Narzędzia

*QT* to zestaw bibliotek dedykowanych dla m.in. języka C++ pozwalający na tworzenie zaawansowanych interfejsów użytkownika. *QT* zawiera również elementy pozwalające na obsługę procesów, sieci i grafiki trójwymiarowej a także integrację z bazami danych, posiada także narzędzia pozwalające na przeprowadzenie lokalizacji dla innych wersji językowych programu. Wykorzystane narzędzia zostały szczegółowo omówione w części pracy poświęconej graficznemu interfejsowi użytkownika. Wybór bibliotek *QT* jako narzędzia do tworzenia *GUI* (*graphic user interface*) był niejako oczywisty ze względu na fakt, iż VLC wykorzystuje jako QT jako jeden z podstawowych interfejsów użytkownika w odtwarzaczu podstawowym. Kluczowa była także opinia Jean-Baptiste Kempfa jednego z autorów i moderatorów biblioteki *VLC*, a także prezydenta i administratora forum jej poświęconego. Po przeanalizowaniu innych dostępnych możliwości uznano, iż jest to jedyny dostępny produkt dający możliwość swobodnego tworzenia paneli np. do oceny obejrzanego filmu.

*QT Creator* jedno z narzędzi udostępnianych w ramach zestawu bibliotek *QT*. Jest to kolejne użyte w pracy środowisko programistyczne pozwalające nie tylko na edycje kodu źródłowego, ale także (w narzędziu *QT Designer*) na projektowanie graficznego interfejsu użytkowania w okienkowym edytorze za pomocą widżetów z biblioteki QT. Za pomocą dodatkowych narzędzi *uic* i *moc* każda klasa korzystająca z sygnałów i slotów *QT* (reprezentująca element GUI) otrzymuje dodatkowe pliki \*.cpp i \*.ui reprezentujące rozmieszczenia okien, ustawienia grafiki, standardowe zdarzenia i przypisane do nich metody.

*VLC-QT* darmowa biblioteka autorstwa Tadej Novaka służąca do połączenia bibliotek *QT* z biblioteką *libvlc*. Pozwala ona na stworzenie prostego odtwarzacza wraz z dowolnym interfejsem pozwalającym na kontrolowanie odtwarzania. *VLC-QT* nakrywa metody i klasy upraszczając użycie *libvlc* w oknach *QT*. Ze względu na ograniczenia spowodowane przez użycie interfejsu *imem* do wczytywanie klatek nieskompresowanych sekwencji wideo, użycie *VLC-QT* zostało ograniczone do użycia widżetu wideo dającego większe możliwości niż standardowe *QFrame*.

# Graficzny interfejs użytkownika

## Wstęp

Przeprowadzanie jakichkolwiek testów z udziałem losowych osób wymaga, aby testerzy byli prowadzeni przez cały test niejako za rękę. Wiążę się to z przemyśleniem i zaprojektowaniem interfejsu użytkownika w taki sposób, aby zapewnić możliwie jak największą czytelność i ergonomie procesu testowania. Interfejs użytkownika musi pozwalać na oglądanie filmów i wystawianie ocen, do interfejsu można wprowadzić dodatkowe panele pozwalające na informowanie testera, czy też pobieranie od niego dodatkowych informacji.

## Konfiguracja środowiska

Przed przystąpieniem do pracy z bibliotekami *QT* należy skonfigurować środowisko programistyczne. Ponieważ projekt został stworzony w środowisku *CLion* opartym na *cmake* należało zmodyfikować plik skrypt konfiguracyjny, dodają do linkera ścieżki do uprzednio zainstalowanej biblioteki *QT*. Dla poprawnego działania narzędzi biblioteki, a także do automatycznej generacji koniecznych do kompilacji plików \*.ui oraz ui\_\*.h należy w skrypcie *CMakeLists* uruchomić następujące parametry:



