04.09.2017

# **Projektzielsetzung**

Innerhalb der nächsten 2 Wochen sollen Bedienkonzepte von Sprachgesteuerten Apps erforscht werden. Das Ziel ist es ein Bedienkonzept für die Website 10.150.50.21/irj/portal zu erstellen. Es ist offen, ob das für die Website in HTML passieren soll oder ob eine App entwickelt werden sollte, die mit der Bedienoberfläche der Website identisch ist.

## **Sprachsteuerung einer HTML5-Seite**

In HTML5 ist es möglich, die Spracheingabe als Formularform zu aktivieren. Das heißt aber nicht, dass auch Sprachassistenten dazu eingebunden werden können. Tastenkombi für Aktivierung ist ctrl shift .

* Nicht alle Feldtypen unterstützen Spracherkennung. Email, url und date weigern sich.
* Nur bei Internetverbindung möglich
* Nur mit Chrome ab Version 25 nutzbar
* experimentell

<http://blog.teamtreehouse.com/accepting-speech-input-html5-forms>

<https://developers.google.com/web/updates/2013/01/Voice-Driven-Web-Apps-Introduction-to-the-Web-Speech-API>

## **Sprachassistenten**

Bei der Entwicklung einer App ist es möglich auf Standardapps des verwendeten Gerätes zurückzugreifen, so etwa Cortana (ab Windows 8), Siri (MacOS), Google Now (Google), Alexa (Amazon), Lucida (Open Source) bzw. die internen Mikrofon(e) und Lautsprecher zu nutzen. Dazu existieren meist sogenannte „Skills“ die das Entwickeln fördern.

Momentan ist Android das am weitesten verbreitete Betriebssystem auf dem Handymarkt. Das sieht man an der nachfolgenden Grafik von [Statista](https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/), die die Verkaufszahlen von Handys von 2009 bis 2017 grafisch darstellt. IOS ist aktuell mit einem knappen Fünftel in den Weltmarktverkäufen vertreten. Da IOS und Android unterschiedliche Entwicklungssprachen und –konzepte verwenden, ist es sinnvoll, plattformübergreifende Entwicklung zu erwägen.



## **Plattformübergreifende Entwicklung**

Dazu hat Windows Universal App Programmierung die Möglichkeit bereitgestellt, mit Visual Studio plus Xamarin Plugin und C#-Kenntnissen für beide Betriebssysteme zugleich zu entwickeln.

-Microsoft Azure Account benötigt

- Testen der App mit Xamarin IOS ist komplex, denn es ist notwendig eine Verbindung zu einem Mac einzurichten.

## **Navigation**

Sprachgesteuerte Anwendungen werden in Bezug auf die Navigationstypen auf Zielgerichtete z.B. „move to unicorn“ und Richtungsorientierte z.B. „move 4 lines down, left 8 words“ unterschieden. Außerdem gibt es eine Unterscheidung zwischen multimodalen Systemen und handfreien Systemen.

Bei den multimodalen Systemen kann der User zwischendurch den Cursor auch mit der Maus oder Tastatur führen. Handfreie Systeme lassen dagegen keine konventionellen Mittel der Steuerung zu, zum Beispiel weil der Benutzer blind ist oder aus anderen Gründen keine Tasten oder Touchs ausführen kann wie zum Beispiel beim Autofahren.

Die Sprachintelligenz erkennt in beiden Fällen Kommandos wie etwa die folgenden:



Abbildung Sprachbasierte Kommandos von TkTalk version 1.0[[1]](#footnote-1)

Eine dritte Art der Navigation stellt eine Art Kompromiss zwischen multimodalen und handfreien Systemen dar: Das Grid-System. Im Gridsystem navigiert der Benutzer mittels Tasten und Kommandos den Cursor auf dem Bildschirm. Dies wurde erstmalig dazu verwendet, blinden Benutzern zu ermöglichen, graphische Elemente zu zeichnen von Hesham M. Kamel und seinen Mitarbeitern in IC2D[[2]](#footnote-2).

Blinde Benutzer sind es gewohnt, das Nummerntastenfeld 3x3 zu verwenden um den Cursor zu positionieren. Mit den folgenden Kommandos können außerdem Buchstaben verwendet werden, um den Cursor zu versetzen oder eines der Felder weiter zu unterteilen (m). Die Palette zeigt eine Auswahl von Standardformen, die in das Feld eingetragen werden können. Dies ist auf ein Formular übertragen eine Liste von häufigen Einträgen.



Abbildung Auswahl an IC2D Funktionstasten

\*o= information welches Objekt der Palette gerade ausgewählt ist (e.g. shape square, positions 1, 3, 9, and 7").

W= where am I key sagt wo der cursor gerade ist (und was dort bereits steht)

Cortana Skills

Tbd

## Azure

Azure ist ein Cloud-dienst der plattformübergreifende Entwicklung ermöglicht. Diese ist für Apps, Webanwendungen und weitere möglich. Bei den Apps besteht die Möglichkeit mehrere Apps auf einem App Service Plan = 1 physisches Gerät zu hosten, wie die folgende Abbildung illustriert.



Abbildung : App Service Plan

+Scale up/down bei mehr Nutzeranfragen möglich

+Verteilen auf verschiedene Instanzen möglich

-Kosten: Für virtuellen Server z.B. ab 54$/Monat, Durchschnittspreise aber eher 150$/Monat

Tutorial für UWP findet sich unter: <https://docs.microsoft.com/de-de/azure/app-service-mobile/app-service-mobile-windows-store-dotnet-get-started>

Möglicherweise ist es zunächst in Visual Studio möglich offline die App zu entwickeln und anschließend hochzuladen. Dies ist bei der Entwicklung anderer Clouddienste gegeben.

Azure kann ebenfalls Bots verwalten. Kosten unbekannt. Es gibt dort ein Template zur Einbindung von LUIS. (Spracherkennungsapi) . Nach einem nutzungsbasiertem Abonnement von Azure kann man folgende Schritte durchlaufen (Anmerkung hierzu- der Bot erfordert einen Server, dieser kann bis zu 1 GB umsonst genutzt werden und verursacht ansonsten ebenfalls Kosten. Die Nachfolgende Abbildung illustriert wie man die Verbindeung zum Botframework Schritt für Schritt aufbaut.



Der Bot lässt sich online, über Dritte oder in Kooperation mit einer eigenen VisualStudio Instanz verwenden.

## Publish C# bot on App Service plan from Visual Studio

To set up publishing from Visual Studio using the .PublishSettings file, perform the following steps:

1. In the Azure Portal, click your Bot Service, click the **BUILD** tab, and click **Download zip file**.
2. Extract the contents of the downloaded zip file to a local folder.
3. In Explorer, find the Visual Studio Solution (.sln) file for your bot, and double-click it.
4. In Visual Studio, click **View**, and click **Solution Explorer**.
5. In the Solution Explorer pane, right-click your project, and click **Publish...** The Publish window opens.
6. In the Publish window, click **Create new profile**, click **Import profile**, and click **OK**.
7. Navigate to your project folder, navigate to the **PostDeployScripts** folder, select the file that ends in .PublishSettings, and click **Open**.

