СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc152798549)

[1 АНАЛИЗ ИСХОДНЫХ ДАННЫХ 5](#_Toc152798550)

[2 ПРОГРАММНОЕ ПРОЕКТИРОВАНИЕ 6](#_Toc152798551)

[3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ 8](#_Toc152798552)

[3.1 Детальная реализация функциональных частей ПО 8](#_Toc152798553)

[3.2 Сопроводительная документация 16](#_Toc152798554)

[3.3 Анализ ПО 17](#_Toc152798555)

[3.4 Тестирование ПО 17](#_Toc152798556)

[ЗАКЛЮЧЕНИЕ 20](#_Toc152798557)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 21](#_Toc152798558)

[ПРИЛОЖЕНИЕ А (обязательное) Техническое задание 22](#_Toc152798559)

[ПРИЛОЖЕНИЕ Б (обязательное) Диаграмма вариантов использования 25](#_Toc152798560)

[ПРИЛОЖЕНИЕ В (обязательное) Таблица тестирования 26](#_Toc152798561)

# ВВЕДЕНИЕ

В эпоху цифровых технологий и всемирной информатизации возникает неотъемлемая потребность в эффективных и многофункциональных средствах для работы с медиа. Аудио и видео контент играют значительную роль в повседневной жизни современного человека, будь то образование, работа, или развлечения. В этом контексте, курсовой проект, посвященный созданию аудио/видео проигрывателя для операционной системы Mac OS, становится актуальным и востребованным.

Цель данного проекта - разработать приложение, которое не только отвечает требованиям функциональности и надежности, но и обладает высоким уровнем удобства использования и эстетической привлекательности. Проигрыватель должен поддерживать основные аудио и видео форматы, предоставлять пользователям гибкие настройки воспроизведения и иметь интуитивно понятный интерфейс.

Проект реализован с использованием фреймворка Qt, который является одним из ведущих инструментов для разработки кроссплатформенных приложений. Qt предлагает мощные инструменты для создания графического интерфейса, работы с мультимедиа и сетевыми технологиями, что делает его идеальным выбором для реализации данного проекта. Важность проекта также обусловлена растущей популярностью операционной системы Mac OS среди пользователей по всему миру. Создание специализированного аудио/видео проигрывателя для этой платформы позволит удовлетворить специфические потребности пользователей Mac OS, которые ценят не только функциональность, но и качество дизайна программного обеспечения.

Таким образом, данный курсовой проект имеет не только академическую ценность, представляя собой комплексное задание по разработке программного продукта, но и практическую значимость, направленную на удовлетворение актуальных потребностей современных пользователей медиасодержимого.

# 1 АНАЛИЗ ИСХОДНЫХ ДАННЫХ

В рамках первого этапа разработки проекта «Аудио/видео проигрыватель для Mac OS» был проведен тщательный анализ исходных данных, который охватывает изучение потребностей целевой аудитории, а также обзор и сравнение существующих аналогов на рынке.

Для определения ключевых функций и характеристик аудио/видео проигрывателя был проведен анализ потребностей пользователей Mac OS. Было выявлено, что для целевой аудитории важны следующие аспекты:

* Поддержка различных медиаформатов: пользователи ожидают, что проигрыватель будет поддерживать широкий спектр аудио и видео форматов, чтобы избежать необходимости использования дополнительных кодеков или конвертации файлов.
* Интуитивно понятный интерфейс: для удобства использования важен чистый, минималистичный дизайн, который позволяет легко находить и управлять медиаконтентом.
* Высокая производительность и стабильность работы: проигрыватель должен эффективно использовать системные ресурсы, обеспечивая плавное воспроизведение без задержек и сбоев.

Для понимания сильных и слабых сторон текущих решений на рынке был проведен анализ популярных медиапроигрывателей для Mac OS, включая VLC Media Player, QuickTime, и IINA. Основные выводы анализа:

* VLC Media Player: хотя VLC поддерживает большое количество форматов и предлагает множество настроек, его интерфейс может показаться перегруженным для новых пользователей.
* QuickTime: этот проигрыватель отличается простотой и интеграцией с Mac OS, однако ограничен в функциональности и поддержке форматов.
* IINA: современный дизайн и хорошая интеграция с Mac OS делают IINA привлекательным выбором, но приложение все еще находится в разработке и может иметь ограниченную поддержку некоторых функций.

Исходя из анализа потребностей пользователей и существующих решений, для нового аудио/видео проигрывателя были определены следующие приоритеты:

* Разработка удобного и интуитивно понятного пользовательского интерфейса, сочетающего в себе эстетику и функциональность.
* Обеспечение поддержки широкого спектра медиаформатов без необходимости дополнительной настройки.
* Гарантирование высокой производительности и стабильности работы приложения на устройствах с Mac OS.

Таким образом, анализ исходных данных позволил выявить ключевые требования к проекту и определить направления для дальнейшей разработки и улучшения аудио/видео проигрывателя.

# 2 ПРОГРАММНОЕ ПРОЕКТИРОВАНИЕ

Программное проектирование аудио/видео проигрывателя для Mac OS охватывает разработку архитектуры приложения, дизайн интерфейса пользователя и выбор технологий для реализации ключевых функций.

Основа архитектуры приложения строится на фреймворке Qt, выбранном за его мощные возможности по созданию кросс-платформенных приложений и поддержку широкого спектра медиаформатов. Структура приложения включает в себя следующие основные компоненты:

* Модуль воспроизведения медиа: отвечает за обработку и воспроизведение аудио и видео файлов. Использует возможности Qt Multimedia для поддержки различных медиаформатов [1].
* Пользовательский интерфейс (UI): разработан с использованием QML и Qt Quick для создания гибкого и адаптивного интерфейса, который легко настраивается под разные размеры экрана и предпочтения пользователя [5].
* Модуль управления: включает функции управления воспроизведением, такие как пауза, перемотка, регулировка громкости и выбор треков.

