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 Математического обеспечения и суперкомпьютерных технологий**

Направление подготовки: «Фундаментальная информатика и информационные технологии»

Профиль подготовки: «Инженерия программного обеспечения»

**ОТЧЕТ**

**По лабораторной работе №5**

по дисциплине «Технология программирования мобильных систем»

**Выполнила:** студентка группы 381906-3

Сорокина Екатерина Вячеславовна

**Руководитель:**

Доцент, кандидат технических наук

Борисов Николай Анатольевич

Нижний Новгород  
2022

**Содержание**

[1. Цель 3](#_Toc122817061)

[2. Постановка задачи 4](#_Toc122817062)

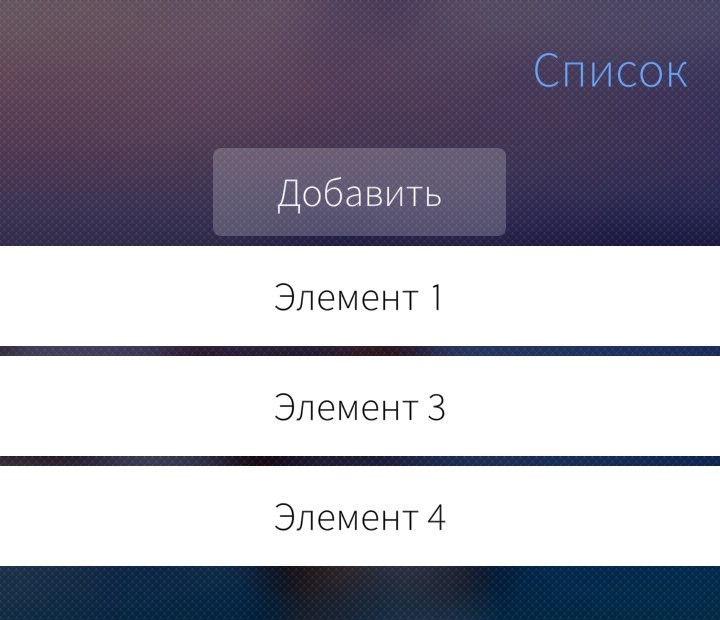
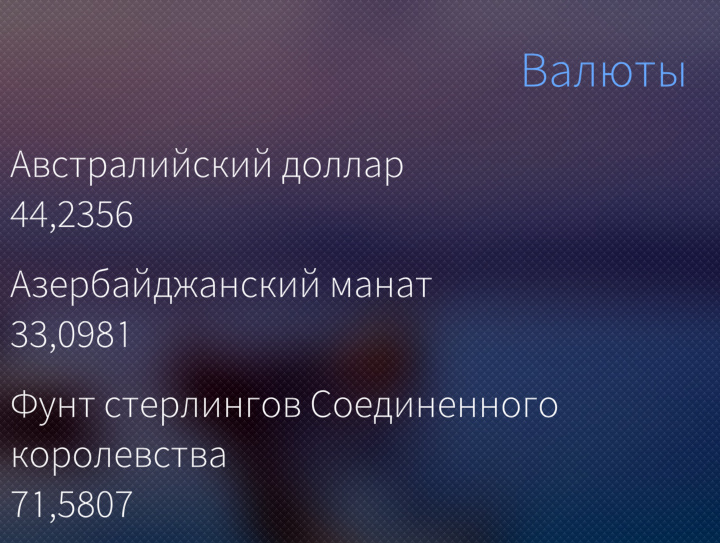
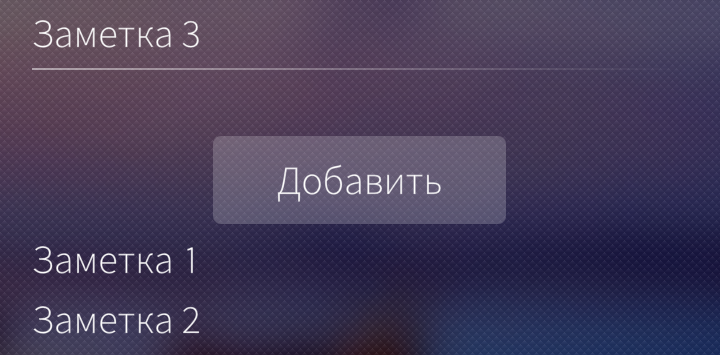
[3. Решение поставленной задачи 6](#_Toc122817063)

[4. Руководство пользователя 12](#_Toc122817064)

[Приложение 16](#_Toc122817065)

1. Цель

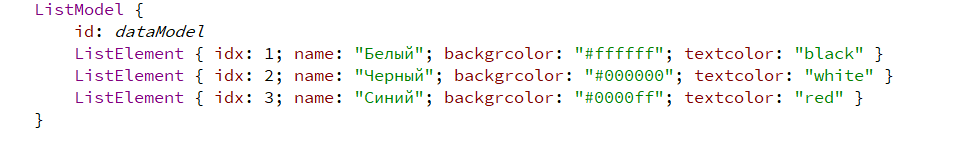
Цель: научиться использовать различные модели для отображения данных в прокручиваемых списках, взаимодействовать с базой данных и управлять настройками приложения.