Rysunek Parametry cmake konieczne dla QT

## Podstawowe połączenie vlc z qt

Odtwarzacz wideo *libvlc\_media\_player* posiada domyślny interfejs (w zależności od ustawień może być oparty o QT) jednakże interfejsu tego nie można w żaden sposób modyfikować. Dlatego też do modyfikowania interfejsu odtwarzacza należy stworzyć własny widżet wideo. Aby to uczynić w najbardziej podstawowy sposób należy utworzyć klasę dziedziczącą z *QWidget*, a następnie utworzyć w niej obiekt typu *QFrame*, który następnie należy połączyć z odtwarzaczem za pomocą metody *libvlc\_media\_player\_set\_xwindow()*, która za argumenty przyjmuje utworzoną wcześniej instancje *libvlc\_media\_player*, a także identyfikator okna, który uzyskujemy wołając metodę *winId()* na obiekcie okna. Następnie można uruchamiać i kończyć wideo tak samo jak robiono to dotychczas bez dodatkowego interfejsu, pamiętając o tym, że nie wolno zniszczyć obiektu okna, do którego przypisany jest odtwarzacz wideo. Standardowy obiekt *QFrame* możemy modyfikować nadając mu zależne od nas wymiary, czy też umieszczając go w layoucie w wybranym przez siebie miejscu otaczając dowolnymi obiektami dodając np. paski do sterowania głośnością. Kolejnym krokiem powinno być wywołanie w głównej funkcji *main()* programu utworzenie obiektu klasy *QApplication*, którego konstruktor przyjmuje standardowe argumenty programu. *QApplication* to klasa służąca do zarządzania aplikacją okienkową, a także podstawowymi opcjami takimi jak np. rozmiary okna podstawowego czy ustawienie trybu pełnoekranowego. Ważną rolą jest także możliwość przenoszenia argumentów i sygnałów sterujących do dalszych paneli programu. Następnym krokiem jest powołanie do życia obiektu klasy macierzystej za pomocą jej konstruktora. Standardowy konstruktor klasy dziedziczącej z *QWidget* nie posiada dodatkowych parametrów, jednakże warto go przeciążyć i przekazać mu argumenty programu, aby następnie przekazać je do konstruktora instancji VLC, jeżeli to konieczne. Stworzony widżet należy wyświetlić metodą show(). Ze względu na konieczność otwierania i zamykanie widżetu wideo, a także stworzenia paneli do oceniania i prowadzenia testera zastosowano bardziej zaawansowane podejście.

## Zaproponowany Interfejs

### Wstęp

Ze względu na silny związek interfejsu użytkownika ze scenariuszami testowymi (końcowa wersja powinna korespondować z opracowanymi scenariuszami pozwalając na ich przeprowadzenie) postanowiono zaprojektować interfejsy dedykowane do konkretnych scenariuszy dopiero po ich określeniu. Ponieważ zdecydowano się na standardowe metody *ACR, PC* oraz menu z wyborem filmów postanowiono opracować *GUI* oparte o modułowe panele podmieniane automatycznie w zależności od wybranej konfiguracji testów.

### Stworzony Interfejs

Interfejs użytkownika wykonano za pomocą środowiska *QT Creator* w narzędziu *QT Designer* pozwalającym na wstawianie komponentów GUI w trybie okienkowym. Ułatwiło to edycje i umiejscowienie elementów dokładnie w tych miejscach w których zamierzano. Całość została ponownie zaimportowana do środowiska *CLion* głównie ze względu na wygodę i przyzwyczajenie do tego środowiska, dającego większe możliwości debugowania, zwłaszcza przy zaimportowanej bibliotece *libvlc*.

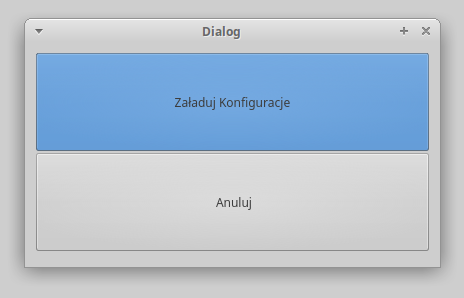
Projektując interfejs kierowano się jego funkcjonalnością. Ze względu na konieczność użytkowania aplikacji zarówno przez administratora testów jak i przez testera należało stworzyć rozdzielne panele testerki i administracyjny. Użytkownik staje przed wyborem czy zamierza konfigurować testy czy też jest testerem i chce je uruchomić.



Rysunek Widok panelu MainWindow

Panel, który widzimy na powyższym zrzucie ekranu jest obiektem typu *MainWindow*. *MainWindow* to klasa stworzona jako główne okno aplikacji, klasa dziedziczy z *QMainWindow*, klasy QT wersji 5 udostępniającej szereg metod to sterowania aplikacją, dziedziczącą z klasyQWid*get*. Ponieważ obiekt *mainWindow* jest tworzony w głównej funkcji programu w konstruktorze dodano parametry wysokości i szerokości okna pobrane uprzednio z ustawień ekranu systemowego. Pobieranie rozdzielczości ekranu odbywa się za pomocą dostarczanych w bibliotece QT klas *QScreen* i *QRect.* Pierwsza z nich to typ wskaźnika, który za pomocą metody *primaryScreen()* z klasy *QApplicatio*n, zostaje ustawiony na obiekt reprezentujący podstawowy ekran w systemie. Następnie za pomocą metody *availableGeometry()* zostają pobrane jego wysokość i szerokość, następnie przekazane do *mainWindow*. Konstruktor *MainWindow* wymusza maksymalizacje okna na ekranie za pomocą metody *showFullScreen(),* a także tworzy przy pomocy słowa kluczowego *new* pusty obiekt typu *PlayerConfigurationsHandler(),* który został omówiony przy okazji obsługi konfiguracji. Na zrzucie ekranu widoczne są dwa przyciski czyli obiekty typu *QPushButton*. Zostały one umieszczone na środku ekranu przy pomocy linii zakotwiczających przyciski w szablonie, co pozwoliło utrzymać je na swoim miejscu dla różnych rozdzielczości.

