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 №3**

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](#_Toc85441990)

[2. Постановка задачи 4](#_Toc85441991)

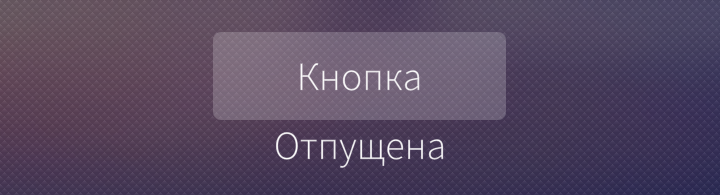
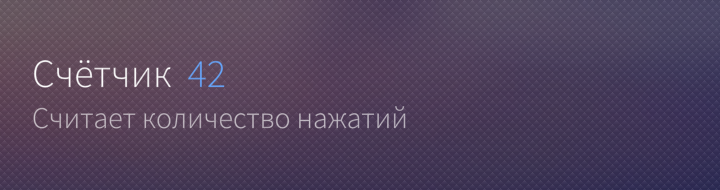
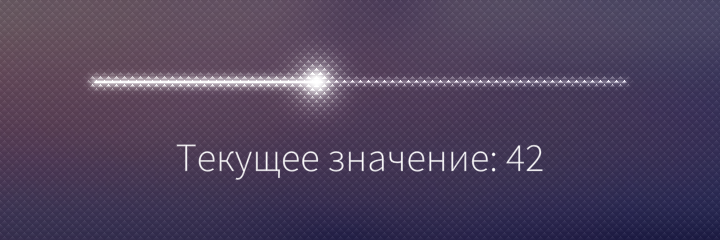
[3. Решение поставленной задачи 6](#_Toc85441992)

[4. Руководство пользователя 9](#_Toc85441993)

[Приложение 13](#_Toc85441994)

1. Цель

Цель: научиться применять типовые элементы интерфейса Sailfish OS

1. Постановка задачи
2. создать текстовое поле для ввода числа с заголовком и подсказкой
3. создать кнопку, которая будет сохранять визуально нажатое состояние, после того, как пользователь нажал на неё один раз
4. создать кнопку и поле с текстом. Поле с текстом должно отображать нажата ли кнопка или нет выводом текста “Нажата” или “Отпущена”
5. создать кнопку со значением, которая будет отображать количество нажатий на неё
6. создать селектор даты, который будет отображать выбранную дату в консоли
7. создать селектор времени, который будет отображать выбранное время в консоли
8. создать поле с выпадающим списком, позволяющее выбрать строку из списка. Результат выбора отобразить в консоли
9. создать переключатель с текстом, в тексте отобразить состояние переключателя “Включен” или “Выключен”
10. создать ползунок и поле с текстом. Поле с текстом должно отображать текущее значение ползунка
11. Решение поставленной задачи

Создаём проект со стандартной заготовкой приложения.

Так как все пункты являются отдельными заданиями, то, чтобы не возникло проблем с размещением всех элементов в одном месте, мы разделим их на отдельные страницы. Для этого в папке <имя проекта>/qml/pages нашего проекта создаём дополнительные файлы через меню (например, правой кнопки мыши) — нажимаем «Добавить новый…», после этого в появившемся окне выбираем создать файл QML (Qt Quick 2). Внутри каждой созданной страницы прописываем import Sailfish.Silica 1.0 и заменяем элемент Item на Page (где это нужно) и прописываем allowedOrientations: Orientation.All. После мы переходим в файл FirstPage.qml и внутри PullDownMenu прописываем недостающие элементы MenuItem по аналогии с уже существующим для SecondPage.qml.

Задание 1.

Внутри Page создаём элемент TextField, внутри которого указываем свойства label (для заголовка поля) и placeholderText (для подсказки при вводе). Можно также довить свойство inputMethodHints: Qt.ImhDigitsOnly для ввода символов, предназначенных для записи чисел.

Задание 2.

Внутри Page создаём элемент Button и указываем его id. В его свойстве text пишем название кнопки. Прописываем метод onClicked, в котором через id кнопки обращаемся к свойству down (указывает нажата кнопка или нет) и устанавливаем его значение как true.

Задание 3.

