**4 ЛАБОРАТОРНАЯ РАБОТА №4**

**«Разработка мобильных приложений для ОС Aurora»**

**4.1 Цель работы**

Изучить и реализовать базовые возможности QML для создания интерактивных мультимедийных элементов, работы с JSON данными и локальной базой данных (LocalStorage), а также освоить управление навигацией в приложении.

* 1. **Вариант задания**

1. Создать 2 кнопки с переходом на страницы SoundEffect.qml и Video.qml:

На странице SoundEffect.qml реализовать SoundEffect для воспроизведения звука с возможностью управления громкостью и количеством повторений с помощью компонента Slider.

На странице Video.qml реализовать компонент Video для воспроизведения видео с возможностью постановки на паузу.

1. Считать данные по ссылке в формате JSON и отобразить на экране приложения id и completed в удобочитаемом формате.
2. Создать приложение, позволяющее добавлять и удалять ФИО людей с использованием базы данных (LocalStorage) и отображать их в списке. Приложение должно отображать на одном экране поле для ввода текста, кнопку для добавления в базу данных и сам список.

**4.3 Ход выполнения работы**

4.3.1 В начале выполнения лабораторной работы была написана программа, имеющая 2 кнопки с переходом на страницу воспроизведения звука и управления громкости и страницу воспроизведения видео. Листинг 4.1 содержит код главной страницы, 4.2 – файла со звуком, а 4.3 – с видео.

Листинг 4.1 – Текст файла главной страницы

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

objectName: "mainPage"

allowedOrientations: Orientation.All

Column {

spacing: 50

anchors.centerIn: parent

Button {

text: "Sound Effect"

onClicked: pageStack.push(Qt.resolvedUrl("Sound.qml"))

backgroundColor: "#785394"

}

Button {

text: "Video"

onClicked: pageStack.push(Qt.resolvedUrl("VideoPlayer.qml"))

backgroundColor: "#785394"

}

}

}

Листинг 4.2 – Текст файла со звуком

import QtQuick 2.0

import Sailfish.Silica 1.0

import QtMultimedia 5.6

Page {

objectName: "mainPage"

allowedOrientations: Orientation.All

SoundEffect {

id: sound

source: "audio.wav"

}

Column {

anchors.centerIn: parent

width: parent.width

Button {

id: btn

text: "Play Sound Effect"

anchors.horizontalCenter: parent.horizontalCenter

onClicked: sound.play()

backgroundColor: "#785394"

}

Slider {

id: volumeSlider

label: "звук"

value: 1.0

valueText: value

minimumValue: 0.0

maximumValue: 1.0

stepSize: 0.1

width: parent.width - Theme.paddingLarge \* 2

onValueChanged: sound.volume = value

color: "#785394"

backgroundColor: "#785394"

}

Slider {

id: repeatSlider

label: "количество повторений"

value: 1.0

valueText: value

minimumValue: 0

maximumValue: 10

stepSize: 1

width: parent.width - Theme.paddingLarge \* 2

onValueChanged: sound.loops = value

color: "#785394"

backgroundColor:"#785394"

}

}

}

Листинг 4.3 – Текст файла с видео

import QtQuick 2.0

import Sailfish.Silica 1.0

import QtMultimedia 5.6

Page {

objectName: "mainPage"

allowedOrientations: Orientation.All

Video {

id: videoPlayer

width: parent.width

height: parent.height

source: "video.mp4"

autoPlay: true

}

Button {

id: playPauseButton

width: parent.width

height: Theme.itemSizeLarge

backgroundColor: "#785394"

text: videoPlayer.playbackState === MediaPlayer.PlayingState ? "Пауза" : "Воспроизведение"

anchors.bottom: parent.bottom

onClicked:{

if (videoPlayer.playbackState === MediaPlayer.PlayingState) {

videoPlayer.pause()

playPauseButton.text = "Воспроизведение"

} else {

videoPlayer.play()

playPauseButton.text = "Пауза"

}

}

}

Button {

id: rewindButton

width: parent.width

height: Theme.itemSizeLarge

text: "Стоп"

backgroundColor: "#785394"

anchors.bottom: playPauseButton.top

onClicked: {

videoPlayer.stop()

}

}

}

Внешний вид главной страницы показан на рисунке 4.1, вид страницы со звуком на рисунке 4.2, а с видео – на рисунке 4.3.