Przyciski typu *QPushButton* to standardowe przyciski z biblioteki *QT*, poza szerokimi możliwościami związanymi z geometrią, czyli położeniem i wymiarami dostarczają szereg metod pozwalających na interakcje z użytkownikiem. Jedną z tych metod jest metoda *on\_pushButton\_clicked(),* która jest wyzwalana w momencie naciśnięcia na aktywny obszar przycisku. W przypadku *MainWindow* zaimplementowano dwie takie metody (do każdego z przycisków osobną). Pierwsza z nich uruchamia panel administracyjny odbywa się to w następujący sposób. Tworzony jest nowy obiekt typu *AdminPanel,* następnie na ty obiekcie wołane są kolejno metody *show(), activateWindow()* oraz *topLevelWidget(),* pozwala to na wyciągnięcie powstałego panelu administracyjnego na wierzch oraz nadanie mu aktywności. Konstruktor *AdminPanel* przyjmuje jako parametr wskaźnik na wspomniany obiekt typu *PlayerConfigurationsHandler*, celem nadpisania pustego obiektu, obiektem przechowującym aktualnie wczytaną konfigurację.

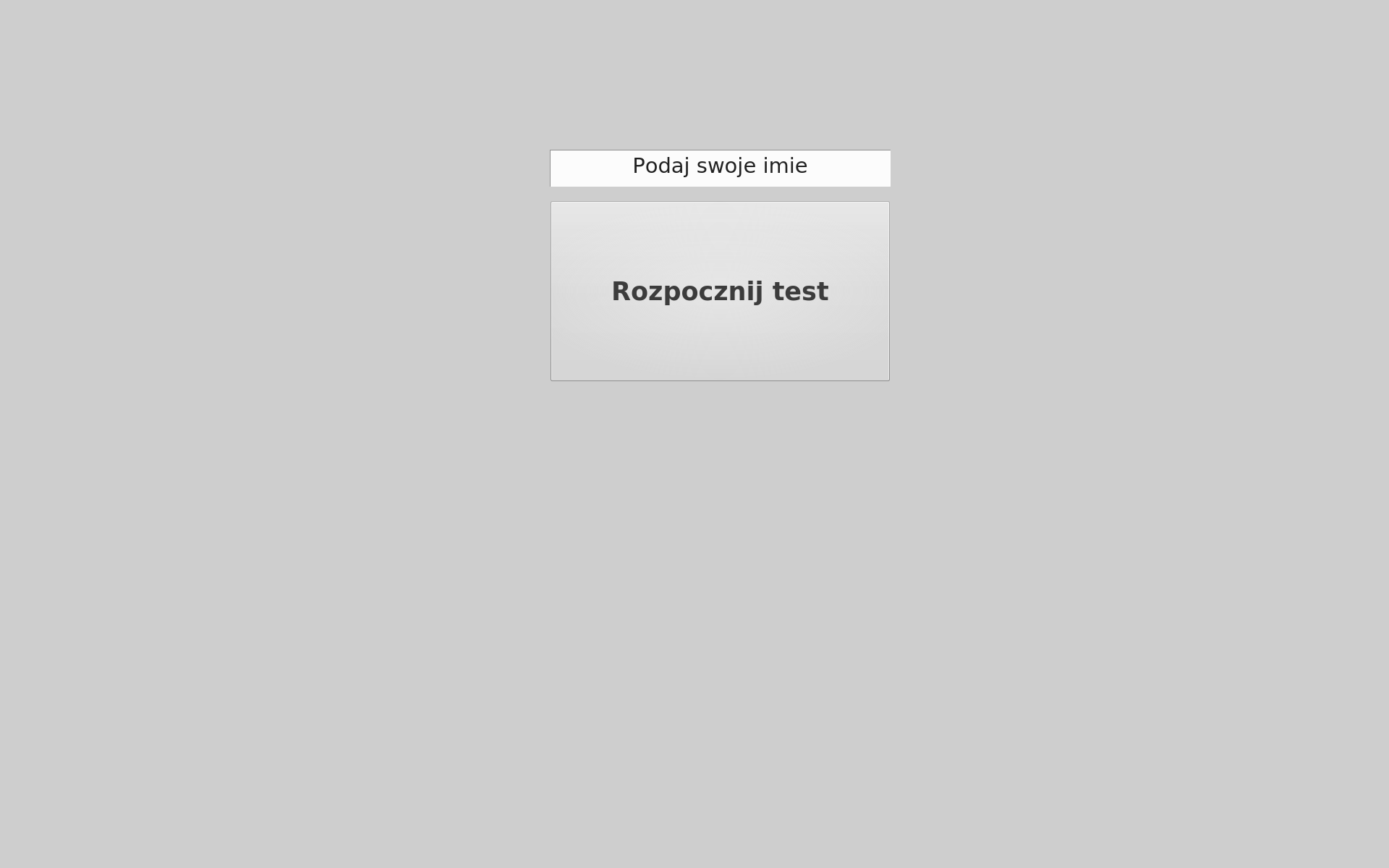


Rysunek Panel administracyjny

Panel administracyjny widoczny na powyższym zrzucie ekranu posiada dwa przyciski z których jeden służy do zamykania go (poprzez zniszczenie obiektu za pomocą *delete*), a drugi uruchamia okno systemowe okno dialogowe za pomocą którego można wybrać plik konfiguracyjny. Następuje to za pomocą metody *getOpenFileName()* która zwraca ścieżkę do wybranego filmu w formacie *QString. QString* ze względu na brak koniczności użycia jest rzutowany do *std::string* metodą *toStdString().* Następnie ścieżka do pliku konfiguracyjnego jest przekazywana jako argument w konstruktorze tworzonego obiektu *PlayerConfigurationsHandler*, konfiguracja zostaje wczytana, przekazany z głównego okna wskaźnik zostaje ustawiony na nowo powstały obiekt, a panel administracyjny zostaje zamknięty.

Drugi z przycisków *MainWindow* jest aktywny tylko w momencie, gdy wczytana jest konfiguracja. Sprawdzanie odbywa się za pomocą wywołania metody *CheckConfiguration()* z obiektu na który wskazuje *playerConfigurationsHandler*. Metoda została opisana w podrozdziale dotyczącym konfiguracji. Naciśnięcie na aktywny przycisk tworzy i uruchamia kolejny panel, ustawiając jego geometrie na taką samą jaka została wpisana w *MainWindow*, skalując go do całości ekranu. Za pomocą szeregu metod okno staje się aktywnym i pełnoekranowym. Nowo powstałe okno jest obiektem typu *UserPanel*.

