МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информатика»

Лабораторная работа №2

по дисциплине

Разработка кроссплатформенных приложений

Выполнил: студент гр. БЭИ2202

Кулешов А. С.

Проверил: зав. кафедры

«Информатика» Халабия Р.Ф.

Москва, 2025 г

**Задание**

— Создать файлы с изображениями кнопок для работы с mp3-плеером, которые будут подключаться к проекту при помощи ресурсного файла.

— Реализовать mp3-плеер при помощи Qt Designer.

— Разработать код программы, позволяющий добавлять, воспроизводить, останавливать треки, а также ставить на паузу воспроизведение и перемещаться между треками.

— Добавить 5 треков и проверить правильность работы написанного программного обеспечения.

**ВЫПОЛНЕНИЕ**

Выполнение заданий представлено на рисунках 1-4.

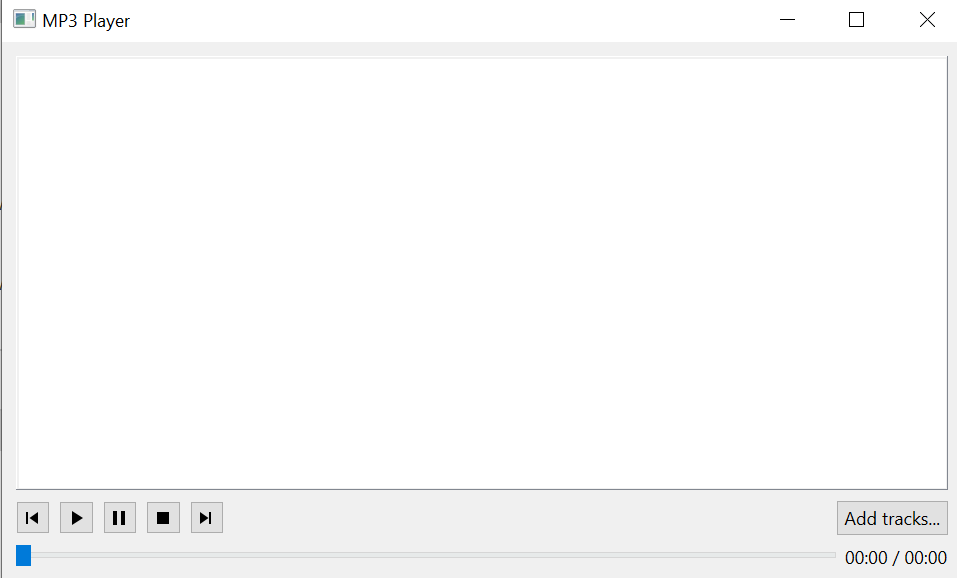
****

Рисунок 1 – Пустой плеер

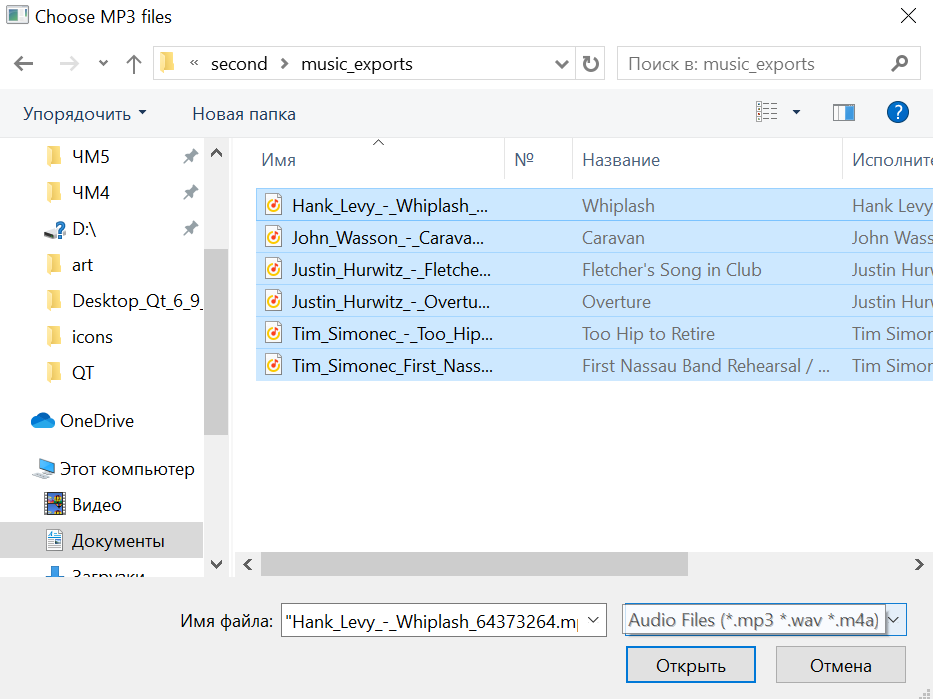


Рисунок 2 – Добавление файлов

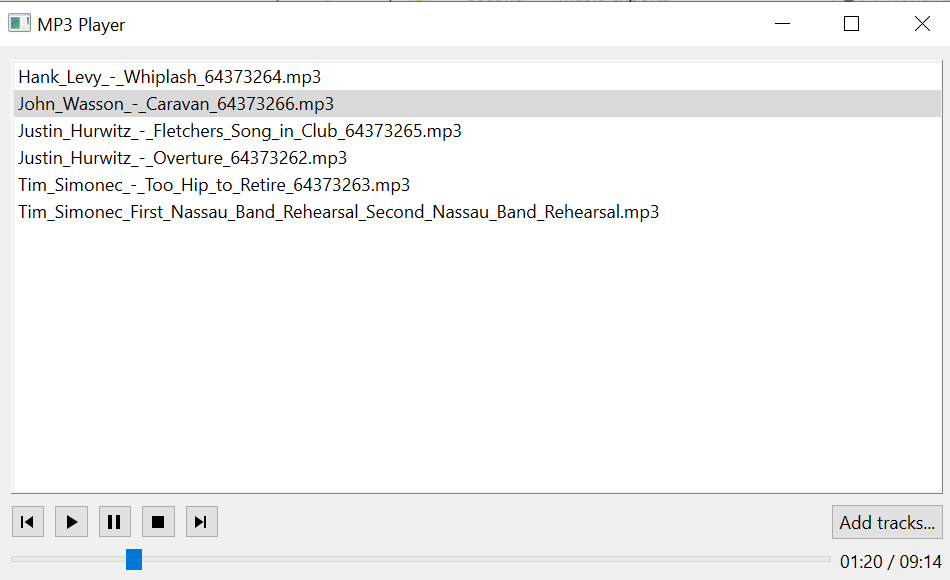
****

Рисунок 3 – Выбранная музыка играет

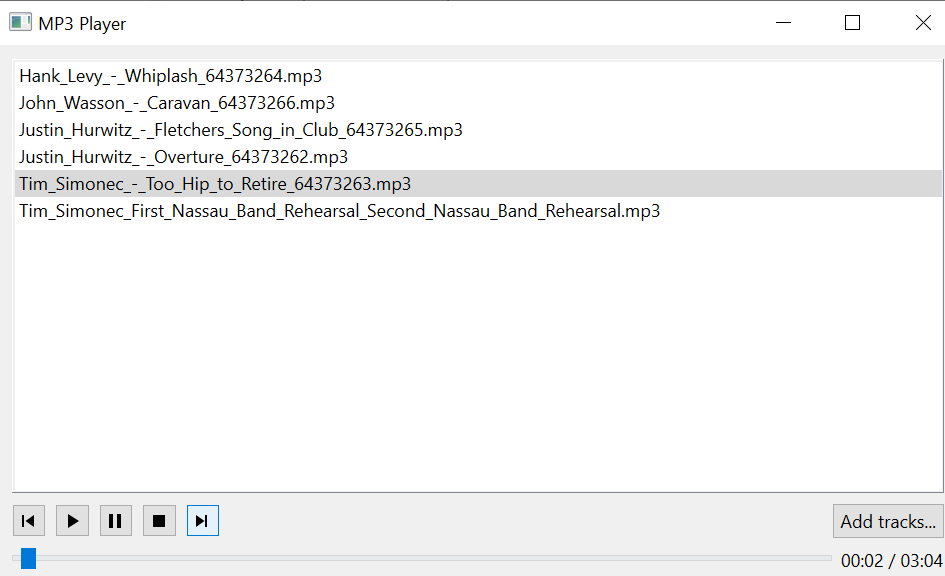
****

Рисунок 4 – Перелистывание треков

**Алгоритм добавления трека**

* Шаг1. Пользователь нажимает кнопку **Add tracks**.
* Шаг2. Открывается диалог выбора файлов.
* Шаг3. Пользователь выбирает один или несколько файлов и подтверждает.
* Шаг4. Для каждого выбранного файла:
  + Шаг4.1. Сохранить полный путь в список trackFiles.
  + Шаг4.2. Добавить название файла (без пути) в виджет списка trackList.
* Шаг5. Если это первый добавленный трек:
  + Шаг5.1. Установить currentIndex = 0.
  + Шаг5.2. Подсветить первую строку в trackList.

**Алгоритм воспроизведения трека**

* Шаг1. Проверить, есть ли треки в списке.
* Шаг2. Если currentIndex == -1, установить currentIndex = 0.
* Шаг3. Вызвать playTrack(currentIndex):
  + Шаг3.1. Установить новый текущий индекс.
  + Шаг3.2. Загрузить трек через setSource(...).
  + Шаг3.3. Запустить воспроизведение play().
  + Шаг3.4. Подсветить текущий трек в списке.
* Шаг4. Обновлять таймер и слайдер по сигналам positionChanged и durationChanged.

**Алгоритм остановки трека**

* Шаг1. Пользователь нажимает кнопку **Stop**.
* Шаг2. Вызвать stop().
* Шаг3. Установить слайдер в начало (0).
* Шаг4. Обновить метку времени на 00:00 / <длительность>.

**Алгоритм паузы**

* Шаг1. Пользователь нажимает кнопку **Pause**.
* Шаг2. Вызвать pause().
* Шаг3. Сохранить текущую позицию.
* Шаг4. При следующем **Play** продолжить с сохранённой позиции.

**Алгоритм перехода к следующему треку**

* Шаг1. Проверить, есть ли треки в списке.
* Шаг2. Вычислить next = (currentIndex + 1) % trackFiles.size().
* Шаг3. Вызвать playTrack(next).

**Алгоритм перехода к предыдущему треку**

* Шаг1. Проверить, есть ли треки в списке.
* Шаг2. Вычислить prev = (currentIndex - 1 + trackFiles.size()) % trackFiles.size().
* Шаг3. Вызвать playTrack(prev).

Листинг 1 — mainwindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QMediaPlayer>

#include <QAudioOutput>

#include <QStringList>

