Protokoll PicDB

# Allgemein

Wir haben ein Bildverwaltungstool gebaut. Diese wurde mit Windows Presentation Foundation (kurz WPF) umgesetzt um eine schöne UI zu bekommen.

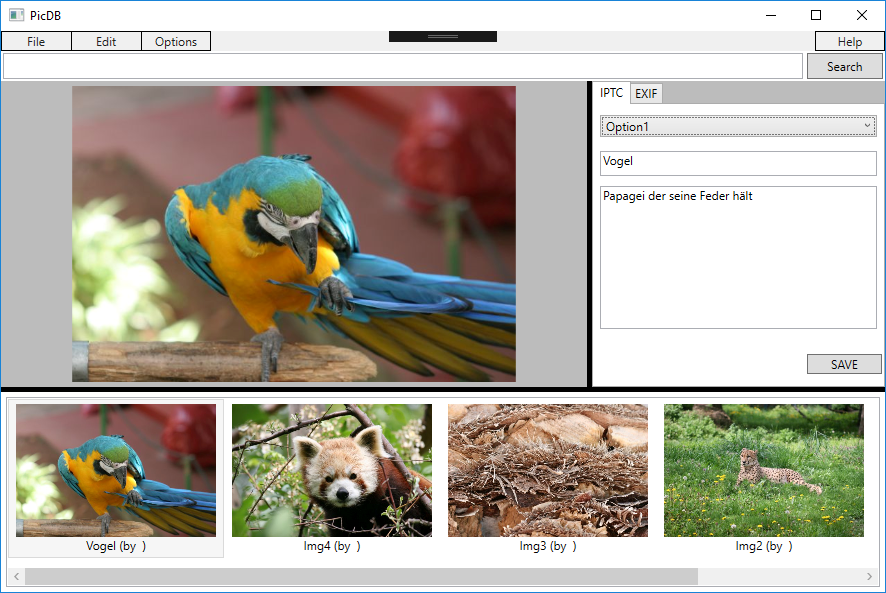
Es werden nicht alle Funktionen umgesetzt, die für Bildverwaltungstools üblich sind, jedoch erhalten wir einen Einblick in die Programmierung größerer Programme.

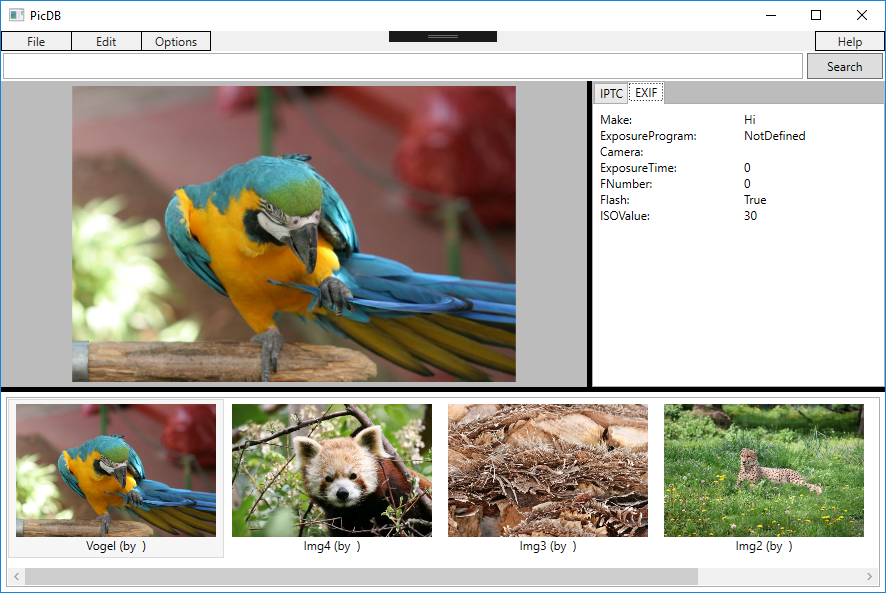
Die Basisfunktionen und die Kommunikation zwischen den Klassen wird mittels vorgegebener Unittests abgedeckt und danach mit der UI verknüpft.

Verwendet wird git um zwischen den Teammitgliedern und dem Lektor Sourcecode auszutauschen und über Jenkins mittels Unit-tests bewertet zu werden.