*UserPanel* przyjmuje w konstruktorze parametry dotyczące wielkości ekranu (przekazywane z *MainWindow*) oraz wskaźnik na obiekt typu *PlayerConfigurationsHandler* przechowujący aktualną konfigurację. Klasa UserPan*el* poza przekazanymi parametrami posiada również kilka istotnych pól. Jednym z nich jest wskaźnik *mTimer* na obiekt typu *QTimer*, który jest używany w konfiguracji numer 3. Klasa *QTimer* dostarcza liczniki pozwalające na odmierzanie chwil czasowych do odświeżania interfejsów bądź uruchamiania zdarzeń w czasie. Wykorzystanie w praktyce zostanie omówione przy konkretnej konfiguracji. Należy jednak nadmienić, że w konstruktorze klasy *UserPanel* zostają wybrane konkretne ustawienia obiektu typu *QTimer*. Typ licznika zostaje ustawiony na *singleShot*, oznacza to, że przypisane zdarzenie jest uruchamiane tylko raz po upłynięciu określonej chwili. W konfiguracji, w której licznik jest wykorzystywany przypisana zostaje metoda *StartPlayback*, omówiona w dalszej części tego podrozdziału. Kolejnymi polami są dwie tablice typu logicznego reprezentujące stany pól wyboru służących do oceny filmów. Ich wykorzystanie zostanie omówione przy opisie systemu oceniania filmów. Klasa *UserPanel* zawiera również numer aktualnie odtwarzanego filmu, oraz pole typu string przechowujące wyniki testów gotowe do zapisania w pliku tekstowym z wynikami. Klasa zawiera także cztery widżety: *startWidget, ratingWidget*, *ratingWidget\_2* oraz *chooseMovieWidget.*



Rysunek Panel startowy testów

Widżet startowy służy podaniu imienia testera mające na celu identyfikacje, czy też odróżnienie od siebie kolejnych osób. Prośba o podanie imienia ma również na celu przywiązanie testera do administratorów, zbudowanie swoistej relacji, daje poczucie większej odpowiedzialności za swoje wyniki niż w przypadku zupełniej aminowości. Po naciśnięciu na przycisk z napisem „Rozpocznij test” wywołana zostaje metoda *on\_pushButton\_clicked()* rozpoczynająca test. Po wywołaniu metody zostaje sczytana zawartość pola typu *TextEdit*, będący miejscem na wpisanie imienia przez użytkownika. Zawartość zostaje wpisana do pola *testsOutputToFileString* klasy *UserPanel* przechowującego zawartość do wpisania w plik wynikowy. Następnie w zależności od numer konfiguracji zostaje wywołana metoda *StartPlayback* uruchamiająca film, bądź zostaje otwarty kolejny widżet. Pozostałe widżety zostaną omówione w dalszej części pracy.

