МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

##### Федеральное государственное автономное образовательное учреждение высшего образования

#### «Национальный исследовательский

#### Нижегородский государственный университет им. Н.И. Лобачевского» (ННГУ)

#### Институт информационных технологий, математики и механики

#### Кафедра МОСТ

##### Направление подготовки: «Программная инженерия»

##### Профиль подготовки: «Разработка программно-информационных систем»

## ОТЧЕТ

### По лабораторной работе №6

##### по дисциплине «Технология программирования мобильных систем»

**Выполнил:** студент группы 381908-4

Кургузиков Кирилл

#### Руководитель:

##### доцент кафедры программной инженерии

##### Борисов Николай Анатольевич

##### Нижний Новгород 2022

**Содержание**

[1. Цель 3](#_Toc97973273)

[2. Постановка задачи 4](#_Toc97973274)

[3. Решение поставленной задачи 6](#_Toc97973275)

[4. Руководство пользователя 8](#_Toc97973276)

[Приложение 13](#_Toc97973277)

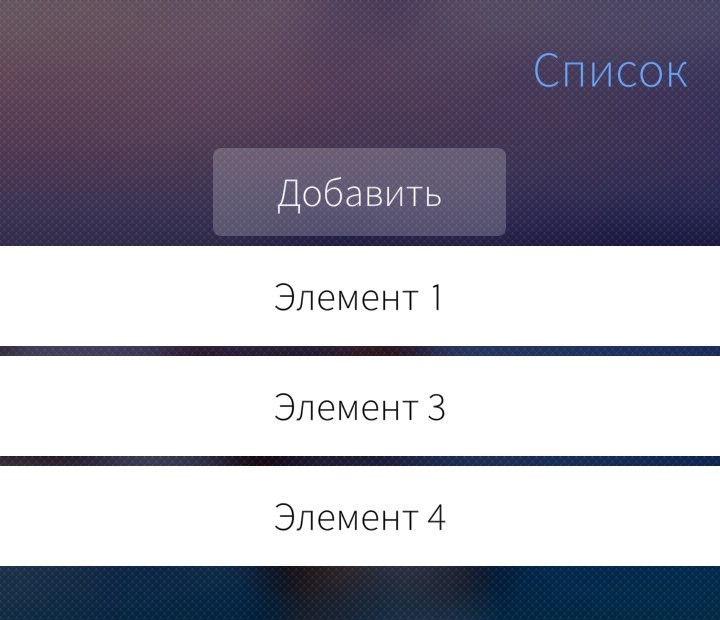
# Цель

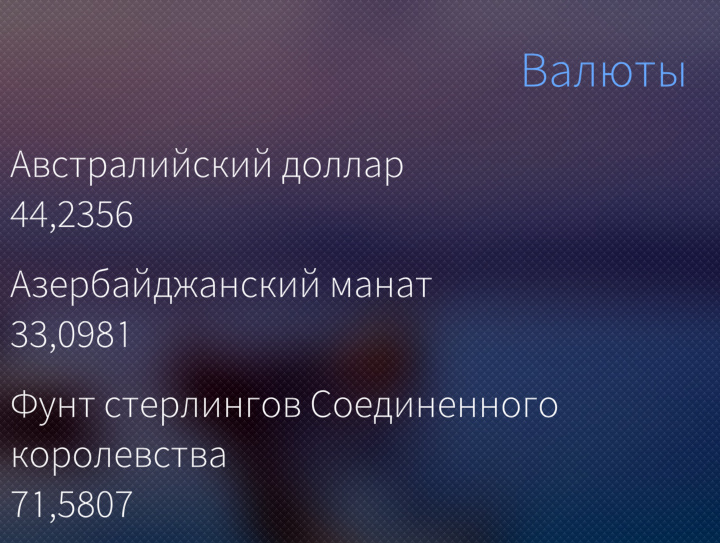
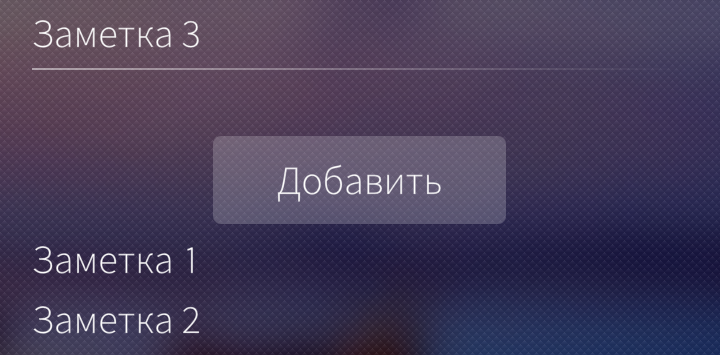
научиться использовать различные модели для отображения данных в прокручиваемых списках, взаимодействовать с базой данных и управлять настройками приложения.