Дизайн интерфейса пользователя разработан с акцентом на простоту и интуитивность. Основные характеристики дизайна включают:

* Минималистичный подход: чистые линии, сдержанная цветовая палитра и отсутствие ненужных элементов.
* Интуитивные элементы управления: легко доступные и понятные кнопки управления воспроизведением, ползунки регулировки громкости и временной шкалы.
* Адаптивный дизайн: интерфейс автоматически адаптируется под размеры экрана пользователя, обеспечивая удобство просмотра на различных устройствах.

Для реализации проекта были выбраны следующие технологии:

* Qt Framework: используется для создания основной структуры приложения, включая модули обработки медиа и пользовательского интерфейса.
* QML и Qt Quick: применяются для разработки динамичного и адаптивного интерфейса пользователя [4].
* Qt Multimedia: обеспечивает поддержку множества аудио и видео форматов, а также различных функций медиавоспроизведения.

В процессе разработки особое внимание уделялось визуализаци ключевых элементов интерфейса:

* Главный экран проигрывателя: Отображает текущий медиаконтент и основные элементы управления. Пример дизайна главного экрана представлен на рисунке 2.1.
* Плейлист: Включает в себя список доступных для воспроизведения аудио и видео файлов. Пример интерфейса плейлиста показан на рисунке 2.2.

