UNIVERSIDAD DE CÓRDOBA

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

SISTEMA INFORMÁTICO PARA EL SEGUIMIENTO DE PUNTERO LÁSER Y SU REPRESENTACIÓN VIRTUAL MEDIANTE VISIÓN ARTIFICIAL

MANUAL DE CÓDIGO

Autor: *Rafael Ulises Baena Herruzo*
Directores: *Juan María Palomo Romero*
Lorenzo Salas Morera

Córdoba, septiembre de 2017

Índice general

Capítulo 1. Introducción

El presente documento tiene como finalidad ofrecer un manual de código de la aplicación "Sistema Informático para el Seguimiento de Puntero Láser y su Representación Virtual mediante Visión Artificial".

La aplicación consta de dos tipos de desarrollos que se han implementado con diferentes tecnologías, las cuales se pasan a describir a continuación.

1.1 Back-end

Es la implementación que se encarga de la parte lógica del programa, en este caso es donde se encuentran los diferentes algoritmos y las estructuras de datos necesarias para poder atender las peticiones que el usuario realice a través de la interfaz de usuario de la aplicación, *front-end.* Este va a estar desarrollado en el lenguaje de programación *C++* y se va a hacer uso de la biblioteca libre de visión artificial *OpenCV*, y del conversor de audio y vídeo *avconv.* El desarrollo íntegro del *back-end,* ha sido realizado con el editor de código *Atom.*

1.2 Front-end

Es la implementación que se encarga de la parte de la interfaz gráfica de la aplicación, y la cual hace de intermediario entre el usuario y el *Back-end.* Va a ser desarrollada en el lenguaje de programación *C++,* haciendo uso del

framework multiplataforma *Qt*. El desarrollo íntegro del *front-end*, ha sido realizado con el IDE *Qt Creator* (*Community*).

Capítulo 2. Instrucciones de compilación

Para realizar la compilación del código fuente, se va a hacer uso del IDE de desarrollo *Qt Creator*.

Para la compilación, se va a utilizar un archivo de configuración que genera *Qt Creator* de manera automática cuando se insertan los archivos correspondientes al código fuente. A el cual, se le han añadido las distintas librerías que van a ser requeridas en el proceso de compilación. A continuación se muestra el contenido de dicho fichero.

2.1 Fichero de configuración

Nombre del fichero: *LaserPress.pro*

```
#-------------------------------------------------
#
# Project created by QtCreator 2017-06-23T14:07:22
#
#-------------------------------------------------

QT       += core gui

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

TARGET = LaserPress
TEMPLATE = app

# The following define makes your compiler emit warnings if you use
# any feature of Qt which as been marked as deprecated (the exact
warnings
# depend on your compiler). Please consult the documentation of the
# deprecated API in order to know how to port your code away from it.
```

```
DEFINES += QT_DEPRECATED_WARNINGS

# You can also make your code fail to compile if you use deprecated
APIs.
# In order to do so, uncomment the following line.
# You can also select to disable deprecated APIs only up to a certain
version of Qt.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all
the APIs deprecated before Qt 6.0.0

INCLUDEPATH += /usr/local/include/opencv
LIBS += -L/usr/local/lib `pkg-config opencv --cflags --libs` -lX11
-lrt

SOURCES += \
    main.cpp \
    ventanaprincipal.cpp \
    grabacion.cpp \
    tracker.cpp

HEADERS += \
    ventanaprincipal.h \
    grabacion.hpp \
    tiempo.hpp \
    tracker.hpp \
    logicclass.hpp

FORMS += \
    ventanaprincipal.ui \

RESOURCES += \
    imagenes.qrc
```

Una vez se han introducido las librerías necesarias y ficheros de código fuente, tal y como indica el fichero de configuración *LaserPress.pro*, la compilación se realiza yendo a la pestaña "Build" del programa *Qt Creator* y realizando clic derecho sobre la opción "Build All" o pulsando la combinación de teclas *Ctrl+Shift+B*.

Capítulo 3. Ficheros del proyecto

El código está organizado en un conjunto de ficheros que corresponden con el diseño procedimental realizado en el M*anual Técnico* del programa. Esto quiere decir, que cada clase tiene su fichero propio correspondiente.

El código fuente de la aplicación se encuentra comentado con las pertinentes aclaraciones de las distintas operaciones y funciones que se realizan.

A continuación se van a mostrar los distintos ficheros que componen el código del programa, junto a una breve descripción de su contenido.

3.1 ventanaprincipal.h

Este archivo contiene la clase encargada de crear la ventana principal, se puede considerar como la parte principal del sistema, implementando una gran cantidad de métodos.

```
/*
 * Proyecto: Sistema Informático para el Seguimiento de Puntero Láser y su
Representación Virtual mediante Visión Artificial
 * Fichero: ventanaprincipal.h
 * Descripción: fichero que contiene la clase que define la interfaz de usuario
de la aplicación.
 * Autor: Rafael Ulises Baena Herruzo (i32baher@uco.es)
 */

#ifndef VENTANAPRINCIPAL_H
#define VENTANAPRINCIPAL_H

#include <QMainWindow>
#include <stdio.h>
#include <iostream>
```

```
#include <unistd.h>

#include "tracker.hpp"
#include "opencv2/highgui.hpp"
#include "opencv2/imgproc.hpp"
#include "grabacion.hpp"
#include "logicclass.hpp"

using namespace std;

namespace Ui {
class VentanaPrincipal;
}

class VentanaPrincipal : public QMainWindow
{
    Q_OBJECT

public:
    explicit VentanaPrincipal(QWidget *parent = 0);
    ~VentanaPrincipal();

    /*
     * OBSERVADORES Y MODIFICADORES DE LA CLASE
     */

    lp::Tracker getTrackModel()const{return _track_model;}

    void setTrackModel(const lp::Tracker &track_model){_track_model =
track_model;}

    lp::Grabacion getGrabacionModel()const{return _grabacion_model;}
    void setGrabacionModel(const lp::Grabacion &grabacion_model){_grabacion_model
= grabacion_model;}

    int getCamaraID()const{return _camara_id;}
    void setCamaraID(const int &camara_id){_camara_id = camara_id;}

    int getSonidoID()const{return _sonido_id;}
    void setSonidoID(const int &sonido_id){_sonido_id = sonido_id;}

    string getRutaArchivo()const{return _ruta_archivo;}
    void setRutaArchivo(const string &ruta_archivo){_ruta_archivo =
ruta_archivo;}

    string getDirSalida()const{return _dir_salida;}
    void setDirSalida(const string &dir_salida){_dir_salida = dir_salida;}

    string getArchivoSalida()const{return _archivo_salida;}
    void setArchivoSalida(const string &archivo_salida){_archivo_salida =
archivo_salida;}

    // Manejo de las rutas para guardar/cargar los archivos en un montaje offline
    string getDirMO()const{return _dir_mo;}
    void setDirMO(const string &dir_mo){_dir_mo = dir_mo;}
```

```
    string getArchMediaMO()const{return _arch_media_mo;}
    void setArchMediaMO(const string &arch_media_mo){_arch_media_mo =
arch_media_mo;}

    string getArchCapMO()const{return _arch_cap_mo;}
    void setArchCapMO(const string &arch_cap_mo){_arch_cap_mo = arch_cap_mo;}

    LogicClass &getLogica(){return _logica;}

    int getFormatoPantalla()const{return _formato_pantalla;}
    void setFormatoPantalla(const int &formato_pantalla){_formato_pantalla =
formato_pantalla;}

    int getExtAncho()const{return _ext_ancho;}
    void setExtAncho(const int &ext_ancho){_ext_ancho = ext_ancho;}

    int getExtAlto()const{return _ext_alto;}
    void setExtAlto(const int &ext_alto){_ext_alto = ext_alto;}

    /*
     * FIN OBSERVADORES Y MODIFICADORES DE LA CLASE
     */

    /* Nombre: tracking
     * Objetivo: método que realiza la captura del puntero láser
     *  y la grabación del audio y vídeo correspondiente a las
     *  diapositivas mostradas por el usuario en la presentación.
     *  Para ello realizará una llamada a los métodos de la clase
     *  Tracker y Grabación.
     * Parámetros de entrada: ninguno
     * Parámetros de salida:
     *                      - valor lógico que inidica si se ha realizado la
ejecución correctamente
     */
    bool tracking();

    /* Nombre: procesar
     * Objetivo: método que realiza el montaje de la presentación,
     *  representará las coordenadas del puntero láser, en la
     *  grabación realizada de las diapositivas y unirá el audio
     *  y vídeo resultantes. Para ello realizará una llamada a
     *  los métodos de la clase Tracker y Grabación.
     * Parámetros de entrada: ninguno
     * Parámetros de salida:
     *                      - valor lógico que inidica si se ha realizado la
ejecución correctamente
     */
    bool procesar();

private slots:
    /*
     * Métodos encargados de dar funcionalidad a los distintos componentes de la
interfaz gráfica
     */

    void on_camaraID_valueChanged(int arg1);
```

```
    void on_camaraID_editingFinished();

    void on_selecArchivo_clicked(bool checked);

    void on_selecCamara_clicked(bool checked);

    void on_sonidoID_editingFinished();

    void on_sonidoID_valueChanged(int arg1);

    void on_lineEdit_editingFinished();

    void on_boton_previsualizar_clicked();

    void on_formato_automatico_clicked(bool checked);

    void on_formato_4_3_clicked(bool checked);

    void on_formato_16_9_clicked(bool checked);

    void on_ext_ancho_editingFinished();

    void on_ext_ancho_valueChanged(int arg1);

    void on_ext_alto_editingFinished();

    void on_ext_alto_valueChanged(int arg1);

    void on_boton_mo_rutadir_clicked();

    void on_lineEdit_mo_archivo_editingFinished();

    void on_lineEdit_mo_captura_editingFinished();

    void on_pushButton_mo_rAudio_clicked();

    void on_lineEdit_mo_archivo2_editingFinished();

    void on_pushButton_mo_rCap_clicked();

    void on_boton_iniciarCaptura_clicked();

    void on_boton_buscarCaptura_clicked();

    void on_boton_buscarDirMontaje_clicked();

    void on_boton_iniciarMontaje_clicked();

private:
    /*
     * Variables miembro
     */
```

```
    Ui::VentanaPrincipal *ui;
    lp::Tracker _track_model;
    lp::Grabacion _grabacion_model;
    int _camara_id;
    int _sonido_id;
    string _ruta_archivo;
    string _dir_salida;
    string _dir_mo;
    string _archivo_salida;
    string _arch_media_mo;
    string _arch_cap_mo;
    LogicClass _logica;
    int _formato_pantalla; // 0 = 4:3 / 1 = 16:9 / -1 = Automático
    int _ext_ancho;
    int _ext_alto;
};

#endif // VENTANAPRINCIPAL_H
```