Внутри Page создаём элемент Button и указываем его id. В его свойстве text пишем название кнопки. Также создаем элемент Text и указываем его id. В его свойстве text пишем «Отпущена». Для определения, когда кнопка отпущена и когда нажата, можем использовать число кликов, которое мы совершили. Для этого создаем переменную для хранения числа нажатий и инициализируем её 0. Прописываем метод onClicked. В нем увеличиваем счетчик переменной на 1. Также через id текстового поля обращаемся к свойству text: если переменная имеет четное значение, то в свойстве text пишем «Отпущена», иначе – «Нажата». Аналогично обращаемся через id кнопки к свойству down и устанавливаем ему значение false при четном числе нажатий и true – при нечетном.

Задание 4.

Внутри Page создаём элемент ValueButton и указываем его id. Создаем свойства label для заголовка и description для описания кнопки. Также прописываем свойство value для хранения значения кнопки, где ссылаемся на значение переменной. Создаем переменную для хранения числа нажатий и инициализируем её 0. Прописываем метод onClicked. В нем увеличиваем счетчик переменной на 1.

Задание 5.

Внутри Page создаём элемент DatePicker и указываем его id. В нем устанавливаем значение true для свойств daysVisible (видимость дней), weeksVisible (видимость недель) и monthYearVisible (видимость текущего месяца и года). Для отображения выбранной даты в консоли прописываем метод onDateTextChanged, где внутри скобок console.log() пишем обращение через id к свойству dateText.

Задание 6.

Внутри Page создаём элемент Item. Внутри создаем элемент TimePicker и указываем его id. Для отображения выбранного времени в консоли прописываем метод onTimeTextChanged, где внутри скобок console.log() пишем обращение через id к свойству timeText. Для видимости текущего времени можно создать элемент Label (лучше поместить его по центру TimePicker), внутри которого прописываем его id и в свойстве text обращаемся к timeText.

Задание 7.

Внутри Page создаём элемент ComboBox и прописываем свойство label для заголовка. Чтобы выпадающий список содержал какие-то строки выбора внутри себя, нужно прописать menu: ContextMenu {}, где внутри фигурных скобок создать нужное количество MenuItem{} и для каждого из них прописать свойство text. Чтобы выбранная строка отображалась в консоли, прописываем метод onCurrentIndexChanged, где внутри скобок console.log() пишем value для отображения значения (т.е. текста) текущей строки.

Задание 8.

Так как есть несколько вариантов переключателей с текстом, то для выполнения задания можно выбрать любой из них. Например, TextSwitch. Для этого внутри Page создаём элемент TextSwitch, в котором указываем его id и свойство text для подписи. Создаем свойство description для описания переключателя, где через id обращаемся к свойству checked (указывает на текущее состояние переключателя). Если checked находится в состоянии true, то в описании переключателя пишем "Включён", иначе - "Выключен".

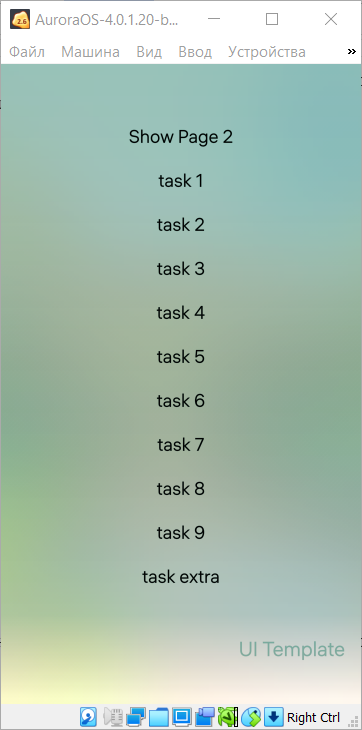
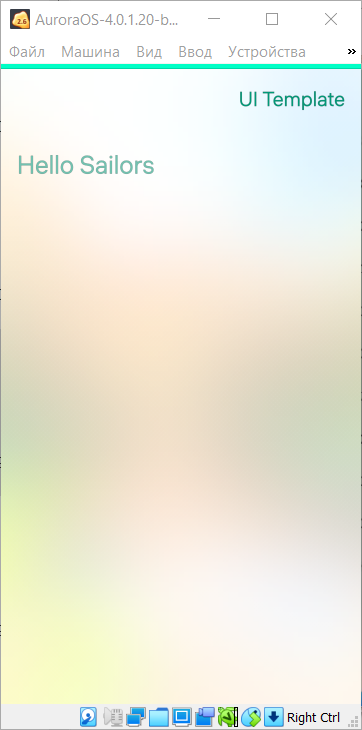
Задание 9.