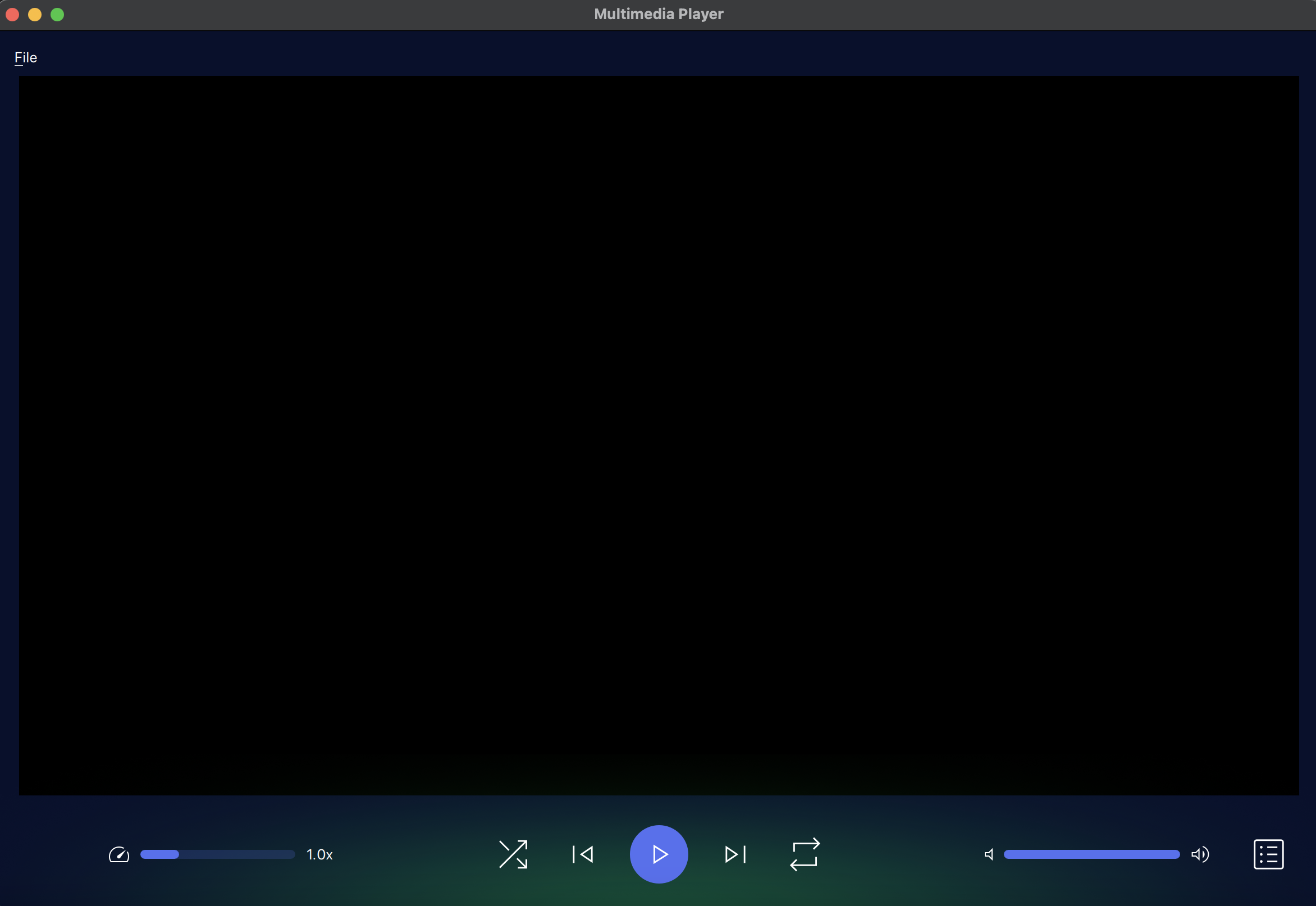


Рисунок 2.1 – Главный экран аудио/видео проигрывателя

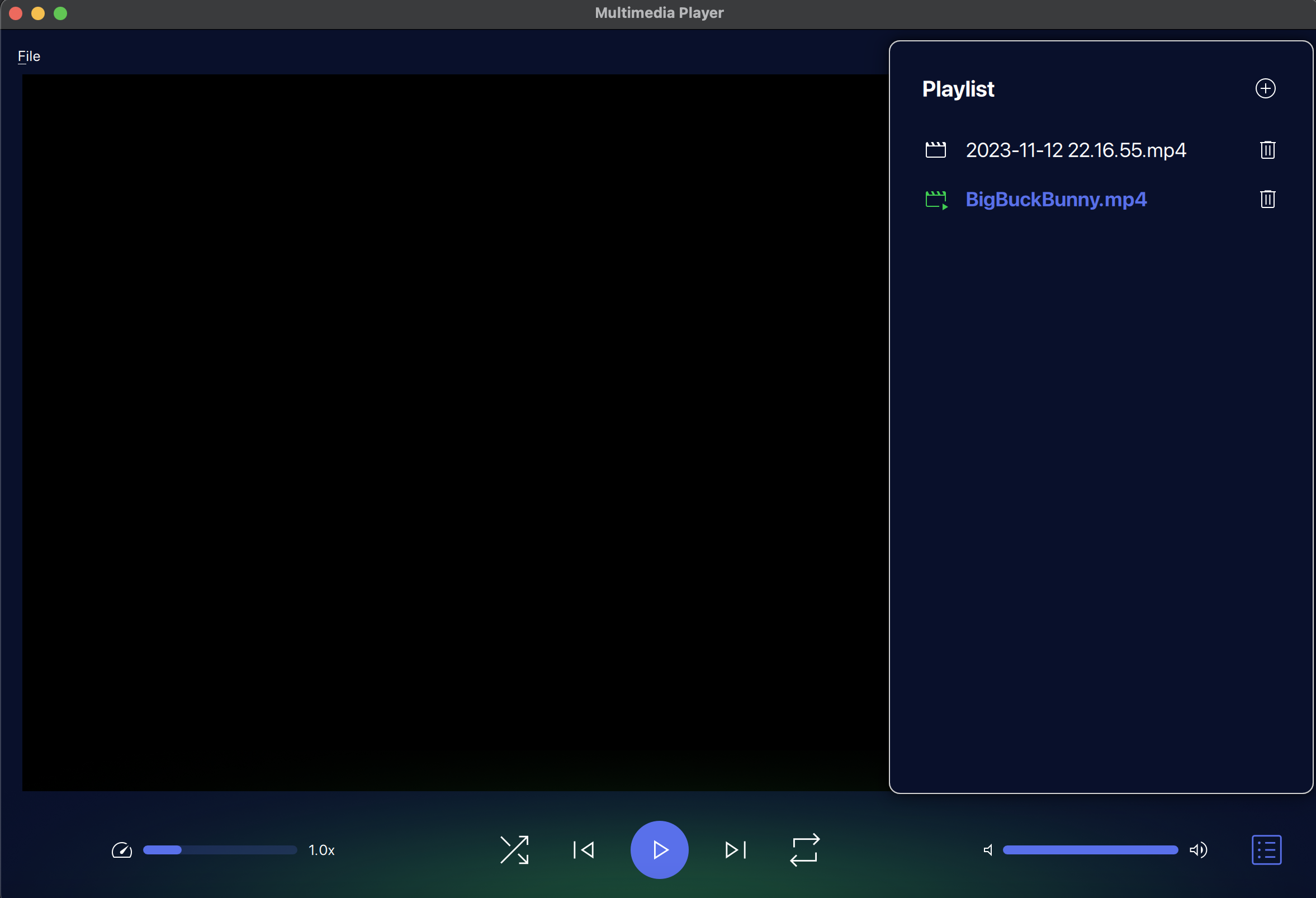


Рисунок 2.2 – Плейлист проигрывателя

Таким образом, программное проектирование аудио/видео проигрывателя для Mac OS было нацелено на создание удобного в использовании и эстетически привлекательного приложения, соответствующего современным стандартам и ожиданиям пользователей.

# 3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

## **3.1 Детальная реализация функциональных частей ПО**

В этом разделе представлено описание основных компонентов аудио/видео проигрывателя для Mac OS, разработанного на базе фреймворка Qt. Основные функциональные части включают модуль воспроизведения медиа, пользовательский интерфейс и модуль управления.

Модуль воспроизведения медиа отвечает за загрузку и обработку аудио и видео файлов. Реализация этого модуля демонстрируется в листинге 3.1, где настроены основные параметры приложения и создан экземпляр QQmlApplicationEngine.

Листинг 3.1 – Основной класс приложения (main.cpp)

#include <QGuiApplication>

#include <QQmlApplicationEngine>

#include <QCommandLineParser>

#include <QDir>

#include <QMediaFormat>

#include <QMimeType>

#include <algorithm>

using namespace Qt::Literals::StringLiterals;

struct NameFilters

{

QStringList filters;

int preferred = 0;

};

static NameFilters nameFilters()

{

QStringList result;

QString preferredFilter;

const auto formats = QMediaFormat().supportedFileFormats(QMediaFormat::Decode);

for (qsizetype m = 0, size = formats.size(); m < size; ++m) {

const auto format = formats.at(m);

QMediaFormat mediaFormat(format);

const QMimeType mimeType = mediaFormat.mimeType();

if (mimeType.isValid()) {

QString filter = QMediaFormat::fileFormatDescription(format) + " ("\_L1;

const auto suffixes = mimeType.suffixes();

for (qsizetype i = 0, size = suffixes.size(); i < size; ++i) {

if (i)

filter += u' ';

filter += "\*."\_L1 + suffixes.at(i);

Продолжение листинга 3.1

}

filter += u')';

result.append(filter);

if (mimeType.name() == "video/mp4"\_L1)

preferredFilter = filter;

}

}

std::sort(result.begin(), result.end());

const int preferred = preferredFilter.isEmpty() ? 0 : int(result.indexOf(preferredFilter));

return { result, preferred };

}

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

{

QGuiApplication app(argc, argv);

QCoreApplication::setApplicationName("MediaPlayer");

QCoreApplication::setOrganizationName("QtProject");

QCoreApplication::setApplicationVersion(QT\_VERSION\_STR);

QCommandLineParser parser;

parser.setApplicationDescription(QCoreApplication::translate("main", "MediaPlayer"));

parser.addHelpOption();

parser.addVersionOption();

parser.addPositionalArgument("url", QCoreApplication::translate("main", "The URL(s) to open."));

parser.process(app);

QQmlApplicationEngine engine;

QObject::connect(&engine, &QQmlApplicationEngine::quit, &app, &QGuiApplication::quit);

QUrl source;

if (!parser.positionalArguments().isEmpty())

source = QUrl::fromUserInput(parser.positionalArguments().at(0), QDir::currentPath());

const auto filters = nameFilters();

QVariantMap initialProperties{

{"source", source},

{"nameFilters", filters.filters},

{"selectedNameFilter", filters.preferred}

};

engine.setInitialProperties(initialProperties);

engine.loadFromModule("MediaPlayer", "Main");

return app.exec();

}

Пользовательский интерфейс разработан с использованием QML и Qt Quick. Пример интерфейса, включая элементы воспроизведения медиа и управления, представлен в листинге 3.2.