1. Постановка задачи
2. 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.
3. 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.
4. Выполнить задание 1 с использованием javascript-модели.
5. Получить и отобразить курсы валют из ресурса ЦБ РФ по адресу <http://www.cbr.ru/scripts/XML_daily.asp>.
6. Выполнить задание 4 с использованием XMLHttpRequest.
7. 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. 
8. Создать приложение с текстовым полем и полем с флажком, значение которых сохраняется в настройках приложения с помощью ConfigurationValue.
9. Выполнить задание 7 с помощью ConfigurationGroup.
10. Решение поставленной задачи

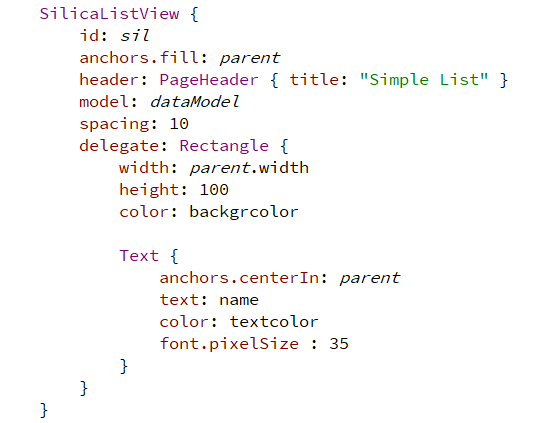
Создаём проект со стандартной заготовкой приложения.

**Задание 1.**

ListModel:

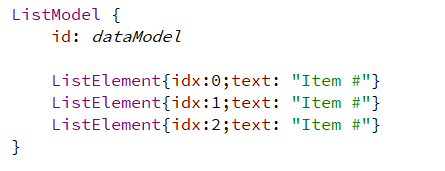


Отображение прямоугольников:

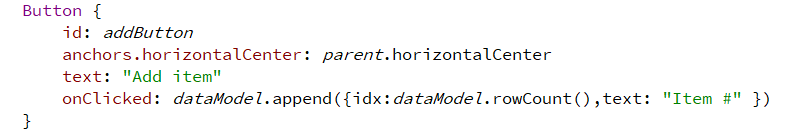


**Задание 2.**

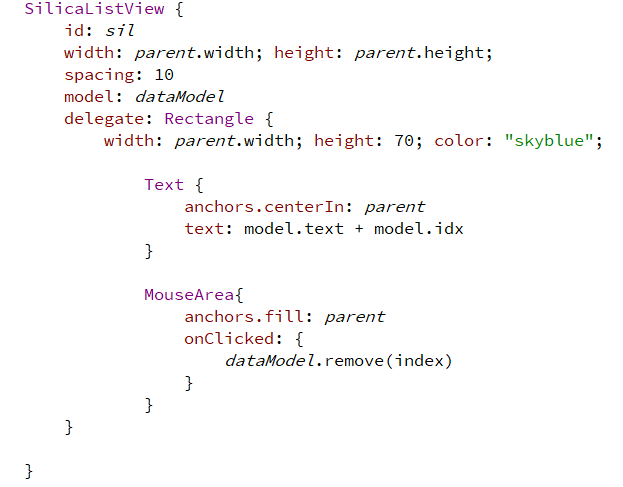
Элементы списка



Добавление прямоугольников

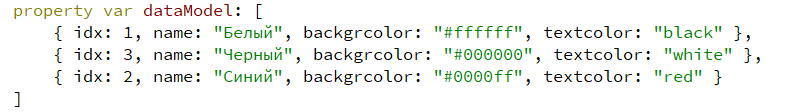


Отображение прямоугольников и удаление через нажатие

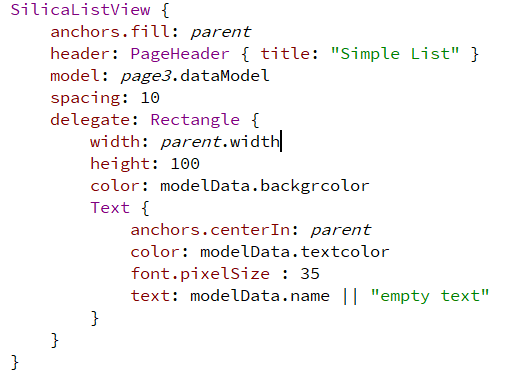


**Задание 3.**

Модель через javascript



Отображение прямоугольников

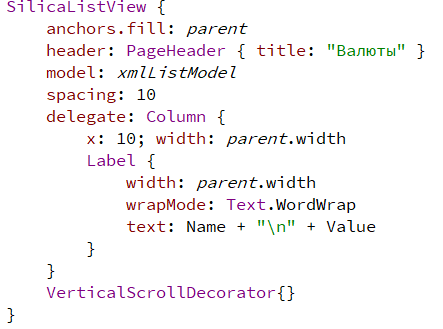


**Задание 4.**

Получение курса валют через XmlListModel



Отображение



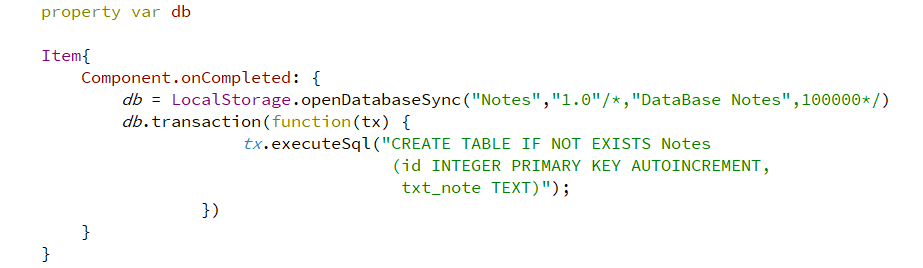
**Задание 5.**

Получение курса валют через XMLHttpRequest и отображение

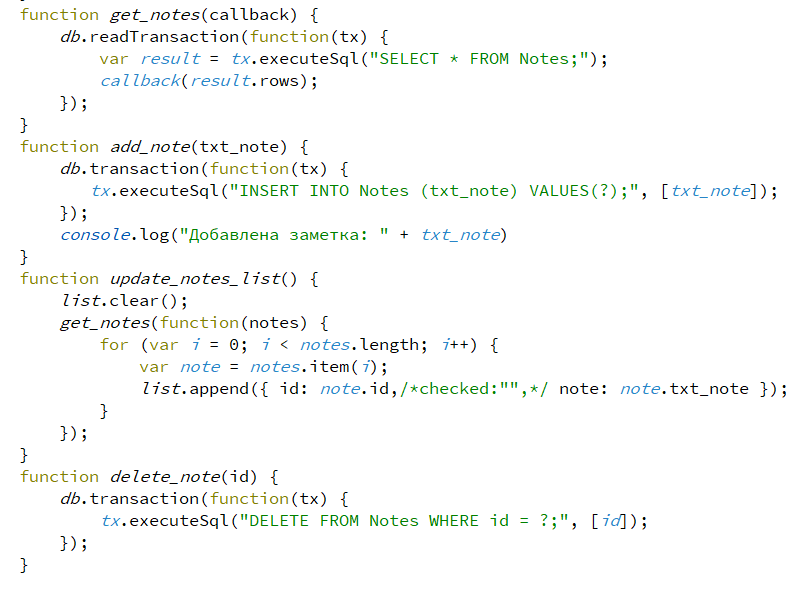


**Задание 6.**

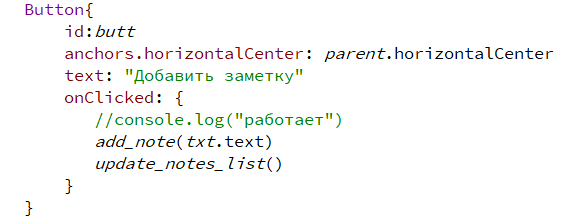
Создание базы данных



Методы для получения, добавления, обновления и удаления заметок



Кнопка добавления новой заметки

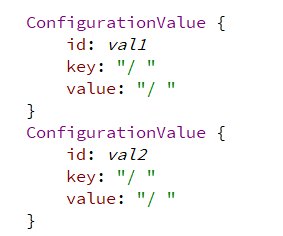


Отображение заметок и удаление через нажатие

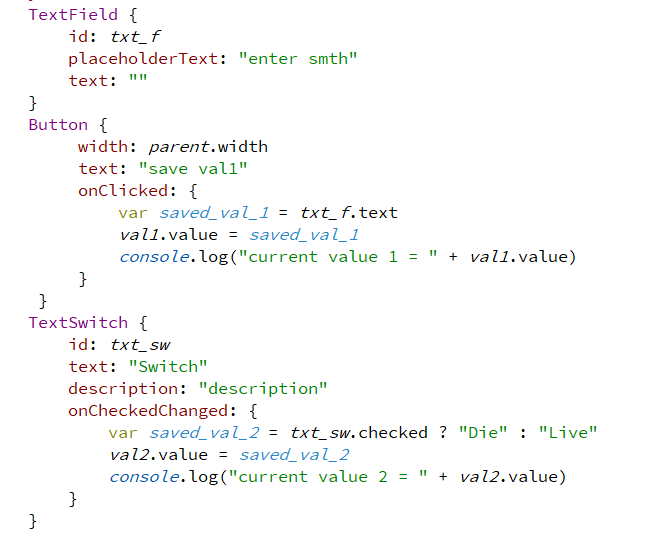


**Задание7.**

ConfigurationValue

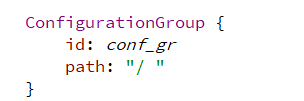


Текстовое поле и поле с флагом (+ их сохранение)



**Задание 8.**

ConfigurationGroup



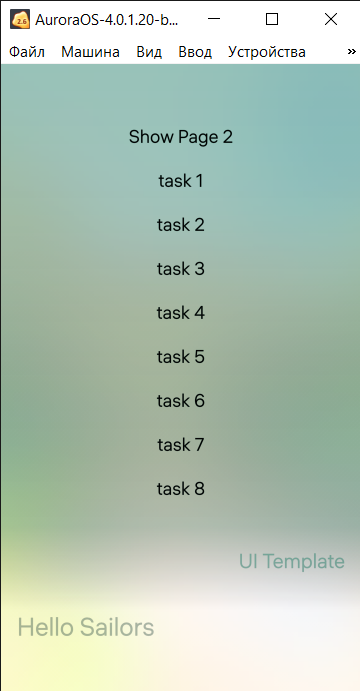
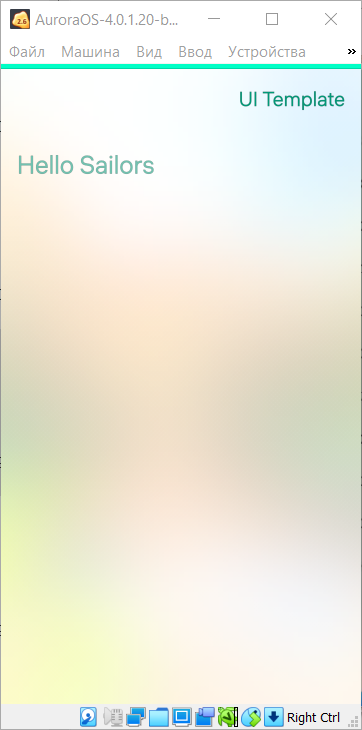
Текстовое поле и поле с флагом (+ их сохранение)