# Постановка задачи

1. Создать приложение, которое позволяет отображать список из прямоугольников с использованием ListModel. В модели должны настраиваться цвет фона и текста внутри прямоугольника. Текст содержит название цвета фона прямоугольника. 

1. Создать приложение, которое позволяет отображать список из прямоугольников. Нажатие на кнопку над списком добавит новый элемент. Нажатие на элемент в списке удалит его из списка. В прямоугольниках должен отображаться порядковый номер, присваиваемый при добавлении в список. При удалении элементов порядковые номера у добавленных прямоугольников остаются неизменными.



1. Выполнить задание 1 с использованием javascript-модели.
2. Получить и отобразить курсы валют из ресурса ЦБ РФ по адресу <http://www.cbr.ru/scripts/XML_daily.asp>.
3. Выполнить задание 4 с использованием XMLHttpRequest.
4. Создать приложение, позволяющее добавлять и удалять заметки с использованием базы данных и отображать их в списке. Текстовое поле служит для ввода текста, кнопка для добавления заметки, нажатие на заметку удалит её. 
5. Создать приложение с текстовым полем и полем с флажком, значение которых сохраняется в настройках приложения с помощью ConfigurationValue.
6. Выполнить задание 7 с помощью ConfigurationGroup.

# Решение поставленной задачи

1. Прямоугольники с использованием ListModel

ListModel {

id: *dataModel*

ListElement { color: "white"; text: "Белый"; color\_text: "Black"}

ListElement { color: "black"; text: "Черный"; color\_text: "White" }

ListElement { color: "blue"; text: "Синий"; color\_text: "White" }

}

Отображение:

Rectangle {

width: *parent*.width

height: 100

color: model.color

Text {

anchors.centerIn: *parent*

text: model.text

color: model.color\_text

}

}

2. Список из прямоугольников, с добавлением и удалением

Добавление:

Button {

id: *addButton*

anchors.horizontalCenter: *parent*.horizontalCenter

text: "Добавить"

onClicked: {

*dataModel*.append({ text: "Элемент " + *sv*.count});

*sv*.count++;

}

}

Прямоугольники, удаление при нажатии:

Rectangle {

width: *parent*.width; height: 70; color: "skyblue";

Text {

anchors.centerIn: *parent*

text: model.text;

}

Button {

anchors.centerIn: *parent*

height: *parent*.height;

width: *parent*.width;

onClicked: *dataModel*.remove(model.index);

}

}

3. Прямоугольники с использованием ListModel и JavaScript

property var dataModel: [

{ color: "white", text: "Белый", color\_text: "Black" },

{ color: "Black", text: "Черный", color\_text: "White" },

{ color: "Blue", text: "Синий", color\_text: "White" }

]

Отображение:

model: *dataModel*

Rectangle {

width: *parent*.width

height: 100

color: modelData.color

Text {

anchors.centerIn: *parent*

text: modelData.text

color: modelData.color\_text

}

}

4. Отображение курсов валют через XmlListModel

XmlListModel {

id: *xmlListModel*

source: "https://www.cbr-xml-daily.ru/daily\_utf8.xml"

query: "/ValCurs/Valute"

XmlRole { name: "Name"; query: "Name/string()"; }

XmlRole { name: "Value"; query: "Value/string()"; }

}

5. Отображение курсов валют через *XMLHttpRequest*

function *loadNews*() {

var *xhr* = new *XMLHttpRequest*();

*xhr*.open('GET', 'https://www.cbr-xml-daily.ru/daily\_utf8.xml', true);

*xhr*.onreadystatechange = function() {

if (*xhr*.readyState === *XMLHttpRequest*.DONE) {

*xmlListModel*.xml = *xhr*.responseText;

