Dokumentacja projektu nr 21:
Napisy na zdjęciach

1. Tytuł projektu i autorzy projektu

Tytuł projektu to **“Napisy na zdjęciach”**. Został on zrealizowany w trzyosobowym zespole w składzie:

- Olga Kubiszyn
- Gabriel Naleźnik
- Jan Zajda

2. Opis projektu

Celem projektu było stworzenie przeglądarki do zdjęć, która umożliwia odczytanie z nich danych EXIF i IPTC oraz umieszcza na nich proste napisy.

3. Założenia wstępne przyjęte w realizacji projektu

- możliwość wskazania katalogu, z którego zdjęcia będą przeglądane.
- po wybraniu katalogu, ekran programu zostaje podzielony na dwie części. Po lewej zostaną wyświetlone miniatury zdjęć z katalogu, a po prawej informacje EXIF i IPTC dla aktualnie wybranego zdjęcia.
- po dwukrotnym kliknięciu na wybranym zdjęciu zostanie ono wyświetlone w powiększeniu w tej części okna, którą zajmowały miniaturki, a informacje dodatkowe pozostają ciągle widoczne
- po ponownym podwójnym kliknięciu, program powraca do trybu wyświetlania miniaturek.
- program generuje plik zawierający listę nazw plików wraz z wybranymi parametrami dla każdego zdjęcia
- program wypisuje wybrane parametry zdjęcia w postaci napisu na zdjęciu i zapisuje tak powstałe zdjęcie pod nową nazwą (czynność ta jest wykonywana w sposób zautomatyzowany dla wszystkich zaznaczonych plików)
- program wczytuje zewnętrzny plik zawierający nazwy plików i proste teksty, które zostają umieszczone na zdjęciach.

4. Analiza projektu

4.1 specyfikacja danych wejściowych

Na wejściu programu można wczytać wybrany katalog ze zdjęciami. Zdjęcia powinny być w formacie jpg (potem inne?). W rozszerzonej wersji programu można także wczytać plik z danymi, które chcemy zapisać na wybranym zdjęciu. Plik ten powinien zawierać nazwę danego zdjęcia oraz dwukropek: “nazwa_zdjęcia:”. Wszystko co po nim następuje zostanie zapisane na zdjęcie aż do nazwy następnego zdjęcia.

4.2 opis oczekiwanych danych wyjściowych

Podstawowymi danymi wyjściowymi są miniaturki zdjęć wyświetlane po lewej stronie oraz informacje EXIF i IPTC dla aktualnie wybranego zdjęcia wyświetlane po prawej stronie. Oprócz tego po dwukrotnym kliknięciu na wybranym zdjęciu, zdjęcie to jest pokazywane w powiększeniu w miejscu, gdzie wcześniej znajdowały się miniaturki. W rozszerzonej wersji danymi wyjściowymi są także wybrane informacje EXIF i IPTC zapisywane do pliku dla zaznaczonych zdjęć. Można też zapisać wybrane lub wczytane z odpowiedniego pliku informacje na zdjęcie i zapisać je jako nowy plik.

4.3 zdefiniowanie struktur danych

Pojedyncze załadowane zdjęcie przechowywane jest w obiekcie klasy LoadedImage. Podczas działania programu, wczytane zdjęcia są przechowywane w kontenerze std::vector. W klasie LoadedImage, bitmapa załadowanego zdjęcia jest zapisana w obiekcie wxBitmap, a wyświetlana jest jako wxBitmapButton.

4.4 specyfikacja interfejsu użytkownika

Komunikacja z użytkownikiem opiera się na korzystaniu z jednego przycisku i 3 opcji z menu:

- przycisk **“Wybierz folder ze zdjęciami”** - użytkownik zostaje poproszony o wskazanie folderu ze zdjęciami, których miniaturki mają zostać wyświetlone w lewej części okna.