Листинг 3.2 - Основной интерфейс пользователя (main.qml)

import QtQuick

// ... [Продолжение импорта модулей]

ApplicationWindow {

id: root

width: 1200

height: 780

minimumHeight: 460

minimumWidth: 640

visible: true

color: Config.mainColor

title: qsTr("Multimedia Player")

// Определение свойств и методов окна

// ... [Продолжение кода]

// MediaPlayer для воспроизведения медиа

MediaPlayer {

id: mediaPlayer

playbackRate: playbackControl.playbackRate

videoOutput: videoOutput

audioOutput: AudioOutput {

id: audio

volume: playbackControl.volume

}

source:new URL("http://commondatastorage.googleapis.com/gtv-videos-bucket/sample/BigBuckBunny.mp4")

function updateMetadata() {

root.metadataInfo.clear()

root.metadataInfo.read(mediaPlayer.metaData)

}

onMetaDataChanged: updateMetadata()

onActiveTracksChanged: updateMetadata()

onErrorOccurred: {

errorPopup.errorMsg = mediaPlayer.errorString

errorPopup.open()

}

onTracksChanged: {

settingsInfo.tracksInfo.selectedAudioTrack = mediaPlayer.activeAudioTrack

settingsInfo.tracksInfo.selectedVideoTrack = mediaPlayer.activeVideoTrack

Продолжение листинга 3.2

settingsInfo.tracksInfo.selectedSubtitleTrack = mediaPlayer.activeSubtitleTrack

updateMetadata()

}

onMediaStatusChanged: {

if ((MediaPlayer.EndOfMedia === mediaStatus && mediaPlayer.loops !== MediaPlayer.Infinite) &&

((root.currentFile < playlistInfo.mediaCount - 1) || playlistInfo.isShuffled)) {

if (!playlistInfo.isShuffled) {

++root.currentFile

}

root.playMedia()

} else if (MediaPlayer.EndOfMedia === mediaStatus && root.playlistLooped && playlistInfo.mediaCount) {

root.currentFile = 0

root.playMedia()

}

}

}

// Определение элементов управления воспроизведением

PlaybackControl {

id: playbackControl

anchors.bottom: parent.bottom

anchors.left: parent.left

anchors.right: parent.right

mediaPlayer: mediaPlayer

isPlaylistVisible: playlistInfo.visible

onPlayNextFile: {

if (playlistInfo.mediaCount) {

if (!playlistInfo.isShuffled){

++root.currentFile

if (root.currentFile > playlistInfo.mediaCount - 1 && root.playlistLooped) {

root.currentFile = 0

} else if (root.currentFile > playlistInfo.mediaCount - 1 && !root.playlistLooped) {

--root.currentFile

return

}

}

root.playMedia()

}

}

onPlayPreviousFile: {

Продолжение листинга 3.2

if (playlistInfo.mediaCount) {

if (!playlistInfo.isShuffled){

--root.currentFile

if (root.currentFile < 0 && isPlaylistLooped) {

root.currentFile = playlistInfo.mediaCount - 1

} else if (root.currentFile < 0 && !root.playlistLooped) {

++root.currentFile

return

}

}

root.playMedia()

}

}

playlistButton.onClicked: !playlistInfo.visible ? root.showOverlay(playlistInfo) : root.closeOverlays()

menuButton.onClicked: menuPopup.open()

}

}

Модуль управления обеспечивает функционал управления воспроизведением, включая паузу, перемотку и регулировку громкости. Пример реализации модуля управления представлен в листинге 3.3.

Листинг 3.3 - Управление воспроизведением (PlaybackControl.qml)

import QtQuick

import QtQuick.Layouts

import QtMultimedia

import Config

Item {

id: root

implicitHeight: 100

required property MediaPlayer mediaPlayer

readonly property int mediaPlayerState: root.mediaPlayer.playbackState

property bool isPlaylistShuffled: false

property bool isPlaylistLooped: false

property bool isPlaylistVisible: false

property url playlistIcon: !root.isPlaylistVisible ? ControlImages.iconSource("Playlist\_Icon") : ControlImages.iconSource("Playlist\_Active", false)

property url shuffleIcon: !root.isPlaylistShuffled ? ControlImages.iconSource("Shuffle\_Icon") : ControlImages.iconSource("Shuffle\_Active", false)

Продолжение листинга 3.3

property alias volume: audio.volume

property alias playbackRate: rate.playbackRate

property alias playlistButton: playlistButton

property alias menuButton: menuButton

signal playNextFile()

signal playPreviousFile()

function changeLoopMode() {

if (root.mediaPlayer.loops === 1 && !root.isPlaylistLooped) {

root.mediaPlayer.loops = MediaPlayer.Infinite

} else if (root.mediaPlayer.loops === MediaPlayer.Infinite) {

root.mediaPlayer.loops = 1

root.isPlaylistLooped = true

} else {

root.mediaPlayer.loops = 1

root.isPlaylistLooped = false

}

}

Item {

anchors.fill: root

RowLayout {

id: playerButtons

anchors.fill: parent

Item {

CustomButton {

id: menuButton

icon.source: ControlImages.iconSource("Menu\_Icon")