You have now configured publishing for this project. To publish your local source code to Azure Bot Service, right-click your project, click **Publish...**, and click the **Publish** button.

Es gibt allerdings Probleme mit dem Emulator, wenn ich lokal testen möchte, ähnlich wie bei <https://github.com/Microsoft/BotBuilder/issues/3461> beschrieben. Memo an mich selbst: vielleicht besser online entwickeln?

VisualStudiOnline verbinden

Accesstoken kk6l6kszjnhwyocww6b2vvzd3wrk4arbrywldsa2tnz2zgsnlbxa

## Ergometrie

Die DIN EN ISO 13407 legt fest, dass die Anwender von Anfang an in das Projekt miteinbezogen werden sollen (s.S.47). Das Papier bekräftigt außerdem während des Entwickelns die Bedeutung von Prototypen.

## Visual Studio

In Visual Studio kann eine UWP-App aus einer blanken Vorlage sehr leicht erstellt werden und mit der sich links am Rand befindlichen Toolbox Elemente der App mittels Cursor angeordnet werden, wie die folgende Abbildung zeigt. Visual Studio 2017 ist als Community Edition frei verfügbar, dies ist jedoch die Enterprise Edition die für die meisten Studenten über das Portal MS Imagine zur Verfügung steht.

Troubleshooting

Fehlermeldung „InitializeComponent() konnte nicht gefunden werden

Lösung: Rechtsklick auf Projekt, Projekt entladen, Rechtsklick auf Projekt, Projekt erneut laden

Projekt debuggt alten Stand

Lösung Auf Rechtsklick auf Projekt, Projekt Bereitstellen klicken und dann erst Laufen lassen



Abbildung : hello World[[3]](#footnote-3)

Wie man hier ebenfalls sieht, gibt es aktuell Probleme bei diesen beiden Editionen- die Android SDK wird nicht gefunden.

Die Gestaltungssprache XAML erinnert sehr an XML. Der Default ist das GridLayout, bei dem die einzelnen Elemente in Reihen und Spalten angeordnet werden. Mithilfe der Klasse Rectangle lässt sich zudem der Hintergrund in farblich unterschiedliche Bereiche einteilen. Canvas.ZIndex hilft dabei zu bestimmen, wie weit nach vorne oder hinten ein Element positioniert werden sollte.

Farbverläufe lassen sich hingegen im Eigenschaften-Fenster rechts unter Pinsel festlegen, wie folgende Abbildung zeigt:



Abbildung : Hintergrundgestaltung in XAML

# Cortana Skills

Zum Entwickeln von Cortana Skills[[4]](#footnote-4) wird folgendes benötigt:

* A developer dashboard (or portal) for registering, configuring, and connecting your skill.
* A development machine on which to create **a bot** built with the Microsoft Bot Framework that provides the functionality for your skill. The Bot Framework supports development in .NET (**C#,** Visual Basic) and Node.js. […]
  + Azure-, Cortana-Skills-, Microsoft-LUIS- und MSA-Account
  + Bot Framework Emulator[[5]](#footnote-5)
  + Visual Studio
* An internet accessible endpoint to host your cloud based bot or service. The [Azure Bot Service](https://azure.microsoft.com/en-us/services/bot-service/), [Azure functions](https://azure.microsoft.com/en-us/services/functions/) or [Azure Web apps](https://azure.microsoft.com/en-us/services/app-service/web/).
* Mindestens 1 Testgerät, das Cortana and Cortana skills unterstützt. Empfohlen wird dazu Win 10 Anniversary Update. Das Testgerät muss mit dem gleichen MSA-Account angemeldet sein mit dem die App programmiert wurde.
  + Bei Android: Android 4.4+ und Cortana version 2.1.6.1547+
  + Bei IOS: iOS 9.0+ und Cortana version 1.9.15+
  + Bei Windows Phone: Windows 10 Mobile Insider Preview Build 14327+.



Abbildung :Persönliche Infos die Cortaner über den User haben könnte

## Bot Framework

Mithilfe des Bot Frameworks ist es möglich u.a. eine Konversation zu erstellen. Ein Bot ist in etwa eine App, die Zugriff auf verschiedene Dienste haben kann, zu zum Beispiel den Kalender wie die folgende Abbildung zeigt[[6]](#footnote-6).



Abbildung : saloon app.

Das Botframework stellt verschiedene Klassen zur Verfügung um einen Bot zu erstellen. So gibt es die Dialogs (in denen der Bot immer wieder auf Eingaben des Users wartet mit await um anschließend seine Antwort zu senden mittels eines Activity-Objectes – meist eine message). Zweitens gibt es die FormFlows (die sich zum Gestalten eines Bots eignen, der Informationen vom User sammelt) und drittens die Bot Framework States (Bots, die zum Speichern von Informationen geeignet sind).

Mittels des Emulators lassen sich die selbstgebastelten Bots testen. Cortana lässt sich einbinden, wenn der Bot registriert ist und e registered and using a valid MS AppId and Password. -> eine MS AppId mit Passwort besitzt.

Es gibt darüber hinaus ein Tool, das das Testen erleichtert durch inverse Containerverkünpfung, namens Autofac[[7]](#footnote-7).

## Dialogs

In Dialogs kann man Verzweigungen wie folgt anlegen:



Navigation in Dialogen ist den traditionellen Verfahren entlehnt, wie die folgende Abbildung zeigt.



Zum Herausfinden, wie die (multiuser)-Dialoge abgewickelt werden ist folgendes Projekt auf GitHub empfehlenswert: <https://github.com/Microsoft/BotBuilder-Samples/tree/master/CSharp/core-BasicMultiDialog>

Jedoch ist das Thema komplex, weswegen hier der Focus auf FormFlows gelegt werden sollte. Es reicht zu sagen, das Dialoge zum Beispiel der LuisDialog auf User Eingaben in der Form von Sprache reagieren können.