QT\_BEGIN\_NAMESPACE

**namespace** **Ui** { **class** **MainWindow**; }

QT\_END\_NAMESPACE

**class** **MainWindow** : **public** QMainWindow

{

Q\_OBJECT

**public**:

MainWindow(QWidget \*parent = **nullptr**);

~MainWindow();

**private** slots:

**void** on\_addButton\_clicked();

**void** **on\_playButton\_clicked**();

**void** **on\_pauseButton\_clicked**();

**void** **on\_stopButton\_clicked**();

**void** **on\_nextButton\_clicked**();

**void** **on\_prevButton\_clicked**();

**void** **on\_trackList\_doubleClicked**(**const** QModelIndex &index);

**void** **positionChanged**(qint64 pos);

**void** **durationChanged**(qint64 dur);

**void** **on\_positionSlider\_sliderMoved**(**int** position);

**private**:

Ui::MainWindow \*ui;

QMediaPlayer \*player;

QAudioOutput \*audioOutput;

QStringList trackFiles; // список файлов

**int** currentIndex; // индекс текущего трека

**void** **loadDefaultTracks**();

**void** **playTrack**(**int** index);

QString **msToString**(qint64 ms);

};

#endif // MAINWINDOW\_H

Листинг 2 — mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QFileDialog>

#include <QDir>

#include <QFileInfo>

#include <QUrl>

#include <QDebug>

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(**new** Ui::MainWindow)

, player(**new** QMediaPlayer(**this**))

, audioOutput(**new** QAudioOutput(**this**))

, currentIndex(**-1**)

{

ui->setupUi(**this**);

player->setAudioOutput(audioOutput);

audioOutput->setVolume(**1.0**);