visible: Config.isMobileTarget

anchors.centerIn: parent

}

Layout.fillWidth: true

Layout.minimumWidth: 40

Layout.maximumWidth: 95

}

PlaybackRateControl {

id: rate

Layout.fillHeight: true

Layout.fillWidth: true

}

Item {

Продолжение листинга 3.3

Layout.fillWidth: true

}

RowLayout {

id: controlButtons

spacing: Screen.primaryOrientation === Qt.LandscapeOrientation ? 25 : 1

Layout.alignment: Qt.AlignHCenter

Layout.fillWidth: true

CustomButton {

id: shuffleButton

icon.source: root.shuffleIcon

visible: Screen.primaryOrientation === Qt.LandscapeOrientation

onClicked: root.isPlaylistShuffled = !root.isPlaylistShuffled

}

CustomButton {

id: previousButton

icon.source: ControlImages.iconSource("Previous\_Icon")

onClicked: root.playPreviousFile()

}

CustomButton {

id: playButton

icon.source: ControlImages.iconSource("Play\_Icon", false)

onClicked: root.mediaPlayer.play()

}

CustomButton {

id: pausedButton

icon.source: ControlImages.iconSource("Stop\_Icon", false)

onClicked: root.mediaPlayer.pause()

}

CustomButton {

id: nextButton

icon.source: ControlImages.iconSource("Next\_Icon")

onClicked: root.playNextFile()

}

CustomButton {

id: loopButton

Продолжение листинга 3.3

icon.source: ControlImages.iconSource("Loop\_Icon")

visible: Screen.primaryOrientation === Qt.LandscapeOrientation

onClicked: root.changeLoopMode()

states: [

State {

name: "noActiveLooping"

when: root.mediaPlayer.loops === 1 && !root.isPlaylistLooped

PropertyChanges {

loopButton.icon.source: ControlImages.iconSource("Loop\_Icon")

}

},

State {

name: "singleLoop"

when: root.mediaPlayer.loops === MediaPlayer.Infinite

PropertyChanges {

loopButton.icon.source: ControlImages.iconSource("Single\_Loop", false)

}

},

State {

name: "playlistLoop"

when: root.isPlaylistLooped

PropertyChanges {

loopButton.icon.source: ControlImages.iconSource("Loop\_Playlist", false)

}

}

]

}

}

Item {

Layout.fillWidth: true

}

AudioControl {

id: audio

Layout.fillHeight: true

Layout.fillWidth: true

}

Item {

Layout.fillWidth: true

Layout.minimumWidth:40

Layout.maximumWidth: 95

Продолжение листинга 3.3

CustomButton {

id: playlistButton

anchors.centerIn: parent

icon.source: root.playlistIcon

}

}

}

}

states: [

State {

name: "playing"

when: root.mediaPlayerState == MediaPlayer.PlayingState

PropertyChanges {

playButton.visible: false

}

PropertyChanges {

pausedButton.visible: true

}

},

State {

name: "paused"

when: root.mediaPlayerState == MediaPlayer.PausedState || root.mediaPlayerState == MediaPlayer.StoppedState

PropertyChanges {

playButton.visible: true

}

PropertyChanges {

pausedButton.visible: false

}

}

]

}

## **3.2 Сопроводительная документация**

Сопроводительная документация по разработанному программному продукту предоставляется в составе технического задания (приложение А) согласно ГОСТ 19.201-78.

Требования к сопроводительной документации устанавливаются государственными стандартами ЕСПД.

## **3.3 Анализ ПО**

В процессе анализа программного обеспечения аудио/видео проигрывателя для Mac OS особое внимание уделялось следующим аспектам:

* Производительность и совместимость: Оценка эффективности использования системных ресурсов и совместимости с различными версиями Mac OS. Приложение оптимизировано для плавного воспроизведения медиафайлов различных форматов без задержек и сбоев.
* Интуитивный пользовательский интерфейс: Анализ показал, что использование QML и Qt Quick способствует созданию гибкого и интуитивно понятного интерфейса, что делает приложение доступным для широкого круга пользователей.
* Функциональность воспроизведения: Оценка обширной поддержки медиаформатов и разнообразных функций управления воспроизведением, включая регулировку громкости, перемотку и выбор треков.

Таким образом, аудио/видео проигрыватель был нацелен на обеспечение высокой функциональности и удобства использования, при этом поддерживая широкий спектр медиаформатов.

## **3.4 Тестирование ПО**

Тестирование программного обеспечения включало следующие ключевые аспекты:

1. Запуск приложения: проверка инициализации и запуска приложения.
2. Воспроизведение медиафайлов: тестирование открытия и воспроизведения различных аудио и видео форматов.
3. Работа интерфейса управления: проверка функционирования элементов управления воспроизведением, включая кнопки паузы, остановки и перемотки.
4. Тестирование адаптивности интерфейса: убедиться, что интерфейс корректно отображается на различных размерах экранов.
5. Стабильность приложения: проверка работы приложения в различных условиях, включая длительное воспроизведение.
6. Работоспособность плейлиста: добавление в плейлист нескольких видео и переключение между ними.

Пример работы интерфейса управления воспроизведением представлен на рисунке 3.1.