Beispiele für LuisDialoge finden ihre Erklärung in der Dokumentation des Botframeworks. <https://docs.microsoft.com/en-us/bot-framework/dotnet/bot-builder-dotnet-luis-dialogs>

## FormFlows

FormFlows unterstützen folgende Typen als Eingabeparameter:

* Integral (sbyte, byte, short, ushort, int, uint, long, ulong)
* Floating point (float, double)
* String
* DateTime
* Enumeration
* List of enumerations

Sie eignen sich besonders für einen festgeschriebenen Ablauf. Ein gutes Beispiel ist der Flight and Hotel Dialog Bot s.a. <https://github.com/Microsoft/BotBuilder-Samples/tree/master/CSharp/core-MultiDialogs>

Es besteht grundsätzlich die Möglichkeit, Bots über diverse Channel zu steuern (so zum Beispiel Skype oder Telegramchats). Als Fortgeschrittene Funktionen von FormFlows steht zur Verfügung die Benutzereingaben auszuwerten oder zu beeinflussne mit Hilfe von

**ATTRIBUTE PURPOSE[[8]](#footnote-8)**

Describe Alter how a field or a value is shown in a template or card

Numeric Restrict the accepted values of a numeric field

Optional Mark a field as optional

Pattern Define a regular expression to validate a string field

Prompt Define the prompt for a field

Template Define the template to use to generate prompts or values in

prompts

Terms Define the input terms that match a field or value

Da ein Bot registriert sein muss um auf Cortana Skills zurückzugreifen habe ich meinen persönlichen MSA-Account verwendet um eine App-ID zu generieren.

myBot21, App-ID: d5ee4efb-3c8e-4df9-a19d-d5407859ec25 Kennwort KZLe7S54SjPmXZJ3Vui3nns

* Dies führte jedoch zu keinem Erfolg, da der Botframework Emulator einen Server benötigt um zu kommunizieren.
* Daher in der kostenpflchtigen Version von Azure einen Bot mit eigenem (virtuellen) Server generiert, der eine gewisse Zahl fon Transaktionen kostenfrei ist.

TestBot21, App-ID 85eadc7c-fe75-40b8-8c1f-b6f215be4755 Kennwort F0ciURSU1KTm0qLD2FVB4mU

Das Problem mit dem Botframework ist, dass Visual Studio aus mir unbekannten Gründen Projektverweise nicht finden kann, die dieses benötigt.



Abbildung : Das Testen eines Botframeworks lief nach einer erfolgreichen Anfangszeit hartnäckig schief

Über die Ursache kann ich nur spekulieren, da ich sie nicht gefunden habe. Ich schließe fehlende Verweise auf NuGet-Pakete aus. Ebenso fehlende Updates von Visual Studio und Fehlkonfigurationen durch mich (da ich Visual Studio resetten habe lassen). In einem Onlineforum habe ich gelesen, dass es möglicherweise mit Windows Update zusammenhängen könnte.



08.09.2017

# Windows Universal Apps

In Windows Universal Apps (im Weiteren als UWP bezeichnet) ist die Verwendung von asynchronen Methodenaufrufen eine Besonderheit um Win 8.1+ Kompabilität zu gewährleisten. Der Rückgabetyp ist nicht immer direkt ersichtlich, da zumeist ein Objekt zurückgegeben wird, dass diesen mit getResult erarbeitet (vgl. https://docs.microsoft.com/en-us/windows/uwp/threading-async/call-asynchronous-apis-in-csharp-or-visual-basic). Die folgende Tabelle zeigt dies anschaulich.



<https://docs.microsoft.com/de-de/windows/uwp/input-and-devices/speech-interactions>

Speech recognition by Cortana in UWP apps

Basically all Cortana interaction requests and answers should be short. UWP is smart enough to determine the right scale of images for the resolution of a particular device. Aber es gibt in der Benennung der Bildressourcen im SourceCode folgende Vorschriften:

The standard naming convention is "foldername/qualifiername-value[\_qualifiername-value]/filename.qualifiername-value[\_qualifiername-value].ext". For example: images/en-US/logo.scale-100\_contrast-white.png is simply referred to in code using the root folder and the filename: images/logo.png. See [Manage language and region](https://msdn.microsoft.com/windows/uwp/globalizing/manage-language-and-region) and [How to name resources using qualifiers](https://msdn.microsoft.com/library/windows/apps/xaml/hh965324).[[9]](#footnote-9)

Es gibt zudem die Möglichkeit Ergebnisse bildlich auszudrücken. Dazu existieren folgende Größenoptionen:

* 68w x 68h
* 68w x 92h
* 280w x 140h

Um Cortana Speech Recognition zu nutzen ist es notwendig eine Cortana Fähigkeit zu erstellen. S.a. Bots und die damit verbundene Problematik. Wenn man diese Assistentin umgehen möchte kann man dagegen einfach Windows SpracherkennungsAPI benutzen. Es gibt dazu ein Tutorial unter <https://github.com/Microsoft/Windows-universal-samples/tree/master/Samples/SpeechRecognitionAndSynthesis>

## Xamarin

Nach der Installation von Xamarin ist es möglich eine UWP, ein Android oder ein IOS-Project in Visual Studio für die 3 übrigen Plattformen bereitzustellen. Für UWP hat die Autorin folgende Anleitung verwendet: <https://developer.xamarin.com/guides/xamarin-forms/platform-features/windows/installation/universal/>

Ein kostenloses ebook über Xamarin hat Charles Petzold online zur Verfügung gestellt[[10]](#footnote-10).

Dieses erklärt unter anderem die Unterschiede zwischen Xamarin shared und portable. Wohingegen bei shared die 3 Plattformen eine zentrale Plattform verwenden und in dieser Zwischensprache kompiliert werden, ist es bei portable so, dass die Plattformen unabhängig existieren laufzeiteffizienter aber pflegeaufwändiger.



Die Autorin hat in der Verwendung von Xamarin.Forms Version 2.3.4.270 verwendet nicht 2.0 wie in der Anleitung vorgeschlagen. Deswegen kommt es möglicherweise im Konfigurationsmanager zu einer abweichenden Darstellung wie folgt:



Das Erstellen einer Xamarin-App funktioniert, allerdings bemerkt die App, dass das Android SDK nicht am gewohnten Ort liegt. Dies ist in meiner Visual Studio Installation leider schon beim Installieren fehlgeschlagen.



11.09.2017

Das Microsoft Developer Center hat mehrere Referenzprojekte erstellt, unter anderem auch eines für Cortana namens [Adventure Works](https://docs.microsoft.com/en-us/cortana/voicecommands/launch-a-foreground-app-with-voice-commands-in-cortana). Siehe auch <https://github.com/Microsoft/Windows-universal-samples/tree/master/Samples/CortanaVoiceCommand>

Für diese UWP-Referenzapp empfiehlt sich beim Entpacken nach Anleitung[[11]](#footnote-11) vorzugehen (Wichtig: Rechtsklick auf zip, unter Eigenschaften>Allgemein einen Haken bei Sicherheit setzen, der die Blockierung des zips von anderen Websites verhindert.

Alternativ wäre auch die Nutzung der GitHub Erweiterung für Visual Studio denkbar, aber diese schlug bei der Autorin mit Fehlern fehl.