3.2 vetanaprincipal.cpp

Fichero que contiene los métodos que dan funcionalidad a la clase VentanaPrincipal.

```
/*
 * Proyecto: Sistema Informático para el Seguimiento de Puntero Láser y su
Representación Virtual mediante Visión Artificial
 * Fichero: ventanaprincipal.cpp
 * Descripción: fichero que contiene los métodos que dan funcionalidad a la clase
VentanaPrincipal.
 * Autor: Rafael Ulises Baena Herruzo (i32baher@uco.es)
 */

#include "ventanaprincipal.h"
#include "ui_ventanaprincipal.h"
#include "opencv2/opencv.hpp"

#include <QFileDialog>
#include <QMessageBox>

VentanaPrincipal::VentanaPrincipal(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::VentanaPrincipal)
{
    ui->setupUi(this);

    LogicClass &logica = getLogica();

    ui->progreso_montaje->setMaximum( logica.max() );
    ui->progreso_montaje->setMinimum( logica.min() );
    connect( &logica, SIGNAL( signalProgress(int) ), ui->progreso_montaje, SLOT(
setValue(int) ) );
    connect( &logica, SIGNAL( signalFrame(QString) ), ui->frame_actual, SLOT(
setText(QString) ) );
```

```
    connect( &logica, SIGNAL( signalNframes(QString) ), ui->nFrames, SLOT(
setText(QString) ) );

    //setWindowFlags(Qt::Window | Qt::WindowMinimizeButtonHint |
Qt::WindowCloseButtonHint);

    ui->selecCamara->setChecked(true);

    setRutaArchivo("");
    setCamaraID(0);
    setSonidoID(0);
    setDirSalida("/tmp/");
    setArchivoSalida("salida.mp4");

    // Inicializa las rutas para guardar las capturas en un montaje offline
    setDirMO("");
    setArchMediaMO("");
    setArchCapMO("");

    //ui->progreso_montaje->setFormat(fromStdString(""));
    ui->progreso_montaje->setValue(0);

    // Se ajusta la detección automática de la relación de aspecto de la
proyección
    ui->formato_automatico->setChecked(true);
    setFormatoPantalla(-1);

    // Inicializa las variables que guardan el ancho y alto de la pantalla
principal
    // cuando se quiere extender la pantalla del ordenador
    setExtAlto(0);
    setExtAncho(0);

    string lista_dis_audio;
    ifstream f("/proc/asound/cards");
    while(!f.eof()) {
      localizacion l;
      char cadena[128];
      f.getline(cadena, 128);
      lista_dis_audio += cadena;
      lista_dis_audio += "\n";
    }
    ui->disp_sonido->setText(QString::fromStdString(lista_dis_audio));
}

VentanaPrincipal::~VentanaPrincipal()
{
    delete ui;
}

bool VentanaPrincipal::tracking(){
    // Si se ha seleccionado la cámara principal
    lp::Tracker track_model = getTrackModel();
    lp::Grabacion grabacion_model(getSonidoID());

    bool salida;
```

```
    if(ui->selecCamara->isChecked())
        salida = track_model.inicializar(getCamaraID());
    else
        salida = track_model.inicializar(getRutaArchivo());

    if(!salida){
        this->setVisible(true);
        QMessageBox::warning(this,"Error","No se pudo abrir la cámara o el
archivo de vídeo (1)");
        return false;
    }

    if(!track_model.estaAbierta()){
        this->setVisible(true);
        QMessageBox::warning(this,"Error","No se pudo abrir la cámara o el
archivo de vídeo (2)");
        return false;
    }

    namedWindow("Capturando presentación (Pulsa ESC para regresar al menú
principal)", WINDOW_NORMAL);

    // Si se ha elegido un montaje offline guarda los archivos de audio y vídeo
en un fichero
    if (getDirMO() != "" && getArchMediaMO() != "" && getArchCapMO() != "")
        grabacion_model.setRuta(getDirMO() + getArchMediaMO());

    int salida_iniciar = grabacion_model.iniciar(Point(getExtAncho(),
getExtAlto()));
    if(salida_iniciar != 0){
        this->setVisible(true);
        switch(salida_iniciar){
            case 1:
                QMessageBox::warning(this,"Error","No se ha podido iniciar la
grabación, la grabación ya se ha iniciado.");
            break;
            case 2:
                QMessageBox::warning(this,"Error","No se ha podido iniciar la
grabación, error al realizar el fork().");
            break;
            case 3:
                QMessageBox::warning(this,"Error","No se ha podido iniciar la
grabación, el archivo de salida ya existe.");
            break;
        }
        return false;
    }

    while(true){
      if(!track_model.capturar())
        break;

      track_model.extraerFondo();

      track_model.operacionesMorfologicas();

      track_model.calcularPosicion();
```

```
      Mat imagen = track_model.dibujarCirculo();

      imshow("Capturando presentación (Pulsa ESC para regresar al menú
principal)", imagen);

      char k = (char)waitKey(30);
      if( k == 27 ) break;
    }

    track_model.release();

    // Si se ha elegido un montaje offline guarda las coordenas del puntero en un
fichero
    if (getDirMO() != "" && getArchMediaMO() != "" && getArchCapMO() != ""){
        if(!track_model.guardarArchivoLoc(getDirMO() + getArchCapMO())){
            QMessageBox::warning(this,"Error","No se ha podido guardar el archivo
\".cap\".");
            return false;
        }
    }

    int salida_parar = grabacion_model.parar();
    if(salida_parar != 0){
        this->setVisible(true);
        switch(salida_parar){
            case 1:
                QMessageBox::warning(this,"Error","La grabación no se ha podido
iniciar.");
            break;
            case 2:
                QMessageBox::warning(this,"Error","No se ha podido parar el
proceso de grabación.");
            break;
            case 3:
                QMessageBox::warning(this,"Error","No se ha grabado bien el audio
o el vídeo.");
            break;
        }
        return false;
    }
    setGrabacionModel(grabacion_model);
    setTrackModel(track_model);
    destroyAllWindows();
    this->setVisible(true);
    QMessageBox::information(this,"Información","¡La captura se ha realizado con
éxito!");

    return true;
}

bool VentanaPrincipal::procesar(){
    lp::Tracker track_model = getTrackModel();
    lp::Grabacion grabacion_model = getGrabacionModel();

    // Si se ha elegido un montaje offline carga los archivos de audio y vídeo
desde un fichero
    if (getDirMO() != "" && getArchMediaMO() != "" && getArchCapMO() != ""){
```

```
        grabacion_model.setRuta(getDirMO() + getArchMediaMO());
        if(!track_model.cargarArchivoLoc(getArchCapMO())){
            QMessageBox::warning(this,"Error","No se ha podido abrir el archivo
\".cap\".");
            return false;
        }
    }

    grabacion_model.setFormatoPantalla(getFormatoPantalla());
    string ruta_salida = getDirSalida() + getArchivoSalida();
    int salida = grabacion_model.procesar(track_model.getLocalizacion(),
track_model.getResolucion(), ruta_salida, getLogica());

    if(salida != 0){
        switch(salida){
            case 1:
                QMessageBox::warning(this,"Error","No se ha podido abrir el
archivo de vídeo.");
            break;
            case 2:
                QMessageBox::warning(this,"Error","No se ha podido crear el
archivo de salida de video.");
            break;
            case 3:
                QMessageBox::warning(this,"Error","No existe el archivo de audio
o vídeo, o ya existe la ruta de salida.");
            break;
            case 4:
                QMessageBox::warning(this,"Error","Fallo en el fork().");
            break;
            case 5:
                QMessageBox::warning(this,"Error","No se ha ejecutado
correctamente el comando avconv paa unir audio y vídeo.");
            break;
        }
        return false;
    }

    QMessageBox::information(this,"Información","La grabación se ha montado
correctamente.");

    return true;
}

void VentanaPrincipal::on_camaraID_valueChanged(int arg1)
{
    setCamaraID(arg1);
    ui->selecArchivo->setChecked(false);
    ui->selecCamara->setChecked(true);
}

void VentanaPrincipal::on_camaraID_editingFinished()
{
    setCamaraID(ui->camaraID->value());
    ui->selecArchivo->setChecked(false);
    ui->selecCamara->setChecked(true);

}
```

```
void VentanaPrincipal::on_selecArchivo_clicked(bool checked)
{
    if(checked)
        ui->selecCamara->setChecked(false);

}

void VentanaPrincipal::on_selecCamara_clicked(bool checked)
{
    if(checked)
        ui->selecArchivo->setChecked(false);
}

void VentanaPrincipal::on_sonidoID_editingFinished()
{
    setSonidoID(ui->sonidoID->value());
}

void VentanaPrincipal::on_sonidoID_valueChanged(int arg1)
{
     setSonidoID(arg1);
}

void VentanaPrincipal::on_lineEdit_editingFinished()
{
    setArchivoSalida(ui->lineEdit->text().toStdString() + ".mp4");
    string s = getDirSalida() + getArchivoSalida();
    ui->ruta_dir->setText(QString::fromStdString(s));
}

void VentanaPrincipal::on_boton_previsualizar_clicked()
{
    VideoCapture cap;

    if(ui->selecCamara->isChecked())
        cap.open(getCamaraID());
    else
        cap.open(getRutaArchivo());

    Mat mat;
    cap >> mat;
    cap.release();

    if(!mat.empty())
    {
        QSize sz = ui->previsualizacion->frameSize();
        cv::resize(mat, mat, Size(sz.width(), sz.height()));

        // Copia la imagen mat
        const uchar *qImageBuffer = (const uchar*)mat.data;
        // Crea QImage con las mismas dimensiones que la imagen mat
        QImage img(qImageBuffer, mat.cols, mat.rows, mat.step,
QImage::Format_RGB888);
        ui->previsualizacion->setPixmap(QPixmap::fromImage(img.rgbSwapped()));
    }
    else
```

```
        QMessageBox::warning(this,"Error","No se ha podido abrir la ruta de
captura seleccionada.");

}

void VentanaPrincipal::on_formato_automatico_clicked(bool checked)
{
    if(checked){
        ui->formato_4_3->setChecked(false);
        ui->formato_16_9->setChecked(false);
        setFormatoPantalla(-1);
    }
    else{
        ui->formato_automatico->setChecked(true);
        ui->formato_4_3->setChecked(false);
        ui->formato_16_9->setChecked(false);
        setFormatoPantalla(-1);
    }
}

void VentanaPrincipal::on_formato_4_3_clicked(bool checked)
{
    if(checked){
        ui->formato_automatico->setChecked(false);
        ui->formato_16_9->setChecked(false);
        setFormatoPantalla(0);
    }
    else{
        ui->formato_automatico->setChecked(true);
        ui->formato_4_3->setChecked(false);
        ui->formato_16_9->setChecked(false);
        setFormatoPantalla(-1);
    }
}

void VentanaPrincipal::on_formato_16_9_clicked(bool checked)
{
    if(checked){
        ui->formato_automatico->setChecked(false);
        ui->formato_4_3->setChecked(false);
        setFormatoPantalla(1);
    }
    else{
        ui->formato_automatico->setChecked(true);
        ui->formato_4_3->setChecked(false);
        ui->formato_16_9->setChecked(false);
        setFormatoPantalla(-1);
    }
}

void VentanaPrincipal::on_ext_ancho_editingFinished()
{
    setExtAncho(ui->ext_ancho->value());
}

void VentanaPrincipal::on_ext_ancho_valueChanged(int arg1)
{
    setExtAncho(arg1);
```

```
}

void VentanaPrincipal::on_ext_alto_editingFinished()
{
    setExtAlto(ui->ext_alto->value());
}

void VentanaPrincipal::on_ext_alto_valueChanged(int arg1)
{
    setExtAlto(arg1);
}

void VentanaPrincipal::on_boton_mo_rutadir_clicked()
{
    QString fileName = QFileDialog::getExistingDirectory(this,
                                                          tr("Seleccione el
directorio donde guardar la captura"), ".");
    setDirMO(fileName.toStdString() + "/");
}

void VentanaPrincipal::on_lineEdit_mo_archivo_editingFinished()
{
    setArchMediaMO(ui->lineEdit_mo_archivo->text().toStdString());
}

void VentanaPrincipal::on_lineEdit_mo_captura_editingFinished()
{
    setArchCapMO(ui->lineEdit_mo_captura->text().toStdString() + ".cap");
}

void VentanaPrincipal::on_pushButton_mo_rAudio_clicked()
{
    QString fileName = QFileDialog::getExistingDirectory(this,
                                                          tr("Seleccione el
directorio donde cargar la captura"), ".");
    setDirMO(fileName.toStdString() + "/");
}

void VentanaPrincipal::on_lineEdit_mo_archivo2_editingFinished()
{
    setArchMediaMO(ui->lineEdit_mo_archivo2->text().toStdString());
}

void VentanaPrincipal::on_pushButton_mo_rCap_clicked()
{
    QString fileName = QFileDialog::getOpenFileName(this,
                                                    tr("Abrir archivo de
captura"), ".",
                                                    tr("Archivos de captura
(*.cap)"));
    setArchCapMO(fileName.toStdString());
}

void VentanaPrincipal::on_boton_iniciarCaptura_clicked()
{
    this->setVisible(false);
```

```
    tracking();
}

void VentanaPrincipal::on_boton_buscarCaptura_clicked()
{

    QString fileName = QFileDialog::getOpenFileName(this,
                                                     tr("Abrir archivo de vídeo"),
".",
                                                     tr("Archivos de vídeo (*.mp4
*.avi *.webm)"));
    setRutaArchivo(fileName.toStdString());
    ui->selecArchivo->setChecked(true);
    ui->selecCamara->setChecked(false);
    ui->ruta_archivo->setText(fileName);
}

void VentanaPrincipal::on_boton_buscarDirMontaje_clicked()
{
    QString fileName = QFileDialog::getExistingDirectory(this,
                                                         tr("Seleccione el
directorio donde guardar el vídeo final"), ".");
    setDirSalida(fileName.toStdString() + "/");
    ui->ruta_dir->setText(fileName);
}

void VentanaPrincipal::on_boton_iniciarMontaje_clicked()
{
    procesar();
}
```