# UI – Fenster

## MainWindow



Das Programm besteht aus einem MainWindow in dem die Bilder angezeigt werden. Dabei wird das aktuell angewählte Bild groß im oberen linken Bereich angezeigt. Beim Start des Programms ist noch kein Bild gewählt, deswegen erscheint der Bereich zu beginn grau. Die Thumbnails im unteren Bereich lassen sich zur Seite scrollen und werden mit dem Dateinamen ohne Endung angezeigt. In der Klammer steht der Name des Photographen. Auf der rechten Seite des Hauptbildes sieht man die IPTC und EXIF- Informationen, die man mit Klick auf die entsprechenden Tabs wechseln kann. Die IPTC-Informationen können geändert werden, die EXIF-Informationen können nicht geändert werden, da diese ja bei der Aufnahme des Bildes erstellt werden und das Ändern nicht sinnvoll ist

Im oberen Bereich befinden sich die typischen Menü-Buttons, die man bei jedem Programm findet. Die meisten dieser Funktionen sind ebenfalls nicht implementiert, sie zeigen jedoch wie umfangreich und erweiterbar unser Programm sein kann.

Unter dem Punkt Edit findet man den Unterpunkt Photographer, der ein neues Fenster öffnet: Das PhotographerWindow

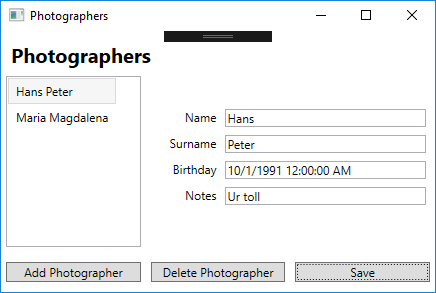
Bei Options findet man die Funktionen „Make Tags“ und „Make PDF“ die Tags zu den Bildern zuweisen und ein PDF des aktuellen Bilds erstellt.

Mit der darunterliegenden Searchbar können Bilder gefiltert bzw. gesucht werden. Hierzu ist ein klick auf den Search-Button notwendig.

Die Schwarzen Balken, die das Fenster aufteilen haben zu Beginn einen fixen Abstand, dieser kann jedoch zur Laufzeit vergrößert und verkleinert werden.

Hierbei skaliert das ausgewählte Bild mit, die Thumbnails bleiben jedoch bei ihrer ursprünglichen Größe.

# PhotographerWindow



Im PhotographerWindow können Photographen bearbeitet und erstellt werden. Auf der linken Seite sieht man die bereits vorhandenen Photographen und auf der rechten Seite die Detail-Ansicht des aktuell ausgewählten Photographen.

Im unteren Bereich sind die Buttons angesiedelt die, die Datensätze steuert.

Wird Add Photographer geklickt wird ein neuer Eintrag bei den Photographen erstellt. Wird jedoch Save geklickt wird der aktuelle Photograph überschrieben. Delete löscht den gewählten Photographen und setzt das aktuelle Element auf null.

# Umsetzung

Es wurden alle Fenster mit WPF / XAML umgesetzt. Hierbei wurde so gut es ging mit Bindings gearbeitet um Source-Code zu reduzieren. Wenn es nicht explizit erwähnt wird befindet sich der Code in den .xaml-Dateien

## MainWindow

Um auf Daten zugreifen zu können die für das Fenster wichtig sind, wird im Window-Tag der DataContext gesetzt. In unserem Fall ist dies das MainWindowViewModel. So können alle Funktionen des MainWindowViewModels im MainWindow verwendet werden. Für das Binding ist das essenziell damit dieses überhaupt funktioniert.

Über das ganze Fenster wird ein Grid gelegt wobei 5 Bereiche entstehen:

* Menü-Leiste
* Search-Leiste
* Hauptbereich mit Bild und IPTC und EXIF-Informationen
* Gridsplitter
* Thumbnails

In der Menü-Leiste befinden sich alle Menü-Buttons des oberen Bereichs. Wir haben dafür das xaml-Element „Menu.ItemPanel“ verwendet, welches ein Template braucht um die Formatierung schön zu gestalten. Für die Aufteilung haben wir wieder ein Grid genommen.

In dem MenuItem kann man einen Header definieren (z.B: File oder Edit) und deren Unterpunkte (z.B. Save, Photographer,…)

Die Search-Zeile besteht nur aus einem Textfeld welches mit einem Binding auf das Textfeld ausgestattet ist und einem Searchbutton der die Search auslöst. Für die Aufteilung wurde wieder ein Grid zur Hilfe genommen.

Der größte Bereich, der Main-Bereich, teilt sich in 3 Teile. Im ersten ist das aktuell gewählte Bild, das mit dem Bereich mitscaliert. Im Image Tag muss dafür nur als Source {Binding CurrentPicture.Filepath} eingetragen werden. Das bewirkt dass immer wenn bei den Thumbnails ein neues Bild angewählt wird, das Bild im Hauptbereich geändert wird.