}

}

*xhr*.send();

}

XmlListModel {

id: *xmlListModel*

query: "/ValCurs/Valute"

XmlRole { name: "Name"; query: "Name/string()"; }

XmlRole { name: "Value"; query: "Value/string()"; }

}

6. Заметки с использованием LocalStorage

Получение заметок из базы данных  
function *selectBooks*() {

*booksListModel*.clear();

dao.retrieveNotes(function(notes) {

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

var *note* = *notes*.item(*i*);

*console*.log(*note*.id + " " + *note*.text\_note);

*booksListModel*.append({id: *note*.id, note\_text: *note*.text\_note });

}

});

}

function *retrieveNotes*(callback) {

*db*.readTransaction(function(tx) {

var *result* = *tx*.executeSql("SELECT \* FROM notes;");

*callback*(*result*.rows);

});

}

Добавление заметки  
function *insertNote*(text\_note) {

*db*.transaction(function(tx) {

*tx*.executeSql("INSERT INTO notes (text\_note) VALUES(?);", [*text\_note*]);

});

}

Удаление заметки  
 function *deleteNote*(text) {

*console*.log("deleted");

*db*.transaction(function(tx) {

*tx*.executeSql("DELETE FROM notes WHERE text\_note = (?);", [*text*]);

});

}

7. ConfigurationValue

ConfigurationValue {

id: *setting\_1*

key: "/apps/app\_name/setting\_1"

defaultValue: "Menu Default"

}

ConfigurationValue {

id: *setting\_2*

key: "/apps/app\_name/setting\_2"

defaultValue: "Menu Default"

}

8. ConfigurationGroup

ConfigurationGroup {

id: *settings*

path: "/apps/app\_name/settings"

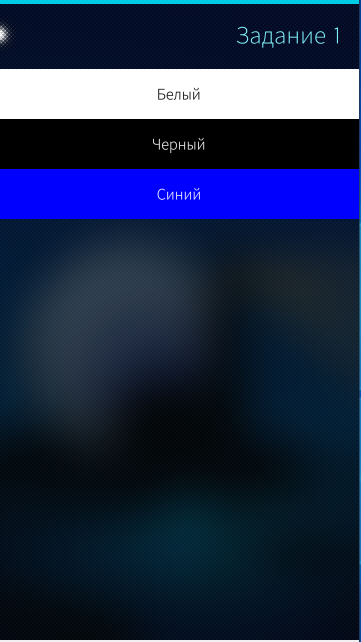
property var tf: "empty"

property bool sw: false

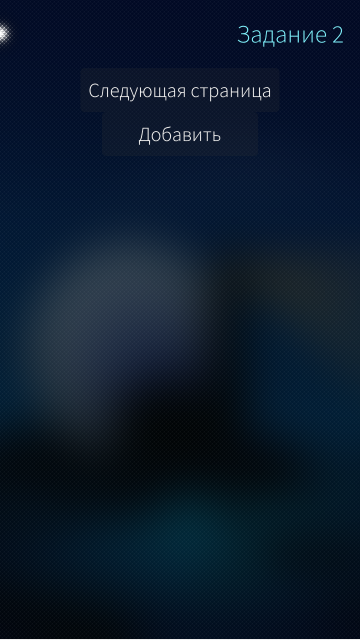
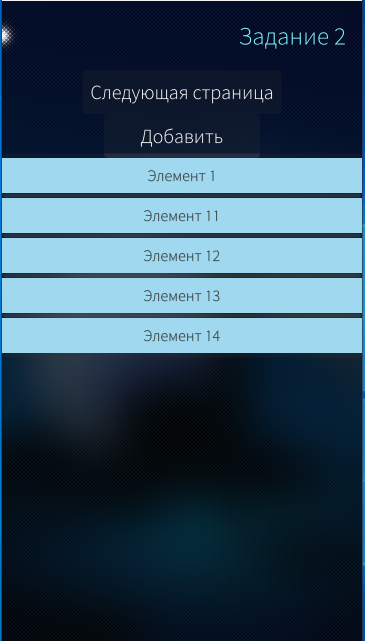
}