Внутри Page создаём элемент Slider и указываем его id. Также пишем его заголовок через label и устанавливаем желаемые значения для свойств maximumValue (максимальное значение ползунка), minimumValue (минимальное значение ползунка), value (текущее значение ползунка) и stepSize (шаг изменения значения). В свойстве valueText ссылаемся на value. Для создания поля с текстом дополнительно создаем элемент Label, в свойстве text которого обращаемся через id ползунка к value.

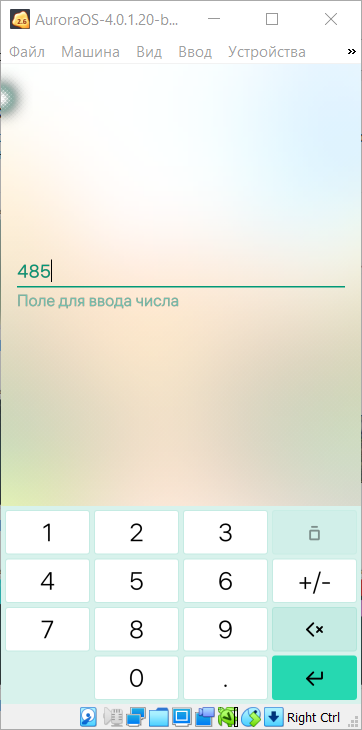
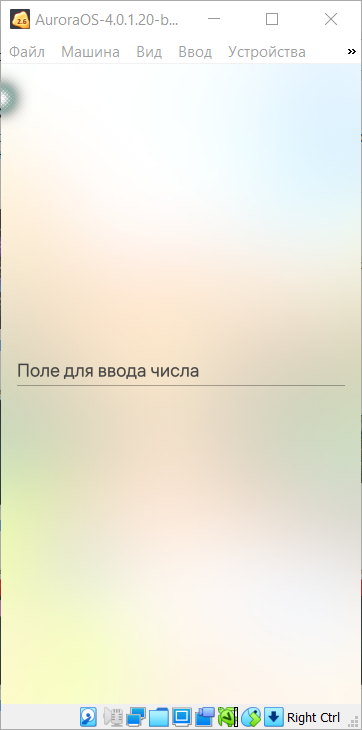
Осталось только сохранить все изменения и запустить эмулятор.

1. Руководство пользователя

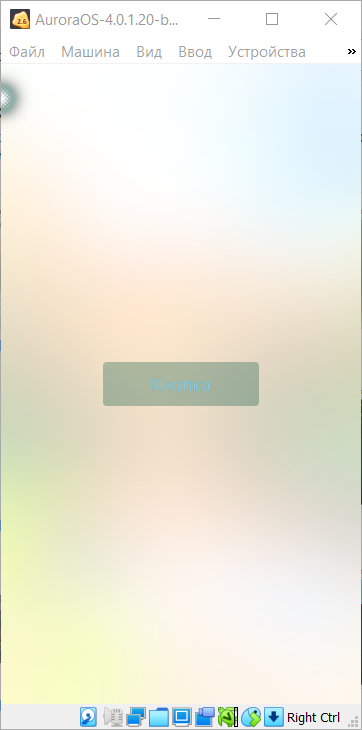
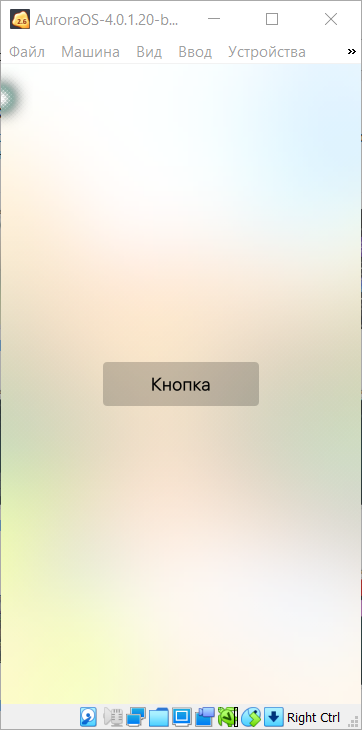
При запуске приложения пользователь попадает на страницу FirstPage. Потянув вниз, открывается меню с нашими страницами.