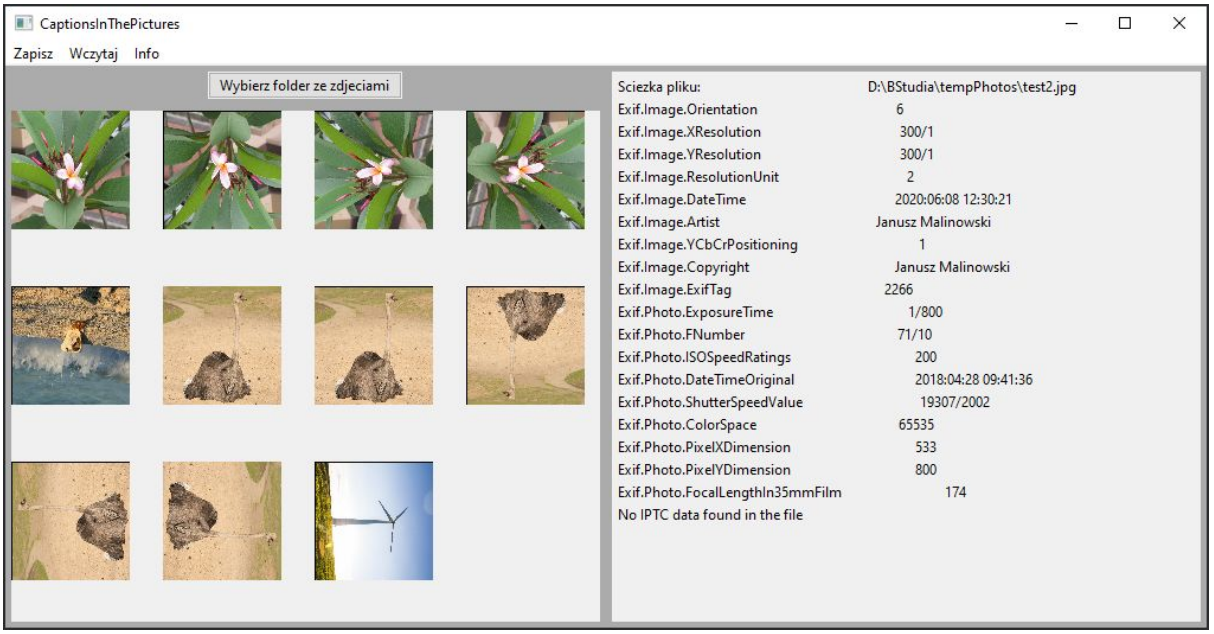
- opcja menu **“Eksportuj dane Exif do pliku”** - użytkownik zostaje poproszony o zaznaczenie z których zdjęć chce zapisać dane exif, wybrać niektóre spośród wszystkich dostępnych informacji oraz zapisać plik tekstowy z danymi w wybranym miejscu na dysku.

- opcja menu **“Eksportuj dane Exif na zdjęciu”** - użytkownik zostaje poproszony o zaznaczenie z których zdjęć chce zapisać dane exif, wybrać niektóre spośród wszystkich dostępnych informacji oraz zapisać początkowe zdjęcie, z informacjami wypisanymi na nim, pod nową nazwą w wybranym miejscu na dysku.