# Руководство пользователя

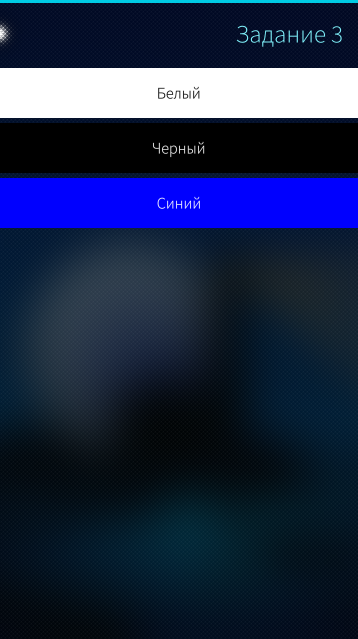
1. Приложение, которое позволяет отображать список из прямоугольников с использованием ListModel. В модели настраиваются цвет фона и текста внутри прямоугольника. Текст содержит название цвета фона прямоугольника.



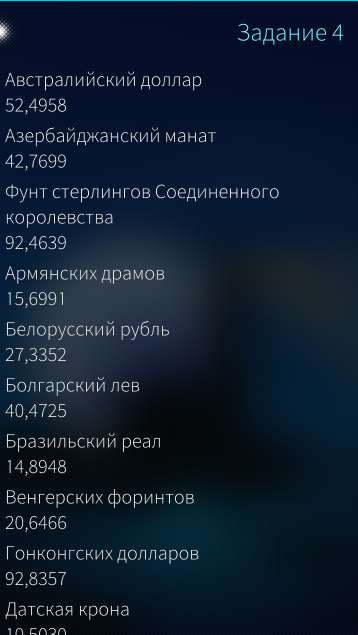
2. Приложение, которое позволяет отображать список из прямоугольников. Нажатие на кнопку над списком добавит новый элемент. Нажатие на элемент в списке удалит его из списка. В прямоугольниках должен отображаться порядковый номер, присваиваемый при добавлении в список. При удалении элементов порядковые номера у добавленных прямоугольников остаются неизменными.

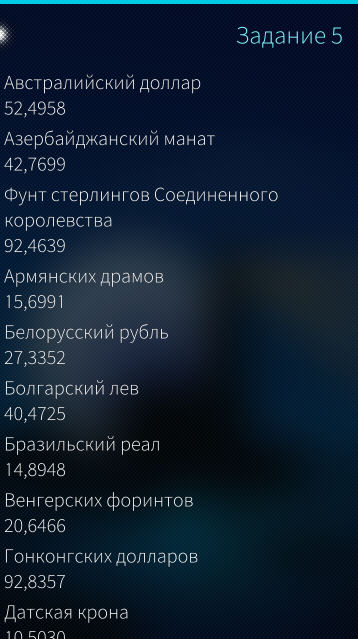
3. Выполнить задание 1 с использованием javascript-модели.



