrag kann mit sämtlichen Datenfeldern geändert werden
AO4	Das Login des Mitarbeiters wird nach drei nachfolgenden Falschanmeldungen automatisch gesperrt.
AO5	Personalisierte Auftragsliste des eingeloggtten Mitarbeiters. Der Mitarbeiter kann sich zusätzlich zur gesamten Auftragsliste nur die von ihm übernommenen Aufträge ansehen.
AO6	Eingeloggte Mitarbeiter können neue Benutzer (Login) eröffnen.
AO7	Eingeloggte Mitarbeiter können ein gesperrtes Login zurücksetzen.
AO8	Gelöscht Aufträge werden nicht aus der Datenbank entfernt, sondern nur als gelöscht markiert.

Zeitplanung/PSP

Mit WBSTool erstellt



ProjektplanungJetstreamServiceWPF.wbst 17.12.2022 11:31:58



Vorgehensweise

1. Informieren

1.1. Ausgangslage/Anforderungen

Als ersten Arbeitsschritt habe ich mich über die Anforderungen und die Ausgangslage dieses Projektes informiert. Hierzu habe ich die zur Verfügung gestellten Unterlagen verwendet.

Durch diese Recherche konnte ich die Anforderungen und das Grundkonzept der Applikation verstehen, und wusste ungefähr, was zu tun war.

1.2. Informieren über WPF/Data, Command Binding

Als Nächstes habe ich mich über Technologien, die ich für dieses Projekt brauchen werde, informiert. Dies war aber auch ein laufender Prozess, dies heisst, dass ich mich in dem ganzen Laufe des Projektes weiter informieren/in das Themengebiet vertiefen musste.

Diese Recherche habe ich einerseits mit dem Unterrichtstoff durchgeführt. Andererseits habe ich aber auch mich im Internet weiter informiert.

2. Planen

2.1. Features/Aufbau Planen

Hier musste ich die Features und den Aufbau der Applikation planen, hier habe ich mich auch schon etwas mit der Benutzerfreundlichkeit und dem Design auseinandergesetzt.

2.2. Zeitplanung und PSP

Als Letztes konnte ich eine Zeitplanung/ein PSP erstellen, da ich jetzt eine grobe Planung meines Projektes hatte.

2.3. Wireframe Entwürfe

In diesem Schritt habe ich zwei Wireframe Entwürfe erstellt, um mich später zu entscheiden. Der erste Entwurf ist komplexer, hat aber auch gewisse Vorteile, da man Erstellen und Modifizieren auf einer Seite durchführen kann.

Entwurf 1:

Authentifikation | Abrufen/Löschen | Erstellen/Modifizieren

General

Auswahl ▼

Status/ID:

Authentifikation

User:

Password:

JWT-Key:

Daten

Name: Service: Alle Aufträge ▼

Email: Status: Alle Aufträge ▼

Telefon: Priorität: Alle Aufträge ▼

Auftrag-Datum: / /

Bemerkungen:

Senden

Output

Entwurf 2:

Auftrag Modifizieren

Allgemeines

Erstellung

Verwaltung

Daten Abrufen

Eintrag Löschen

Eintrag Modifizieren

Error Codes and Information

Name (job title)	Age	Nickname	Employee
Giacomo Guilizzoni Founder & CEO	40	Peldi	<input checked="" type="radio"/>
Marco Botton Tuttofare	38		<input checked="" type="checkbox"/>
Mariah MacLachlan Better Half	41	Patata	<input type="checkbox"/>
Valerie Liberty Head Chef	:)	Val	<input checked="" type="checkbox"/>
Data Grid Docs			<input type="checkbox"/>

3. Entscheiden

3.1. Für Versionierungsplattform/Tools entscheiden

Hier musste ich mich entscheiden, welche Versionierungsplattform/andere Tools ich verwende. Für das Versionieren des Codes habe ich GitHub verwendet, und für das Schreiben des Codes habe ich den IDE Visual Studio verwendet.

GitHub: <https://github.com/alexanderernst/JetstreamServiceNETWPF>

3.2. Für Wireframe/Design entscheiden

Hier habe ich mich für den zweiten Designentwurf entschieden, da dieser Benutzerfreundlicher ist und auch verständlicher ist.