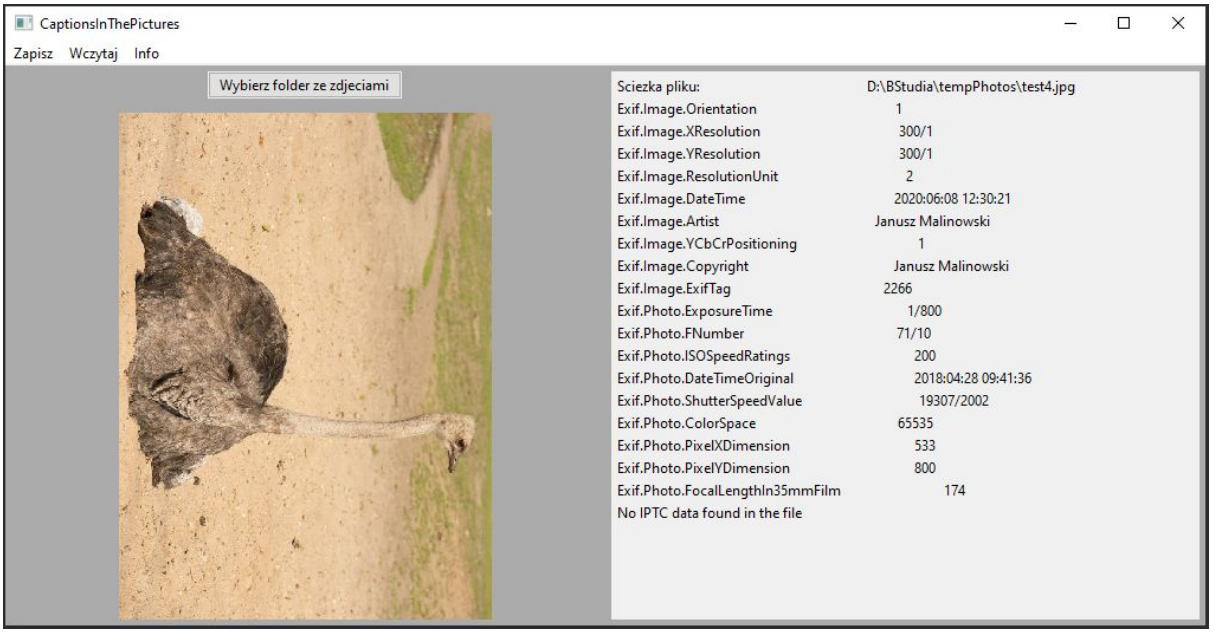
- opcja menu **“Wczytaj napisy z pliku”** - użytkownik zostaje poproszony o wybranie pliku tekstowego z danymi, które chcemy zapisać na wybranych zdjęciach. Program umieszcza podane informacje w postaci napisu na zdjęciu i zapisuje nowe zdjęcie pod nową nazwą, w miejscu wybranym przez użytkownika.

Oprócz tego użytkownik może dwukrotnie kliknąć na wybrane zdjęcie, aby je powiększyć. Po dwukrotnym kliknięciu na powiększone zdjęcie, wyświetlone zostaną ponownie miniaturki.

Rysunek 1: Podstawowy widok okna



Rysunek 2: Widok aplikacji po wczytaniu zdjęć z katalogu



Rysunek 3: Widok aplikacji po powiększeniu zdjęcia

4.5 wyodrębnienie i zdefiniowanie zadań

Zadaniem przewodnim projektu było zrealizowanie wszystkich wymagań podstawowych i rozszerzonych. Projekt został zrealizowany z następującą kolejnością wykonanych zadań:

- Integracja biblioteki wxWidgets z biblioteką Exiv2
- Wypisywanie na ekran danych exif z wybranego zdjęcia
- Realizacja interaktywnych miniaturek zdjęć z wybranego katalogu
- Zapisywanie danych exif i iptc w pliku tekstowym
- Zapisywanie napisów na zdjęciu

4.6 decyzja o wyborze narzędzi programistycznych

Program został wykonany w języku C++ oraz przy wykorzystaniu następujących narzędzi programistycznych:

- wxWidgets (wersja 3.1.3)
- wxFormBuilder (wersja 3.9.0)
- exiv2 (wersja 0.26.0)

Wykorzystane przez zespół zintegrowane środowiska programistyczne to Visual Studio 2017 oraz Visual Studio 2019.

Biblioteka wxWidgets została dobrze poznana podczas zajęć laboratoryjnych i posłużyła w celu skonstruowania interaktywnego GUI. Przy projektowaniu interfejsu pomógł bardzo intuicyjny edytor wxFormBuilder. Biblioteka exiv2 została wykorzystana w celu odczytania danych exif i iptc z wybranych zdjęć. Zdecydowaliśmy się z niej skorzystać, gdyż w bardzo prosty sposób pozwala pobierać ze zdjęcia metadane.

5. Podział pracy i analiza czasowa

W celu ułatwienia dostępu do kodu źródłowego aplikacji, został wykorzystany system kontroli wersji git, a na stronie https://github.com/BlackDilvish/CaptionsInThePictures znajduje się repozytorium z projektem, gdzie każdy z członków grupy miał wgląd do najnowszych zmian w strukturze projektu.