*StartPlayback()* to metoda odpowiedzialna za główną funkcjonalność programu czyli odtwarzanie nieskompresowanych sekwencji wideo. Filmy są uruchamiane według wartości pola *iActualPlayedMovie* które jest iterowane z każdym wywołaniem metody lub w przypadku konfiguracji oznaczonej numerem 2 ustawiane poprzez wybranie odpowiedniej pozycji na liście. W metodzie tworzone są obiekty typu *RawDataHandler, FramesHandler* oraz *ThreadsHandler*, oraz *OptionsHandler*. Funkcjonalność poszczególnych obiektów została opisana w podrozdziale dotyczącym wstrzykiwania klatek wideo. Należy zwrócić uwagę na ustawienia opcji instancji *LibVLC*. Opcje zostają ustawione tak samo jak w przypadku odtwarzania wideo bez własnego interfejsu jednak parametry takie jak rozdzielczość czy ilość klatek na sekundę zostają wczytane z obiektu typu *PlayerConfigurationsHandler* do którego *UserPanel* posiada referencje. W metodzie tworzony jest również obiekt typu *Controler*, którego zastosowanie również zostało omówione w poprzednich podrozdziałach. Na obiekt ten zostaje ustawiony wskaźnik współdzielony klasy *shared\_ptr*. Jest to konieczne, aby zachować dostęp do obiektu dla innego wątku. Kolejnym krokiem jest stworzenie kolejnego panelu tym razem typu *VideoPanel,* który zostanie opisany w kolejnym akapicie. Metoda *StartPlayback()* posiada również instrukcje warunkowe wynikające z różnych konfiguracji zgodne z przepływem związanego z nią scenariusza testowego. Wywołana zostaje również metoda *StopPlayBackThread()* na obiekcie *threadsHandler*. Jest ona odpowiedzialna za zniszczenie okna *VideoPanel* w momencie zakończenia odtwarzania filmu co zostało już wspomniane we wcześniejszych podrozdziałach.

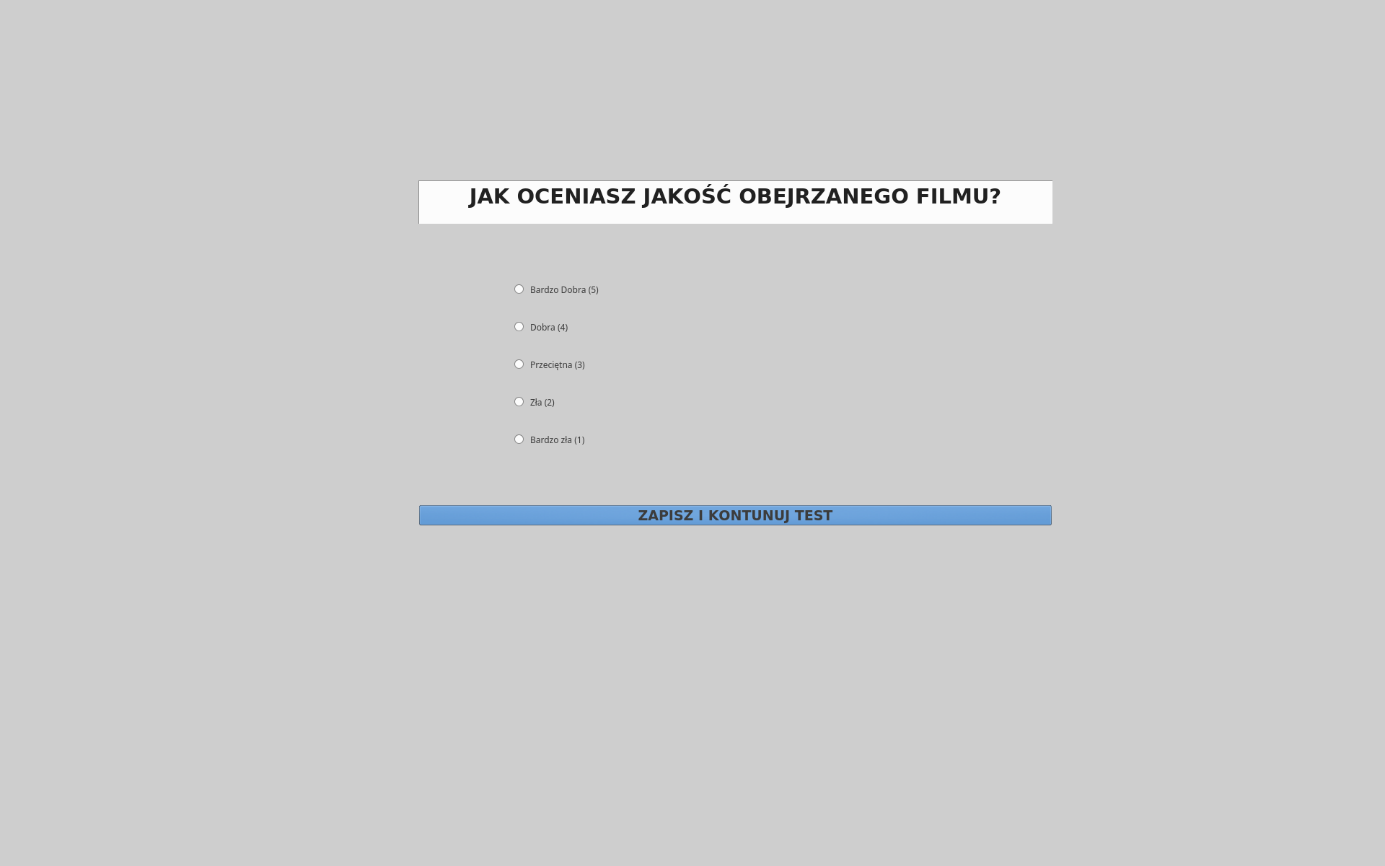
*VideoPanel* to klasa odpowiedzialna za powiązanie *libvlc\_media\_player* z oknem odtwarzacza, odbywa się to w konstruktorze tej klasy zgodnie z opisem przedstawionym w przypadku podstawowego połączenia odtwarzacza z *QFrame*, jednakże zamiast *QFrame* użyto obiektu klasy *VlcWidgetVideo*, aby uzależnić go od opcji instancji *libvlc\_instance*. Konstruktor wraz z metodą przypisania odtwarzacza do widżetu został pokazany na poniższym rysunku.