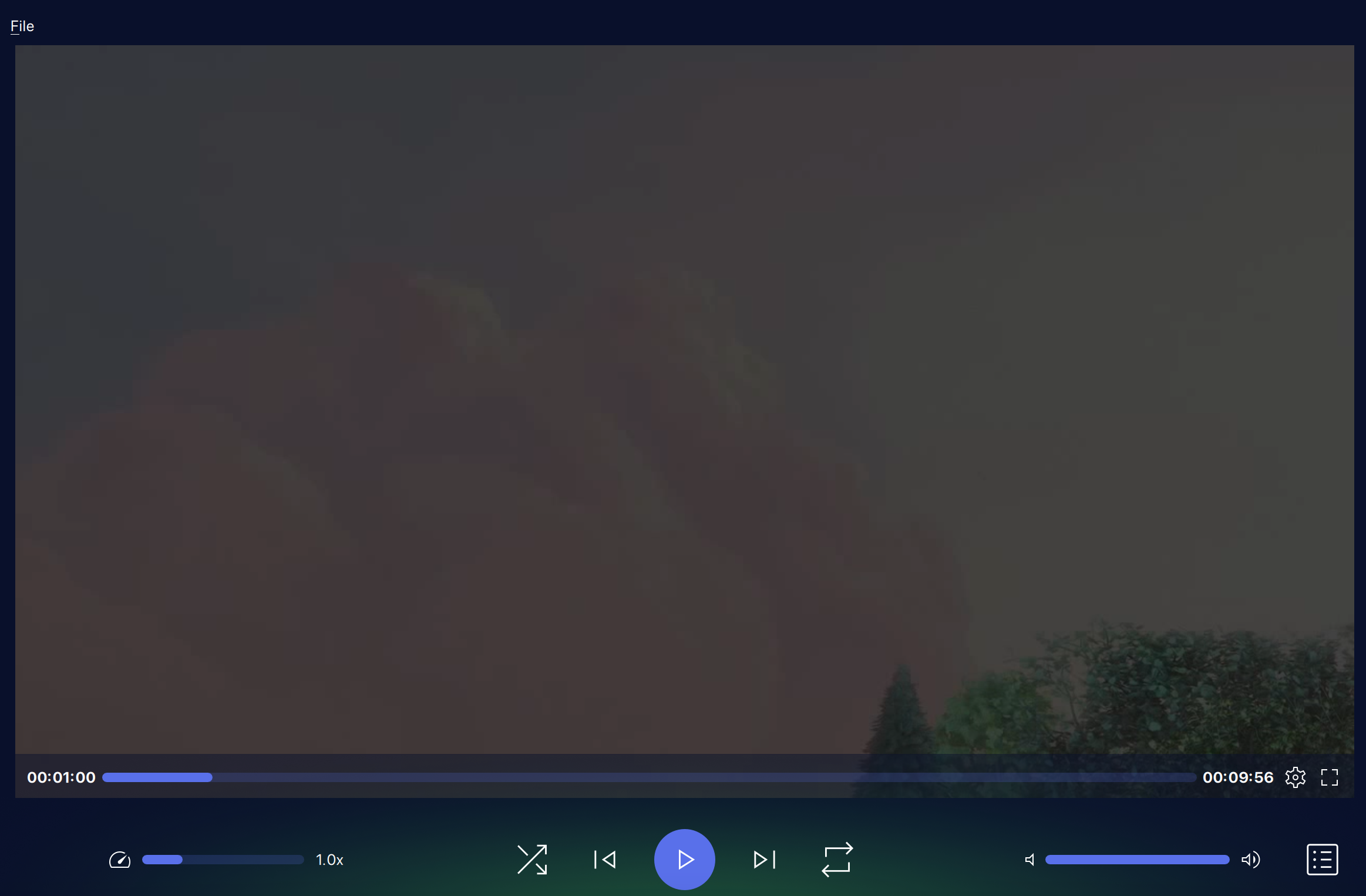


Рисунок 3.1 – Интерфейс управления воспроизведением

Пример адаптивности интерфейса к различным размерам экранов представлен на рисунке 3.2.



Рисунок 3.2 – Адаптивность интерфейса

Пример добавления видео в плейлист представлен на рисунке 3.3.