Każdy z członków grupy zajął się następującymi zadaniami:

- **Olga Kubiszyn:** Wyświetlanie powiększonego zdjęcia, miniaturki, wygląd napisów na zdjęciu, GUI, dokumentacja
- **Gabriel Naleźnik:** Pobieranie danych exif i iptc ze zdjęcia, GUI, miniaturki zdjęć, napisy na zdjęciu, dokumentacja
- **Jan Zajda:** Zapisywanie danych exif do pliku, zapisywanie wczytanych z pliku napisów na zdjęciu, GUI, dokumentacja

6. Opracowanie i opis niezbędnych algorytmów

6.1. Algorytm wyboru koloru napisów na zdjęciu

Aby napisy na zdjęciach były widoczne zastosowaliśmy algorytm dostosowujący ich kolor. W punkcie, od którego ma się zaczynać napis jest pobierany kolor w modelu RGB. Następnie składniki R, G, B są obliczane jako

$$X = 255 - Y,$$

gdzie X - składowa koloru napisu, Y - składowa koloru w danym punkcie na zdjęciu.
Czyli ostatecznie

$$(R', G', B') = (255 - R, 255 - G, 255 - B).$$

Czyli otrzymujemy przeciwne kolory (zamiast białego czarny). Algorytm ten sprawdza się dobrze dla większości przypadków, jednak dla takich kolorów jak np. (125, 125, 125) napisy nie będą widoczne. Także jeśli otoczenie wybranego punktu na zdjęciu jest w dwóch “skrajnych” kolorach np. biały i czarny, to tylko część napisów będzie dobrze widoczna.

6.2. Algorytm dostosowywania wielkości zdjęcia

Po dwukrotnym kliknięciu na miniaturkę można zobaczyć zdjęcie w powiększeniu. Jego rozmiar jest dostosowywany do rozmiaru lewej części okna, czyli rozmiaru obiektu m_leftSizer. Jeśli szerokość (rozmiar w poziomie) oryginalnego zdjęcia jest większa od wysokości (rozmiaru w pionie) to nowa szerokość ustawiana jest jako szerokość obiektu m_leftSizer, a nowa wysokość zdjęcia jest wyliczana jako

$$new\_h = size.x \cdot h/w,$$

gdzie new_h - nowa wysokość, size.x - szerokość obiektu m_leftSizer, h - oryginalna wysokość zdjęcia, w - oryginalna szerokość zdjęcia.
Jeśli wysokość oryginalnego zdjęcia jest większa od szerokości to wysokość ustawiamy jako wysokość m_leftSizer, natomiast szerokość