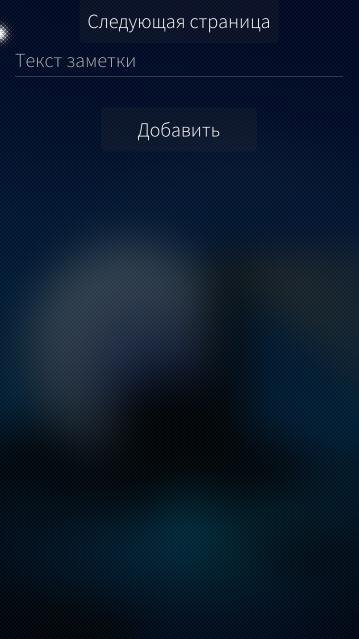
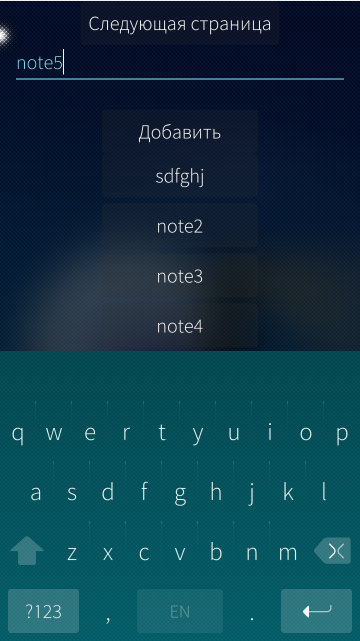
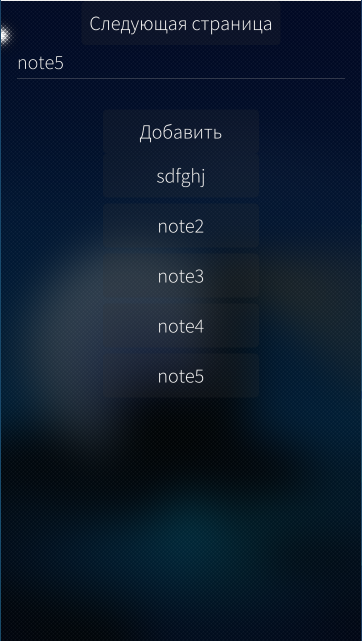
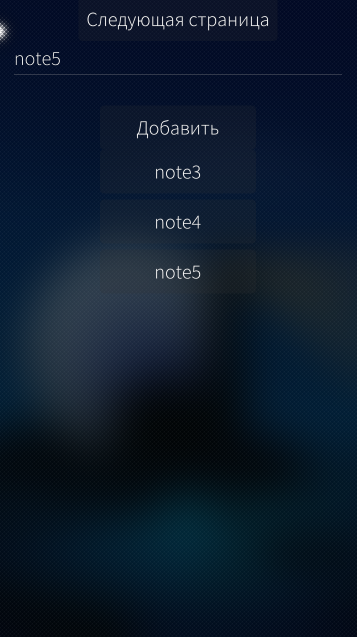
4. Получить и отобразить курсы валют из ресурса ЦБ РФ по адресу <http://www.cbr.ru/scripts/XML_daily.asp>.



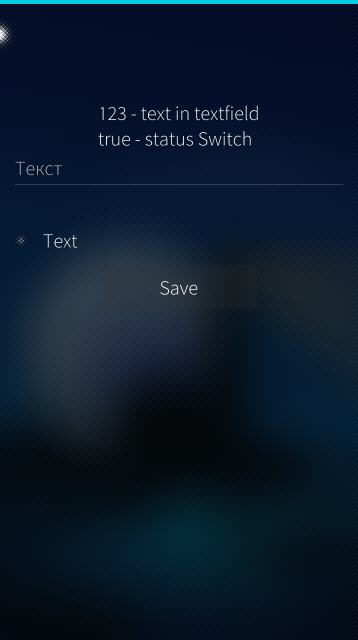
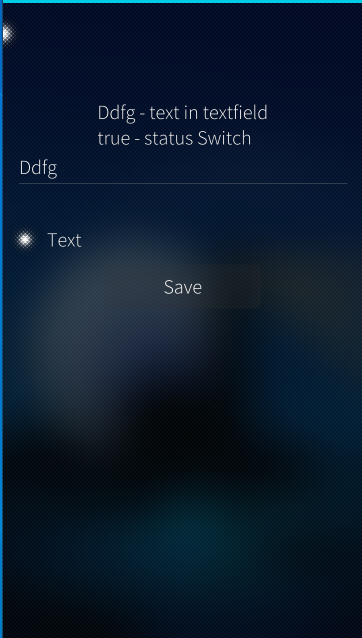
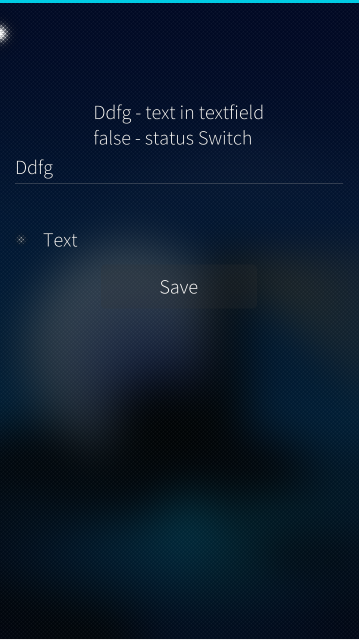
5. Выполнить задание 4 с использованием XMLHttpRequest.



