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 №4**

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)

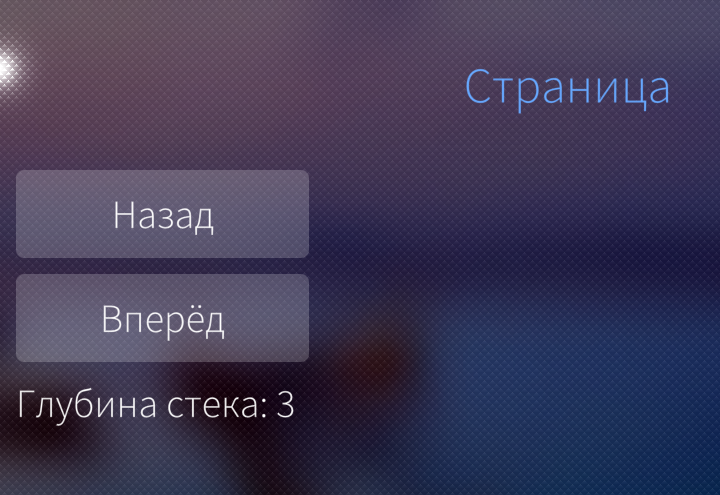
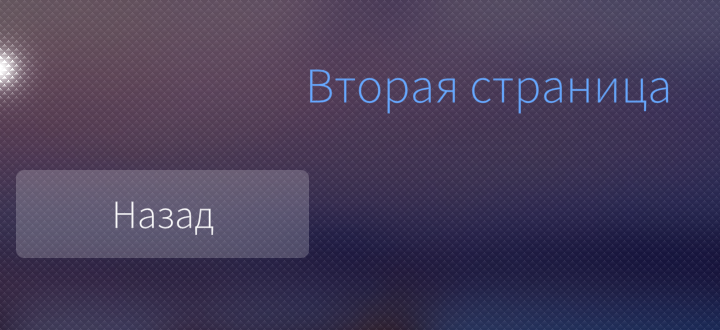
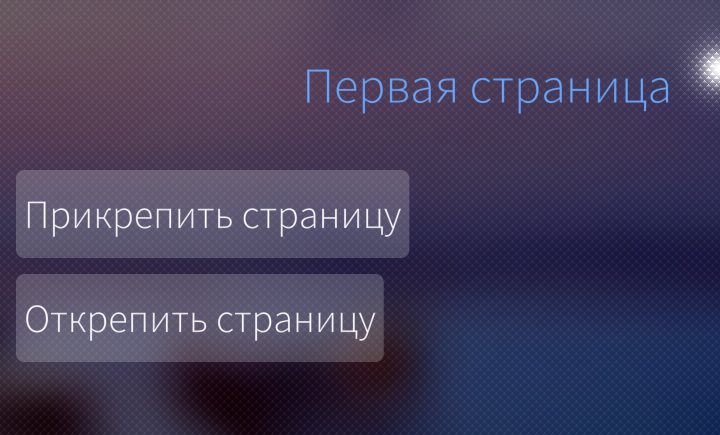
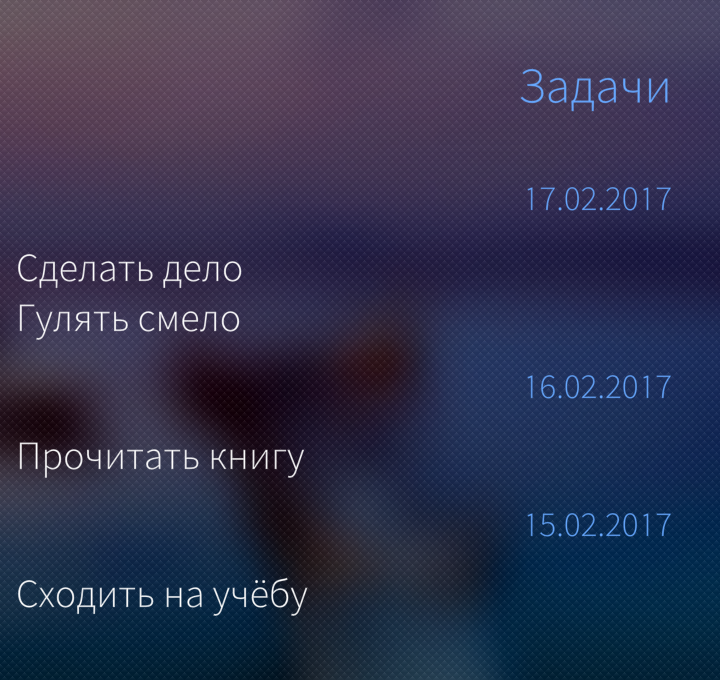
[3. Решение поставленной задачи 6](#_Toc85441992)