3.3 tracker.hpp

Este archivo contiene la clase encargada de realizar el seguimiento del puntero láser y la obtención de sus correspondientes coordenadas.

```
    /*
 * Proyecto: Sistema Informático para el Seguimiento de Puntero Láser y su
Representación Virtual mediante Visión Artificial
 * Fichero: tracker.hpp
 * Descripción: fichero que contiene la clase encargada de realizar el
seguimiento del puntero láser y la obtención de sus correspondientes coordenadas.
 * Autor: Rafael Ulises Baena Herruzo (i32baher@uco.es)
 */

#ifndef TRACKER_HPP
#define TRACKER_HPP

#include "opencv2/video/background_segm.hpp"
```

```
#include "opencv2/videoio.hpp"
#include "tiempo.hpp"
#include <vector>

//#include <opencv2/core/utility.hpp>
//#include "opencv2/features2d.hpp"
//#include "opencv2/core.hpp"

using namespace std;
using namespace cv;

// Estructura que guarda la localización del puntero en un tiempo dado
struct localizacion{
  Point2f posicion;
  uint64_t tiempo;
};

namespace lp{
    class Tracker{

    public:

      // Constructores de la clase Tracker.
      Tracker(const string &ruta_archivo);

      Tracker(const int &camara_id);

      Tracker(){}

      // Destructor de la clase Tracker
      ~Tracker(){}

      /* Nombre: inicializar
       * Objetivo: método encargado de inicializar la clase, realizando el
tracking desde un archivo
       * Parámetros de entrada:
       *                      - ruta_archivo: nombre del archivo al que se
realizará el tracking del puntero láser
       * Parámetros de salida:
       *                      - valor lógico que inidica si se ha realizado la
ejecución correctamente
       */
      bool inicializar(const string &ruta_archivo);

      /* Nombre: inicializar
       * Objetivo: método encargado de inicializar la clase, realizando el
tracking desde una cámara
       * Parámetros de entrada:
       *                      - camara_id: número de la cámara desde la que se
realizará el tracking del puntero láser
       * Parámetros de salida:
       *                      - valor lógico que inidica si se ha realizado la
ejecución correctamente
       */
      bool inicializar(const int &camara_id);
```

```
      /* Nombre: capturar
       * Objetivo: método que realiza la captura de un frame de la cámara,
devuelve true si se ha capturado correctamente.
       * Parámetros de entrada: ninguno
       * Parámetros de salida:
       *                       - valor lógico que inidica si se ha realizado la
ejecución correctamente
       */
      bool capturar();

      /* Nombre: release
       * Objetivo: método que finaliza la captura y cierra la cámara o archivo
       * Parámetros de entrada: ninguno
       * Parámetros de salida: ninguno
       */
      void release();

      /* Nombre: extraerFondo
       * Objetivo: método que extrae el fondo estático de la imagen para así
poder detectar el puntero láser
       * Parámetros de entrada: ninguno
       * Parámetros de salida: ninguno
       */
      void extraerFondo();

      /* Nombre: operacionesMorfologicas
       * Objetivo: método que realiza una serie de operaciones morfológicas a la
máscara obtenida después de la extracción del fondo, la cual contiene al puntero
láser físico
       * Parámetros de entrada:
       *                       - niters: valor entero que indica el número de
iteraciones a aplicar en el algoritmo
       * Parámetros de salida: ninguno
       */
      void operacionesMorfologicas(int niters = 1);

      /* Nombre: calcularPosicion
       * Objetivo: método que calcula la posición de el puntero láser, dada la
máscara con las operaciones morfológicas ya realizadas
       * Parámetros de entrada:
       *                       - anterior: punto de dos dimensiones,
correspondiente al punto anteriormente captado
       * Parámetros de salida:
       *                       - punto de dos dimensiones, correspondiente a la
posición del puntero láser
       */
      Point2f calcularPosicion(Point2f anterior = Point2f(0,0));

      /* Nombre: dibujarCirculo
       * Objetivo: método que dibuja un círculo alrededor del puntero láser en la
imagen original
       * Parámetros de entrada:
       *                       - radio: valor entero correspondiente al radio del
círculo
       *                       - ancho: valor entero correspondiente al ancho del
círculo
       * Parámetros de salida:
       *                       - Imagen con el círculo dibujado alrededor del
```

```
puntero láser
	 */
	Mat dibujarCirculo(int radio = 5, int ancho = 2);

	/* Nombre: calcularResolucion
	 * Objetivo: método que calcula la resolución correspondiente a la
superficie de proyección seleccionada por el usuario
	 * Parámetros de entrada: ninguno
	 * Parámetros de salida:
	 *                      - punto de dos dimensiones correspondiente a la
resolución de la superficie de proyección
	 */
	inline Point2f calcularResolucion()const{Rect2d r = getRegionInteres();
return Point2f(r.width, r.height);}

	/* Nombre: guardarArchivoLoc
	 * Objetivo: método que guarda en un archivo la localización del puntero
láser respecto al tiempo
	 * Parámetros de entrada:
	 *                      - ruta: string que contiene la ruta del archivo
	 * Parámetros de salida:
	 *                      - valor lógico que inidica si se ha realizado la
ejecución correctamente
	 */
	bool guardarArchivoLoc(const string &ruta);

	/* Nombre: cargarArchivoLoc
	 * Objetivo: método que carga desde un archivo la localización del puntero
láser respecto al tiempo
	 * Parámetros de entrada:
	 *                      - ruta: string que contiene la ruta del archivo
	 * Parámetros de salida:
	 *                      - valor lógico que inidica si se ha realizado la
ejecución correctamente
	 */
	bool cargarArchivoLoc(const string &ruta);

	/*
	 * OBSERVADORES Y MODIFICADORES PÚBLICOS DE LA CLASE
	 */

	inline bool estaAbierta()const{return _capturaAbierta;}

	inline Mat getImagen()const{return _img.clone();}

	inline Mat getFgimg()const{return _fgimg.clone();}

	inline Mat getFgmask()const{return _fgmask.clone();}

	inline Point2f getPosicion()const{return _posicion;}

	inline vector<localizacion> getLocalizacion()const{return _vloc;}

	inline Rect2d getRegionInteres()const{return _region_de_interes;}

	/*
```

```
       * FIN OBSERVADORES Y MODIFICADORES PÚBLICOS DE LA CLASE
       */

      inline Point2f getResolucion()const{return _resolucion;}
      inline void setResolucion(const Point2f resolucion){_resolucion =
resolucion;}

    private:
      VideoCapture _captura;
      bool _capturaAbierta;
      Ptr<BackgroundSubtractor> _bg_model;
      Mat _img, _fgimg, _fgmask;
      Point2f _posicion;
      vector <localizacion> _vloc;
      Clock _reloj;
      bool _esArchivo; // True si el tracking es desde un archivo de video
      Rect2d _region_de_interes;
      Point2f _resolucion;

      /*
       * OBSERVADORES Y MODIFICADORES PRIVADOS DE LA CLASE
       */

      inline VideoCapture getCaptura()const{return _captura;}
      inline void setCaptura(const VideoCapture &captura){_captura = captura;}

      inline void setEstaAbierta(const bool &capturaAbierta){_capturaAbierta =
capturaAbierta;}

      inline Ptr<BackgroundSubtractor> getBgModel()const{return _bg_model;}
      inline void setBgModel(const Ptr<BackgroundSubtractor> &bg_model){_bg_model
= bg_model;}

      inline void setImagen(const Mat &img){_img = img;}
      inline void setFgimg(const Mat &fgimg){_fgimg = fgimg;}
      inline void setFgmask(const Mat &fgmask){_fgmask = fgmask;}

      inline void setPosicion(const Point2f &posicion){_posicion = posicion;}
      inline void setLocalizacion(const vector<localizacion> &vloc){_vloc =
vloc;}

      inline Clock getReloj()const{return _reloj;}
      inline void setReloj(const Clock &reloj){_reloj = reloj;}

      inline bool esArchivo()const{return _esArchivo;}
      inline void setEsArchivo(const bool &esArchivo){_esArchivo = esArchivo;}

      inline void setRegionInteres(const Rect2d &region_de_interes)
{_region_de_interes = region_de_interes;}

      /*
       * FIN OBSERVADORES Y MODIFICADORES PRIVADOS DE LA CLASE
       */

      /* Nombre: iniciarReloj
       * Objetivo: método que inicia el cronómetro
```

```
        * Parámetros de entrada: ninguno
        * Parámetros de salida: ninguno
        */
       void iniciarReloj();

       /* Nombre: obtenerSuperficieProyeccion
        * Objetivo: método que se encarga de pedirle al usuario que seleccione la
superficie donde se están proyectando las diapositivas
        * Parámetros de entrada: ninguno
        * Parámetros de salida:
        *                      - valor lógico que inidica si se ha realizado la
ejecución correctamente
        */
       bool obtenerSuperficieProyeccion();

    };
}

#endif
```

3.4 tracker.cpp

Fichero que contiene los métodos que dan funcionalidad a la clase Tracker.

```
/*
 * Proyecto: Sistema Informático para el Seguimiento de Puntero Láser y su
Representación Virtual mediante Visión Artificial
 * Fichero: tracker.cpp
 * Descripción: fichero que contiene los métodos que dan funcionalidad a la clase
Tracker.
 * Autor: Rafael Ulises Baena Herruzo (i32baher@uco.es)
 */

#include "opencv2/videoio.hpp"
#include "opencv2/video/background_segm.hpp"
#include "opencv2/imgproc.hpp"
#include "opencv2/highgui.hpp"
#include "tracker.hpp"
#include <opencv2/tracking.hpp>

#include <iostream>
#include <fstream>

using namespace std;
using namespace cv;

namespace lp{
    Tracker::Tracker(const string &ruta_archivo){
        inicializar(ruta_archivo);
    }
```

```
    Tracker::Tracker(const int &camara_id){
        inicializar(camara_id);
    }

    bool Tracker::inicializar(const string &ruta_archivo){
      VideoCapture captura;

      if(!captura.open(ruta_archivo))
          return false;
      setEsArchivo(true);

      setEstaAbierta(captura.isOpened());
      setCaptura(captura);

      Ptr<BackgroundSubtractor> bg_model = createBackgroundSubtractorMOG2(16, 45,
false).dynamicCast<BackgroundSubtractor>();
      setBgModel(bg_model);

      if(!obtenerSuperficieProyeccion())
          return false;

      iniciarReloj();
      return true;
    }

    bool Tracker::inicializar(const int &camara_id){
      VideoCapture captura;

      if(!captura.open(camara_id))
          return false;

      setEsArchivo(false);

      setEstaAbierta(captura.isOpened());
      setCaptura(captura);

      Ptr<BackgroundSubtractor> bg_model = createBackgroundSubtractorMOG2(16, 45,
false).dynamicCast<BackgroundSubtractor>();
      setBgModel(bg_model);

      if(!obtenerSuperficieProyeccion())
          return false;
      iniciarReloj();
      return true;
    }

    void Tracker::iniciarReloj(){
      Clock reloj;
      reloj.start();
      setReloj(reloj);
    }

    bool Tracker::capturar(){
      VideoCapture captura = getCaptura();

      // Si la fuente de captura es un archivo, se realiza una captura temporal
```

```
de los frames
      // esto se hace para imitar de forma realista una captura en directo.
      if(esArchivo()){
        Clock reloj = getReloj();
        const int fps = 25;
        const double pos = ((double)reloj.elapsed()/1000)*fps/1000;
        captura.set(CV_CAP_PROP_POS_FRAMES,pos);
      }

      Mat img;
      captura >> img;

      Rect2d region_de_interes = getRegionInteres();

      setImagen(img(region_de_interes));
      setCaptura(captura);

      return !img.empty();
    };

    void Tracker::release(){
        VideoCapture cap = getCaptura();
        cap.release();
        setCaptura(cap);
    }

    void Tracker::extraerFondo(){
      Mat fgimg = getFgimg();
      Mat img = getImagen();
      Mat fgmask = getFgmask();

      if( fgimg.empty() )
        fgimg.create(img.size(), img.type());

      // actualiza el modelo

      Ptr<BackgroundSubtractor> bg_model = getBgModel();
      bg_model->apply(img, fgmask, -1);

      fgimg = Scalar::all(0);
      img.copyTo(fgimg, fgmask);

      setBgModel(bg_model);
      setFgimg(fgimg);
      setFgmask(fgmask);
      setImagen(img);
    };

    void Tracker::operacionesMorfologicas(int niters){
      Mat temp = getFgmask();
      dilate(temp, temp, Mat(), Point(-1,-1), niters*2);
      erode(temp, temp, Mat(), Point(-1,-1), niters*1);
      dilate(temp, temp, Mat(), Point(-1,-1), niters*2);

      setFgmask(temp);
```

```
    };

    Point2f Tracker::calcularPosicion(Point2f anterior){
      Mat mascara = getFgmask();
      Point2f posicion, punto;
      float radio;

      int mayor_area=0;
      vector<vector<Point> > contornos;
      findContours(mascara, contornos, CV_RETR_EXTERNAL, CV_CHAIN_APPROX_SIMPLE);

      if( contornos.size() > 0 ){
        // Encuentra el mayor contorno y lo uso para calcular el menor circulo
que lo encierra y el centroide
        for(unsigned int j = 0; j< contornos.size(); j++){    // Itera a través
de cada contorno

              double a=contourArea(contornos[j],false);  // Calcula el área del
contorno
              if(a>mayor_area){
                  mayor_area=a;
                  minEnclosingCircle(contornos[j], posicion, radio);
              }
        }
      }
      else{
        posicion = anterior;
      }

      // Genera los datos para la nueva localizacion
      Clock reloj = getReloj();
      localizacion loc = {posicion, reloj.elapsed()};

      // Inserta la nueva posición al vector de localizacion (posicion / tiempo)
      vector<localizacion> vloc = getLocalizacion();
      vloc.push_back(loc);
      setLocalizacion(vloc);

      setPosicion(posicion);
      return posicion;
    }

    Mat Tracker::dibujarCirculo(int radio, int ancho){
      Mat img = getImagen();
      Point2f posicion = getPosicion();
      circle(img, Point(posicion.x, posicion.y), radio, Scalar(0, 0, 255),
ancho);
      return img;
    }

    bool Tracker::obtenerSuperficieProyeccion(){
        VideoCapture captura = getCaptura();

        Mat img;
        captura >> img;

        if(img.empty())
```

```
            return false;

        bool fromCenter = false;
        bool showCrosshair = false;
        Rect2d r = selectROI("Seleccione la superficie de proyeccion", img,
fromCenter, showCrosshair);
        setRegionInteres(r);
        // Obtiene la resolución de la pantalla seleccionada
        setResolucion(calcularResolucion());
        destroyAllWindows();

        return true;
    }

    bool Tracker::guardarArchivoLoc(const string &ruta){
        vector<localizacion> loc = getLocalizacion();
        ofstream f(ruta);
        if(!f.is_open())
            return false;

        Point2f resolucion = getResolucion();
        // En la primera línea del archivo se guarda la resolución de la captura
        f<<resolucion.x<<" "<<resolucion.y<<endl;
        for(int i=0;i<loc.size();i++) {
          f<<loc[i].posicion.x<<" "<<loc[i].posicion.y<<" "<<loc[i].tiempo<<endl;
        }
        f.close();
        return true;
    }

    bool Tracker::cargarArchivoLoc(const string &ruta){
        vector<localizacion> loc;
        ifstream f(ruta);
        if(!f.is_open())
            return false;

        char cadena[128];

        Point2f resolucion;
        f>>cadena;
        resolucion.x = atoi(cadena);
        f>>cadena;
        resolucion.y = atoi(cadena);
        setResolucion(resolucion);

        while(!f.eof()) {
           localizacion l;

           f>>cadena;
           l.posicion.x = atoi(cadena);
           f>>cadena;
           l.posicion.y = atoi(cadena);

           f>>cadena;
           l.tiempo = atof(cadena);
           loc.push_back(l);
```

```
        }
        loc.pop_back();
        setLocalizacion(loc);
        f.close();
        return true;
    }
}
```