Rysunek Konstruktor panelu VideoPanel z przypisaniem odtwarzacza

Jak zostało wspomniane we wcześniejszym fragmencie tego podrozdziału w zależności od wybranej konfiguracji uruchamiane są różne widżety do oceniania są nimi *ratingPanel* i *ratingPanel2*. Zostały one stworzone jako osobne widoki, bez tworzenia klas ze względu na brak konieczności posiadania osobnej logiki. Oba widżety posiadają niemal identyczną konstrukcję. W górnej części znajduję się pasek z pytaniem zadawanym do testera, a w środkowej znajdują się przyciski opcji (typu *QRadioButton*) przyciski zostały umieszone w jednym obszarze interfejsu co pozwala na stworzenie grupy. Grupa przycisków automatycznie łączy je nadając im parametr ekskluzywności, oznacza to że tylko jeden z nich może być zaznaczony w danej chwili, a zaznaczenie innego powoduje wyłączenie poprzedni zaznaczonego. Parametr ten można zmieniać przy pomocy metody *setAutoExclusive(),* która przyjmuje argument typu logicznego oznaczający stan docelowy parametru. Metoda ta została użyta w metodzie *UncheckToggles()* w klasie *UserPanel*, która została omówiona poniżej. Każdy z przycisków opcji posiada własną metodę *on\_buttonNumber\_toggled(),* która zostaje wywołana przy każdorazowej zmianie stanu przycisku. Wywołanie metody ustawia element tablicy pola *states* (lub *states2* w scenariuszu numer 3) o numerze przycisku w zależności od stanu. W dolnej części widżetów do oceniania znajdują się przyciski służące do kontynuacji przepływu testu. Przycisk kontynuacji testu posiada różne zachowania zależne od etapu testu i konfiguracji. Jego zachowanie w konkretnych sytuacjach zostanie omówione w dalszej części rozdziału. Istotną w każdym scenariuszu rolą jest jednak zbieranie po kliknięciu informacji o wybranej przez użytkownika ocenie filmu. Odbywa się to przy pomocy pętli która przegląda odpowiednią tablice *states* w poszukiwaniu elementu ustawionego na wartość *„true*”. Numer zaznaczonego przycisku opcji oznacza wystawioną przez testera ocenę. Wystawione oceny zostają rzutowane na stringa i zapisane do pliku w metodzie *WriteToFile().*

*UncheckToggles()* to metoda pomocnicza służąca odświeżaniu przycisków typu *QRadioButton* na widżetach do oceny. Funkcjonalność tej metody jest standardową funkcjonalnością wyłączającą wszystkie przyciski opcji. Wyłącznie aktywnego przycisku opcji nie ogranicza się jednak do ustawienia stanu *QRadioButton* na nieaktywny, ponieważ w przypadku ustawionej opcji ekskluzywności odznaczenia jednego z przycisków jest możliwe tylko w przypadku zaznaczenia innego, dlatego też w metodzie *UncheckToggles()* konieczne jest wyłączenie automatycznej ekskluzywności wszystkich przycisków w grupie. Jest to możliwe przy pomocy wspomnianej już metody *setAutoExclusive()* z klasy *QRadioButton*. Następnie na każdym z przycisków opcji należy wywołać metodę *setChecked()* z parametrem „*false*” przełączając wszystkie w stan wyłączony. Na koniec pozostaje ponowne włączenie automatycznej ekskluzywności.



Rysunek Widok oceny filmu

Widoczny na powyższym zrzucie ekranu panel to panel typu *UserPanel* z włączonym widżetem *ratingWidget*. Na widżecie widzimy pięć zgrupowanych przycisków opcji – przyciski posiadają opisy tekstowe będące reprezentacją wystawianych ocen, oraz wspomniane przycisk kontynuacji testu. Widżet ten jest widoczny po zakończeniu się każdego z odtwarzanych filmów w scenariuszach oznaczonych w konfiguracji jako 1 i 2 (czyli *ACR* i opcji wyboru filmu z listy).