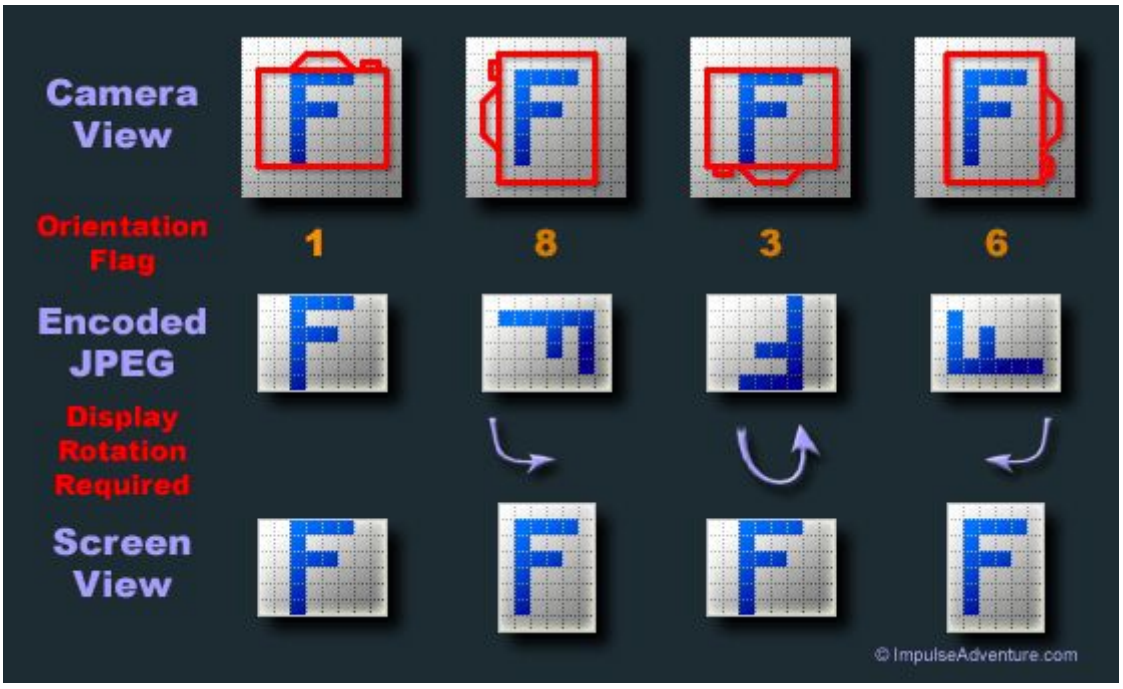
$$new\_w = size.y \cdot w/h,$$

gdzie new_w - nowa szerokość, size.y - wysokość obiektu m_leftSizer, w - oryginalna szerokość zdjęcia, h - oryginalna wysokość zdjęcia.
Dzięki temu algorytmowi proporcje zdjęcia są zachowane.

6.3. Algorytm obrotu miniaturki zdjęcia o odpowiedni kąt

Po załadowaniu zdjęć z katalogu, wczytane bitmapy nie uwzględniały tego czy zdjęcie zostało obrócone w programie do edycji i było wyświetlane w pozycji podstawowej. Aby uwzględnić obrót programie, zaimplementowany został algorytm wywoływany w konstruktorze klasy LoadedImage, który najpierw sprawdzał wartość metadanej exif.Image.Orientation. Jeżeli ta była różna od -1 (brak informacji o rotacji) lub 1 (pozycja standardowa obrotu), zdjęcie było obracane o odpowiedni kąt.
Dla kodu:

- 3: obrót o 180 stopni
- 6: obrót o 270 stopni
- 8: obrót o 90 stopni



Rysunek 4: Tablica reprezentująca powiązanie kodu rotacji i obrotu zdjęcia

7. Kodowanie

Poniżej znajduje się opis wykorzystanych w programie klas, ich metod i zmiennych składowych:

- ➔ **MyApp** - klasa uruchamiająca program:
  - ◆ **MyApp()** - konstruktor domyślny,
  - ◆ **~MyApp()** - destruktor domyślny,
  - ◆ **virtual bool OnInit()** - metoda tworząca główne okno programu klasy CaptionsMyFrame i wywołująca wxInitAllImageHandlers(), aby możliwa była praca z wxImage,
  - ◆ **virtual int OnExit()** - metoda kończąca program i zwracająca 0 ,
  - ◆ **wxFrame* m_frame** - główne okno programu.

➔ **MyFrame** - klasa wygenerowana przez wxFormBuilder - klasa bazowa dla głównego okna programu:
- **wxButton* m_btnChooseDirectory** - przycisk umożliwiający wybranie katalogu, z którego chcemy przeglądać zdjęcia,
- **wxRichTextCtrl* m_tbExifInfo** - panel, na którym wyświetlane są informacje EXIF i IPTC,
- **wxMenuBar* m_menubar2** - menu, na którym są dostępne opcje:
  - **wxMenu* m_menuSave** - zapis tekstu do pliku lub na zdjęcie,
  - **wxMenu* m_menuLoad** - wczytywanie tekstu z pliku,
  - **wxMenu* m_menuInfo** - informacje na temat programu,
- wirtualne metody, które należy nadpisać w klasie pochodnej - zostały opisane w metodach klasy CaptionsMyFrame,
- **MyFrame()** - konstruktor tworzący i łączący wszystkie potrzebne elementy i wydarzenia,
- **~MyFrame()** - destruktor rozłączający wydarzenia.