3.5 grabacion.hpp

Este fichero contiene la clase encargada de realizar la grabación de las dispositivas en formato digital y del audio generado durante la presentación. También se encarga del montaje de esta, es decir, teniendo las coordenadas del puntero láser, representa estas en el archivo de vídeo que contiene las diapositivas grabadas y finalmente une el resultado con el archivo de audio.

```
/*
 * Proyecto: Sistema Informático para el Seguimiento de Puntero Láser y su
Representación Virtual mediante Visión Artificial
 * Fichero: grabacion.hpp
 * Descripción: fichero que contiene la clase encargada de realizar la grabación
de las dispositivas en formato digital y del audio generado durante la
presentación. También se encarga del montaje de esta, es decir, teniendo las
coordenadas del puntero láser, representa estas en el archivo de vídeo que
contiene las diapositivas grabadas y finalmente unir el resultado con el archivo
de audio.
 * Autor: Rafael Ulises Baena Herruzo (i32baher@uco.es)
 */

#ifndef GRABACION_HPP
#define GRABACION_HPP

#include <fstream>
#include <vector>
#include "tracker.hpp"
#include "logicclass.hpp"

using namespace std;

static const char alphanum[] =
"0123456789"
"!@#$%^&*"
"ABCDEFGHIJKLMNOPQRSTUVWXYZ"
"abcdefghijklmnopqrstuvwxyz";

namespace lp{
    class Grabacion{

    public:

      // Constructor de la clase Grabacion.
      Grabacion(int hw_sonido);
```

```
      Grabacion(){}

      // Destructor de la clase Grabacion
      ~Grabacion(){}

      /* Nombre: iniciar
       * Objetivo: método que se encarga de iniciar la captura del audio y las
diapositivas de la presentación (realizando una grabación del escritorio)
       * Parámetros de entrada:
       *                      - resolucion_pprincipal: punto de dos dimensiones
que contiene la resolución de la pantalla principal en caso de usar la pantalla
extendida
       * Parámetros de salida:
       *                      - valor entero que contiene el código de salida
obtenido.
       */
      // Devuelve 0 en caso de salida normal. En caso de error devuelve:
      // 1 Error debido a que la grabación ya está iniciada
      // 2 Error debido a un fallo en el fork()
      // 3 Error debido a que la ruta del fichero de salida ya existe.
      int iniciar(const Point &resolucion_pprincipal = Point(0,0));

      /* Nombre: parar
       * Objetivo: método que se encarga de parar la captura del audio y vídeo
       * Parámetros de entrada: ninguno
       * Parámetros de salida:
       *                      - valor entero que contiene el código de salida
obtenido.
       */
      // Devuelve 0 en caso de salida normal. En caso de error devuelve:
      // 1 Error debido a que la grabación no se ha iniciado
      // 2 Error debido a que el comando kill no se ha ejecutado correctamente
      // 3 Error debido a que no se ha grabado bien el audio o video
      int parar();

      /* Nombre: procesar
       * Objetivo: método que se encarga de dibujar las coordenadas del puntero
láser en las diapositivas grabadas, y posteriormente unirá el audio y el video
generado en un archivo de vídeo
       * Parámetros de entrada:
       *                      - loc: vector que contiene la localización de las
coordenadas del puntero láser en función del tiempo
       *                      - resolución: punto de dos dimensiones que contiene
la resolución de la superficie de proyección
       *                      - ruta_saluda: string que almacena la ruta de salida
del archivo procesado
       *                      - logica: objeto de tipo LogicClass que servirá para
ir informando a la interfaz de usuario del progreso
       * Parámetros de salida:
       *                      - valor entero que contiene el código de salida
obtenido.
       */
      // Devuelve 0 en caso de salida normal. En caso de error devuelve:
      // 1 Error debido a que no se ha podido abrir el archivo de video
      // 2 Error debido a que no se ha podido crear el archivo de salida de video
      // 3 Error debido a que no existe el archivo de audio o vídeo, o ya existe
la ruta de salida
      // 4 Error debido a un fallo en el fork()
```

```
      // 5 Error debido a que no se ha ejecutado correctamente el comando avconv
paa unir audio y vídeo
      int procesar(const vector <localizacion> &loc, const Point2f &resolucion,
const string &ruta_salida, LogicClass &logica);

      /*
       * OBSERVADORES Y MODIFICADORES PÚBLICOS DE LA CLASE
       */

      // 0 = 4:3 / 1 = 16:9 / -1 = automático
      inline int getFormatoPantalla()const{return _formato_pantalla;}
      inline void setFormatoPantalla(const int &formato_pantalla)
{_formato_pantalla = formato_pantalla;}

      inline string getRuta()const{return _ruta_tmp;}
      inline void setRuta(const string &ruta_tmp){_ruta_tmp = ruta_tmp;}

      /*
       * FIN OBSERVADORES Y MODIFICADORES PÚBLICOS DE LA CLASE
       */

    private:
      pid_t _pid_audio, _pid_video;
      int _ancho, _alto;
      string _ruta_tmp;
      bool _iniciada;
      int _hw_sonido;
      int _formato_pantalla; // 0 = 4:3 / 1 = 16:9 / -1 = Automático

      /*
       * OBSERVADORES Y MODIFICADORES PRIVADOS DE LA CLASE
       */

      inline pid_t getPidAudio()const{return _pid_audio;}
      inline void setPidAudio(const pid_t &pid){_pid_audio = pid;}

      inline pid_t getPidVideo()const{return _pid_video;}
      inline void setPidVideo(const pid_t &pid){_pid_video = pid;}

      inline int getAncho()const{return _ancho;}
      inline void setAncho(const int &ancho){_ancho = ancho;}

      inline int getAlto()const{return _alto;}
      inline void setAlto(const int &alto){_alto = alto;}

      inline bool getIniciada()const{return _iniciada;}
      inline void setIniciada(const bool &iniciada){_iniciada = iniciada;}

      inline int getHwSonido()const{return _hw_sonido;}
      inline void setHwSonido(const int &hw_sonido){_hw_sonido = hw_sonido;}

      /*
       * FIN OBSERVADORES Y MODIFICADORES PRIVADOS DE LA CLASE
       */

      /* Nombre: existeArchivo
```

```
        * Objetivo: método que se encarga de comprobar si existe un archivo dado
        * Parámetros de entrada:
        *                       - filename: string que contiene el nombre del
archivo
        * Parámetros de salida:
        *                       - valor lógico que indica si la ejecución se ha
realizado correctamente
        */
       inline bool existeArchivo(const string &filename){ifstream f(filename);
return f.is_open();}

       /* Nombre: string2char
        * Objetivo: método que se encarga de transformar un objeto de tipo string
a tipo char
        * Parámetros de entrada:
        *                       - str: string
        * Parámetros de salida:
        *                       - el string convertido a char
        */
       char *string2char(const string &str);

       /* Nombre: combinar
        * Objetivo: método que se encarga de unir el audio grabado con el vídeo
correspondiente a la grabación de las diapositivas ya procesadas con el puntero
láser representado en ellas
        * Parámetros de entrada:
        *                       - ruta_salida: string que almacena la ruta de salida
del archivo final
        * Parámetros de salida:
        *                       - valor entero que contiene el código de salida
obtenido.
        */
       int combinar(const string &ruta_salida);

       /* Nombre: combinar
        * Objetivo: método que se encarga de generar un nombre aleatorio para los
archivos que se usarán durante el proceso de grabación y procesado
        * Parámetros de salida:
        *                       - string que contiene un nombre aleatorio
        */
       string nombreAleatorio();

    };
}

#endif
```