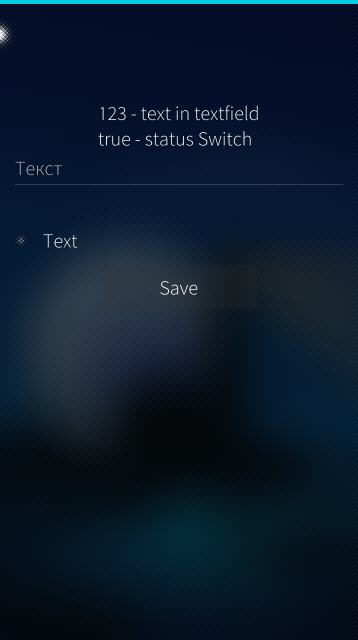
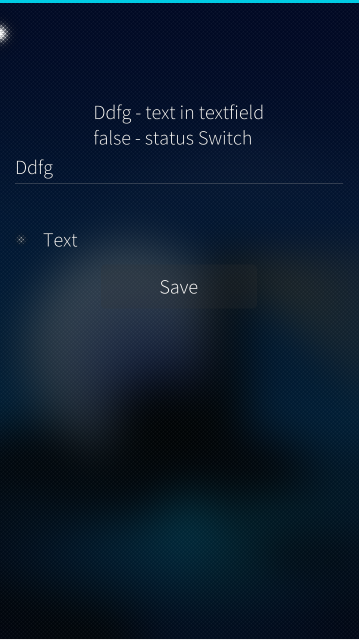
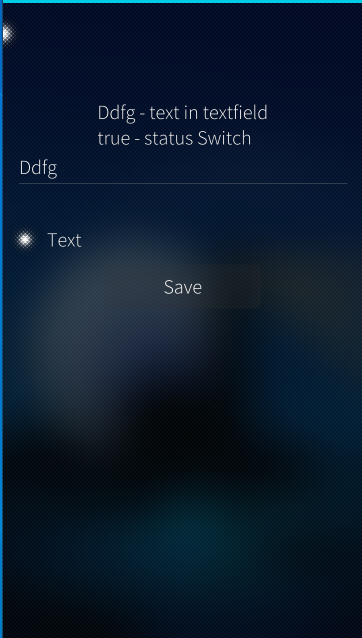
6. Приложение, позволяющее добавлять и удалять заметки с использованием базы данных и отображать их в списке. Текстовое поле служит для ввода текста, кнопка для добавления заметки, нажатие на заметку удалит её.

7. приложение с текстовым полем и полем с флажком, значение которых сохраняется в настройках приложения с помощью ConfigurationValue.

8. Выполнить задание 7 с помощью ConfigurationGroup.

# Приложение

Страница 1

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page*

allowedOrientations: Orientation.All

ListModel {

id: *dataModel*

ListElement { color: "white"; text: "Белый"; color\_text: "Black"}

ListElement { color: "black"; text: "Черный"; color\_text: "White" }

ListElement { color: "blue"; text: "Синий"; color\_text: "White" }

}

SilicaListView {

PullDownMenu {

MenuItem {

text: *qsTr*("Show Page 2")

onClicked: pageStack.animatorPush(*Qt*.resolvedUrl("Page\_2.qml"))

}

}

anchors.fill: *parent*

header: PageHeader { title: "Задание 1" }

model: *dataModel*

delegate: Rectangle {

width: *parent*.width

height: 100

color: model.color

Text {

anchors.centerIn: *parent*

text: model.text

color: model.color\_text

}

}

}

}

Страница 2

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page*;

ListModel { id: *dataModel*; }

Column {

height: *parent*.height;

width: *parent*.width;

PageHeader { title: "Задание 2" }

Button {

id: *addButton1*

anchors.horizontalCenter: *parent*.horizontalCenter

text: "Следующая страница"

onClicked: pageStack.animatorPush(*Qt*.resolvedUrl("Page\_3.qml"))

}

Button {

id: *addButton*

anchors.horizontalCenter: *parent*.horizontalCenter

text: "Добавить"

onClicked: {

*dataModel*.append({ text: "Элемент " + *sv*.count});

*sv*.count++;

}

}

SilicaListView {

id: *sv*;

property int count: 1;

width: *parent*.width; height: *parent*.height

model: *dataModel*

spacing: 10;

delegate: Rectangle {

width: *parent*.width; height: 70; color: "skyblue";