W przypadku konfiguracji pierwszej po naciśnięciu przycisku oznaczonego jako *„ZAPISZ I KONTYNUUJ TEST*” następuje uruchomienie kolejnego filmu czyli wywołanie wcześniej opisanej metody *StartPlayback()* oraz wyczyszczenie stanów przycisków opcji za pomocą metody *UncheckToggles().* Kolejne filmy są wybierane są według kolejności z konfiguracji. Po zakończeniu filmu użytkownik ponownie zostaje postawiony przed widżetem *ratingWidget* aby ocenić jakość kolejnego nagrania, aż do momentu gdy zostaną odtworzone wszystkie filmy z konfiguracji. Po odtworzeniu ostatniej z wyznaczonych sekwencji wideo napis na przycisku zostaje zmieniony na *„ZAKOŃCZ TEST”.* Zmieniając tym samym swoją funkcjonalność. Po kliknięciu wywołana zostaje metoda *WriteToFile(),* a okno zostaje zniszczone pozwalając na powrót do głównego panelu.

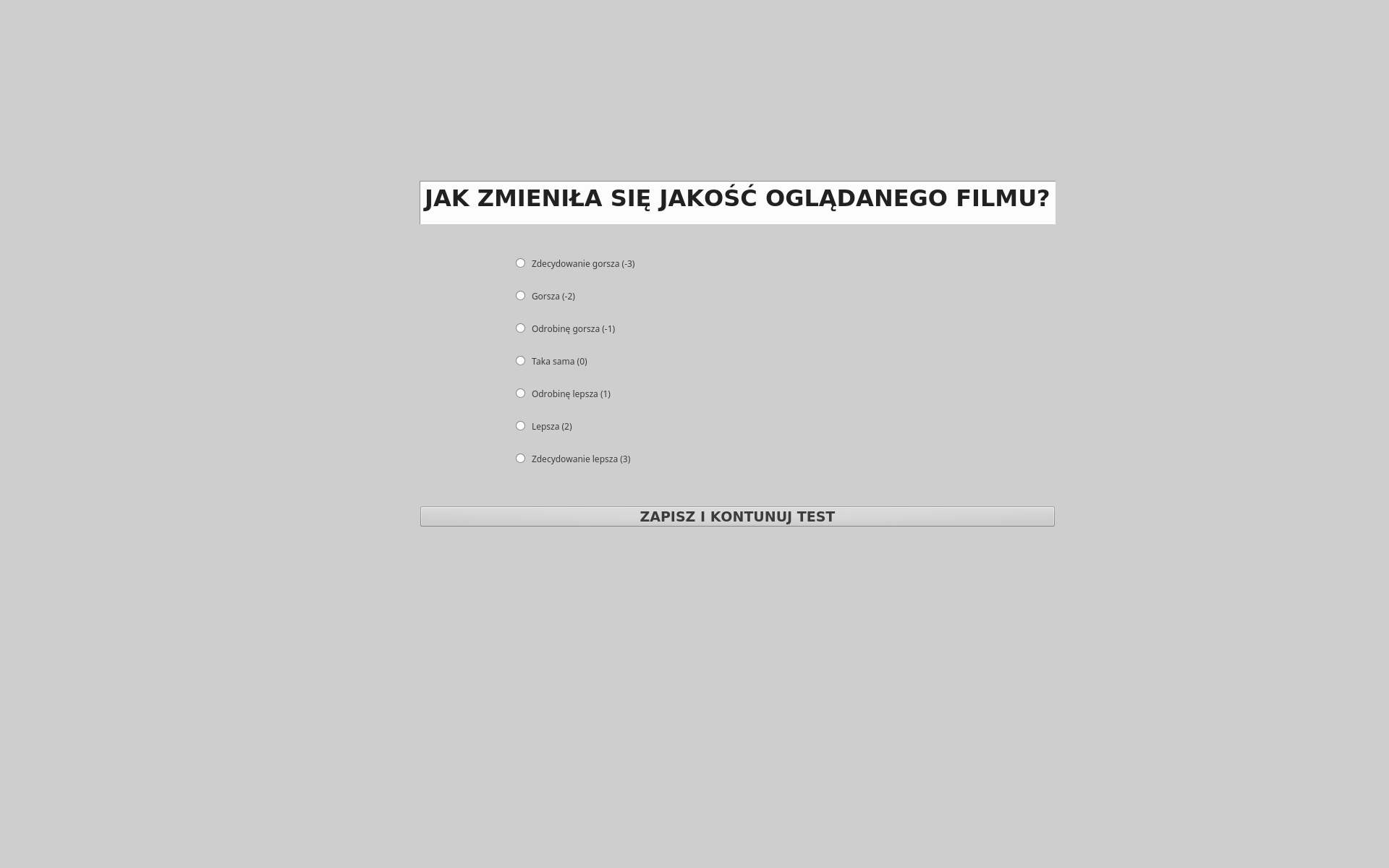
Dla konfiguracji oznaczonej numerem dwa przebieg testu jest zupełnie inny co wymusza inne działanie interfejsu użytkownika. W tej konfiguracji tester rozpoczyna test od menu pozwalającego mu na wybór filmu z listy. Odbywa się to za pomocą widżetu *chooseVideoWidget* przedstawionego na poniższym zrzucie ekranu.