Ist man dann einmal so weit das Projekt entpackt zu haben, reicht im cs Ordner ein Klick auf die im cs-Ordner befindliche Datei .sln um das Projekt in Visual Studio zu laden. Beim Abspielen der internen Simulatoren und beim Öffnen der .cs-Dateien werden jede Menge Fehler angezeigt. Dies liegt teils an fehlenden Nuget-Paketen. Diese werden jedoch nachinstalliert. Mehrere Male Neu-Build und Antivirensystem-Einstellungsänderungen (falls Hinweise zum Netzwerkkartenzugriff kommen) lassen ein betriebsbereiten Emulator entstehen.



Abbildung Antwort auf Frage an Cortana: "When is my trip to London?"

VCD-Datei

Jedes C#-Projekt, das Cortana verwendet, sollte eine .xml Datei gespeichert haben, die die gültigen Kommandos aufzählt, also zum Beispiel

<?xml version="1.0" encoding="utf-8" ?>

<VoiceCommands xmlns="http://schemas.microsoft.com/voicecommands/1.2">

<CommandSet xml:lang="en-us" Name="AdventureWorksCommandSet\_en-us">

<AppName> Adventure Works </AppName>

<Example> Show trip to London </Example>

<Command Name="showTripToDestination">

<Example> Show trip to London </Example>

<ListenFor RequireAppName="BeforeOrAfterPhrase"> show [my] trip to {destination} </ListenFor>

<ListenFor RequireAppName="ExplicitlySpecified"> show [my] {builtin:AppName} trip to {destination} </ListenFor>

<Feedback> Showing trip to {destination} </Feedback>

<Navigate />

</Command>

<Command Name="whenIsTripToDestination">

<Example> When is my trip to Las Vegas?</Example>

<ListenFor RequireAppName="BeforeOrAfterPhrase"> when is [my] trip to {destination}</ListenFor>

<ListenFor RequireAppName="ExplicitlySpecified"> when is [my] {builtin:AppName} trip to {destination} </ListenFor>

<Feedback> Looking for trip to {destination}</Feedback>

<VoiceCommandService Target="AdventureWorksVoiceCommandService"/>

</Command>

<PhraseList Label="destination">

<Item>London</Item>

<Item>Las Vegas</Item>

<Item>Melbourne</Item>

<Item>Yosemite National Park</Item>

</PhraseList>

</CommandSet>

Wie anhand des Codes ersichtlich werden in dieser xml-datei CommandSets für unterstützte Sprachen angelegt. Es muss zudem ein Servicehandler implementiert sein, hier zum Beispiel Adventure Works Command service.

ServiceHandler

ModelKlasse

Die Modelklasse (im Referenzprojekt Trip) erbt von INotifyPropertyChanged ruft NotifyPropertyChange auf wenn einzelne Felder geändert wurden.

private void NotifyPropertyChanged(string propertyName)

{

PropertyChangedEventHandler handler = PropertyChanged;

if (handler != null)

{

PropertyChangedEventArgs args = new PropertyChangedEventArgs(propertyName);

handler(this, args);

}

}

Trip wird in einer Model-Klasse namens TripStore gespeichert. Das Attribut dazu ist vom Typ ObservableCollection<Trip>. Die App hat einen Stammordner auf dem Speicherplatz des Gerätes angelegt, in der die möglichen Cortana-Befehle liegen. Dieser kann mit Windows.Storage.ApplicationData.Current.LocalFolder angezapft werden.

Memo an mich selbst: das Begrüßen mit eigenem Namen geht noch nicht, DIE muss die Bibliotheken richtiger verlinken. Hier Website mit Codebeispiel: <https://stackoverflow.com/questions/18283255/how-to-implement-user-identity-name-in-global-asax-on-a-new-web-application>

Da gibt es spezielle Bibliotheken von Cortana zu, s.a. <https://docs.microsoft.com/en-us/cortana/reference/contextual-info>

ILB Baukasten

Der ILB Baukasten ist eine HTML-Seite die neben der Adresse und der Testart bzw. dem Förderprogramm 4 Antagsseiten unterstützt, wie folgt dargestellt, die hochgeladen werden sollten



Auf der Seite zum Formular ausfüllen müssen folgende Angaben gemacht werden:

* [1. Angaben zum antragstellenden Unternehmen](http://10.150.50.21/formularservice/formular/A_FOREX_ANTRAG_ESF_2/appl/d556026e-991d-11e7-9fb1-27c0f1da4ec4/#SECTION772)
* [2. Sitz des Antragstellers](http://10.150.50.21/formularservice/formular/A_FOREX_ANTRAG_ESF_2/appl/d556026e-991d-11e7-9fb1-27c0f1da4ec4/#SECTION593)
* [3. Rechtsform und Geschäftsform des Antragstellers](http://10.150.50.21/formularservice/formular/A_FOREX_ANTRAG_ESF_2/appl/d556026e-991d-11e7-9fb1-27c0f1da4ec4/#SECTION200)
* [4. Angaben zum angemeldeten Gewerbe](http://10.150.50.21/formularservice/formular/A_FOREX_ANTRAG_ESF_2/appl/d556026e-991d-11e7-9fb1-27c0f1da4ec4/#SECTION225)
* [5. Registereintragungen](http://10.150.50.21/formularservice/formular/A_FOREX_ANTRAG_ESF_2/appl/d556026e-991d-11e7-9fb1-27c0f1da4ec4/#SECTION650)
* [6. Gesetzliche Vertreter](http://10.150.50.21/formularservice/formular/A_FOREX_ANTRAG_ESF_2/appl/d556026e-991d-11e7-9fb1-27c0f1da4ec4/#SECTION290)
* [7. Weitere Angaben zum Antragsteller](http://10.150.50.21/formularservice/formular/A_FOREX_ANTRAG_ESF_2/appl/d556026e-991d-11e7-9fb1-27c0f1da4ec4/#SECTION790)
* [8. Angaben zur Antragstellung](http://10.150.50.21/formularservice/formular/A_FOREX_ANTRAG_ESF_2/appl/d556026e-991d-11e7-9fb1-27c0f1da4ec4/#SECTION432)
* [9. Ort der beantragten Weiterbildungsmaßnahme/n](http://10.150.50.21/formularservice/formular/A_FOREX_ANTRAG_ESF_2/appl/d556026e-991d-11e7-9fb1-27c0f1da4ec4/#SECTION523)
* [10. Daten zur beantragten Weiterbildungsmaßnahme/n](http://10.150.50.21/formularservice/formular/A_FOREX_ANTRAG_ESF_2/appl/d556026e-991d-11e7-9fb1-27c0f1da4ec4/#SECTION990)
* [11. Anzahl Mitarbeiter/innen & zu qual. Teilnehmer/innen](http://10.150.50.21/formularservice/formular/A_FOREX_ANTRAG_ESF_2/appl/d556026e-991d-11e7-9fb1-27c0f1da4ec4/#SECTION578)
* [12. Einzelne Weiterbildungsinhalte & fav. Bildungsanbieter](http://10.150.50.21/formularservice/formular/A_FOREX_ANTRAG_ESF_2/appl/d556026e-991d-11e7-9fb1-27c0f1da4ec4/#SECTION248)
* [13. Gesamtfinanzierung](http://10.150.50.21/formularservice/formular/A_FOREX_ANTRAG_ESF_2/appl/d556026e-991d-11e7-9fb1-27c0f1da4ec4/#SECTION869)
* [14. Bestätigungen](http://10.150.50.21/formularservice/formular/A_FOREX_ANTRAG_ESF_2/appl/d556026e-991d-11e7-9fb1-27c0f1da4ec4/#SECTION824)
* [15. Erklärungen](http://10.150.50.21/formularservice/formular/A_FOREX_ANTRAG_ESF_2/appl/d556026e-991d-11e7-9fb1-27c0f1da4ec4/#SECTION725)
* [Anlagen](http://10.150.50.21/formularservice/formular/A_FOREX_ANTRAG_ESF_2/appl/d556026e-991d-11e7-9fb1-27c0f1da4ec4/#SECTION762)
  + [4.2 Zahl der in das Projekt eingetretenen TN](http://10.150.50.21/formularservice/formular/A_FOREX_ANTRAG_ESF_2/appl/d556026e-991d-11e7-9fb1-27c0f1da4ec4/#SECTION239)
  + [4.3 Zahl der eingetretenen (TN) nach Altersgruppen](http://10.150.50.21/formularservice/formular/A_FOREX_ANTRAG_ESF_2/appl/d556026e-991d-11e7-9fb1-27c0f1da4ec4/#SECTION355)
  + [4.4 Zahl der eingetretenen (TN) nach best. Bevölkerungsgruppen](http://10.150.50.21/formularservice/formular/A_FOREX_ANTRAG_ESF_2/appl/d556026e-991d-11e7-9fb1-27c0f1da4ec4/#SECTION159)
  + [4.5 Zahl der eingetretenen (TN) nach ihrem höchsten Bildungsabschluss/-stand](http://10.150.50.21/formularservice/formular/A_FOREX_ANTRAG_ESF_2/appl/d556026e-991d-11e7-9fb1-27c0f1da4ec4/#SECTION451)