Der 2. Teil im Hauptbereich ist für einen Gridsplitter vorgesehen. Dieser bewirkt dass man die Teile in deren Größe verändern kann. Hier gibt man nur an ob er Vertikal oder Horizontal eingestellt werden soll und welche Bereiche in deren Größe verändert werden sollen. In unserem Fall wären das PreviousAndNext also das linke und rechte Feld. Um es als Gridsplitter zu identifizieren haben wir uns für die Hintergrundfarbe schwarz entschieden.

Der 3. Teil ist für IPTC und EXIF-Informationen gedacht. Per Default werden die IPTC-Informationen angezeigt. Das kann jedoch durch Klicken auf den Tab EXIF geändert werden. Dies setzten wir mit einem TabControl-Element um. Im IPTC-Tab werden Titel des aktuellen Bilds, sowie Notizen als Textfeld und der Photograph des Bildes als ComboBox angezeigt. Der Save-Button ist momentan nicht implementiert, da durch das Binding gleich alles aktualisiert wird. Der Plan ist es mit dem Save-Button einem Datenbank-Befehl abzusetzen, der die Änderungen in die Datenbank schreibt.

Für die einzelnen Felder werden Bindings für PicTitle und PicNotes verwendet um diese immer aktuell zu halten.

Der EXIF-Part verwendet nur eine große Textbox, die man nicht bearbeiten kann.

In dieser wird Multibinding-Element verwendet um die Informationen als Liste wiederzuspiegeln.

Im Multibinding-Element wird angegeben wie der string auszusehen hat. Die Überschriften der Informationen stehen zuerst dann getrennt durch die notwendige Anzahl an Tabstopps der Binding-Wert gefolgt von einem Zeilenumbruch.

Tabstopps werden im xaml mit &#x09; dargestellt. Jeder Wert der Bindings wird über eine geschwungene Klammer mit dem Index des Bindings repräsentiert zum Beispiel {0} oder{4}. Ein Zeilenumbruch wird im xaml mit &#x0a; umgesetzt.

Der 4. Bereich ist wieder ein Grid-Splitter. Nur ist dieser jetzt horizontal

Der 5. Und letzte Bereich des MainWindows sind die Thumbnails. Diese wurden mit einer ListBox umgesetzt. Im Template geben wir an dass er die elemente Horizontal angeordnet werden sollen.

Die einzelnen Elemente sind zusammenhängende Bilder mit dazugehörenden Bildtitel. Auch hier werden Bindings verwendet

Das erste bereits im ListBox-Element bei SelectedItem. Hier wird das CurrentPicture gesetzt und so weiß das Programm welches Bild oben angezeigt werden muss.

Als ItemSource wird die List der List vom MainWindowViewModel herangezogen.

Durch ein weites Binding am DisplayName kann man eine Änderung in den IPTC-Informationen gleich in den Thumbnails sehen.

Im MainWindow.xaml.cs sind die Funktionen der Buttons angesiedelt. Prinzipiell wäre es auch möglich gewesen diese mit einem Binding auf eine Command-Klasse umzusetzen, aber wir wollten alle Funktionen von WPF zumindest einmal ausprobiert haben. Deswegen sind hier Click-Eventhandler implementiert worden.

Um ein neues Fenster, nämlich das PhotographerWindow zu öffnen muss man ein Objekt der Klasse PhotographerWindow deklarieren und den Befehl ShowDialog() ausführen.

## MainWindowViewModel

Hier liegen Funktionen und Parameter auf die die das MainWindow zugreift.

Der Construktor setzt das CurrentPicture auf null und befüllt die pictureList mit Bildern.

Das CurrentPicture liefert das aktuelle Bild zurück.

Die List-Funktion liefert ein neues Objekt der PictureListViewModel zurück.

Die Search-Funktion liefert ein neues Objekt der SearchViewModel zurück.

## PhotographerWindow

Genau wie beim MainWindow wird der DataContext auch hier angegeben um die Bindings und Funktionen verwenden zu können. Hier wird nur auf das PhotographerWindowViewModel verwiesen.

Das PhotographerWindow besteht aus 4 Bereichen:

* Oben die Überschrift
* Links oben die Liste an Photographen
* Rechts oben die Details des ausgewählten Photographs
* Unten die Buttonleiste

Um diese Aufteilung zu gewährleisten verwendeten wir erneut ein Grid.

Die Überschrift ist ein Label, dass wir mit FontSize und FontWeight angepasst haben.

Die Liste an Photographen funktioniert ähnlich wie die der Thumbnails, nur dass diese nicht horizontal angeordnet werden sondern vertikal.

Als Element in der Liste ist ein weiteres Grid in dem vorname und nachname angezeigt werden.

Im Detail Bereich werden mit Stackpanels die Überschriften als TextBlock und die Daten als TextBox verwendet.

Die Buttons im unteren Bereich werden genau so angestoßen wie im MainWindow