➔ **CaptionsMyFrame** - główna klasa programu dziedzicząca z MyFrame:
- **void m_btnChooseDirectoryOnButtonClick(wxCommandEvent& event)** - metoda obsługująca kliknięcie przycisku m_btnChooseDirectory - wyświetla się okienko do wyboru folderu, następnie tworzone jest okno m_scrolledWindow, na którym umieszczany jest m_buttonsSizer z miniaturkami zdjęć.
- **void m_itemExportToTxtOnMenuSelection(wxCommandEvent& event)** - event wywoływany po wciśnięciu opcji menu m_itemExportToTxt, pozwala wybrać zdjęcia i ich metadane, a następnie zapisuje te informacje w pliku txt
- **void m_itemExportToImageOnMenuSelection(wxCommandEvent& event)** - event wywoływany po wciśnięciu opcji menu m_itemExportToImage, pozwala wybrać zdjęcia i ich metadane, a następnie zapisuje metadane na wybranych zdjęciach w postaci napisów. Jeśli jedno ze zdjęć jest aktualnie powiększone to wybrane metadane odnoszą się bezpośrednio do niego (etap wyboru zdjęć zostaje pominięty).
- **void m_menuLoadCaptionsOnMenuSelection(wxCommandEvent& event)** - event wywoływany po wciśnięciu opcji menu m_menuLoadCaptions, pozwala wczytać plik tekstowy z napisami, a następnie umieszcza je na określonych w pliku zdjęciach.
- **void m_menuAuthorsOnMenuSelection(wxCommandEvent& event)** - event wywoływany po wciśnięciu opcji menu m_menuAuthors, pokazuje informacje o autorach aplikacji
- **CaptionsMyFrame(wxWindow* parent)** - konstruktor głównej klasy programu, inicjalizuje nazwę głównego okna programu
- **std::vector<std::string> getImages(const wxString& dirPath)** - funkcja zwracająca kontener zawierający nazwy wszystkich plików znajdujących się w katalogu, którego ścieżka przekazywana jest w parametrze wywołania
- **wxSizer* m_leftSizer** - wskaźnik do sizera zawierającego m_btnChooseDirectory,

- **wxGridSizer* m_buttonsSizer** - sizer, do którego dodawane są miniaturki zdjęć,
- **wxScrolledWindow* m_scrolledWindow** - okno, na które umieszczany jest m_buttonsSizer,
- **std::pair <wxString, bool> openSelectWindow(int index)** - do funkcji przekazywany jest indeks jednego z załadowanych zdjęć. Następnie użytkownik wybiera interesujące go metadane ze zdjęcia. Funkcja zwraca wybrane metadane zdjęcia w postaci pary składającej się z obiektu klasy wxString i zmiennej typu bool. Zmienna bool przyjmuje wartość true kiedy użytkownik potwierdza swój wybór klikając OK oraz false gdy wybiera Cancel.
- **void openSaveWindow(int index, const std::stringstream& toSave)** - funkcja zapisująca przekazane napisy “toSave” na zdjęciu o indeksie “index” i zapisuje nowe zdjęcie pod nową nazwą.
- **int isFilename(std::string str) const** - jeżeli przekazany jako argument string jest nazwą jednego ze zdjęć w katalogu, to zwraca indeks tego zdjęcia. W przeciwnym wypadku zwraca -1.
- **std::vector <std::unique_ptr<LoadedImage>> m_loadedImages** - vector zdjęć z wybranego katalogu,
- **std::vector <wxString> m_name** - wektor zawierający nazwy załadowanych zdjęć z wybranego katalogu.