Rysunek Menu wyboru filmów

Widżet składa się z napisu, rozwijalnej listy (pole wyboru) a także dwóch przycisków oznaczonych w sposób widoczny na powyższym zrzucie ekranu. Przycisk zakończenia testu posiada funkcjonalność zbliżoną do funkcjonalności zakończenia testu o pierwszym numerze konfiguracji różnica polega na sposobie zapisu ocen wystawionych przez użytkownika. Oceny są przypisywane do filmów poprzez klasę *MovieProperties* i jej pole *rate*, które zostały opisane w części dotyczącej konfiguracji. Następnie z wektora obiektów typu *MovieProperties* wartości są odczytywane w pętli i zapisywane do pliku tekstowego, a samo okno podobnie jak w poprzednio jest niszczone. Drugi z przycisków znajdujących się na widżecie służy do uruchamiania wybranego z listy filmu poprzez metodę *StartPlayback().* Wybór filmu odbywa się poprzez wybranie pozycji z pola wyboru. Pole wyboru jest reprezentowane przez obiekt *QComboBox,* który udostępnia między innymi metodę *on\_activated* wywoływaną przy każdym rozwinięciu pola. Metoda ta ustawia parametr index na wartość wybraną z listy. Parametr ten zostaje użyty do ustawienia pola *iActualPlayedMovie* reprezentującego aktualnie wybranego filmu. Wywołując w następnym kroku metodę *StartPlayback()* uruchamiany zostaje film o tym numerze. Pozycje w polu wyboru reprezentują kolejne filmy identyfikując je po nazwach. Obok nazwy w polu znajduję się aktualnie wybrana ocena nadana przez użytkownika danej sekwencji, jeżeli żadna ocena nie została wybrana w polu widoczne jest zero. Użytkownik zgodnie z koncepcją scenariusza może według uznania obejrzeć każdy filmu kilkukrotnie zmieniając bądź podtrzymując ocenę.

Trzecia opcja konfiguracyjna pozwala na realizacje scenariusza opartego o porównanie dwóch sekwencji np. *DCR* czy *PC.* Zrealizowano scenariusz *PC*, dla którego ze względu na konieczność zmiany skali oceniania należało dodać kolejny widżet. Było to wygodniejsze niż przebudowywanie widżetu *ratingWidget* z kodu w trakcie działania aplikacji. Stworzono widżet *ratingWidget2*.



Rysunek Widok oceny porównawczej pary filmów

Widżet *ratingWidget2* widoczny na powyższym zrzucie ekranu został zaprojektowany w identyczny sposób co widżet *ratingWidget*. Najważniejsza zmiana w widocznej części interfejsu to zmiana ilości przycisków opcji oraz tekstów identyfikujących możliwe do nadania oceny. Ponieważ test ma charakter porównawczy testerowi zadano inne pytanie w polu tekstowym. Ze względu na fakt, iż w skali występują oceny ujemne wystąpiła konieczność przesunięcia wartości ocen, które dotychczas były równe numerom przycisków w dół, aby uzyskać konieczne wartości. Istotne zmiany w tej konfiguracji następują w metodzie *StartPlayback(),* a także w konstruktorze całego panelu *UserPanel* co zostało już wspomniane. Konfiguracja trzecia wymusza konieczność uruchamiania drugiego filmu od razu po pierwszym. Ponieważ zmiana odtwarzanego filmu wymusza zmianę instancji *libvlc\_media* co z kolei zmusza do zmiany instancji *libvlc\_media\_player* uznano, iż łatwiejszym rozwiązaniem będzie zniszczenie starego panelu *VideoPanel* i utworzenie kolejnego, dlatego też funkcja *StartPlayback* zostaje wywołana ponownie po czasie odtwarzania poprzedniej sekwencji poprzez zdarzenie oparte na liczniku czasu typu *QTimer*. Dopiero po dwóch sekwencjach widoczny jest widżet pozwalający na ocenę. Ocena jest przekazywana do zapisu do pliku w ten sam sposób co w przypadku konfiguracji pierwszej.

Stworzony w ten sposób interfejs pozwala na przeprowadzanie testów w trzech wybranych konfiguracjach. Interfejs można dodatkowo rozszerzyć o lokalizację za pomocą udostępnianego w ramach biblioteki *QT* narzędzia *QT Linguist* pozwalającego na tłumaczenie. Narzędzie to można również wykorzystać do podmiany pół tekstowych co pomogłoby skonstruować inne scenariusze testowe. Całość interfejsu zostało utrzymane w szarych stonowanych kolorach niepowodujących rozproszenia uwagi testera, jednakże korzystano z podstawowych, szablonowych grafik, dlatego też istotnym krokiem w rozwoju aplikacji byłaby wymiana grafiki na dedykowane.