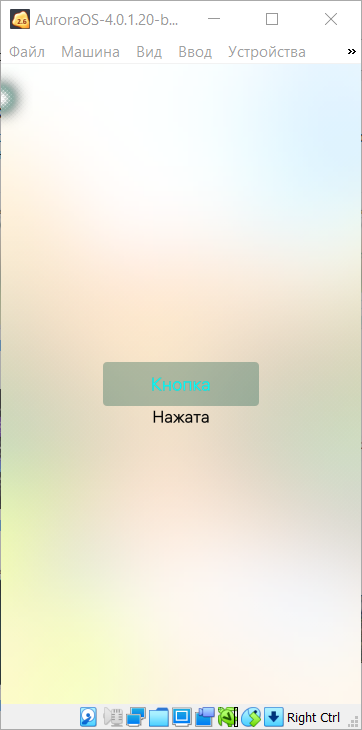
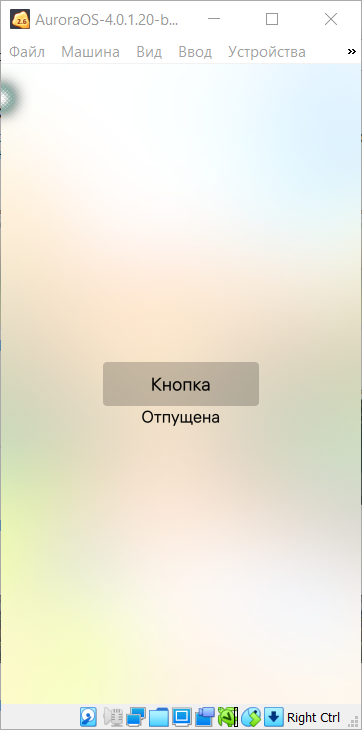
При нажатии на каждую из страниц, она загрузит нам наше задание. Страница 1:



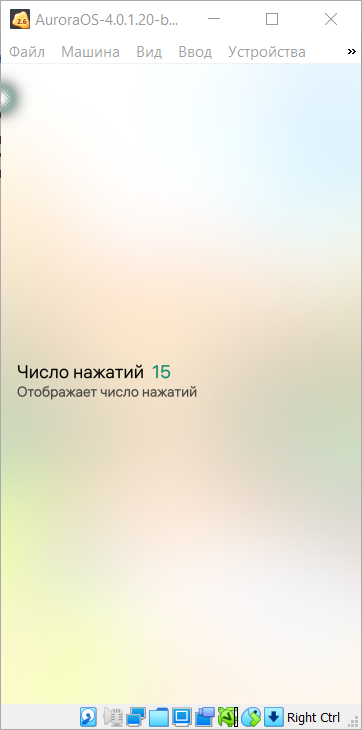
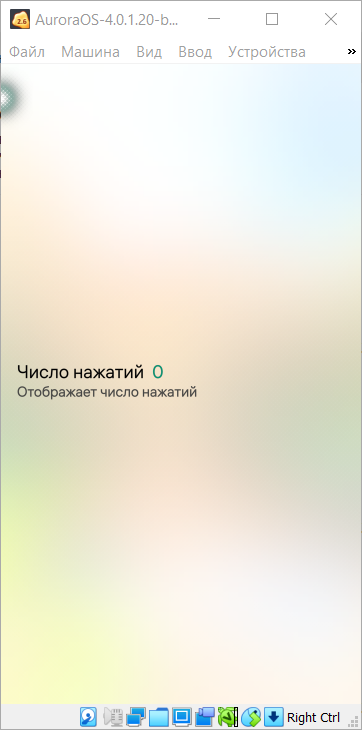
Страница 2:



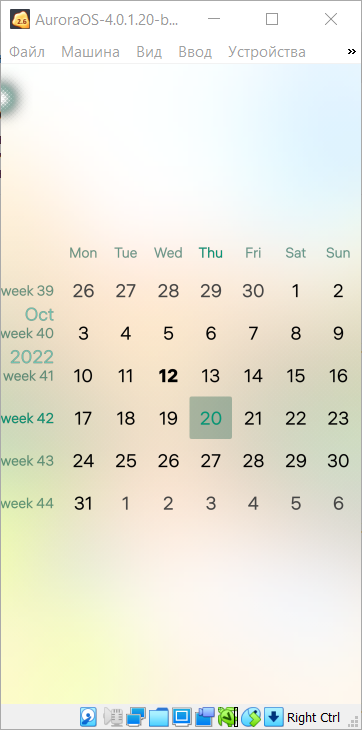
Страница 3:



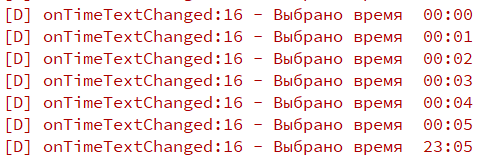
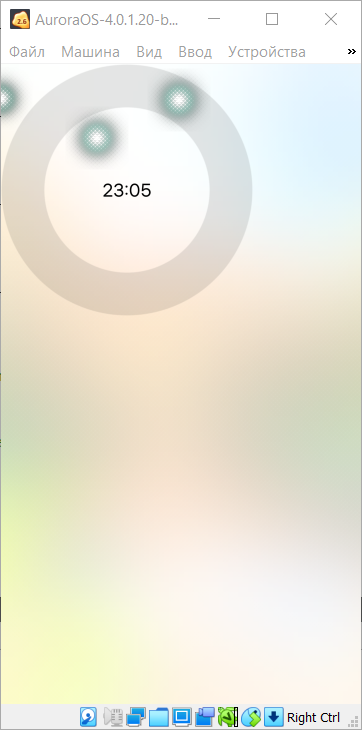
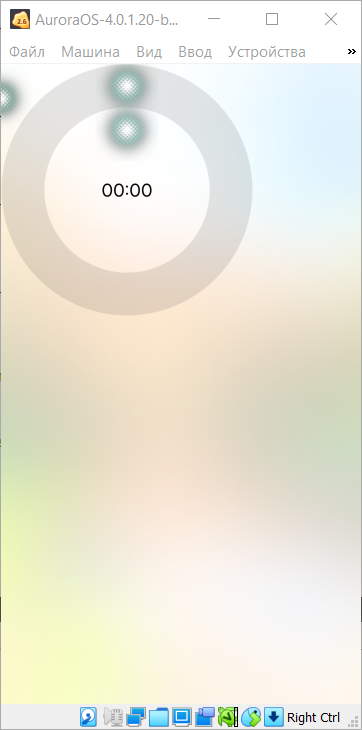
Страница 4:



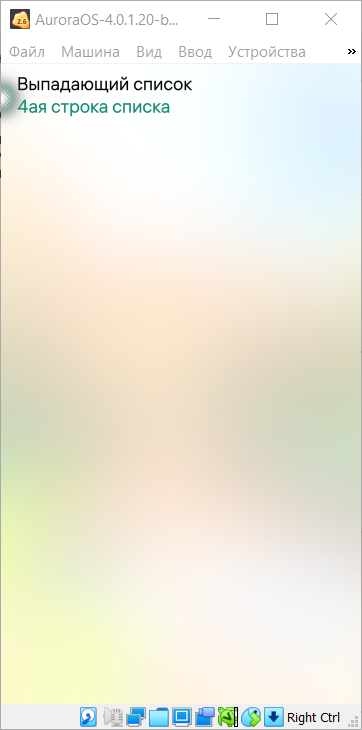
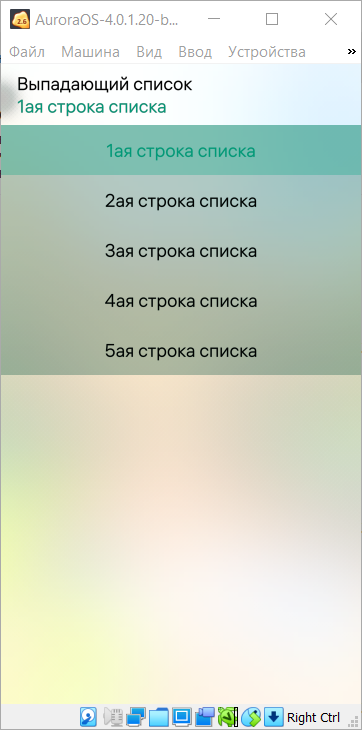
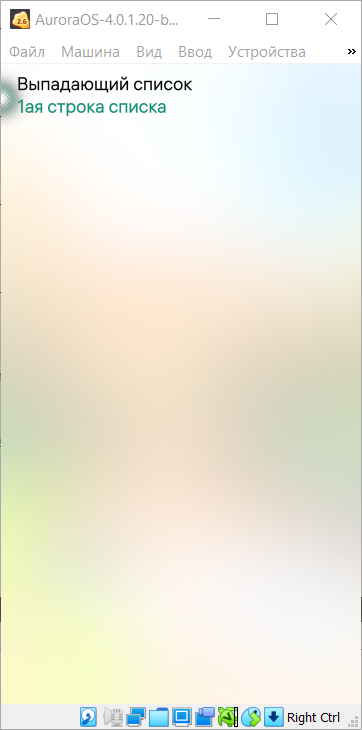
Страница 5:



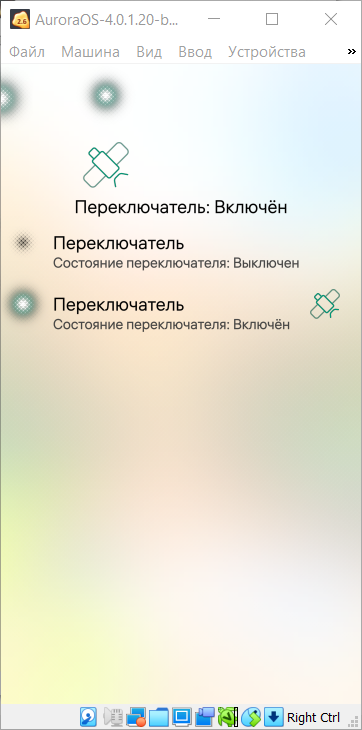
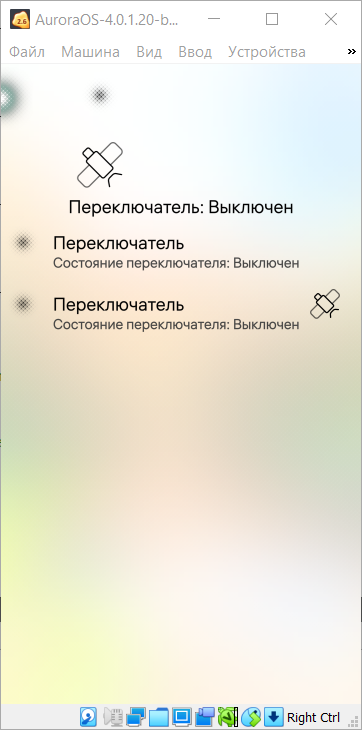
Страница 6:



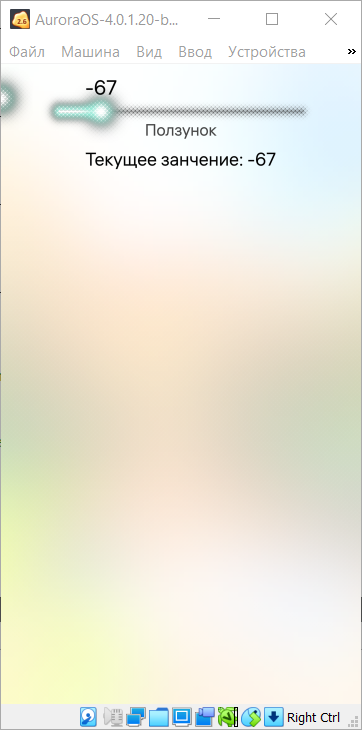
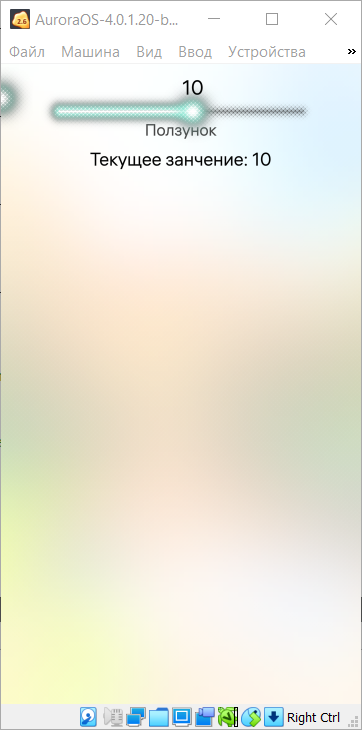
Страница 7:



Страница 8:



Страница 9:



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"))

}

MenuItem {

text: *qsTr*("task 9")

onClicked: *pageStack*.push(*Qt*.resolvedUrl("task\_9.qml"))