Осталось только сохранить все изменения и запустить эмулятор.

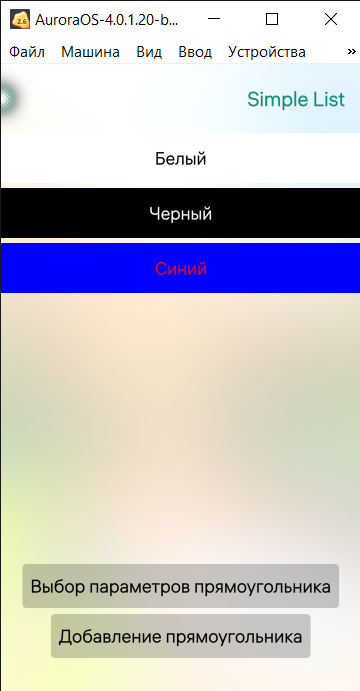
1. Руководство пользователя

При запуске приложения пользователь попадает на страницу FirstPage. Потянув вниз, открывается меню с нашими страницами.

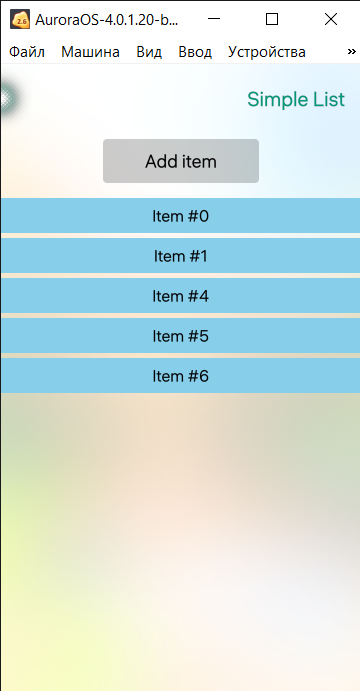


При нажатии на каждую из страниц, она загрузит нам наше задание.

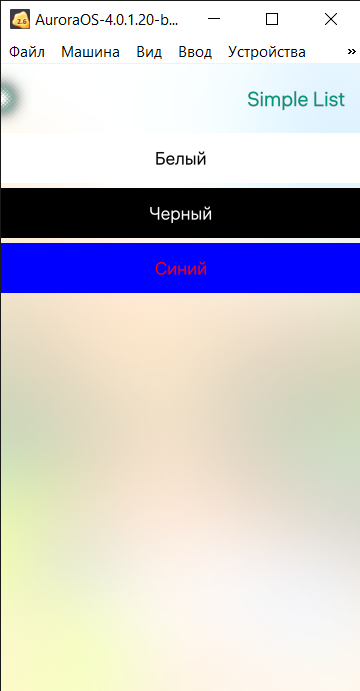
Задание 1:



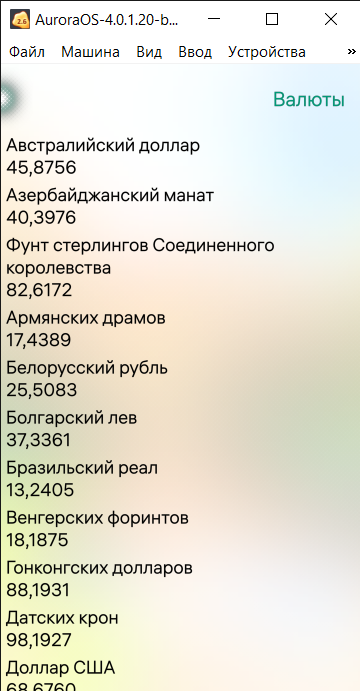
Задание 2:



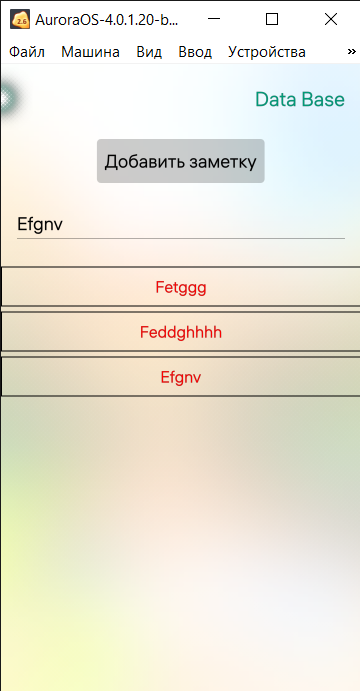
Задание 3:



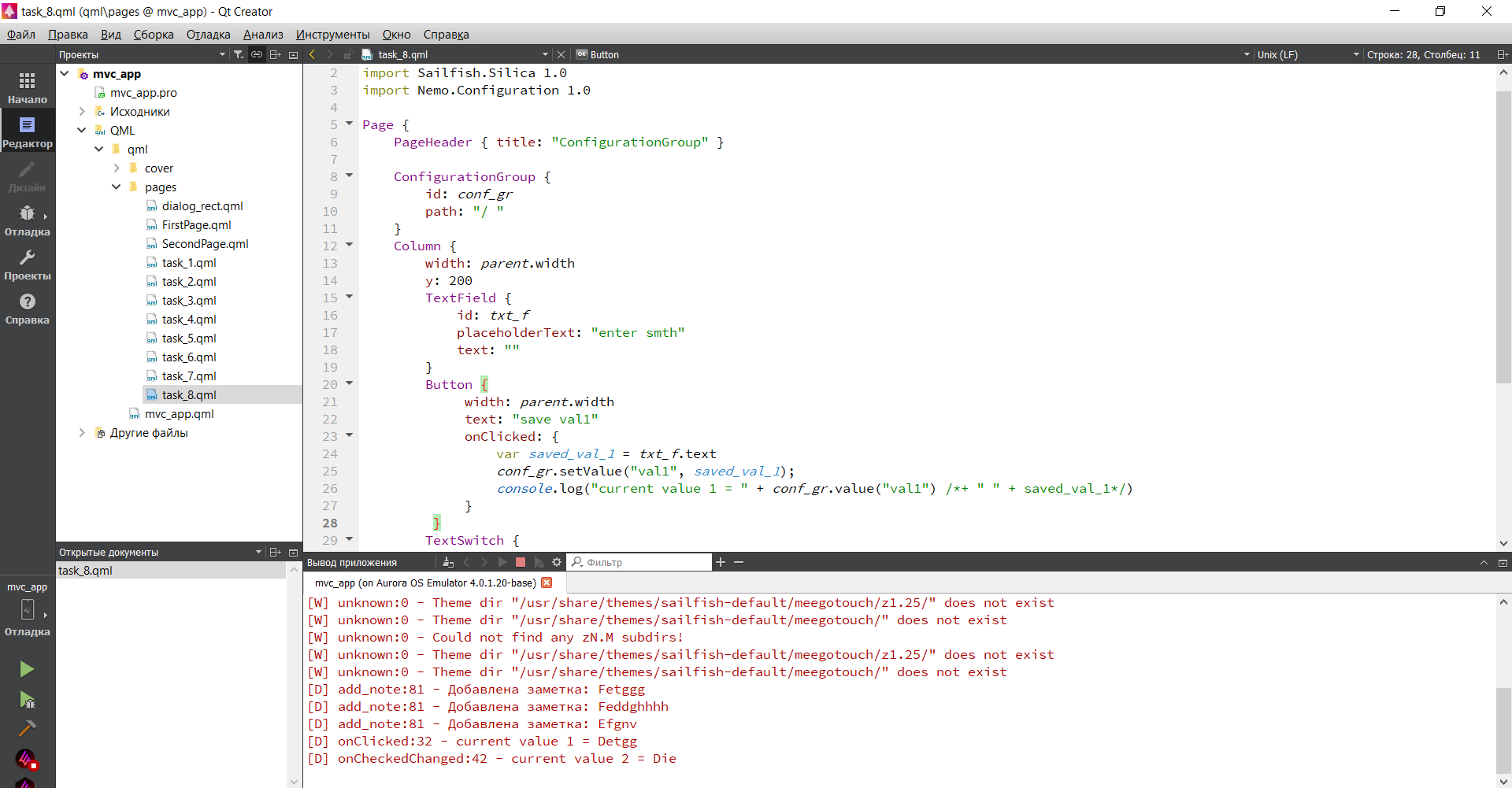
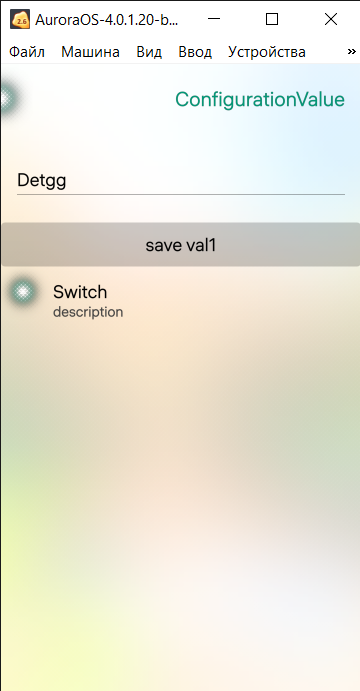
Задание 4 и 5:



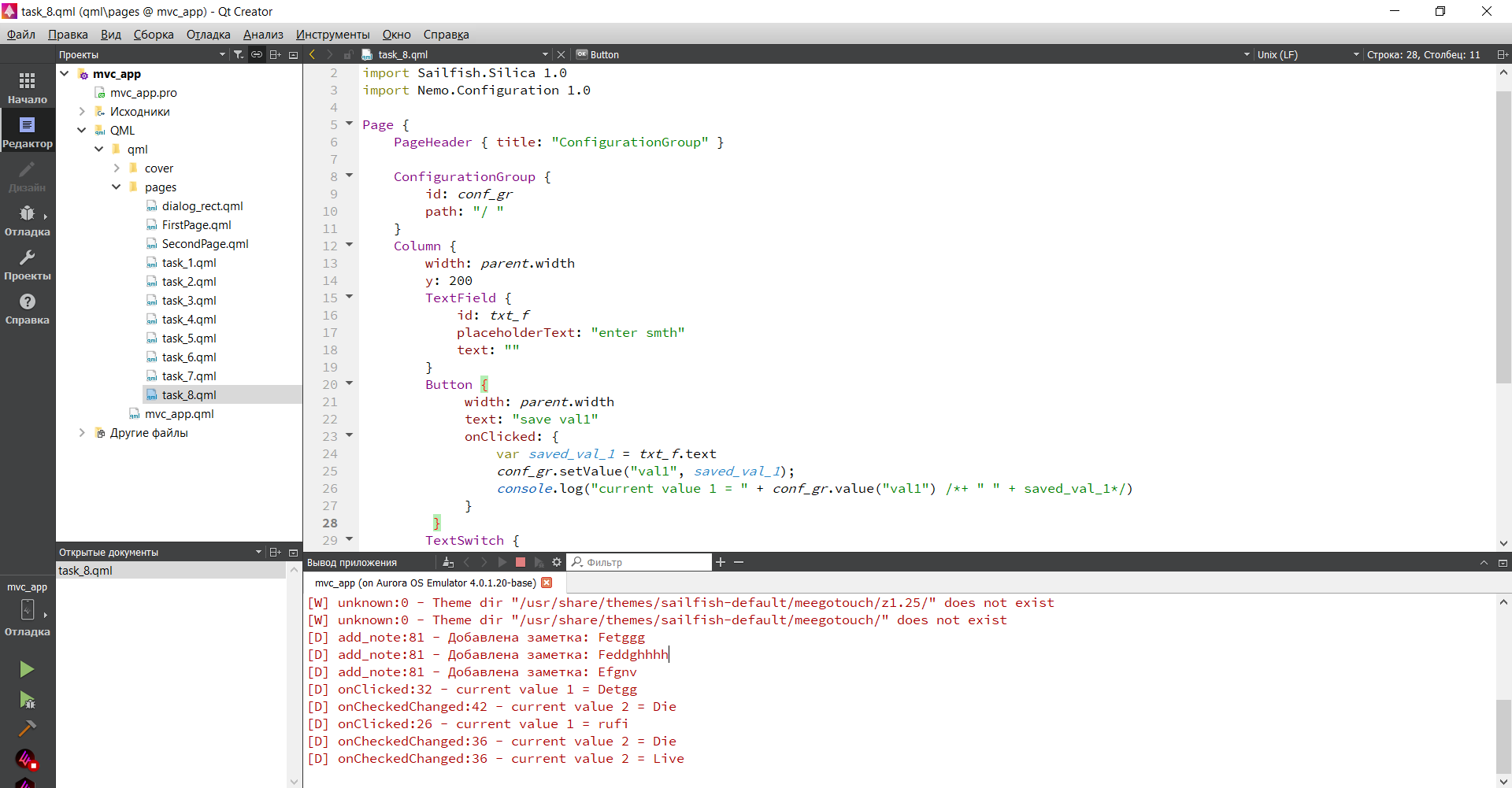
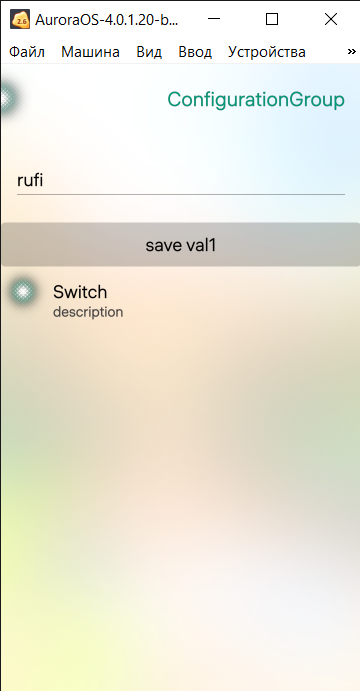
Задание 6:



Задание 7:



Задание 8:



Приложение

// FirstPage.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page*

// The effective value will be restricted by ApplicationWindow.allowedOrientations

allowedOrientations: Orientation.All

// To enable PullDownMenu, place our content in a SilicaFlickable

SilicaFlickable {

anchors.fill: *parent*

// PullDownMenu and PushUpMenu must be declared in SilicaFlickable, SilicaListView or SilicaGridView

PullDownMenu {

MenuItem {

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

onClicked: *pageStack*.push(*Qt*.resolvedUrl("SecondPage.qml"))

}

MenuItem{

text: *qsTr*("task 1")

onClicked: *pageStack*.push(*Qt*.resolvedUrl("task\_1.qml"))

}

MenuItem{

text: *qsTr*("task 2")

onClicked: *pageStack*.push(*Qt*.resolvedUrl("task\_2.qml"))

}

MenuItem{

text: *qsTr*("task 3")

onClicked: *pageStack*.push(*Qt*.resolvedUrl("task\_3.qml"))

}

MenuItem{

text: *qsTr*("task 4")

onClicked: *pageStack*.push(*Qt*.resolvedUrl("task\_4.qml"))

}

MenuItem{

text: *qsTr*("task 5")

onClicked: *pageStack*.push(*Qt*.resolvedUrl("task\_5.qml"))

}

MenuItem{

text: *qsTr*("task 6")

onClicked: *pageStack*.push(*Qt*.resolvedUrl("task\_6.qml"))

}

MenuItem {

text: *qsTr*("task 7")

onClicked: *pageStack*.push(*Qt*.resolvedUrl("task\_7.qml"))

}

MenuItem {

text: *qsTr*("task 8")

onClicked: *pageStack*.push(*Qt*.resolvedUrl("task\_8.qml"))

}

}

// Tell SilicaFlickable the height of its content.

contentHeight: *column*.height

// Place our content in a Column. The PageHeader is always placed at the top

// of the page, followed by our content.