TextBoxes in Visual Studio, Orientation

Für die Übergänge zwischen Portrait und Landscape ist bislang die Holzhammer-Methode implementiert. Die Navigation von einer Detaildatenmaske zur anderen funktioniert noch nicht. Folgende Möglichkeit für das Feld Gemeindeschlüssel gefunden:

<TextBlock>

<Span xml:space="preserve"><Run>Open </Run><Hyperlink NavigateUri="http://www.bing.com">Bing</Hyperlink><Run> in your browser.</Run></Span>

</TextBlock>

* Soll bewirken, dass die Nachschlagewebsite aus dem Formular heraus geöffnet werden kann.
* Weitere Nützlichkeiten zur Validierung sind .MaxLength und .InputScope XAML-Attribute

Es ist leider nicht möglich in XAML ohne selbstgeschriebene Validationsmethoden auf Bestimmte Formate zu beschränken, z.B. nur Texteingabe. Folgende URL gibt überblick über Methoden, die dazu eingebunden werden müssten: <https://www.codeproject.com/articles/220519/numbers-or-characters-only-textbox-validation-in-c>

Quelle zum Nachschlagen des Gemeindeverzeichnis war übrigens ein überalterter Link. Müsste eigentlich sein: <http://www.statistik-portal.de/Statistik-Portal/gemeindeverz.asp>

## Validation

Für die Validation ist es wichtig Datenbindung in UWP zu verstehen. Im Wesentlichen werden Benutzereingaben in die Eingabemasken mit der INotifiPropertyChanged-Schnittstelle an das entsprechende Datenmodell gebunden. Im MMVM-Pattern, alias Model-ModelView-Model-Pattern, sind aber noch weitere Klassen beteiligt, wie das folgende Ablaufdiagramm zeigt:

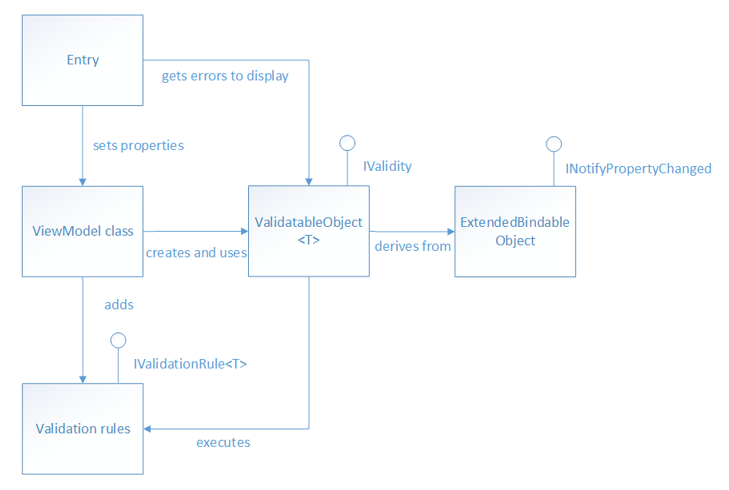


Abbildung : Validierende Klassen des MVMM Programmiermusters

Das Binden an eigene Validatoren bzw. Validationsregeln kann wie zum Beispiel unter <https://stackoverflow.com/questions/6087468/wpf-validation-of-textbox-what-should-the-binding-look-like> beschrieben, stattfinden.

In UWP gibt es dazu von Usern geschriebene nuget-Erweiterungspakete die eine Validierung der Kontrollelemente zulassen. Beispielsweise das Windows Runtime Xaml Toolkit[[12]](#footnote-12) von xyzzer. Dieses unterstützt den User beim Ausfüllen von TextBoxes farblich mit Feedback ob ein Pflichtfeld ausgefüllt wurde oder nicht, was die minimal erforderliche Länge angeht und kann bei den Typen auf Numeric testen. Es besitzt leider keine spezifischen Tags zum Validieren von email, URL oder alphanumerischem Content. Das Feld Pattern ist leider nur ein Behelf. Dafür enthält dieses Nuget-Package einige weitere Kontroll-Elemente, zum Beispiel die [Auto-Suggest-TextBox](https://docs.microsoft.com/en-us/windows/uwp/controls-and-patterns/auto-suggest-box) oder den WebBrowser, ein erweitertes WebView, das GoBack-Button anbietet, Favoriten und ähnliche ansonsten nicht in WebView enthaltene Elemente. Leider ist die Dokumentation[[13]](#footnote-13) dürftig.