Zusätzlich habe ich beim zweiten Designentwurf mehr Erweiterungsmöglichkeiten durch die Toolbar und kann z.B. auch eine Suche integrieren.

Um den zweiten Designentwurf weiter zu verbessern habe ich auch noch einen Tab «Benutzer» hinzugefügt, unter welchem man Benutzer sehen und verwalten kann.

Zur Benutzerfreundlichkeit habe ich für jeden Button in der Toolbar auch noch einen Tooltip hinzugefügt.

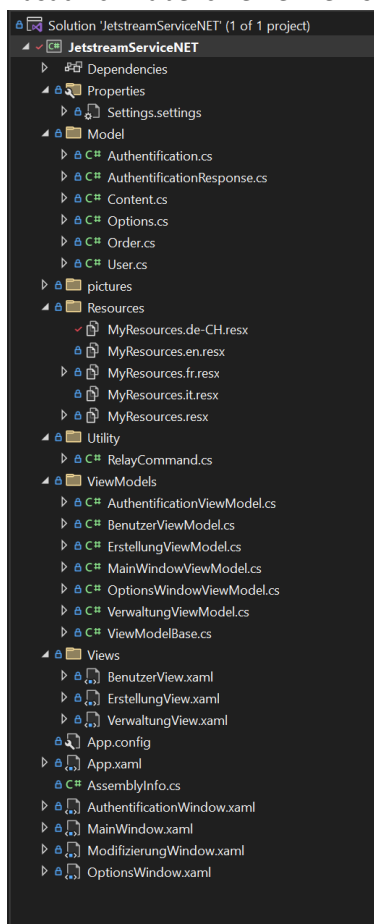
3.3. Features/Aufbau definieren

Hier habe ich mich entschieden, dass Projekt mit MVVM umzusetzen, dies heisst, dass ich Model Klassen, Views (Fenster) und ViewModel Klassen für die jeweiligen Fenster/Views habe.

Die Modellklassen verwende ich für die API Requests und um Daten anzuzeigen/auszulesen.

Bei den Zusatzfeatures habe ich mich entschieden die Suche zu integrieren und eine Einstellung XML Datei zu integrieren, welche man mit einem Fenster in der Applikation modifizieren kann.

Zusätzlich habe ich eine Mehrsprachfähigkeit mit Ressourcen Dateien eingebaut.