}

MenuItem {

text: *qsTr*("task extra")

onClicked: *pageStack*.push(*Qt*.resolvedUrl("extra\_task.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 {

id: *page1*

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

allowedOrientations: Orientation.All

TextField{

id: *textField*

anchors.horizontalCenter: *parent*.horizontalCenter

anchors.verticalCenter: *parent*.verticalCenter

label: "Поле для ввода числа"

placeholderText: *label*

inputMethodHints: *Qt*.ImhDigitsOnly

}

}

//task\_2.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page2*

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

allowedOrientations: Orientation.All

Button {

id: *butt*

anchors.centerIn: *parent*

text: "Кнопка"

highlightColor: "skyblue"

onClicked: {

//console.log("Вы нажали кнопку")

*butt*.down = true

}

}

}

//task\_3.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page3*

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

allowedOrientations: Orientation.All

Button {

id: *butt*

anchors.centerIn: *parent*

text: "Кнопка"

highlightColor: "cyan"

property int count: 0

onClicked: {

*butt*.count++;

*txt*.text = (*butt*.count % 2 == 0) ? "Отпущена" : "Нажата"

*butt*.down = (*butt*.count % 2 == 0) ? false : true

}

Text {

id: *txt*

anchors.top: *parent*.bottom

color: "black"

font.pointSize: 25

anchors.horizontalCenter: *parent*.horizontalCenter

text: "Отпущена"

}

}

}

//task\_4.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page4*

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

allowedOrientations: Orientation.All

ValueButton {

id: *valButt*

anchors.verticalCenter: *parent*.verticalCenter

anchors.horizontalCenter: *parent*.horizontalCenter

label: "Число нажатий"

description: "Отображает число нажатий"

value: *valButt*.count

property int count: 0

onClicked: {

*valButt*.count++;

}

}

}

//task\_5.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page5*

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

allowedOrientations: Orientation.All

DatePicker{

id: *datePick*

anchors.verticalCenter: *parent*.verticalCenter

daysVisible: true

weeksVisible: true

monthYearVisible: true

onDateTextChanged: *console*.log("Выбрана дата: " + *datePick*.dateText)

}

}

//task\_6.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page6*

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

allowedOrientations: Orientation.All

Item {

height: *timePick*.height

width: *timePick*.width

TimePicker {

id: *timePick*

anchors.centerIn: *parent*

onTimeTextChanged: *console*.log("Выбрано время " + *timePick*.timeText)

}

Label{

id: *tm*

anchors.centerIn: *timePick*

text: *timePick*.timeText

}

}

}

// task\_7.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page7*

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

allowedOrientations: Orientation.All

ComboBox{

id: *comb*

label: "Выпадающий список"

menu: ContextMenu {

MenuItem{text: "1ая строка списка"}

MenuItem{text: "2ая строка списка"}

MenuItem{text: "3ая строка списка"}

MenuItem{text: "4ая строка списка"}

MenuItem{text: "5ая строка списка"}

}

onCurrentIndexChanged: *console*.log("Вы изменили выбор на строку с индексом [" + *currentIndex* + "]. Она содержит следующий текст: " + *value*)

}

}

// task\_8.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page8*

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

allowedOrientations: Orientation.All

Column {

id: *coll1*

anchors.horizontalCenter: *parent*.horizontalCenter

Switch {

id: *sw1*

icon.source: "image://theme/icon-l-gps?"

+ (*checked* ? Theme.highlightColor

: Theme.primaryColor)

}

Label {

text: "Переключатель: " + (*sw1*.checked ? "Включён" : "Выключен")

}

}

Column {

id: *coll2*

width: *parent*.width

anchors.top: *coll1*.bottom

TextSwitch {

id: *sw2*

text: "Переключатель"

description: "Состояние переключателя: " + (*sw2*.checked ? "Включён" : "Выключен")

}

}

IconTextSwitch {

id: *sw3*

anchors.top: *coll2*.bottom

text: "Переключатель"

description: "Состояние переключателя: " + (*sw3*.checked ? "Включён" : "Выключен")

icon.source: "image://theme/icon-m-gps?" + (*checked* ? Theme.highlightColor : Theme.primaryColor)

}

}

// task\_9.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page9*

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

allowedOrientations: Orientation.All

Slider {

id: *sl*

width: *parent*.width

label: "Ползунок"

maximumValue: 100

minimumValue: -100

value: 10

stepSize: 1

valueText: *value*

}

Label {

anchors.horizontalCenter: *parent*.horizontalCenter

anchors.top: *sl*.bottom

text: "Текущее занчение: " + *sl*.value

}

}