3.6 grabacion.cpp

Fichero que contiene los métodos que dan funcionalidad a la clase Grabacion.

```
/* Proyecto: Sistema Informático para el Seguimiento de Puntero Láser y su
Representación Virtual mediante Visión Artificial
 * Fichero: grabacion.cpp
 * Descripción: fichero que contiene los métodos que dan funcionalidad a la clase
Grabacion.
 * Autor: Rafael Ulises Baena Herruzo (i32baher@uco.es)
 */

#include <string>
#include <iostream>
#include <vector>
#include "opencv2/highgui.hpp"
#include "opencv2/imgproc.hpp"
#include "grabacion.hpp"

extern "C"
{
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <X11/Xlib.h>
#include <string.h>
}

namespace lp{
    Grabacion::Grabacion(int hw_sonido){
      Display* disp = XOpenDisplay(NULL);
      Screen* scrn = XScreenOfDisplay(disp,0);
      int alto = scrn->height;
      int ancho  = scrn->width;

      setHwSonido(hw_sonido);
      setAncho(ancho);
      setAlto(alto);
      setIniciada(false);
      setFormatoPantalla(-1); // Pone el formato de pantalla por defecto en 16:9

      // Genera una ruta temporal donde se guardará el audio y video generado
      string ruta_tmp = "/tmp/" + nombreAleatorio();
      setRuta(ruta_tmp);
    }

    string Grabacion::nombreAleatorio(){
      srand(time(0));
      string str;
      int stringLength = sizeof(alphanum) - 1;
      for(unsigned int i = 0; i < 16; ++i){
      str += alphanum[rand() % stringLength];
```

```
      }
      return str;
    }

    int Grabacion::iniciar(const Point &resolucion_pprincipal){
      // Comprueba que no se ha iniciado la grabación este parada.
      if(getIniciada() == true)
        return 1;

      // Comprueba que tanto el fichero de video como el de sonido no existan
      string ruta_video = getRuta() + ".mp4";
      string ruta_sonido = getRuta() + ".mp3";
      if(existeArchivo(ruta_video) || existeArchivo(ruta_sonido))
        return 3;

      pid_t pid_audio, pid_video;

      // Se realizan los forks correspondientes a la captura de audio y de
pantalla.
      switch(pid_video = fork()){
        case -1:{
            return 2;
          }
        case 0 :{
          string s_resolucion;
          string s_ajuste;
          // Pantalla extendida - si la pantalla principal es menor que el alto
total
          if(resolucion_pprincipal.x != 0 && resolucion_pprincipal.y < getAlto())
{
            s_resolucion = to_string(getAncho()-resolucion_pprincipal.x) + "x" +
to_string(getAlto());
            s_ajuste = ":0.0+" + to_string(resolucion_pprincipal.x);
          }
          // Si la pantalla principal tiene el mismo alto que la total, se
calcula el alto de la extendida
          else if(resolucion_pprincipal.x != 0 && resolucion_pprincipal.y ==
getAlto()){
            // Cálculo realizado para una segunda pantalla con una relación de
aspecto 16:9 o 4:3
              int alto_ajustado = (getAncho() - resolucion_pprincipal.x)*3/4;
              if(alto_ajustado > getAlto())
                  alto_ajustado = (getAncho() - resolucion_pprincipal.x)*9/16;

              s_resolucion = to_string(getAncho()-resolucion_pprincipal.x) + "x"
+ to_string(alto_ajustado);
              s_ajuste = ":0.0+" + to_string(resolucion_pprincipal.x);
          }
          else{
            s_resolucion = to_string(getAncho()) + "x" + to_string(getAlto());
            s_ajuste = ":0.0+0";
          }
          cout<<s_resolucion<<endl;
          cout<<s_ajuste<<endl;

          char * resolucion = string2char(s_resolucion);
          char * ajuste = string2char(s_ajuste);
```

```
          char * ruta_salida = string2char(ruta_video);

          char *args[] = {(char*)"avconv", (char*)"-f", (char*)"x11grab",
(char*)"-r", (char*)"25",
          (char*)"-s", resolucion, (char*)"-i", ajuste, (char*)"-q:v",
(char*)"0", ruta_salida, (char*)0};

          int error = execvp("avconv", args);
          if(error == -1){
              cout << "Ha ocurrido un error al ejecutar el comando" << endl;
          }
          exit(-1);
        }
      }

      switch(pid_audio = fork()){
        case -1:{
            return 2;
          }
        case 0 :{
          char *ruta_salida = string2char(ruta_sonido);

          int hw_sonido = getHwSonido();
          string str = "hw:" + to_string(hw_sonido);
          char *hw = string2char(str);

          char *args[] = {(char*)"avconv", (char*)"-f", (char*)"alsa", (char*)"-
i", hw, ruta_salida, (char*)0};

          int error = execvp("avconv", args);
          if(error == -1){
              cout << "Ha ocurrido un error al ejecutar el comando" << endl;
          }
          exit(-1);
        }
      }
      setPidAudio(pid_audio);
      setPidVideo(pid_video);
      setIniciada(true);
      return 0;
    }

    int Grabacion::parar(){
      // Comprueba que la grabación este iniciada
      if(getIniciada() == false)
        return 1;

      pid_t pid_audio = getPidAudio(), pid_video = getPidVideo();

      string comando = "/bin/kill -SIGINT " + to_string(pid_audio) + " " +
to_string(pid_video);

      // Le manda la señal SIGINT a los procesos correspondientes con la
grabación de audio y video
      int error = system(comando.c_str());
      if(error != 0)
        return 2;
```

```
      string ruta_video = getRuta() + ".mp4";
      string ruta_sonido = getRuta() + ".mp3";
      if(!existeArchivo(ruta_video) || !existeArchivo(ruta_sonido))
        return 3;

      return 0;
    }

    char *Grabacion::string2char(const string &str){
      char * cstr = new char [str.length()+1];
      strcpy (cstr, str.c_str());
      return cstr;
    }

    int Grabacion::combinar(const string &ruta_salida){
      string ruta_video = getRuta() + ".mkv";
      string ruta_audio = getRuta() + ".mp3";

      if(!existeArchivo(ruta_video) || !existeArchivo(ruta_audio) ||
existeArchivo(ruta_salida))
        return 3;

      char * ruta_v = string2char(ruta_video);
      char * ruta_a = string2char(ruta_audio);
      char * ruta_s = string2char(ruta_salida);

      char *args[] = {(char*)"avconv", (char*)"-i", ruta_v, (char*)"-i", ruta_a,
      (char*)"-c:v", (char*)"copy", (char*)"-c:a", (char*)"copy", ruta_s,
(char*)0};

      pid_t pid;
      switch(pid = fork()){
        case -1:{
            return 4;
          }
        case 0 :{
          int error = execvp("avconv", args);
          if(error == -1){
              return 5;
          }
        }
        default:
            wait();
      }

      return 0;
    }

    int Grabacion::procesar(const vector <localizacion> &loc, const Point2f
&resolucion, const string &fichero_salida, LogicClass &logica){
      string ruta_video = getRuta() + ".mp4";
      VideoCapture captura(ruta_video);
      if(!captura.isOpened())
        return 1;

      int ancho = captura.get(CV_CAP_PROP_FRAME_WIDTH);
```

```
      int alto = captura.get(CV_CAP_PROP_FRAME_HEIGHT);
      double relacion_aspecto = ancho/alto;

      int ancho_formateado = ancho; // Guarda el ancho calibrado al formato de
pantalla.
      int alto_formateado = alto; // Guarda el alto calibrado al formato de
pantalla.
      // Obtiene el formato de pantalla a través de la resolución de la
presentación
      int formato_pantalla = resolucion.x / resolucion.y;
      int desplazamiento_x = 0;
      int desplazamiento_y = 0;

      // Detección automática del formato de pantalla de la presentación
      if(getFormatoPantalla() == -1){
          // Realiza el que mas se acerque a la relación de aspecto de la
presentación
          if(relacion_aspecto == 16/9 && abs(formato_pantalla - 4/3) <
abs(formato_pantalla - 16/9)){
              ancho_formateado = alto*4/3;
              desplazamiento_x += (ancho - ancho_formateado)/2; // Se coge la
diferencia y se divide en dos márgenes.
          }
          else if(relacion_aspecto == 4/3 && abs(formato_pantalla - 4/3) >
abs(formato_pantalla - 16/9)){
              alto_formateado = ancho*16/9;
              desplazamiento_y += (alto - alto_formateado)/2; // Se coge la
diferencia y se divide en dos márgenes.
          }
      }
      // Formato de pantalla manual 16:9 o 4:3
      else{
          formato_pantalla = getFormatoPantalla();
          if(relacion_aspecto == 16/9 && formato_pantalla == 0){
              ancho_formateado = alto*4/3;
              // Una vez se tiene el ancho adecuado para que la relación sea de
4:3
              // se debe realizar un desplazamiento en el eje x, para que el
puntero
              // no se salga fuera de la diapositiva.
              desplazamiento_x += (ancho - ancho_formateado)/2; // Se coge la
diferencia y se divide en dos márgenes.
          }
          // Si el formato de pantalla es 4:3 y la presentación está a 16:9
          // Si el formato de pantalla fuese 16:9 no afectaría que la
presentación esté a 16:9
          else if(relacion_aspecto == 4/3 && formato_pantalla == 1){
              alto_formateado = ancho*16/9;
              // Una vez se tiene el ancho adecuado para que la relación sea de
4:3
              // se debe realizar un desplazamiento en el eje x, para que el
puntero
              // no se salga fuera de la diapositiva.
              desplazamiento_y += (alto - alto_formateado)/2; // Se coge la
diferencia y se divide en dos márgenes.
          }
      }

      const int fps = captura.get(CV_CAP_PROP_FPS);
```

```
      const int nFrames = captura.get(CV_CAP_PROP_FRAME_COUNT);

      logica.emitNframes(nFrames);

      string ruta_salida = getRuta() + ".mkv";
      VideoWriter salida(ruta_salida, CV_FOURCC('X','2','6','4'), fps,
Size(ancho, alto));
      if(!salida.isOpened())
        return 2;

      int fcount = 0;
      for(unsigned int i=0; i<loc.size(); i++){
        // Obtiene la posición relativa (al tiempo) del frame al que corresponde
dicha coordenada
        const int pos = ((double)loc[i].tiempo/1000)*fps/1000;

        for(; fcount<pos && fcount<nFrames; fcount++){
          Mat img;
          captura >> img;

          // Dibuja el puntero
          if(loc[i].posicion.x != 0 && loc[i].posicion.y != 0){
            // Realiza una interpolación lineal para ajustar las coordenadas del
puntero a los nuevos ancho y alto.
            Point2f pos((loc[i].posicion.x*ancho_formateado/resolucion.x)
+desplazamiento_x, (loc[i].posicion.y*alto_formateado/resolucion.y)
+desplazamiento_y);
            circle(img, Point(pos.x, pos.y), 4, Scalar(0, 0, 255), 8);
            circle(img, Point(pos.x, pos.y), 12, Scalar(0, 0, 255), 2);
          }
          // Actualiza la barra de progreso de la interfaz

          salida << img;
        }

        logica.emitProgress(fcount*100/nFrames);
        logica.emitFrame(fcount);
      }
      logica.emitProgress(100);
      logica.emitFrame(nFrames);
      //ui->frame_actual->setText(QString::fromStdString(to_string(nFrames)));

      return combinar(fichero_salida);
    }
}
```

3.7 logicclass.hpp

Este fichero contiene la clase encargada de enlazar la parte lógica del programa, con la interfaz de usuario. Su función principal será mostrar el progreso a la hora de realizar el montaje.

```
/*
 * Proyecto: Sistema Informático para el Seguimiento de Puntero Láser y su
Representación Virtual mediante Visión Artificial
 * Fichero: logicclass.hpp
 * Descripción: fichero que contiene la clase encargada de enlazar la parte
lógica del programa, con la interfaz de usuario. Su función principal será
mostrar el progreso a la hora de realizar el montaje.
 * Autor: Rafael Ulises Baena Herruzo (i32baher@uco.es)
 */

#ifndef LOGICCLASS_HPP
#define LOGICCLASS_HPP

#include <QObject>
#include <string>

class LogicClass : public QObject
{
    Q_OBJECT
public:
    explicit LogicClass(QObject *parent = 0){}

    // max: obtendrá el valor máximo de la barra de progreso
    int max(){ return 100; }

    // min: obtendrá el valor mínimo de la barra de progreso
    int min(){ return 0; }

    // EmitProgress: emitirá el porcentaje del progreso de montaje
    void emitProgress(int value){ emit signalProgress(value); }

    // emitFrame: emitirá el número de frame por el que va el proceso de montaje
    void emitFrame(int value){ emit
signalFrame(QString::fromStdString(std::to_string(value))); }

    // emitNframes: emitirá el número total de frames
    void emitNframes(int value){ emit
signalNframes(QString::fromStdString(std::to_string(value))); }

signals:

    // signalProgress: asignará el porcentaje del progreso de montaje
    void signalProgress(int);

    // signalFrame: asignará el número de frame por el que va el proceso de
montaje
    void signalFrame(QString);

    // signalNframes: establecerá el número total de frames
```

```
    void signalNframes(QString);

public slots:

};

#endif // LOGIC_HPP
```