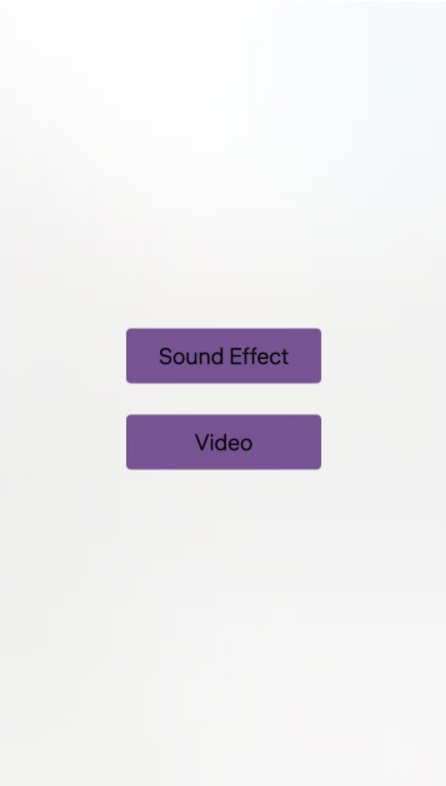


Рисунок 4.1 – Внешний вид главной страницы



Рисунок 4.2 – Внешний вид страницы со звуком



Рисунок 4.3 – Внешний вид страницы с видео

4.3.2 Далее была написана программа, которая считывает данные по ссылке в формате JSON и отображает на экране приложения id и completed в удобочитаемом формате, код этой программы отражен в листинге 4.4.

Листинг 4.4 – Текст второй программы

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

objectName: "mainPage"

allowedOrientations: Orientation.All

backgroundColor: "#ffebcd"

ListModel {

id: jsonModel

}

ListView {

anchors.fill: parent

model: jsonModel

spacing: 130

delegate: Item {

Row {

spacing: 30

x: 30

Text {

text: "ID: " + model.id

color: "#ff6800"

font.pointSize: 30

font.family: "monospace"

}

Text {

text: "Completed: " + model.completed

color: "#4d0000"

font.pointSize: 30

font.family: "monospace"

}

}

}

}

function processData(data) {

for(var i = 0; i < data.length; i++) {

jsonModel.append({

"id": data[i].id,

"completed": data[i].completed

});

}

}

Component.onCompleted: {

var request = new XMLHttpRequest();

request.onreadystatechange = function() {

if(request.readyState === XMLHttpRequest.DONE) {

if(request.status === 200) {

var response = JSON.parse(request.responseText);

processData(response);

} else {

console.error("Failed to fetch JSON data", request.status, request.statusText);

}

}

}

request.open("GET", "https://jsonplaceholder.typicode.com/todos", true);

request.send();

}

}

Внешний вид получаемых выходных данных второй программы показан на рисунке 4.4.

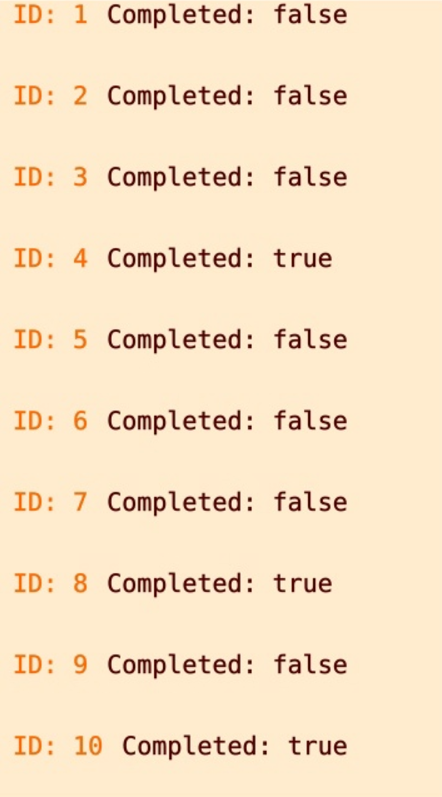


Рисунок 4.4 – Выходные данные второго приложения

4.3.3 В последнем пункте было написано приложение, которое позволяет добавлять и удалять ФИО людей с использованием базы данных (LocalStorage) и отображать их в списке. Также оно должно отображать на одном экране поле для ввода текста, кнопку для добавления в базу данных и сам список. Листинг 4.5 содержит код описанного приложения.