Column {

id: *column*

width: *page*.width

spacing: Theme.paddingLarge

PageHeader {

title: *qsTr*("UI Template")

}

Label {

x: Theme.horizontalPageMargin

text: *qsTr*("Hello Sailors")

color: Theme.secondaryHighlightColor

font.pixelSize: Theme.fontSizeExtraLarge

}

}

}

}

// task\_1.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

ListModel {

id: *dataModel*

ListElement { idx: 1; name: "Белый"; backgrcolor: "#ffffff"; textcolor: "black" }

ListElement { idx: 2; name: "Черный"; backgrcolor: "#000000"; textcolor: "white" }

ListElement { idx: 3; name: "Синий"; backgrcolor: "#0000ff"; textcolor: "red" }

}

SilicaListView {

id: *sil*

anchors.fill: *parent*

header: PageHeader { title: "Simple List" }

model: *dataModel*

spacing: 10

delegate: Rectangle {

width: *parent*.width

height: 100

color: backgrcolor

Text {

anchors.centerIn: *parent*

text: name

color: textcolor

font.pixelSize : 35

}

}

}

property string txt\_rect: ""

property string txt\_color: ""

property string bg\_color: ""

Button{

y: 1000

//anchors.top: sil.bottom

anchors.horizontalCenter: *parent*.horizontalCenter

text: "Выбор параметров прямоугольника"

onClicked: {

var *dialog\_rect* = pageStack.push(*Qt*.resolvedUrl("dialog\_rect.qml")/\*,{saved\_text: full\_text}\*/)

*dialog\_rect*.accepted.connect(function(){

*txt\_rect* = *dialog\_rect*.txt\_rec;

*txt\_color* = *dialog\_rect*.txt\_col;

*bg\_color* = *dialog\_rect*.bg\_col;

})

}

}

Button{

y: 1100

//anchors.top: sil.bottom

anchors.horizontalCenter: *parent*.horizontalCenter

text: "Добавление прямоугольника"

onClicked: {

*dataModel*.append({idx:*dataModel*.rowCount()+1, name: *txt\_rect*, backgrcolor:*bg\_color*, textcolor: *txt\_color*})

}

}

}

//dialog\_rect.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Dialog {

property string txt\_rec: *txt*.text

property string txt\_col: *txt\_c*.text

property string bg\_col: *bg\_c*.text

Column {

width: *parent*.width

spacing: Theme.paddingMedium

DialogHeader { }

TextField {

id: *txt*

width: *parent*.width

text: ""

placeholderText: "текст прямоугольника"

}

TextField{

id: *txt\_c*

width: *parent*.width

text: ""

placeholderText: "цвет текста прямоугольника"

}

TextField{

id: *bg\_c*

width: *parent*.width

text: ""

placeholderText: "цвет фона прямоугольника"

}

}

}

//task\_2.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page2*

PageHeader {id: *header* ; title: "Simple List"}

ListModel {

id: *dataModel*

ListElement{idx:0;text: "Item #"}

ListElement{idx:1;text: "Item #"}

ListElement{idx:2;text: "Item #"}

}

Column{

y: 150

spacing: 30

width: *parent*.width

Button {

id: *addButton*

anchors.horizontalCenter: *parent*.horizontalCenter

text: "Add item"

onClicked: *dataModel*.append({idx:*dataModel*.rowCount(),text: "Item #" })

}

SilicaListView {

id: *sil*

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

spacing: 10

model: *dataModel*

delegate: Rectangle {

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

Text {

anchors.centerIn: *parent*

text: model.text + model.idx

}

MouseArea{

anchors.fill: *parent*

onClicked: {

*dataModel*.remove(index)

}

}

}

}

}

}

//task\_3.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page3*

property var dataModel: [

{ idx: 1, name: "Белый", backgrcolor: "#ffffff", textcolor: "black" },

{ idx: 3, name: "Черный", backgrcolor: "#000000", textcolor: "white" },

{ idx: 2, name: "Синий", backgrcolor: "#0000ff", textcolor: "red" }

]

SilicaListView {

anchors.fill: *parent*

header: PageHeader { title: "Simple List" }

model: *page3*.dataModel

spacing: 10

delegate: Rectangle {

width: *parent*.width

height: 100

color: modelData.backgrcolor

Text {

anchors.centerIn: *parent*

color: modelData.textcolor

font.pixelSize : 35

text: modelData.name || "empty text"

}

}

}

}

//task\_4.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

import QtQuick.XmlListModel 2.0

Page {

XmlListModel {

id: *xmlListModel*

source: "http://www.cbr.ru/scripts/XML\_daily.asp"

query: "/ValCurs/Valute"