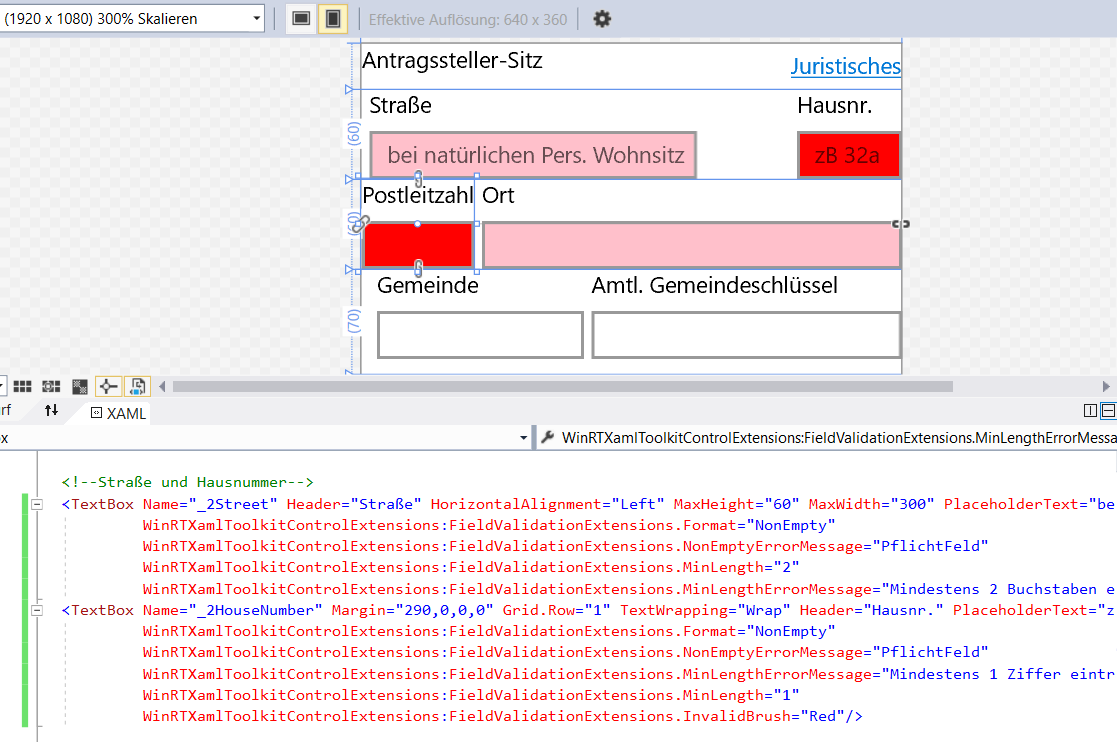


Abbildung : Validierung mittels des WinRTXamlToolkit

Eine Alternative die auch auf alphanumeric und zum Beispiel Telefonnummer testen kann könnte im Nuget-Paket WINUX.UWP hinterlegt sein. Unter Einbindung bestimmter Namespaces kann darauf zugegriffen werden, wie in <https://www.youtube.com/watch?v=AqKwCZmv-ao> beschrieben.

Daneben gibt es die Möglichkeit, eigene Validationsregeln zu definieren, wie im Beispiel zur Klasse ValidationRule [[14]](#footnote-14)angedeutet:

public class AgeRangeRule : ValidationRule

{

private int \_min;

private int \_max;

public AgeRangeRule()

{

}

public int Min

{

get { return \_min; }

set { \_min = value; }

}

public int Max

{

get { return \_max; }

set { \_max = value; }

}

public override ValidationResult Validate(object value, CultureInfo cultureInfo)

{

int age = 0;

try

{

if (((string)value).Length > 0)

age = Int32.Parse((String)value);

}

catch (Exception e)

{

return new ValidationResult(false, "Illegal characters or " + e.Message);

}

if ((age < Min) || (age > Max))

{

return new ValidationResult(false,

"Please enter an age in the range: " + Min + " - " + Max + ".");

}

else

{

return new ValidationResult(true, null);

}

}

}

Zudem ist es möglich über Regex-Strings und spezielle Ereignishandler wie Key\_Down und Text\_Changed zu validieren. Grob gesagt wird ein Regex durch die Zeichen^und $ begrenzt. In {} kann die minimal bzw. maximal zulässige Länge des Textes angegeben werden und in [] die zulässigen Zeichen. Ein Regex wie ^[A-Z0-9.\_%+-]{1,50}@[A-Z0-9.-].[A-Z]{2,}$ soll eine Emailadresse validieren können.

Fehlerbenachrichtigen lassen sich auch über die Implementation des IDataErrorInfo-Interfaces behandeln. Das evt. Zur Prism-Bibliothek gehört. Quelle: <https://msdn.microsoft.com/en-us/library/gg405484(v=pandp.40).aspx>

Probleme:

* Invertierung des Regex funktioniert nicht
* Code Contracts lässt bei fehlenden Daten abstürzen, aber Benutzer kann „ignorieren“ wählen und das Programm weiterlaufen lassen.
* Zur Laufzeit zu checken hat unerwartete Auswirkungen. Es ist mir bislang nicht gelungen, bestimmte Keys von der Eingabe bei bestimmten Feldern auszuschließen. Es ist auch die Frage, ob das sinnvoll ist, denn Copy-Paste oder Stimmeingabe sind nicht vom Key\_Down Event betrofffen. Der einzige sinnvolle Ansatz ist da TextChanged, denke ich
* Es ist unerwartet kompliziert die Farbumrandung eines Feldes zu ändern, dazu muss zum Beispiel Rectangle genutzt werden. Leichter die Hintergrundfarbe zu ändern.

Notizen 20.09.2017

->Feedback: Erst mal egal welche Felder

->Anforderung: Cortana soll aus Feldbezeichnung und Typ ableiten, welche Informationen gefragt sind

-> Anforderung: über symbolische Sprache recherchieren, die Cortana vielleicht verwendet um Strategie zu entwickeln z.B. Suche mir Tankstellen im Umkreis von 10 km

->weiche Anforderung: HTML Schnittstelle zu Cortana suchen



25.09.2017

## Visual Studio Online

Visual Studio Online ist ein Versionierungssystem, das auf Git basiert. Es kann kostenfrei genutzt werden und hat die Möglichkeit Azure Bots online zu testen. Zwar besteht diese Möglichkeit auch lokal über den Bot Framework Emulator, hat bei mir aber aus bislang unbekannten Gründen die Verbindung nie authentifizieren können.

Speech Synthesis Markup Language

Intermediäre Sprache für die Ausgabe zum Beispiel von Text-To-Speach. XML ähnlich, arbeitet eng mit der Klasse SpeechSynthesizer zusammen. Das Startrek-Races-Dictionary[[15]](#footnote-15) zeigt eine Verwendung. Es gibt diverse xml-Attributähnliche Tags die die Aussprache steuern[[16]](#footnote-16).