// автоподключения работают через on\_\*\_clicked, вручную лишние connect не нужны

connect(player, &QMediaPlayer::positionChanged, **this**, &MainWindow::positionChanged);

connect(player, &QMediaPlayer::durationChanged, **this**, &MainWindow::durationChanged);

connect(ui->positionSlider, &QSlider::sliderMoved, **this**, &MainWindow::on\_positionSlider\_sliderMoved);

ui->positionSlider->setRange(**0**,**0**);

loadDefaultTracks();

}

MainWindow::~MainWindow()

{

**delete** ui;

}

**void** MainWindow::on\_addButton\_clicked()

{

QStringList files = QFileDialog::getOpenFileNames(

**this**, tr("Choose audio files"),

QString(), tr("Audio Files (\*.mp3 \*.wav \*.m4a)")

);

**for** (**const** QString &f : files) {

trackFiles.append(f);

ui->trackList->addItem(QFileInfo(f).fileName());

}

**if** (currentIndex == **-1** && !trackFiles.isEmpty()) {

currentIndex = **0**;

ui->trackList->setCurrentRow(**0**);

}

}

**void** MainWindow::playTrack(**int** index)

{

**if** (index < **0** || index >= trackFiles.size()) **return**;

currentIndex = index;

player->setSource(QUrl::fromLocalFile(trackFiles.at(currentIndex)));

player->play();

ui->trackList->setCurrentRow(currentIndex);

}

**void** MainWindow::on\_playButton\_clicked()

{

**if** (trackFiles.isEmpty()) **return**;

**if** (currentIndex < **0**) currentIndex = **0**;

playTrack(currentIndex);

}

**void** MainWindow::on\_pauseButton\_clicked()

{

player->pause();

}

**void** MainWindow::on\_stopButton\_clicked()

{

player->stop();

}

**void** MainWindow::on\_nextButton\_clicked()

{

**if** (trackFiles.isEmpty()) **return**;

**int** next = (currentIndex + **1**) % trackFiles.size();

playTrack(next);

}

**void** MainWindow::on\_prevButton\_clicked()

{

**if** (trackFiles.isEmpty()) **return**;

**int** prev = (currentIndex - **1** + trackFiles.size()) % trackFiles.size();

playTrack(prev);

}

**void** MainWindow::on\_trackList\_doubleClicked(**const** QModelIndex &index)

{

playTrack(index.row());

}

**void** MainWindow::positionChanged(qint64 pos)

{

ui->positionSlider->setValue(**static\_cast**<**int**>(pos));

QString cur = msToString(pos);

QString total = msToString(player->duration());

ui->timeLabel->setText(cur + " / " + total);

}

**void** MainWindow::durationChanged(qint64 dur)

{

ui->positionSlider->setRange(**0**, **static\_cast**<**int**>(dur));

}

**void** MainWindow::on\_positionSlider\_sliderMoved(**int** position)

{

player->setPosition(position);

}

**void** MainWindow::loadDefaultTracks()

{

QDir d(QDir::currentPath() + "/tracks");

**if** (!d.exists()) **return**;

QStringList list = d.entryList(QStringList() << "\*.mp3" << "\*.wav" << "\*.m4a", QDir::Files);

**for** (**const** QString &f : list) {

QString full = d.filePath(f);

trackFiles.append(full);

ui->trackList->addItem(f);

}

**if** (!trackFiles.isEmpty()) {

currentIndex = **0**;

ui->trackList->setCurrentRow(**0**);

}

}

QString MainWindow::msToString(qint64 ms)

{

**int** s = **static\_cast**<**int**>(ms/**1000**);

**int** minutes = s/**60**;

**int** seconds = s%**60**;

**return** QString::asprintf("%02d:%02d", minutes, seconds);

}

Листинг 3 — main.cpp

#include "mainwindow.h"

#include <QApplication>

**int** **main**(**int** argc, **char** \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

w.show();

**return** a.exec();

}

**Вывод**

Было написано приложение для работы с аудиофайлами, их воспроизведением, паузой, добавлением новых аудиофайлов, переходу к следующим и предыдущим, т.е. аудиоплеер. Сделано это было при помощи Qt Designer, на ЯП C++