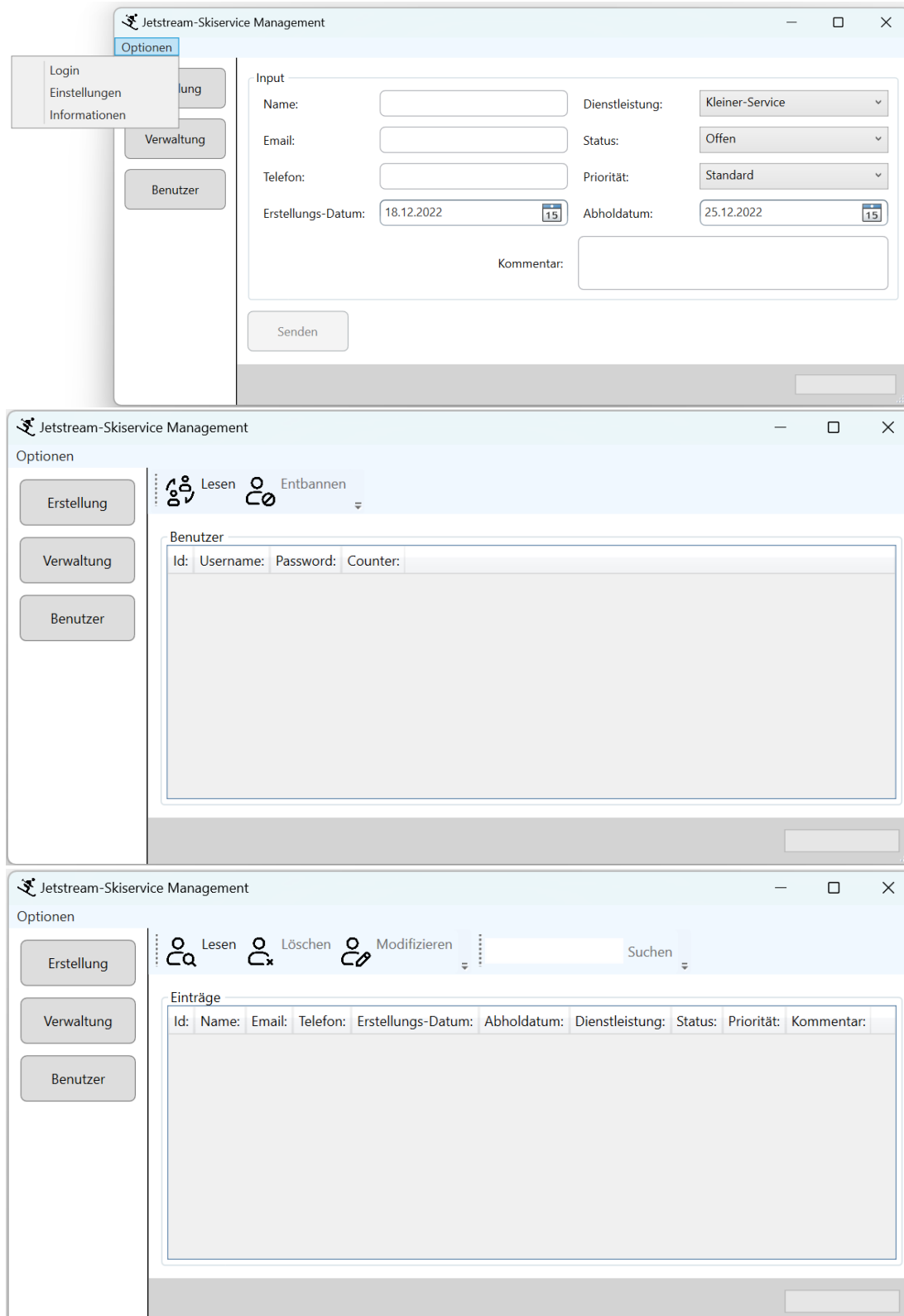
4. Realisieren

4.1. WPF Design erstellen

Als Erstes habe ich das WPF Design erstellt, hier habe ich das Design erstellt, dazu habe ich Views (UserControl) mit ViewModel Klassen verwendet.

Hier habe ich eine Statusbar, Progress Bar, Toolbar und eine Suche hinzugefügt.

Zusätzlich habe ich in diesem Schritt WPF Styles erstellt, um die Knöpfe und Textboxen abzurunden.



4.2. Data Binding/Command Binding integrieren

In diesem Schritt habe ich das Data und das Command Binding integriert. Dies habe ich durch MVVM mit ViewModel Klassen getan, hierzu habe ich die RelayCommand Klasse verwendet.

Für das Data Binding verwende ich Models mit dem Interface INotifyPropertyChanged.

Für das Command Binding verwende ich hingegen die RelayCommand Klasse, und habe eine CanExecute Methode, welche überprüft, ob man diesen Command ausführen können soll und eine Execute Methode, in der die eigentliche Logik ausgeführt wird, wenn die CanExecute Methode dies erlaubt.

Dies hat den Vorteil, dass ich Knöpfe ausgrauen kann, wenn Textfelder z.B. leer sind

Als Nächstes habe ich dieses Data/Command Binding im XAML Code integriert, hierzu verwende ich DataContext für das Angeben des ViewModels, ItemSource für das Data Binding und die Command Property für das Command Binding.

4.3. Logik erstellen

Als Nächstes habe ich die Logik in dem ViewModel erstellt, hier habe ich auch die HTTP Requests mit RestSharp erstellt.

Diese Logik ist nicht besonders kompliziert, das einzige spezielle ist, dass ich anstatt Listen, ObservableCollection verwende, da nur diese mit dem DataGrid funktionieren.

Die HTTP Requests habe ich mit RestSharp umgesetzt, da ich diese Technologie lernen wollte.