➔ **LoadedImage** - klasa reprezentująca jedno załadowane zdjęcie:

- **LoadedImage(const std::string& path, wxWindow* parent, wxRichTextCtrl* textPanel, wxSizer* sizer, wxGridSizer* m_buttonsSizer, wxScrolledWindow *scrolledWindow)** - konstruktor ustawiający wartości składników klasy w tym m.in. bitmapę m_bmpImage i bitmap button m_btnImage; jeśli obrazek ma parametr EXIF Orientation to jest obracany w odpowiedni sposób; eventy klikania są wiązane z m_btnImage,
- **wxBitmapButton* GetButton() const** - zwraca wskaźnik do m_btnImage,
- **const wxBitmap* GetBitmap() const** - zwraca stały wskaźnik do m_bmpImage,
- **std::vector < std::pair <wxString, wxString> > getInfoArr() const** - funkcja zwracająca kontener zawierający pary składające się z atrybutu metadanych i jego wartości
- **bool isBig() const** - funkcja zwraca true jeżeli dane zdjęcie jest aktualnie powiększone oraz false w przeciwnym wypadku
- **void m_btnLoadedImageOnButtonClick(wxCommandEvent& event)** - wypisuje dane EXIF i IPTC na m_textPanel,
- **void m_btnLoadedImageDoubleClick(wxMouseEvent& event)** - ukrywa m_scrolledWindow z miniaturkami, tworzy m_btnBig w odpowiednim rozmiarze zależnym od rozmiaru m_leftSizer i wiąże event podwójnego kliknięcia z m_btnBig,
- **void m_btnLoadedImageDoubleClickBack(wxMouseEvent& event)** - pokazuje m_scrolledWindow i usuwa m_btnBig,

- **int rotationCode() const** - funkcja zwracająca wartość kodu rotacji określonego przez pole exif.Image.Orientation. Jeżeli zdjęcie nie posiada tego pola, zwracana jest wartość -1.
- **void rotateBitmap(int code)** - obraca zdjęcie o kąt określony przez wartość kodu rotacji przekazanej jako argument funkcji
- **std::string getExifInfo() const** - zwraca dane exif zawarte w zdjęciu
- **std::string getIptcInfo() const** - zwraca dane iptc zawarte w zdjęciu
- **std::string m_path** - ścieżka do danego zdjęcia
- **wxRichTextCtrl* m_textPanel** - wskaźnik do CaptionsMyFrame::m_tbExifInfo - panelu z tekstem,
- **std::unique_ptr<wxBitmap> m_bmpImage** - bitmapa ze zdjęciem,
- **std::unique_ptr<wxBitmapButton> m_btnImage** - bitmap button ze zdjęciem,
- **std::unique_ptr<wxBitmapButton> m_btnBig** - bitmap button z powiększonym zdjęciem,
- **wxSizer* m_leftSizer** - wskaźnik do CaptionsMyFrame::m_leftSizer zawierającego przycisk m_btnChooseDirectory i m_scrolledWindow,,
- **wxGridSizer* m_buttonsSizer** - wskaźnik do CaptionsMyFrame::m_buttonsSizer zawierającego miniaturki,
- **wxWindow* m_parent** - wskaźnik na główne okno programu - m_frame klasy CaptionsMyFrame,
- **wxScrolledWindow* m_scrolledWindow** - wskaźnik do CaptionsMyFrame::m_scrolledWindow.

8. Testowanie

Pojedyncze procedury były testowane w trakcie ich implementacji w sposób manualny w celu uniknięcia powielania błędów w kolejnych etapach projektu. W celu wyświetlania informacji o aktualnie testowanych metodach, wykorzystane zostały obiekty klasy wxMessageBox. Wyświetlanie i interakcja z miniaturkami zdjęć była testowana za pomocą wcześniej przygotowanego, przez każdego z członków zespołu, zbioru zdjęć testowych. Na nich też testowane całościowo były funkcjonalności generowania pliku z wybranymi metadanymi oraz zapisywaniem napisów na zdjęciach.

9. Wdrożenie, raport i wnioski

Zarówno wymagania podstawowe jak i rozszerzone projektu zostały spełnione. Interakcja z użytkownikiem została przetestowana pod większością skrajnych przypadków, tak by uniknąć powstawania niepożądanych błędów. W przyszłości można skupić się nad wydajnością aplikacji i przyśpieszyć ładowanie zdjęć z katalogu.