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

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

}

SilicaListView {

anchors.fill: *parent*

header: PageHeader { title: "Валюты" }

model: *xmlListModel*

spacing: 10

delegate: Column {

x: 10; width: *parent*.width

Label {

width: *parent*.width

wrapMode: Text.WordWrap

text: Name + "\n" + Value

}

}

VerticalScrollDecorator{}

}

}

//task\_5.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

import QtQuick.XmlListModel 2.0

Page {

// loadValutes()

XmlListModel {

id: *xmlListModel*

query: "/ValCurs/Valute"

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

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

}

SilicaListView {

anchors.fill: *parent*

header: PageHeader { title: "Валюты" }

model: *xmlListModel*

delegate: Column {

x: 10; width: *parent*.width

Label {

width: *parent*.width

wrapMode: Text.WordWrap

text: Name + "\n" + Value

}

}

VerticalScrollDecorator{}

}

Component.onCompleted: *loadValutes*()

function *loadValutes*() {

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();

}

}

//task\_6.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

import QtQuick.LocalStorage 2.0

Page {

id: *page6*

PageHeader {id: *header* ; title: "Data Base"}

property var db

Item{

Component.onCompleted: {

*db* = LocalStorage.openDatabaseSync("Notes","1.0"/\*,"DataBase Notes",100000\*/)

*db*.transaction(function(tx) {

*tx*.executeSql("CREATE TABLE IF NOT EXISTS Notes

(id INTEGER PRIMARY KEY AUTOINCREMENT,

txt\_note TEXT)");

})

}

}

Column{

id:*col*

spacing: 50

y: 150

width: *parent*.width

Button{

id:*butt*

anchors.horizontalCenter: *parent*.horizontalCenter

text: "Добавить заметку"

onClicked: {

//console.log("работает")

*add\_note*(*txt*.text)

*update\_notes\_list*()

}

}

TextField{

id:*txt*

width: *parent*.width

color: "black"

placeholderText: "введите текст заметки"

}

SilicaListView{

id:*notes\_list*

model: ListModel{id: *list*}

anchors.top: *col*.bottom

//anchors.topMargin: 200

height: 1000

orientation: *Qt*.Vertical

spacing: 10

delegate: Rectangle{

width: *page6*.width

height: 80

border.width: 2

color: "transparent"

border.color: "black"

Text{

text: note

color: "red"

anchors.centerIn: *parent*

}

MouseArea{

anchors.fill: *parent*

onClicked: {

*delete\_note*(model.id)

*update\_notes\_list*()

}

}

}

VerticalScrollDecorator{flickable: *notes\_list*}

}

}

function *get\_notes*(callback) {

*db*.readTransaction(function(tx) {

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

*callback*(*result*.rows);

});

}

function *add\_note*(txt\_note) {

*db*.transaction(function(tx) {

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

});

*console*.log("Добавлена заметка: " + *txt\_note*)

}

function *update\_notes\_list*() {

*list*.clear();

*get\_notes*(function(notes) {

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

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

*list*.append({ id: *note*.id,/\*checked:"",\*/ note: *note*.txt\_note });

}

});

}

function *delete\_note*(id) {

*db*.transaction(function(tx) {

*tx*.executeSql("DELETE FROM Notes WHERE id = ?;", [*id*]);

});

}

}

// task\_7.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

import Nemo.Configuration 1.0

Page {

PageHeader { title: "ConfigurationValue" }

ConfigurationValue {

id: *val1*

key: "/ "

value: "/ "

}

ConfigurationValue {

id: *val2*

key: "/ "

value: "/ "

}

Column {

width: *parent*.width

y: 200

TextField {

id: *txt\_f*

placeholderText: "enter smth"

text: ""

}

Button {

width: *parent*.width

text: "save val1"

onClicked: {

var *saved\_val\_1* = *txt\_f*.text

*val1*.value = *saved\_val\_1*

*console*.log("current value 1 = " + *val1*.value)

}

}

TextSwitch {

id: *txt\_sw*

text: "Switch"

description: "description"

onCheckedChanged: {

var *saved\_val\_2* = *txt\_sw*.checked ? "Die" : "Live"

*val2*.value = *saved\_val\_2*

*console*.log("current value 2 = " + *val2*.value)

}

}

}

}

// task\_8.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

import Nemo.Configuration 1.0

Page {

PageHeader { title: "ConfigurationGroup" }

ConfigurationGroup {

id: *conf\_gr*

path: "/ "

}

Column {

width: *parent*.width

y: 200

TextField {

id: *txt\_f*

placeholderText: "enter smth"

text: ""

}

Button {

width: *parent*.width

text: "save val1"

onClicked: {

var *saved\_val\_1* = *txt\_f*.text

*conf\_gr*.setValue("val1", *saved\_val\_1*);

*console*.log("current value 1 = " + *conf\_gr*.value("val1") /\*+ " " + saved\_val\_1\*/)

}

}

TextSwitch {

id: *txt\_sw*

text: "Switch"

description: "description"

onCheckedChanged: {

var *saved\_val\_2* = *txt\_sw*.checked ? "Die" : "Live"

*conf\_gr*.setValue("val2", *saved\_val\_2*);

*console*.log("current value 2 = " + *conf\_gr*.value("val2") /\*+ " " + saved\_val\_2\*/)

}

}

}

}