4.4. Zusatzfeatures erstellen

Als Letztes habe ich die Zusatzfeatures entwickelt, dazu gehört die Suchfunktion, Einstellungsdatei und die Ressourcen Dateien für die Mehrsprachfähigkeit.

Für die Suchfunktion habe ich den LINQ Where Befehl verwendet und das Resultat in eine Variable vom Typ IEnumerable gespeichert und diese dann in eine ObservableCollection konvertiert. Dies musste ich so durchsetzen, da man das Resultat eines Where Befehls nicht in einer ObservableCollection speichern kann.

Um die Einstellungsdatei zu erstellen, habe ich einfach eine .settings Datei mit ihrer zugehörigen XML-Datei erstellt. Diese Datei kann man dann von dem C# Code abrufen oder schreiben.

Wichtig ist hier das die Properties in der .settings Datei auf die Scope User gesetzt sind, denn nur so können wir Sie modifizieren.

Application settings allow you to store and retrieve property settings and other information for your application dynamically.

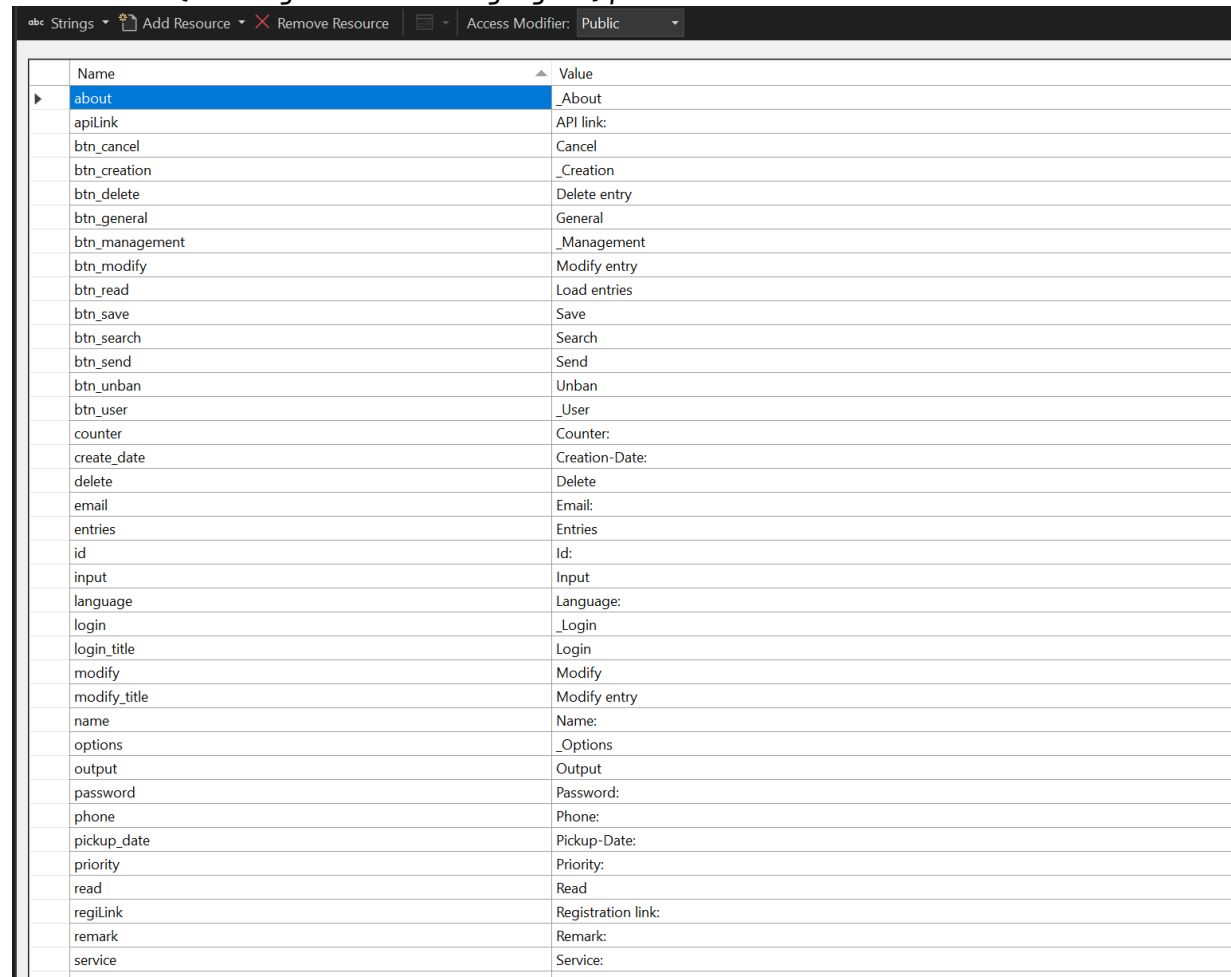
	Name	Type	Scope	Value
	LanguageID	string	User	DE-CH
	WindowsHeight	string	User	600
	WindowsWidth	string	User	400
	JWTToken	string	User	jwt
	APILink	string	User	https://localhost:7253
	registrationLink	string	User	/api/Registration/
	userLink	string	User	/api/User/
*				

```
Settings.Default.userLink = Options.UserLink;  
Settings.Default.Save();
```