3.8 tiempo.hpp

Este archivo contiene la clase se encarga de medir el tiempo transcurrido.

```
/*
 * Proyecto: Sistema Informático para el Seguimiento de Puntero Láser y su
Representación Virtual mediante Visión Artificial
 * Fichero: tiempo.hpp
 * Descripción: fichero que contiene la clase encargada de medir el tiempo.
 * Autor: Rafael Ulises Baena Herruzo (i32baher@uco.es)
 */

#ifndef TIEMPO_HPP
#define TIEMPO_HPP

/*
 * A pesar de que se puede trabajar con nanosegundos, se devuelven los tiempos en
microsegundos
 * Tipo de dato: struct timespec
 * La estructura struct timespec representa el tiempo transcurrido. Está
declarada en sys/time.h y tiene los siguientes miembros:
 * time_t tv_sec: representa el número de segundos enteros de tiempo transcurrido
 * long tv_nsec: este es el resto de tiempo transcurrido (una fracción de
segundo), representado como un número en nanosegundos.
 */

// http://man7.org/linux/man-pages/man2/clock_gettime.2.html

// Ojo hay que compilar g++ -Wall main.cpp -lrt para incluir las librerías de
tiempos.

#include <cassert>
#include <ctime>
#include <cstdio>
#include <cstring> //Para usar memset
#include <iostream>
#include <stdint.h> // Para usar uint64_t

  class Clock
  {
  private:
    timespec _start;
    timespec _stop;
    bool _isStarted;
```

```
    /* Nombre: stop
     * Objetivo: pone el tiempo actual en la variable _stop, pero continua
corriendo el tiempo
     * Parámetros de entrada: ninguno
     * Parámetros de salida: ninguno
     */
    void stop ()
    {
      assert (_isStarted);
      clock_gettime (CLOCK_REALTIME, &_stop);
      //_isStarted=false;
    }

  public:
    Clock ()
    {
      memset(&_start,0,sizeof(timespec));
      memset(&_stop,0,sizeof(timespec));
      _isStarted=false;
    }

    /* Nombre: start
     * Objetivo: inicia el reloj
     * Parámetros de entrada: ninguno
     * Parámetros de salida: ninguno
     */
    void start ()
    {
      assert (!isStarted());
      clock_gettime (CLOCK_REALTIME, &_start);
      _isStarted=true;
    }

    /* Nombre: restart
     * Objetivo: reinicia el reloj
     * Parámetros de entrada: ninguno
     * Parámetros de salida: ninguno
     */
    void restart ()
    {
      clock_gettime (CLOCK_REALTIME, &_start);
      _isStarted=true;
    }

    /* Nombre: isStarted
     * Objetivo: comprueba si está el reloj iniciado
     * Parámetros de entrada: ninguno
     * Parámetros de salida: true si el reloj esta iniciado
     */
    bool isStarted() const
    {
      return _isStarted;
    }

    /* Nombre: elapsed
     * Objetivo: devuelve el tiempo transcurrido desde el inicio, hasta la
llamada a esta función.
     * Parámetros de entrada: ninguno
```

```
     * Parámetros de salida: true si el reloj esta iniciado
     */
    uint64_t elapsed()
    {
      assert (_isStarted);
      stop();
      uint64_t startT = (uint64_t)_start.tv_sec * 1000000LL +
(uint64_t)_start.tv_nsec / 1000LL;
      uint64_t stopT = (uint64_t)_stop.tv_sec * 1000000LL +
(uint64_t)_stop.tv_nsec / 1000LL;
      return stopT-startT;
    }
  };

#endif
```

3.9 main.cpp

Fichero que se encarga de ejecutar la ventana principal e iniciar el programa.

```
/*
 * Proyecto: Sistema Informático para el Seguimiento de Puntero Láser y su
Representación Virtual mediante Visión Artificial
 * Fichero: main.cpp
 * Descripción: fichero que se encarga de ejecutar la ventana principal e iniciar
el programa
 * Autor: Rafael Ulises Baena Herruzo (i32baher@uco.es)
 */

#include <stdio.h>
#include <iostream>
#include <unistd.h>
#include "opencv2/highgui.hpp"
#include "opencv2/imgproc.hpp"
#include "tracker.hpp"
#include "grabacion.hpp"

#include "ventanaprincipal.h"
#include <QApplication>

using namespace std;
using namespace cv;

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    VentanaPrincipal w;
    w.show();

    return a.exec();
}
```