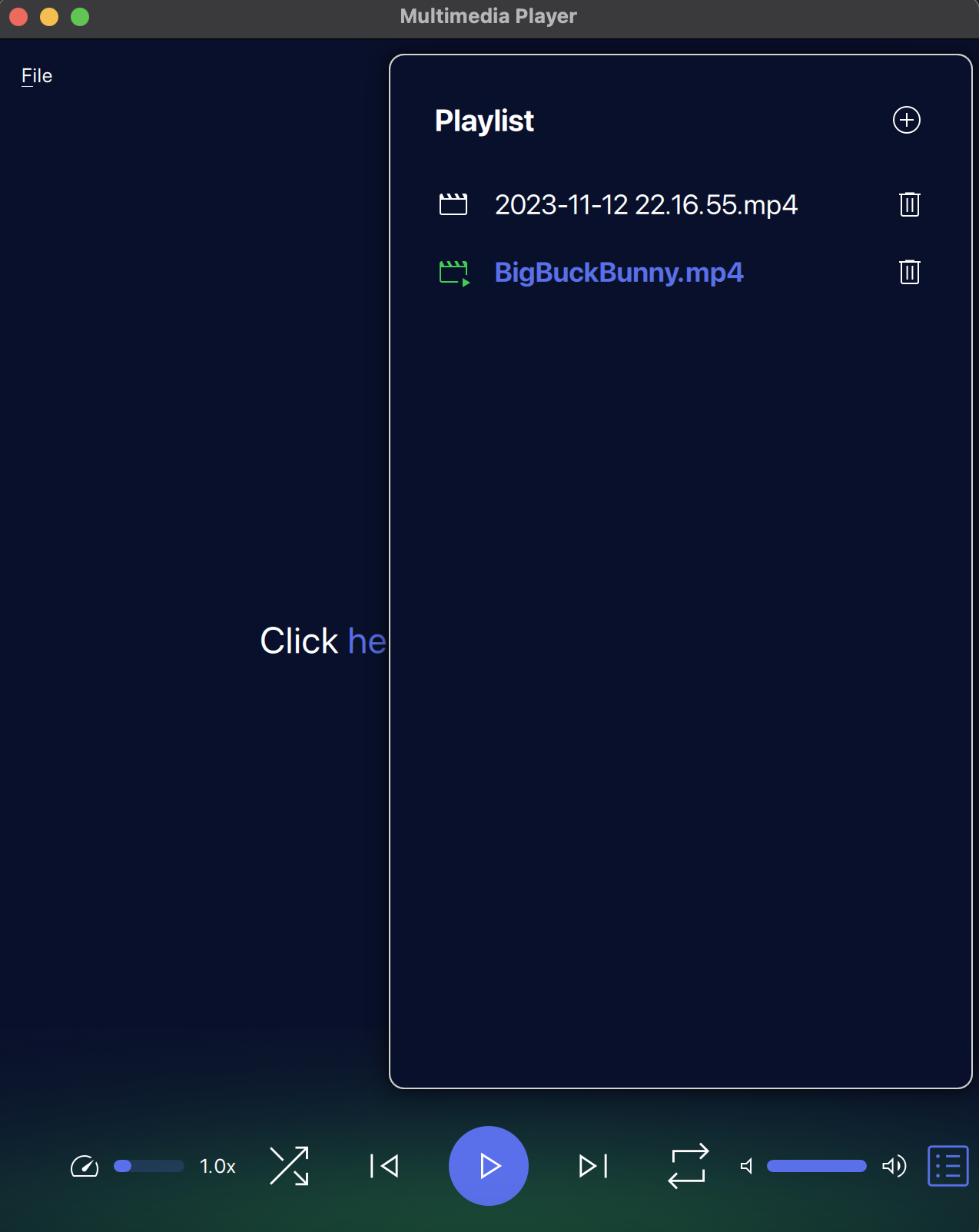


Рисунок 3.3 – Добавление видео в плейлист

# ЗАКЛЮЧЕНИЕ

В результате работы над курсовым проектом «Аудио/видео проигрыватель для Mac OS» на базе Qt, было разработано полнофункциональное приложение для воспроизведения медиафайлов. Проект реализован на языке C++ с использованием фреймворка Qt, что обеспечило высокую переносимость и адаптивность приложения к операционной системе Mac OS.

В ходе разработки были освоены ключевые навыки программирования на C++ в контексте создания кросс-платформенных приложений. Был достигнут глубокий уровень понимания работы с медиафайлами, включая их загрузку, обработку и воспроизведение, а также были разработаны интуитивно понятные и удобные для пользователя интерфейс и элементы управления.

Ключевым аспектом проекта стало создание адаптивного пользовательского интерфейса с использованием QML и Qt Quick, что позволило обеспечить удобство использования приложения на различных устройствах. Приложение поддерживает широкий спектр медиаформатов, предоставляя пользователю гибкие возможности для воспроизведения аудио и видео контента.

В процессе тестирования приложения были выявлены и успешно устранены некоторые недочеты, что способствовало повышению его стабильности и надежности. В результате, приложение демонстрирует высокую производительность и отзывчивость, соответствуя всем заявленным требованиям и ожиданиям.

Таким образом, разработанный аудио/видео проигрыватель для Mac OS является функциональным и удобным решением, сочетающим в себе передовые технологии программирования и удобный интерфейс, что делает его привлекательным для широкого круга пользователей.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Qt Documentation [Электронный ресурс] – Режим доступа: https://doc.qt.io. Дата доступа: 05.11.2023.

2 C++ Reference [Электронный ресурс] – Режим доступа: https://en.cppreference.com. Дата доступа: 05.11.2023.

3 Stack Overflow [Электронный ресурс] – Режим доступа: https://stackoverflow.com. Дата доступа: 05.11.2023.

4 QML Tutorial [Электронный ресурс] – Режим доступа: https://qmlbook.github.io. Дата доступа: 06.11.2023.

5 Qt Multimedia Overview [Электронный ресурс] – Режим доступа: https://doc.qt.io/qt-5/qtmultimediaoverview.html. Дата доступа: 05.11.2023.

6 Qt Quick Controls [Электронный ресурс] – Режим доступа: https://doc.qt.io/qt-5/qtquickcontrols-index.html. Дата доступа: 05.11.2023.

# ПРИЛОЖЕНИЕ А

(обязательное)

**Техническое задание**

**Введение**

Наименование программного продукта – «Аудио/видео проигрыватель для Mac OS». Данное приложение разработано для воспроизведения медиафайлов на операционной системе Mac OS и написано с использованием фреймворка Qt.

**А.1 Основание для разработки**

Программа разрабатывается в рамках курсового проекта студента учебного заведения «Полоцкий государственный университет имени Ефросинии Полоцкой». Основанием для разработки является задание курсового проекта.

**А.2 Назначение разработки**

Функциональное и эксплуатационное назначение программы – обеспечение пользователей Mac OS удобным и функциональным средством для воспроизведения аудио и видео файлов.

**А.3 Требования к программному продукту**

**А.3.1 Требования к функциональным характеристикам**

Для программы «Аудио/видео проигрыватель для Mac OS» выдвигаются следующие требования:

* Поддержка медиаформатов: воспроизведение широкого спектра аудио и видео форматов.
* Пользовательский интерфейс: интуитивно понятный и адаптивный интерфейс пользователя.
* Управление воспроизведением: функции паузы, остановки, перемотки, регулировки громкости и выбора треков.

**А.3.2 Требования к надежности**

* Устойчивость к возможным сбоям и ошибкам.
* Стабильность воспроизведения.

**А.3.3 Условия эксплуатации**

Эксплуатация программы должна осуществляться на устройствах с операционной системой Mac OS. Пользователи должны иметь базовые навыки работы с компьютером и операционной системой.

**А.3.4 Требования к составу и параметрам технических средств**

Минимальные системные требования для стабильной работы приложения:

1. Процессор Intel или Apple M1 и выше.
2. Минимум 4 ГБ оперативной памяти.

**А.3.5 Требования к информационной и программной совместимости**

Программное средство должно удовлетворять следующему требованию: Mas OS Mojave и выше.

**А.3.6 Требования к маркировке и упаковке**

Требования к маркировке и упаковке отсутствуют.

**А.3.7 Требования к транспортированию и хранению**

Программное средство должно храниться на электронном носителе в

виде исполняемого файла.