Листинг 4.5 – Текст третьей программы

import QtQuick 2.0

import Sailfish.Silica 1.0

import QtQuick.LocalStorage 2.0

Page {

objectName: "mainPage"

allowedOrientations: Orientation.All

backgroundColor: "#f8f8ff"

property string surname: ""

property string name: ""

property string secondName: ""

property var noteModel: ListModel {

ListElement {

note: "Донец Николай Олегович"

}

ListElement {

note: "Крюкова Ксения Максимовна"

}

}

function addNote() {

if (surname !== "" && name !== "" && secondName !== "") {

noteModel.append({"note": surname + "" + name + "" + secondName})

}

}

function deleteNote(index) {

noteModel.remove(index)

}

SilicaFlickable {

anchors.fill: parent

Column {

id: column

y: 30

width: parent.width

TextField {

id: noteSurname

placeholderText: "Фамилия"

width: parent.width

onTextChanged: surname = text

EnterKey.onClicked: noteName.focus = true

placeholderColor: noteSurname.highlighted ? "#423189" : Theme.darkSecondaryColor

}

TextField {

id: noteName

placeholderText: "Имя"

onTextChanged: name = text

EnterKey.onClicked: noteSecondName.focus = true

placeholderColor: noteSurname.highlighted ? "#423189" : Theme.darkSecondaryColor

}

TextField {

id: noteSecondName

placeholderText: "Отчество"

onTextChanged: secondName = text

EnterKey.onClicked: focus = false

placeholderColor: noteSurname.highlighted ? "#423189" : Theme.darkSecondaryColor

}

Button {

id: addButton

text: "Добавить"

anchors.horizontalCenter: parent.horizontalCenter

color: "#423189"

highlightBackgroundColor: "#b4a8e0"

highlightColor:"#b69eff"

onClicked: addNote()

}

}

ListView {

id: noteListView

width: parent.width

height: parent.height/2

model: noteModel

anchors.top: column.bottom

delegate: Item {

width: parent.width

height: Theme.itemSizeMedium

Row {

x: 20

Label {

text: model.note

wrapMode: Text.Wrap

width: 600

anchors.verticalCenter: parent.verticalCenter

color: Theme.primaryColor

font.family: "Liberation Mono"

font.pixelSize: Theme.fontSizeSmall

}

IconButton {

id: btnDelete

icon.source: "delete.png"

onClicked: deleteNote(index)

}

}

}

}

}

function initializeDatabase() {

var db = LocalStorage.openDatabaseSync("notesDB", "1.0", "Notes Database", 1000000)

db.transaction(function(tx) {

tx.executeSql('CREATE TABLE IF NOT EXISTS notes(note TEXT)')

console.log("Таблица создана")

})

}

Component.onCompleted: initializeDatabase()

}

Внешний вид полученного в этом пункте приложения показан на рисунке 4.5.

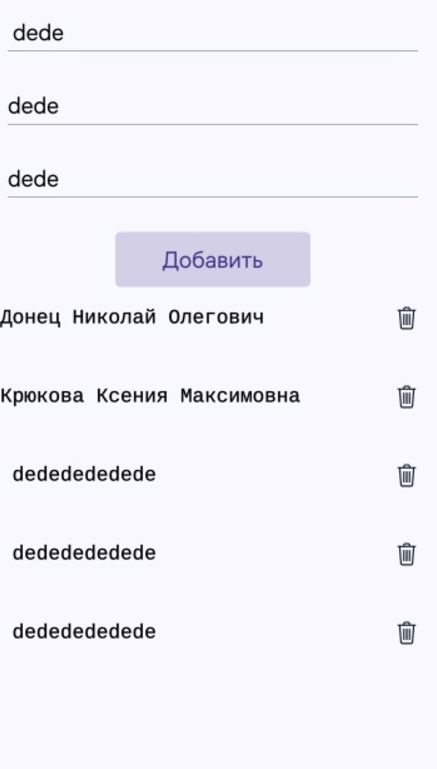


Рисунок 4.5 – Внешний вид третьего приложения

**Выводы**

В ходе лабораторной работы были выполнены следующие задачи: разработаны страницы с кнопками для перехода к мультимедийным элементам SoundEffect и Video, реализован воспроизводитель звука с управлением громкостью и количеством повторений, а также видеоплеер с возможностью постановки на паузу. Было разработано приложение для чтения данных из JSON и отображения их в удобочитаемом формате, а также приложение для управления базой данных с использованием LocalStorage, позволяющее добавлять и удалять ФИО людей. Эти задачи способствовали углубленному изучению возможностей QML и его компонентов, а также навыков работы с данными и локальной базой данных.