3.10 ventanaprincipal.ui

Este fichero contiene el código *xml*, el cual define todos los componentes gráficos de la interfaz de usuario de la aplicación.

```
    <?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>VentanaPrincipal</class>
 <widget class="QMainWindow" name="VentanaPrincipal">
  <property name="enabled">
   <bool>true</bool>
  </property>
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>800</width>
    <height>480</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Laser Press</string>
  </property>
  <property name="styleSheet">
   <string notr="true">#VentanaPrincipal{
background-color: rgb(46, 52, 54);}</string>
  </property>
  <property name="animated">
   <bool>true</bool>
  </property>
  <widget class="QWidget" name="centralwidget">
   <property name="layoutDirection">
    <enum>Qt::LeftToRight</enum>
   </property>
   <layout class="QVBoxLayout" name="verticalLayout_2">
    <item>
     <widget class="QTabWidget" name="tabWidget">
      <property name="enabled">
       <bool>true</bool>
      </property>
      <property name="autoFillBackground">
       <bool>false</bool>
      </property>
      <property name="currentIndex">
       <number>0</number>
      </property>
      <widget class="QWidget" name="tab">
       <attribute name="title">
        <string>Captura</string>
       </attribute>
       <widget class="QGroupBox" name="groupBox">
        <property name="geometry">
         <rect>
          <x>10</x>
          <y>20</y>
```

```
          <width>461</width>
          <height>171</height>
         </rect>
        </property>
        <property name="font">
         <font>
          <pointsize>10</pointsize>
          <weight>75</weight>
          <bold>true</bold>
         </font>
        </property>
        <property name="title">
         <string>Seleccione la ruta de captura</string>
        </property>
        <widget class="QPushButton" name="boton_buscarCaptura">
         <property name="geometry">
          <rect>
           <x>349</x>
           <y>40</y>
           <width>81</width>
           <height>25</height>
          </rect>
         </property>
         <property name="toolTip">
          <string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;Al pulsar sobre
este botón podrá seleccionar un archivo de vídeo. Dicho archivo de vídeo, debe
tener un formato soportado y deberá corresponder con una presentación grabada
mediante cualquier medio.&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
         </property>
         <property name="whatsThis">
          <string/>
         </property>
         <property name="text">
          <string>Buscar</string>
         </property>
        </widget>
        <widget class="QSpinBox" name="camaraID">
         <property name="geometry">
          <rect>
           <x>350</x>
           <y>120</y>
           <width>81</width>
           <height>41</height>
          </rect>
         </property>
         <property name="toolTip">
          <string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;Seleccione el
identificador de su cámara, tenga en cuenta que en un caso general, la cámara
principal del ordenador tiene un ID igual a 0; sin embargo, si tiene más cámaras
conectadas a su ordenador el ID puede variar, dependiendo de que cámara desee
seleccionar.&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
         </property>
        </widget>
        <widget class="QCheckBox" name="selecArchivo">
         <property name="geometry">
          <rect>
           <x>60</x>
```

```
          <y>40</y>
          <width>231</width>
          <height>23</height>
         </rect>
        </property>
        <property name="text">
         <string>Seleccionar la ruta del archivo :</string>
        </property>
       </widget>
       <widget class="QCheckBox" name="selecCamara">
        <property name="geometry">
         <rect>
          <x>60</x>
          <y>130</y>
          <width>221</width>
          <height>23</height>
         </rect>
        </property>
        <property name="text">
         <string>Seleccionar ID de la cámara :</string>
        </property>
       </widget>
       <widget class="QLabel" name="label">
        <property name="geometry">
         <rect>
          <x>0</x>
          <y>30</y>
          <width>51</width>
          <height>51</height>
         </rect>
        </property>
        <property name="contextMenuPolicy">
         <enum>Qt::DefaultContextMenu</enum>
        </property>
        <property name="text">
         <string/>
        </property>
        <property name="pixmap">
         <pixmap resource="imagenes.qrc">:/iconos/icons/archivo.png</pixmap>
        </property>
        <property name="scaledContents">
         <bool>true</bool>
        </property>
       </widget>
       <widget class="QLabel" name="label_2">
        <property name="geometry">
         <rect>
          <x>10</x>
          <y>120</y>
          <width>41</width>
          <height>41</height>
         </rect>
        </property>
        <property name="text">
         <string/>
        </property>
        <property name="pixmap">
```

```
        <pixmap resource="imagenes.qrc">:/iconos/icons/camara.png</pixmap>
       </property>
       <property name="scaledContents">
        <bool>true</bool>
       </property>
      </widget>
      <widget class="QLabel" name="label_16">
       <property name="geometry">
        <rect>
         <x>60</x>
         <y>80</y>
         <width>54</width>
         <height>21</height>
        </rect>
       </property>
       <property name="text">
        <string>Ruta:</string>
       </property>
      </widget>
      <widget class="QLabel" name="ruta_archivo">
       <property name="geometry">
        <rect>
         <x>100</x>
         <y>80</y>
         <width>351</width>
         <height>20</height>
        </rect>
       </property>
       <property name="text">
        <string/>
       </property>
      </widget>
     </widget>
     <widget class="QLabel" name="label_3">
      <property name="geometry">
       <rect>
        <x>560</x>
        <y>240</y>
        <width>131</width>
        <height>121</height>
       </rect>
      </property>
      <property name="text">
       <string/>
      </property>
      <property name="pixmap">
       <pixmap resource="imagenes.qrc">:/iconos/icons/play.png</pixmap>
      </property>
      <property name="scaledContents">
       <bool>true</bool>
      </property>
     </widget>
     <widget class="QPushButton" name="boton_iniciarCaptura">
      <property name="geometry">
       <rect>
        <x>540</x>
```

```
          <y>350</y>
          <width>181</width>
          <height>31</height>
         </rect>
        </property>
        <property name="font">
         <font>
          <pointsize>12</pointsize>
          <weight>75</weight>
          <bold>true</bold>
         </font>
        </property>
        <property name="toolTip">
         <string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;Iniciará la
captura de la presentación mediante el medio seleccionado (fichero o cámara).
También, por otra parte, comenzará la grabación del contenido que se esté
mostrando en el proyector y el audio del
entorno.&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
        </property>
        <property name="text">
         <string>Iniciar captura</string>
        </property>
       </widget>
       <widget class="QGroupBox" name="groupBox_2">
        <property name="geometry">
         <rect>
          <x>10</x>
          <y>210</y>
          <width>461</width>
          <height>181</height>
         </rect>
        </property>
        <property name="font">
         <font>
          <pointsize>10</pointsize>
          <weight>75</weight>
          <bold>true</bold>
         </font>
        </property>
        <property name="title">
         <string>Seleccione su dispositivo de entrada de sonido</string>
        </property>
        <widget class="QScrollArea" name="scrollArea">
         <property name="geometry">
          <rect>
           <x>10</x>
           <y>30</y>
           <width>441</width>
           <height>91</height>
          </rect>
         </property>
         <property name="widgetResizable">
          <bool>true</bool>
         </property>
         <widget class="QWidget" name="scrollAreaWidgetContents">
          <property name="geometry">
           <rect>
            <x>0</x>
```

```
          <y>0</y>
          <width>439</width>
          <height>89</height>
         </rect>
        </property>
        <layout class="QVBoxLayout" name="verticalLayout">
         <item>
          <widget class="QLabel" name="disp_sonido">
           <property name="layoutDirection">
            <enum>Qt::LeftToRight</enum>
           </property>
           <property name="autoFillBackground">
            <bool>false</bool>
           </property>
           <property name="text">
            <string/>
           </property>
          </widget>
         </item>
        </layout>
       </widget>
      </widget>
      <widget class="QLabel" name="label_4">
       <property name="geometry">
        <rect>
         <x>10</x>
         <y>130</y>
         <width>41</width>
         <height>41</height>
        </rect>
       </property>
       <property name="text">
        <string/>
       </property>
       <property name="pixmap">
        <pixmap resource="imagenes.qrc">:/iconos/icons/sonido.png</pixmap>
       </property>
       <property name="scaledContents">
        <bool>true</bool>
       </property>
      </widget>
      <widget class="QLabel" name="label_5">
       <property name="geometry">
        <rect>
         <x>60</x>
         <y>140</y>
         <width>211</width>
         <height>17</height>
        </rect>
       </property>
       <property name="text">
        <string>Seleccionar ID del dispositivo:</string>
       </property>
      </widget>
      <widget class="QSpinBox" name="sonidoID">
       <property name="geometry">
```

```
          <rect>
           <x>350</x>
           <y>130</y>
           <width>81</width>
           <height>41</height>
          </rect>
         </property>
         <property name="toolTip">
          <string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;Seleccione el ID
de su dispositivo de entrada de audio. Para saber cuál es dicho ID, en la caja de
arriba tiene información sobre los dispositivos de audio de su
ordenador.&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
         </property>
        </widget>
       </widget>
       <widget class="QGroupBox" name="groupBox_5">
        <property name="geometry">
         <rect>
          <x>480</x>
          <y>20</y>
          <width>291</width>
          <height>171</height>
         </rect>
        </property>
        <property name="font">
         <font>
          <pointsize>10</pointsize>
          <weight>75</weight>
          <bold>true</bold>
         </font>
        </property>
        <property name="title">
         <string>Previsualización</string>
        </property>
        <widget class="QLabel" name="previsualizacion">
         <property name="geometry">
          <rect>
           <x>0</x>
           <y>20</y>
           <width>291</width>
           <height>151</height>
          </rect>
         </property>
         <property name="text">
          <string/>
         </property>
        </widget>
       </widget>
       <widget class="QPushButton" name="boton_previsualizar">
        <property name="geometry">
         <rect>
          <x>550</x>
          <y>190</y>
          <width>171</width>
          <height>25</height>
         </rect>
        </property>
        <property name="font">
```

```
      <font>
       <pointsize>10</pointsize>
       <weight>75</weight>
       <bold>true</bold>
      </font>
     </property>
     <property name="text">
      <string>Previsualizar / Refrescar</string>
     </property>
    </widget>
   </widget>
   <widget class="QWidget" name="tab_4">
    <attribute name="title">
     <string>Montaje</string>
    </attribute>
    <widget class="QGroupBox" name="groupBox_3">
     <property name="geometry">
      <rect>
       <x>10</x>
       <y>20</y>
       <width>461</width>
       <height>201</height>
      </rect>
     </property>
     <property name="font">
      <font>
       <pointsize>10</pointsize>
       <weight>75</weight>
       <bold>true</bold>
      </font>
     </property>
     <property name="title">
      <string>Seleccione la ruta de destino</string>
     </property>
     <widget class="QLabel" name="label_6">
      <property name="geometry">
       <rect>
        <x>70</x>
        <y>40</y>
        <width>221</width>
        <height>21</height>
       </rect>
      </property>
      <property name="text">
       <string>Seleccione la ruta del directorio :</string>
      </property>
     </widget>
     <widget class="QPushButton" name="boton_buscarDirMontaje">
      <property name="geometry">
       <rect>
        <x>349</x>
        <y>40</y>
        <width>81</width>
        <height>25</height>
       </rect>
      </property>
      <property name="text">
```

```
          <string>Buscar</string>
         </property>
        </widget>
        <widget class="QLabel" name="label_7">
         <property name="geometry">
          <rect>
           <x>10</x>
           <y>30</y>
           <width>41</width>
           <height>41</height>
          </rect>
         </property>
         <property name="text">
          <string/>
         </property>
         <property name="pixmap">
          <pixmap resource="imagenes.qrc">:/iconos/icons/carpeta.png</pixmap>
         </property>
         <property name="scaledContents">
          <bool>true</bool>
         </property>
        </widget>
        <widget class="QLabel" name="label_8">
         <property name="geometry">
          <rect>
           <x>10</x>
           <y>90</y>
           <width>41</width>
           <height>51</height>
          </rect>
         </property>
         <property name="text">
          <string/>
         </property>
         <property name="pixmap">
          <pixmap resource="imagenes.qrc">:/iconos/icons/lapiz.png</pixmap>
         </property>
         <property name="scaledContents">
          <bool>true</bool>
         </property>
        </widget>
        <widget class="QLabel" name="label_9">
         <property name="geometry">
          <rect>
           <x>70</x>
           <y>100</y>
           <width>221</width>
           <height>31</height>
          </rect>
         </property>
         <property name="text">
          <string>Introduza el nombre del archivo :</string>
         </property>
        </widget>
        <widget class="QLineEdit" name="lineEdit">
         <property name="geometry">
          <rect>
```

```
          <x>290</x>
          <y>100</y>
          <width>121</width>
          <height>25</height>
         </rect>
        </property>
        <property name="text">
         <string/>
        </property>
       </widget>
       <widget class="QLabel" name="label_10">
        <property name="geometry">
         <rect>
          <x>420</x>
          <y>100</y>
          <width>31</width>
          <height>20</height>
         </rect>
        </property>
        <property name="text">
         <string>.mp4</string>
        </property>
       </widget>
       <widget class="QLabel" name="ruta_dir">
        <property name="geometry">
         <rect>
          <x>70</x>
          <y>166</y>
          <width>381</width>
          <height>21</height>
         </rect>
        </property>
        <property name="font">
         <font>
          <weight>50</weight>
          <italic>true</italic>
          <bold>false</bold>
         </font>
        </property>
        <property name="text">
         <string/>
        </property>
       </widget>
       <widget class="QLabel" name="label_11">
        <property name="geometry">
         <rect>
          <x>20</x>
          <y>166</y>
          <width>54</width>
          <height>21</height>
         </rect>
        </property>
        <property name="text">
         <string>Ruta:</string>
        </property>
       </widget>
      </widget>
```

```
<widget class="QGroupBox" name="groupBox_4">
 <property name="geometry">
  <rect>
   <x>10</x>
   <y>240</y>
   <width>461</width>
   <height>151</height>
  </rect>
 </property>
 <property name="font">
  <font>
   <pointsize>10</pointsize>
   <weight>75</weight>
   <bold>true</bold>
  </font>
 </property>
 <property name="title">
  <string>Progreso</string>
 </property>
 <widget class="QProgressBar" name="progreso_montaje">
  <property name="geometry">
   <rect>
    <x>30</x>
    <y>90</y>
    <width>401</width>
    <height>23</height>
   </rect>
  </property>
  <property name="value">
   <number>24</number>
  </property>
 </widget>
 <widget class="QLabel" name="label_13">
  <property name="geometry">
   <rect>
    <x>30</x>
    <y>36</y>
    <width>121</width>
    <height>21</height>
   </rect>
  </property>
  <property name="text">
   <string>Procesando frame </string>
  </property>
 </widget>
 <widget class="QLabel" name="frame_actual">
  <property name="geometry">
   <rect>
    <x>160</x>
    <y>37</y>
    <width>54</width>
    <height>20</height>
   </rect>
  </property>
  <property name="font">
   <font>
    <pointsize>12</pointsize>
   </font>
```

```
      </property>
      <property name="text">
       <string>0</string>
      </property>
      <property name="alignment">
       <set>Qt::AlignCenter</set>
      </property>
     </widget>
     <widget class="QLabel" name="label_14">
      <property name="geometry">
       <rect>
        <x>230</x>
        <y>36</y>
        <width>21</width>
        <height>21</height>
       </rect>
      </property>
      <property name="text">
       <string>de</string>
      </property>
     </widget>
     <widget class="QLabel" name="nFrames">
      <property name="geometry">
       <rect>
        <x>260</x>
        <y>40</y>
        <width>61</width>
        <height>16</height>
       </rect>
      </property>
      <property name="font">
       <font>
        <pointsize>12</pointsize>
       </font>
      </property>
      <property name="text">
       <string>0</string>
      </property>
      <property name="alignment">
       <set>Qt::AlignCenter</set>
      </property>
     </widget>
     <widget class="QLabel" name="label_15">
      <property name="geometry">
       <rect>
        <x>330</x>
        <y>37</y>
        <width>54</width>
        <height>20</height>
       </rect>
      </property>
      <property name="text">
       <string>frames</string>
      </property>
     </widget>
    </widget>
```

```
        <widget class="QPushButton" name="boton_iniciarMontaje">
         <property name="geometry">
          <rect>
           <x>530</x>
           <y>270</y>
           <width>191</width>
           <height>41</height>
          </rect>
         </property>
         <property name="font">
          <font>
           <pointsize>12</pointsize>
           <weight>75</weight>
           <bold>true</bold>
          </font>
         </property>
         <property name="toolTip">
          <string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;Iniciará la
captura de la presentación mediante el medio seleccionado (fichero o cámara).
También, por otra parte, comenzará la grabación del contenido que se esté
mostrando en el proyector y el audio del
entorno.&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
         </property>
         <property name="text">
          <string>Iniciar montaje</string>
         </property>
        </widget>
        <widget class="QLabel" name="label_12">
         <property name="geometry">
          <rect>
           <x>540</x>
           <y>110</y>
           <width>161</width>
           <height>141</height>