Text {

anchors.centerIn: *parent*

text: model.text;

}

Button {

anchors.centerIn: *parent*

height: *parent*.height;

width: *parent*.width;

onClicked: *dataModel*.remove(model.index);

}

}

}

}

}

Страница 3

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page*

allowedOrientations: Orientation.All

SilicaFlickable {

anchors.fill: *parent*

contentHeight: *column*.height

PullDownMenu {

MenuItem {

text: *qsTr*("Show task 4")

onClicked: pageStack.animatorPush(*Qt*.resolvedUrl("PageDialogCalendar.qml"))

}

}

Column {

id: *column*

width: *page*.width

spacing: Theme.paddingLarge

PageHeader {

title: *qsTr*("Диалог с текстом")

}

Button {

text: "Открыть диалог"

anchors.horizontalCenter: *parent*.horizontalCenter

onClicked: {

var *dialog* = pageStack.push(*Qt*.resolvedUrl("DialogText.qml"))

*dialog*.accepted.connect(function() { *lb*.text = "Результат: " + *dialog*.res

})

}

}

Label {

id: *lb*;

anchors.horizontalCenter: *parent*.horizontalCenter;

text: "Результат пока пуст";

}

}

}

}import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

property var dataModel: [

{ color: "white", text: "Белый", color\_text: "Black" },

{ color: "Black", text: "Черный", color\_text: "White" },

{ color: "Blue", text: "Синий", color\_text: "White" }

]

SilicaListView {

PullDownMenu {

MenuItem {

text: *qsTr*("Show Page 4")

onClicked: pageStack.animatorPush(*Qt*.resolvedUrl("Page\_4.qml"))

}

}

anchors.fill: *parent*

header: PageHeader { title: "Задание 3" }

model: *dataModel*

spacing: 10

delegate: Rectangle {

width: *parent*.width

height: 100

color: modelData.color

Text {

anchors.centerIn: *parent*

text: modelData.text

color: modelData.color\_text

}

}

}

}

Страница 4

import QtQuick 2.0

import Sailfish.Silica 1.0

import QtQuick.XmlListModel 2

Page {

XmlListModel {

id: *xmlListModel*

source: "https://www.cbr-xml-daily.ru/daily\_utf8.xml"

query: "/ValCurs/Valute"

XmlRole { name: "Name"; query: "Name/string()"; }

XmlRole { name: "Value"; query: "Value/string()"; }

}

SilicaListView {

PullDownMenu {

MenuItem {

text: *qsTr*("Show Page 5")

onClicked: pageStack.animatorPush(*Qt*.resolvedUrl("Page\_5.qml"))

}

}

anchors.fill: *parent*

header: PageHeader { title: "Задание 4" }

model: *xmlListModel*

spacing: 10;

delegate: Column {

x: 10; width: *parent*.width - 2 \* *x*

Label {

width: *parent*.width

wrapMode: Text.WordWrap

text: Name + '\n' + Value

}

}

}

}

Страница 5

import QtQuick 2.0

import Sailfish.Silica 1.0

import QtQuick.XmlListModel 2

Page {

function *loadNews*() {

var *xhr* = new *XMLHttpRequest*();

*xhr*.open('GET', 'https://www.cbr-xml-daily.ru/daily\_utf8.xml', true);

*xhr*.onreadystatechange = function() {

if (*xhr*.readyState === *XMLHttpRequest*.DONE) {

*xmlListModel*.xml = *xhr*.responseText;

}

}

*xhr*.send();

}

XmlListModel {

id: *xmlListModel*

query: "/ValCurs/Valute"

XmlRole { name: "Name"; query: "Name/string()"; }

XmlRole { name: "Value"; query: "Value/string()"; }

}

SilicaListView {

PullDownMenu {

MenuItem {

text: *qsTr*("Show Page 6")

onClicked: pageStack.animatorPush(*Qt*.resolvedUrl("Page\_6.qml"))

}

}

anchors.fill: *parent*

header: PageHeader { title: "Задание 5" }

model: *xmlListModel*

spacing: 10;

delegate: Column {

x: 10; width: *parent*.width - 2 \* *x*

Label {

width: *parent*.width

wrapMode: Text.WordWrap

text: Name + '\n' + Value

}

}

}

Component.onCompleted: *loadNews*()