Als Letztes habe ich noch die Mehrsprachfähigkeit integriert, dies habe ich über Ressourcen Dateien (.resx) getan, die ausgewählte Sprache wird jeweils in der Settings Datei gespeichert und abgerufen und kann auch von dem Programm verändert werden.

Damit wir Zugang auf diese Ressourcen haben muss die Datei auf Public gesetzt sein, um die Sprache zu ändern, verwenden wir den Befehl:

```
Thread.CurrentThread.CurrentUICulture = new
CultureInfo(Settings.Default.LanguageID);
```



Name	Value
about	_About
apiLink	API link:
btn_cancel	Cancel
btn_creation	_Creation
btn_delete	Delete entry
btn_general	General
btn_management	_Management
btn_modify	Modify entry
btn_read	Load entries
btn_save	Save
btn_search	Search
btn_send	Send
btn_unban	Unban
btn_user	_User
counter	Counter:
create_date	Creation-Date:
delete	Delete
email	Email:
entries	Entries
id	Id:
input	Input
language	Language:
login	_Login
login_title	Login
modify	Modify
modify_title	Modify entry
name	Name:
options	_Options
output	Output
password	Password:
phone	Phone:
pickup_date	Pickup-Date:
priority	Priority:
read	Read
regilink	Registration link:
remark	Remark:
service	Service:
...	...

5. Kontrollieren

5.1. WPF Tool testen

Um die WPF Applikation zu testen habe ich alle HTTP Methoden, Zusatzfeatures und auch die Authentifikation getestet.

Um die Benutzerfreundlichkeit zu testen bin ich das Programm nochmals durchgegangen und habe kontrolliert, dass alles verständlich ist und wenn nötig Tooltips vorhanden sind.

Zusätzlich habe ich auch kontrolliert, dass das Programm eine gute Fehlerbehandlung bietet, das Fenster Resizable ist, die Applikation nicht abstürzt und dass Knöpfe bei Invaliden Daten deaktiviert sind.

6. Auswerten

6.1. Reflexion/Fazit zu Projekt

Ich denke, ich habe die Hauptanforderungen dieses Projektes geschafft, da diese Applikation völlig funktionsfähig ist und auch alle Anforderungen erfüllt.

Ich habe durch dieses Projekt auch sehr viel über WPF, Data/Command Binding und C# gelernt, und konnte mein Wissen um einiges erweitern.

Was meiner Meinung in diesem Projekt nicht so gut lief, war das Design, ich brauchte viel mehr Zeit, um ein benutzerfreundliches Design zu entwickeln, als ich dachte, dies denke ich lernt man aber auch mit der Zeit.

Im Allgemeinen lief dieses Projekt aber schon sehr gut und ich habe viel Neues gelernt.

6.2. Dokumentation fertigstellen/Präsentation erstellen

Als aller letzten Arbeitsschritt habe ich die Dokumentation nochmals überarbeitet.