          </rect>
         </property>
         <property name="text">
          <string/>
         </property>
         <property name="pixmap">
          <pixmap resource="imagenes.qrc">:/iconos/icons/montaje.png</pixmap>
         </property>
         <property name="scaledContents">
          <bool>true</bool>
         </property>
        </widget>
       </widget>
       <widget class="QWidget" name="tab_2">
        <attribute name="title">
         <string>Ajustes</string>
        </attribute>
        <widget class="QGroupBox" name="groupBox_6">
         <property name="geometry">
          <rect>
           <x>10</x>
           <y>10</y>
           <width>461</width>
```

```
        <height>191</height>
       </rect>
      </property>
      <property name="font">
       <font>
        <pointsize>10</pointsize>
        <weight>75</weight>
        <bold>true</bold>
       </font>
      </property>
      <property name="title">
       <string>Seleccionar la relación de aspecto de la presentación</string>
      </property>
      <widget class="QLabel" name="label_17">
       <property name="geometry">
        <rect>
         <x>10</x>
         <y>30</y>
         <width>41</width>
         <height>51</height>
        </rect>
       </property>
       <property name="text">
        <string/>
       </property>
       <property name="pixmap">
        <pixmap resource="imagenes.qrc">:/iconos/icons/proyector.png</pixmap>
       </property>
       <property name="scaledContents">
        <bool>true</bool>
       </property>
      </widget>
      <widget class="QCheckBox" name="formato_automatico">
       <property name="geometry">
        <rect>
         <x>70</x>
         <y>50</y>
         <width>361</width>
         <height>23</height>
        </rect>
       </property>
       <property name="text">
        <string>Detectección automática.</string>
       </property>
      </widget>
      <widget class="QLabel" name="label_18">
       <property name="geometry">
        <rect>
         <x>10</x>
         <y>90</y>
         <width>51</width>
         <height>51</height>
        </rect>
       </property>
       <property name="text">
        <string/>
```

```
        </property>
        <property name="pixmap">
         <pixmap resource="imagenes.qrc">:/iconos/icons/4-3.png</pixmap>
        </property>
        <property name="scaledContents">
         <bool>true</bool>
        </property>
       </widget>
       <widget class="QCheckBox" name="formato_4_3">
        <property name="geometry">
         <rect>
          <x>70</x>
          <y>100</y>
          <width>181</width>
          <height>23</height>
         </rect>
        </property>
        <property name="text">
         <string>Relación de aspecto 4:3</string>
        </property>
       </widget>
       <widget class="QLabel" name="label_19">
        <property name="geometry">
         <rect>
          <x>10</x>
          <y>140</y>
          <width>51</width>
          <height>51</height>
         </rect>
        </property>
        <property name="text">
         <string/>
        </property>
        <property name="pixmap">
         <pixmap resource="imagenes.qrc">:/iconos/icons/16-9.png</pixmap>
        </property>
        <property name="scaledContents">
         <bool>true</bool>
        </property>
       </widget>
       <widget class="QCheckBox" name="formato_16_9">
        <property name="geometry">
         <rect>
          <x>70</x>
          <y>150</y>
          <width>201</width>
          <height>23</height>
         </rect>
        </property>
        <property name="text">
         <string>Relación de aspecto 16:9</string>
        </property>
       </widget>
      </widget>
      <widget class="QGroupBox" name="groupBox_7">
       <property name="geometry">
        <rect>
```

```
          <x>480</x>
          <y>10</y>
          <width>291</width>
          <height>191</height>
         </rect>
        </property>
        <property name="font">
         <font>
          <pointsize>10</pointsize>
          <weight>75</weight>
          <bold>true</bold>
         </font>
        </property>
        <property name="title">
         <string>Usar pantalla extendida</string>
        </property>
        <widget class="QLabel" name="label_20">
         <property name="geometry">
          <rect>
           <x>70</x>
           <y>40</y>
           <width>211</width>
           <height>31</height>
          </rect>
         </property>
         <property name="text">
          <string>Introduza el ancho y alto
de su pantalla principal :</string>
         </property>
        </widget>
        <widget class="QLabel" name="label_21">
         <property name="geometry">
          <rect>
           <x>10</x>
           <y>90</y>
           <width>54</width>
           <height>17</height>
          </rect>
         </property>
         <property name="text">
          <string>Ancho :</string>
         </property>
        </widget>
        <widget class="QSpinBox" name="ext_ancho">
         <property name="geometry">
          <rect>
           <x>90</x>
           <y>85</y>
           <width>111</width>
           <height>31</height>
          </rect>
         </property>
         <property name="maximum">
          <number>10000</number>
         </property>
        </widget>
        <widget class="QSpinBox" name="ext_alto">
```

```
        <property name="geometry">
         <rect>
          <x>90</x>
          <y>135</y>
          <width>111</width>
          <height>31</height>
         </rect>
        </property>
        <property name="maximum">
         <number>10000</number>
        </property>
       </widget>
       <widget class="QLabel" name="label_22">
        <property name="geometry">
         <rect>
          <x>10</x>
          <y>140</y>
          <width>54</width>
          <height>17</height>
         </rect>
        </property>
        <property name="text">
         <string>Alto :</string>
        </property>
       </widget>
       <widget class="QLabel" name="label_23">
        <property name="geometry">
         <rect>
          <x>10</x>
          <y>30</y>
          <width>51</width>
          <height>51</height>
         </rect>
        </property>
        <property name="text">
         <string/>
        </property>
        <property name="pixmap">
         <pixmap resource="imagenes.qrc">:/iconos/icons/ancho-alto.png</pixmap>
        </property>
        <property name="scaledContents">
         <bool>true</bool>
        </property>
       </widget>
      </widget>
      <widget class="QGroupBox" name="groupBox_8">
       <property name="geometry">
        <rect>
         <x>10</x>
         <y>210</y>
         <width>761</width>
         <height>181</height>
        </rect>
       </property>
       <property name="font">
        <font>
         <pointsize>10</pointsize>
```

```
          <weight>75</weight>
          <bold>true</bold>
         </font>
        </property>
        <property name="title">
         <string>Ajustes para la realización de un montaje offline</string>
        </property>
        <widget class="Line" name="line">
         <property name="geometry">
          <rect>
           <x>430</x>
           <y>30</y>
           <width>20</width>
           <height>131</height>
          </rect>
         </property>
         <property name="orientation">
          <enum>Qt::Vertical</enum>
         </property>
        </widget>
        <widget class="QLabel" name="label_24">
         <property name="geometry">
          <rect>
           <x>10</x>
           <y>30</y>
           <width>401</width>
           <height>17</height>
          </rect>
         </property>
         <property name="text">
          <string>Ajustes de captura, seleccione donde guardar la captura
:</string>
         </property>
        </widget>
        <widget class="QLabel" name="label_25">
         <property name="geometry">
          <rect>
           <x>460</x>
           <y>30</y>
           <width>301</width>
           <height>17</height>
          </rect>
         </property>
         <property name="text">
          <string>Ajustes de montaje, seleccione los archivos :</string>
         </property>
        </widget>
        <widget class="QLabel" name="label_28">
         <property name="geometry">
          <rect>
           <x>10</x>
           <y>110</y>
           <width>41</width>
           <height>31</height>
          </rect>
         </property>
         <property name="text">
```

```
          <string/>
         </property>
         <property name="pixmap">
          <pixmap resource="imagenes.qrc">:/iconos/icons/lapiz.png</pixmap>
         </property>
         <property name="scaledContents">
          <bool>true</bool>
         </property>
        </widget>
        <widget class="QPushButton" name="boton_mo_rutadir">
         <property name="geometry">
          <rect>
           <x>290</x>
           <y>70</y>
           <width>81</width>
           <height>25</height>
          </rect>
         </property>
         <property name="text">
          <string>Buscar</string>
         </property>
        </widget>
        <widget class="QLabel" name="label_26">
         <property name="geometry">
          <rect>
           <x>10</x>
           <y>60</y>
           <width>41</width>
           <height>31</height>
          </rect>
         </property>
         <property name="text">
          <string/>
         </property>
         <property name="pixmap">
          <pixmap resource="imagenes.qrc">:/iconos/icons/carpeta.png</pixmap>
         </property>
         <property name="scaledContents">
          <bool>true</bool>
         </property>
        </widget>
        <widget class="QLabel" name="label_29">
         <property name="geometry">
          <rect>
           <x>70</x>
           <y>110</y>
           <width>211</width>
           <height>31</height>
          </rect>
         </property>
         <property name="text">
          <string>Introduza el nombre del archivo :</string>
         </property>
        </widget>
        <widget class="QLineEdit" name="lineEdit_mo_archivo">
         <property name="geometry">
          <rect>
```

```
           <x>300</x>
           <y>110</y>
           <width>71</width>
           <height>25</height>
          </rect>
         </property>
         <property name="text">
          <string/>
         </property>
        </widget>
        <widget class="QLabel" name="label_27">
         <property name="geometry">
          <rect>
           <x>70</x>
           <y>70</y>
           <width>221</width>
           <height>21</height>
          </rect>
         </property>
         <property name="text">
          <string>Seleccione la ruta del directorio :</string>
         </property>
        </widget>
        <widget class="QLabel" name="label_30">
         <property name="geometry">
          <rect>
           <x>380</x>
           <y>110</y>
           <width>54</width>
           <height>31</height>
          </rect>
         </property>
         <property name="text">
          <string>.mp4
.mp3</string>
         </property>
        </widget>
        <widget class="QLabel" name="label_31">
         <property name="geometry">
          <rect>
           <x>70</x>
           <y>146</y>
           <width>201</width>
           <height>31</height>
          </rect>
         </property>
         <property name="text">
          <string>Archivo de captura :</string>
         </property>
        </widget>
        <widget class="QLineEdit" name="lineEdit_mo_captura">
         <property name="geometry">
          <rect>
           <x>300</x>
           <y>150</y>
           <width>71</width>
           <height>25</height>
```

```
          </rect>
         </property>
         <property name="text">
          <string/>
         </property>
        </widget>
        <widget class="QLabel" name="label_32">
         <property name="geometry">
          <rect>
           <x>380</x>
           <y>150</y>
           <width>54</width>
           <height>17</height>
          </rect>
         </property>
         <property name="text">
          <string>.cap</string>
         </property>
        </widget>
        <widget class="QLabel" name="label_33">
         <property name="geometry">
          <rect>
           <x>460</x>
           <y>60</y>
           <width>201</width>
           <height>21</height>
          </rect>
         </property>
         <property name="text">
          <string>Seleccione el directorio :</string>
         </property>
        </widget>
        <widget class="QLabel" name="label_34">
         <property name="geometry">
          <rect>
           <x>460</x>
           <y>90</y>
           <width>141</width>
           <height>41</height>
          </rect>
         </property>
         <property name="text">
          <string>Introduzca el nombre
del archivo:</string>
         </property>
        </widget>
        <widget class="QLabel" name="label_35">
         <property name="geometry">
          <rect>
           <x>460</x>
           <y>140</y>
           <width>201</width>
           <height>31</height>
          </rect>
         </property>
         <property name="text">
          <string>Archivo de captura :</string>
         </property>
```

```
        </widget>
        <widget class="QPushButton" name="pushButton_mo_rAudio">
         <property name="geometry">
          <rect>
           <x>660</x>
           <y>60</y>
           <width>81</width>
           <height>25</height>
          </rect>
         </property>
         <property name="text">
          <string>Buscar</string>
         </property>
        </widget>
        <widget class="QLabel" name="label_36">
         <property name="geometry">
          <rect>
           <x>10</x>
           <y>150</y>
           <width>41</width>
           <height>31</height>
          </rect>
         </property>
         <property name="text">
          <string/>
         </property>
         <property name="pixmap">
          <pixmap resource="imagenes.qrc">:/iconos/icons/camara.png</pixmap>
         </property>
         <property name="scaledContents">
          <bool>true</bool>
         </property>
        </widget>
        <widget class="QLabel" name="label_37">
         <property name="geometry">
          <rect>
           <x>710</x>
           <y>100</y>
           <width>54</width>
           <height>31</height>
          </rect>
         </property>
         <property name="text">
          <string>.mp4
.mp3</string>
         </property>
        </widget>
        <widget class="QLineEdit" name="lineEdit_mo_archivo2">
         <property name="geometry">
          <rect>
           <x>610</x>
           <y>100</y>
           <width>91</width>
           <height>25</height>
          </rect>
         </property>
```

```
        </widget>
        <widget class="QPushButton" name="pushButton_mo_rCap">
         <property name="geometry">
          <rect>
           <x>660</x>
           <y>140</y>
           <width>81</width>
           <height>25</height>
          </rect>
         </property>
         <property name="text">
          <string>Buscar</string>
         </property>
        </widget>
       </widget>
      </widget>
      <widget class="QWidget" name="tab_3">
       <attribute name="title">
        <string>Información</string>
       </attribute>
       <widget class="QGroupBox" name="groupBox_9">
        <property name="geometry">
         <rect>
          <x>70</x>
          <y>20</y>
          <width>641</width>
          <height>321</height>
         </rect>
        </property>
        <property name="title">
         <string/>
        </property>
        <widget class="QLabel" name="label_39">
         <property name="geometry">
          <rect>
           <x>210</x>
           <y>140</y>
           <width>341</width>
           <height>41</height>
          </rect>
         </property>
         <property name="font">
          <font>
           <family>Ubuntu Condensed</family>
           <pointsize>12</pointsize>
           <weight>75</weight>
           <bold>true</bold>
          </font>
         </property>
         <property name="text">
          <string>Autor: Rafael Ulises Baena Herruzo</string>
         </property>
        </widget>
        <widget class="QLabel" name="label_38">
         <property name="geometry">
          <rect>
           <x>0</x>
```

```
           <y>30</y>
           <width>641</width>
           <height>91</height>
          </rect>
         </property>
         <property name="font">
          <font>
           <family>Ubuntu Condensed</family>
           <pointsize>14</pointsize>
           <weight>75</weight>
           <italic>true</italic>
           <bold>true</bold>
          </font>
         </property>
         <property name="text">
          <string>Laser Press: SISTEMA INFORMÁTICO PARA EL SEGUIMIENTO
 DE PUNTERO LÁSER Y SU REPRESENTACIÓN
VIRTUAL MEDIANTE VISIÓN ARTIFICIAL</string>
         </property>
         <property name="alignment">
          <set>Qt::AlignCenter</set>
         </property>
        </widget>
        <widget class="QLabel" name="label_40">
         <property name="geometry">
          <rect>
           <x>190</x>
           <y>190</y>
           <width>341</width>
           <height>41</height>
          </rect>
         </property>
         <property name="font">
          <font>
           <family>Ubuntu Condensed</family>
           <pointsize>11</pointsize>
           <weight>75</weight>
           <bold>true</bold>
          </font>
         </property>
         <property name="text">
          <string>Fecha de última compilación: Agosto de 2017</string>
         </property>
        </widget>
        <widget class="QLabel" name="label_41">
         <property name="geometry">
          <rect>
           <x>180</x>
           <y>240</y>
           <width>381</width>
           <height>41</height>
          </rect>
         </property>
         <property name="font">
          <font>
           <family>Ubuntu Condensed</family>
           <pointsize>11</pointsize>
```

```
          <weight>75</weight>
          <bold>true</bold>
         </font>
        </property>
        <property name="text">
         <string>Correo electrónico de contacto: i32baher@uco.es</string>
        </property>
       </widget>
       <widget class="QLabel" name="label_43">
        <property name="geometry">
         <rect>
          <x>10</x>
          <y>270</y>
          <width>151</width>
          <height>51</height>
         </rect>
        </property>
        <property name="text">
         <string/>
        </property>
        <property name="pixmap">
         <pixmap resource="imagenes.qrc">:/iconos/icons/logo.png</pixmap>
        </property>
        <property name="scaledContents">
         <bool>true</bool>
        </property>
       </widget>
      </widget>
      <widget class="QLabel" name="label_42">
       <property name="geometry">
        <rect>
         <x>10</x>
         <y>10</y>
         <width>31</width>
         <height>71</height>
        </rect>
       </property>
       <property name="text">
        <string/>
       </property>
       <property name="pixmap">
        <pixmap resource="imagenes.qrc">:/iconos/icons/info.png</pixmap>
       </property>
       <property name="scaledContents">
        <bool>true</bool>
       </property>
      </widget>
     </widget>
    </widget>
   </item>
  </layout>
 </widget>
 <widget class="QStatusBar" name="statusbar"/>
 <widget class="QMenuBar" name="menubar">
  <property name="geometry">
   <rect>
    <x>0</x>
```

```
     <y>0</y>
     <width>800</width>
     <height>22</height>
    </rect>
   </property>
  </widget>
 </widget>
 <resources>
  <include location="imagenes.qrc"/>
 </resources>
 <connections/>
</ui>
```