**А.4 Требования к программной документации**

Программная документация по приложению должна быть предоставлена в следующем составе:

1 техническое задание. Согласно ГОСТ 19.201-78;

2 пояснительная записка. Согласно ГОСТ 19.101-77.

Требования к перечисленным программным документам устанавли-ваются государственными стандартами ЕСПД.

**А.5 Стадии и этапы разработки**

1. Анализ исходных данных, постановка задачи проектирования, разработка технического задания.
2. Разработка архитектуры, интерфейса и структуры программы.
3. Реализация, тестирование программы.
4. Разработка программной документации.

**А.6 Порядок контроля и приемки**

Контроль и приемка программы осуществляются по результатам тестирования функциональности и стабильности работы. Методы испытаний включают визуальный контроль и выполнение тестовых сценариев для проверки соответствия функциональности техническому заданию.

# ПРИЛОЖЕНИЕ б

(обязательное)

**Диаграмма вариантов использования**

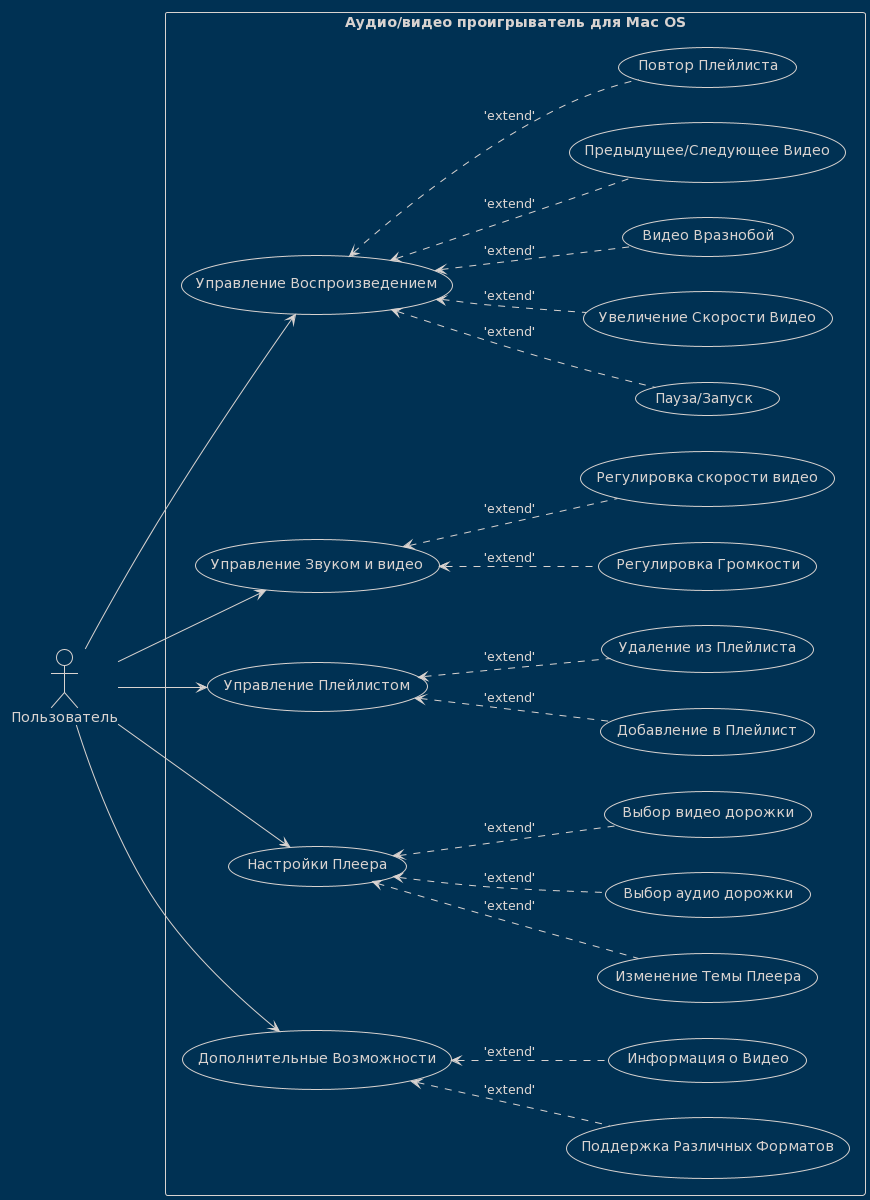
****

Рисунок Б.1 – Диаграмма вариантов использования

# ПРИЛОЖЕНИЕ В

(обязательное)

Таблица В.1 – Способы проверок с указанием ожидаемых результатов испытаний

|  |  |  |  |
| --- | --- | --- | --- |
| Тестовый вариант | Входные данные | Ожидаемый результат | Результат тестирования |
| Запуск проигрывателя | Вход в программу | Открытие главного окна проигрывателя с доступными функциями управления | Тест пройден успешно |
| Управление воспроизведением | Нажатие кнопок управления воспроизведением | Пауза, воспроизведение, переключение на следующее/предыдущее видео | Тест пройден успешно |
| Регулировка громкости | Перемещение ползунка громкости | Изменение уровня громкости аудио | Тест пройден успешно |
| Увеличение скорости видео | Выбор скорости воспроизведения | Ускорение воспроизведения видео | Тест пройден успешно |
| Включение режима повтора плейлиста | Нажатие кнопки повтора плейлиста | Повторное воспроизведение плейлиста после его окончания | Тест пройден успешно |
| Добавление видео в плейлист | Выбор видео для добавления | Видео добавляется в плейлист и доступно для воспроизведения | Тест пройден успешно |
| Удаление видео из плейлиста | Выбор видео для удаления | Видео удаляется из плейлиста и исчезает из списка доступных для воспроизведения | Тест пройден успешно |
| Изменение темы плеера | Выбор темы из настроек | Интерфейс плеера изменяется в соответствии с выбранной темой | Тест пройден успешно |
| Выход из программы | Нажатие пункта меню «Выход» | Закрытие программы | Тест пройден успешно |