Dies ist wichtig um einem Bot das Sprechen beizubringen, wie im Microsoft Tutorial <https://docs.microsoft.com/en-us/cortana/tutorials/bot-skills/teach-your-bot-to-speak> besprochen.

Rollerdice-example: App ID: 17ef92b2-6d65-4964-8810-dde3106c3cf3

## Git

Git kommunikation scheint von den Proxy Einstellungen abhängig zu sein. Diese sind mit folgendem Code über die git Bash abänderbar:

git config --global http.proxy http://proxy2.inndom.intern:3128

git config --global https.proxy <http://proxy2.inndom.intern:3128>

* Wobei es sich um die Proxyadressen:Portnummern von innobis handelt.

c6008a8d-c202-4f80-8925-238c2b5ea93e LuisBot Appid kennwort 5vpir8BN5uDQgt64VZ0OGza Registrierung

In der Git-Extension von Visual Studio gibt es keinen forced push. Daher muss manuell ein erzwungener Push erfolgen, wenn es Probleme mit der Synchronisation in einem nicht von einem Team verwalteten Projekt gibt. Speziell in folgendem Fall:



Abbildung : Synchronisationsprobleme in Visual Studio

In der Git Bash hilft dann folgender Befehl git push –f origin master. Hier einmal demonstriert:



Abbildung : Lösen der Synchronisationproblematik

Zum Ignorieren der Zertifizierungsdatei des verwendeten git-Servers kann verwendet werden: git config http.sslVerify "false"

Cortana UWP

In UWp ist es möglich eine solche App mit Cortana zu steuern als Vordergrund oder Hintergrundprozess[[17]](#footnote-17). Die PhrasenListe ist dynamisch anpassbar laut <https://docs.microsoft.com/en-us/cortana/voicecommands/dynamically-modify-voice-command-definition-vcd-phrase-lists>

## LUIS.ai

Luis = Language understanding intelligent service ist eine Online AI die für eine bestimmte App ein Intent-Modell aus Sprachelementen (=Entities) erstellen kann. Dieses Modell kann in die App eingebunden werden und mir Äußerungen (sogenannten Utterances) trainiert werden. Eine Batchdatei ist eine Datei die mindestens 1000 solcher Äußerungen enthält. Das Luis-Modell muss mit einem Azure Key registriert sein.



Abbildung : Quelle: https://www.luis.ai/home

Der Vorteil von LUIS ist, dass das System in der Lage ist nach einigen Interaktionen aktiv zu lernen, indem es die Äußerungen untersucht. Anschließend macht es Vorschläge an den Entwickler, welche Äußerungen übernommen und beschriftet werden sollten.

|  |  |  |  |
| --- | --- | --- | --- |
| Botname | App-ID | Kennwort | endpoint |
| LuisTalkBot | 0b15fa50-1197-4f5e-a21d-e22d1ed36f01 | 6yywromoi1vgz326GkhsNKB | https://LuisTalkBot.azurewebsites.net/api/messages |
| StockBot | 93dfe8db-873d-46aa-9604-d5c22df499ad | a5de48ea62014f2bbdc4ad05943f2081 (Subskriptionsschlüssel) | <https://westus.api.cognitive.microsoft.com/luis/v2.0/apps/93dfe8db-873d-46aa-9604-d5c22df499ad?subscription-key=a5de48ea62014f2bbdc4ad05943f2081&timezoneOffset=0&verbose=true&q=> |
| ListenToMeBot | 57c99374-f736-4f77-a952-1a1fe90500be | a5de48ea62014f2bbdc4ad05943f2081 | [https://westus.api.cognitive.microsoft.com/luis/v2.0/apps/57c99374-f736-4f77-a952-1a1fe90500be?subscription-key=a5de48ea62014f2bbdc4ad05943f2081&verbose=true&timezoneOffset=0&q=](https://westus.api.cognitive.microsoft.com/luis/v2.0/apps/57c99374-f736-4f77-a952-1a1fe90500be?subscription-key=a5de48ea62014f2bbdc4ad05943f2081&verbose=true&timezoneOffset=0&q=" \t "_blank) |

Die Einbindung des LUIS-Modells in den Code kann über verschiedene Wege erfolgen. Zum einen ist es möglich Anfragen an die URL des Modells zu schicken, wie zum Beispiel im StockApp-LUIS-Beispiel[[18]](#footnote-18) beschrieben. Dabei wird die LUIS-App-ID und der Subskriptions-Key in der URL inklusive der Anfrage an die LUIS-domain übergeben.

Anmerkung zu dem Beispiel: Das Tutorial weist stellenweise Methoden auf in Messagescontroller, die nicht mehr in der Klasse Messages enthalten sind. Dieses Problem lässt sich durch Erstellen eines Dialogs und Auslagern der …

Eine UWP-App lässt sich programmatisch mit einem Bot verknüpfen, indem sie diesen erstellt. Das Beispielprojekt [BestFriend](http://metulev.com/meet-the-speech-platform-in-windows-10/?utm_source=Direct) von Nikola Metulev nutzt dieses Konzept. Nach dem Importieren dazu in Visual Studio 2015 Community rechtsklick auf Projekt und „Als Startup-Projekt kennzeichnen“. Dann weiter wie gewohnt (Build, Deploy).

Der Verbindungsaufbau zum http-Client in diesem Beispiel erfordert mehrere Zwischenschritte. Ein Verbindungsaufbau über DataContractJsonSerializer ist leichter.