}

Страница 6

import QtQuick 2.0

import Sailfish.Silica 1.0

import QtQuick.LocalStorage 2.0

//import "../service"

Page {

height: parent.height;

width: parent.width;

Column{

height: *parent*.height;

width: *parent*.width;

Button {

id: *addButton1*

anchors.horizontalCenter: *parent*.horizontalCenter

text: "Следующая страница"

onClicked: pageStack.animatorPush(*Qt*.resolvedUrl("Page\_7.qml"))

}

TextField {

id: *tf*;

placeholderText: "Текст заметки"

}

Button {

anchors.horizontalCenter: *parent*.horizontalCenter

text: "Добавить"

onClicked: {

dao.insertNote(*tf*.text);

*selectBooks*();

}

}

SilicaListView {

height: *parent*.height;

width: *parent*.width;

anchors.horizontalCenter: *parent*.horizontalCenter

model: ListModel { id: *booksListModel* }

delegate: ListItem {

Button {

anchors.horizontalCenter: *parent*.horizontalCenter

text: note\_text

onClicked: {

dao.deleteNote(*text*);

*selectBooks*();

}

}

}

}

}

function *selectBooks*() {

*booksListModel*.clear();

dao.retrieveNotes(function(notes) {

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

var *note* = *notes*.item(*i*);

*console*.log(*note*.id + " " + *note*.text\_note);

*booksListModel*.append({id: *note*.id, note\_text: *note*.text\_note });

}

});

}

Component.onCompleted: *selectBooks*();

}

Страница 7

import QtQuick 2.0

import Sailfish.Silica 1.0

import Nemo.Configuration 1.0

Page {

ConfigurationValue {

id: *setting\_1*

key: "/apps/app\_name/setting\_1"

defaultValue: "Menu Default"

}

ConfigurationValue {

id: *setting\_2*

key: "/apps/app\_name/setting\_2"

defaultValue: "Menu Default"

}

SilicaListView {

height: *parent*.height;

width: *parent*.width;

PullDownMenu {

MenuItem {

text: *qsTr*("Show Page 8")

onClicked: pageStack.animatorPush(*Qt*.resolvedUrl("Page\_8.qml"))

}

}

Column {

y: 200;

height: *parent*.height;

width: *parent*.width;

Label {

anchors.horizontalCenter: *parent*.horizontalCenter

text: *setting\_1*.value + " - text in textfield" + "\n" + *setting\_2*.value + " - status Switch"

}

TextField {

id: *tf1*

anchors.horizontalCenter: *parent*.horizontalCenter

placeholderText: "Текст"

}

TextSwitch {

id: *sw1*

width: 100;

text: "Text"

}

Button {

anchors.horizontalCenter: *parent*.horizontalCenter

text: "Save"

onClicked: {

*setting\_1*.value = *tf1*.text;

*setting\_2*.value = *sw1*.checked;

}

}

}

}

}

Страница 8

import QtQuick 2.0

import Sailfish.Silica 1.0

import Nemo.Configuration 1.0

Page {

ConfigurationGroup {

id: *settings*

path: "/apps/app\_name/settings"

property var tf: "empty"

property bool sw: false

}

SilicaListView {

height: *parent*.height;

width: *parent*.width;

PullDownMenu {

MenuItem {

text: *qsTr*("Show Page 1")

onClicked: *pageStack*.animatorPush(*Qt*.resolvedUrl("Page\_1.qml"))

}

}

Column {

y: 200;

height: *parent*.height;

width: *parent*.width;

Label {

anchors.horizontalCenter: *parent*.horizontalCenter

text: *settings*.tf + " - text in textfield" + "\n" + *settings*.sw + " - status Switch"

}

TextField {

id: *tf1*

anchors.horizontalCenter: *parent*.horizontalCenter

placeholderText: "Текст"

}

TextSwitch {

id: *sw1*

width: 100;

text: "Text"

}

Button {

anchors.horizontalCenter: *parent*.horizontalCenter

text: "Save"

onClicked: {

*settings*.tf = *tf1*.text;

*settings*.sw = *sw1*.checked;

}

}

}

}

}