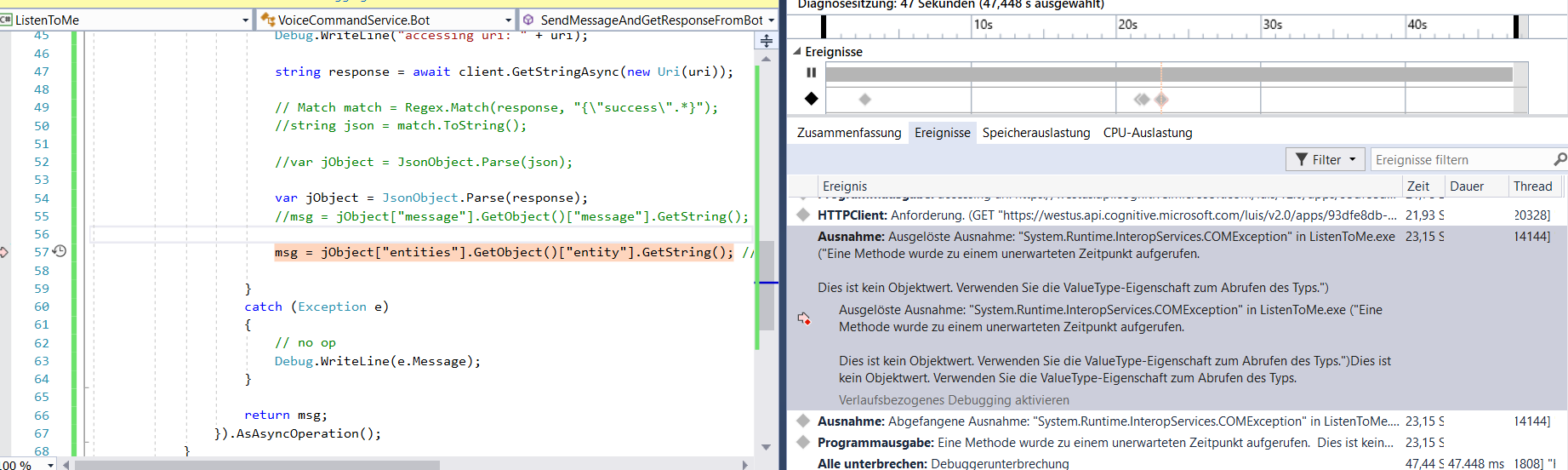


Abbildung :JSON problem

Im LUIS.ai Portal gibt es bereits eine von erkannten Intents:

|  |  |  |
| --- | --- | --- |
| None | 0 |  |
| Utilities.Cancel | 10 |  |
| Utilities.Confirm | 10 |  |
| Utilities.FinishTask | 10 |  |
| Utilities.Goback | 10 |  |
| Utilities.Help | 10 |  |
| Utilities.Repeat | 10 |  |
| Utilities.ShowNext | 10 |  |
| Utilities.ShowPrevious | 10 |  |
| Utilities.StartOver | 9 |  |
| Utilities.Stop | 10 |  |
| address | 4 |  |
| echo | 4 |  |
| name |  |  |

(Wobei die letzen 2 selbstgeschriebene Intents sind zum Testen).

## XAML – die GUI

Die unterschiedlichen Bildschirmgrößen (z.B. Tablet width 650 pixel, Deskop 1024 und Handy 240 Pixel Bildschirmbreite) führen dazu, dass das Layout auf verschiedene Größen optimiert werden sollte. Obschon es möglich ist für bestimmte Gerätefamilien Abfragen zu erstellen, ist der AdaptiveTrigger[[19]](#footnote-19) in XAML eine bessere Lösung, zumal Fenster vom User absichtlich skaliert werden können.

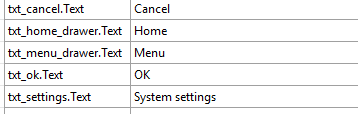
### Lokalisation

Soll die App für mehrere Sprachen verfügbar sein, empfiehlt sich das Anlegen von String-Ressourcen-Dateien (.resx) und das Verweisen darauf. Dabei ist zu beachten, dass in XAML die Texte von Kontrollelementen folgend verknüpft werden[[20]](#footnote-20):

var loader = new ResourceLoader();

var resourceString = loader.GetString("txt\_ok");

txt.Text = resourceString;

[](https://i.stack.imgur.com/khJRN.png)

Man kann dann den TextBlock auf die Ressource wie folgt verweisen lassen:

<TextBlock x:Uid="txt\_cancel" />

Auch dies ist möglich, hängt aber von dem Pfad der Ressource im Projekt ab:

<TextBlock x:Uid="/Resources/txt\_settings" />

Zur Lokalisation in XAML gehört auch das Anpassen von Zahlen, Währungen und Daten als Format. Hierzu folgender Link: <https://docs.microsoft.com/en-us/windows/uwp/globalizing/use-global-ready-formats>

1. Vgl. <file:///I:/Temp/Friederike/Hands-Free_Speech-Based_Navigation_During_Dictatio.pdf> S.11 [↑](#footnote-ref-1)
2. Vgl. <https://pdfs.semanticscholar.org/aec4/93e7d1fa639d269078e376a097ec20f4b3cd.pdf> S.3 [↑](#footnote-ref-2)
3. Vgl UWP Tutorial von Collin Blake, s.a. https://www.youtube.com/watch?v=8lLJt0LkB3s&list=PLi2hbezQRVS0cPMeW3uDlUHnO\_rPvJCV9&index=4 [↑](#footnote-ref-3)
4. Vgl. https://docs.microsoft.com/en-us/cortana/tutorials/setup-dev-env [↑](#footnote-ref-4)
5. Durch Download der exe-Datei unter <https://github.com/Microsoft/BotFramework-Emulator/releases/tag/v3.5.31> kostenlos möglich [↑](#footnote-ref-5)
6. Vgl. <file:///I:/Temp/Friederike/Bot_Framework.pdf> S.6 [↑](#footnote-ref-6)
7. Vgl. <http://autofac.readthedocs.io/en/latest/getting-started/index.html> Gesehehen am 06.09.17 [↑](#footnote-ref-7)
8. Vgl. Bot\_Framework.pdf, S. 153 [↑](#footnote-ref-8)
9. Vgl. <https://docs.microsoft.com/en-us/cortana/voicecommands/voicecommand-design-guidelines> [↑](#footnote-ref-9)
10. Vgl. <https://developer.xamarin.com/guides/xamarin-forms/creating-mobile-apps-xamarin-forms/> [↑](#footnote-ref-10)
11. <https://docs.microsoft.com/de-de/windows/uwp/get-started/get-uwp-app-samples> Gesehen am 11.09.17 [↑](#footnote-ref-11)
12. Vgl. <https://www.nuget.org/packages/WinRTXamlToolkit.UWP/> [↑](#footnote-ref-12)
13. Vgl. <https://github.com/xyzzer/WinRTXamlToolkit> [↑](#footnote-ref-13)
14. Vgl. <https://msdn.microsoft.com/en-us/library/system.windows.controls.validationrule(v=vs.110).aspx> [↑](#footnote-ref-14)
15. Vgl. <https://www.youtube.com/watch?v=ll9oo5RoX9o> [↑](#footnote-ref-15)
16. Vgl. <https://docs.microsoft.com/en-us/cortana/reference/ssml> [↑](#footnote-ref-16)
17. Vgl. <https://docs.microsoft.com/en-us/cortana/voicecommands/launch-a-background-app-with-voice-commands-in-cortana> [↑](#footnote-ref-17)
18. Vgl. <http://www.c-sharpcorner.com/article/an-interactive-bot-application-with-luis-using-microsoft-bot/> [↑](#footnote-ref-18)
19. Vgl. <https://www.captechconsulting.com/blogs/uwp-adaptive-layout-best-practices-part-i-adaptive-triggers> [↑](#footnote-ref-19)
20. <https://stackoverflow.com/questions/36130859/access-resx-string-resource-within-xaml-layout-in-uwp> [↑](#footnote-ref-20)