[4. Руководство пользователя 9](#_Toc85441993)

[Приложение 13](#_Toc85441994)

1. Цель

Цель: научиться организовывать многостраничное приложение, использовать контейнеры Silica, вытягиваемые меню и обложку приложения.

1. Постановка задачи
2. Создать приложение, которое будет отображать страницу с двумя кнопками “Назад” и “Вперёд”. Первая удалит текущую страницу со стека, вторая добавит новую. Также на экране нужно отображать текущую глубину стека. 
3. Создать приложение из двух страниц. Первая страница содержит две кнопки “Добавить страницу” и “Убрать страницу”. Первая кнопка добавит вторую страницу как прикреплённую, вторая кнопка её удалит. На второй странице должна быть кнопка для возврата на первую страницу без закрытия второй. 
4. Создать приложение с одной кнопкой и текстовом поле. После нажатия на кнопку отображается диалог для ввода текста. После согласия с результатом введённый текст отображается в текстовое поле.
5. Создать приложение с одной кнопкой и текстовым полем. После нажатия на кнопку отображается диалог для выбора даты. После согласия с результатом ввода выбранная дата отображается в текстовое поле.
6. Создать приложение с одной кнопкой и текстовым полем. После нажатия на кнопку отображается диалог для выбора времени. После согласия с результатом ввода выбранное время отображается в текстовом поле.
7. Создать приложение со списком SilicaListView, из задач на неделю. Задачи должны содержать дату и описание. В списке задачи группировать по датам. 
8. Создать приложение с SilicaWebView для доступа к вашему любимому сайту.
9. Использовать SlideshowView для просмотра и перелистывания задач на неделю. На одном слайде – одна задача.
10. Создать приложение с вытягиваемыми меню сверху и снизу и текстовым полем. После выбора какого-либо элемента меню, его название отобразить в текстовом поле.
11. Создать приложение со списком и контекстным меню. После выбора элемента контекстного меню отобразить в консоли название выбранного элемента меню и индекс элемента списка.
12. Создать приложение с обложной-счётчиком. На обложке отобразить текущий счёт и две кнопки для добавления единицы к счёту и для сброса счётчика.



1. 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 Column, внутри которого создаем элементы PageHeader (для заголовка), 2 элемента Button (для навигации) и элемент Text (для текстового поля). Для первой кнопки прописываем её название, а также в методе onClicked вызываем pageStack.pop() для удаления страницы. Для 2 кнопки всё аналогично, только вместо pop() вызываем push(), где в скобках указываем страницу для вызова (в нашем случае название файла qml). В текстовом поле в свойстве text передаём глубину стека через свойство depth.

Задание 2.

Файл основной страницы.

Внутри Page создаём элемент Column, внутри которого создаем элементы PageHeader (для заголовка), 2 элемента Button (для навигации). Для первой кнопки прописываем её название, а также в методе onClicked вызываем pageStack.popAttached() для удаления прикрепленной страницы. Для 2 кнопки всё аналогично, только вместо popAttached() вызываем pushAttached(), где в скобках указываем страницу для вызова (в нашем случае название файла qml прикрепленной страницы).

Файл прикрепленной страницы.

Внутри Page создаём элемент Column, внутри которого создаем элементы PageHeader (для заголовка), элемент Button (для навигации). Для кнопки прописываем её название, а также в методе onClicked вызываем pageStack.navigateBack() для возврата на 1 страницу.

Задание 3.

Файл для вызова диалога.

Внутри Page создаём пустую строковую переменную, элемент Button и элемент Text. В элементе Text прописываем свойство text, куда передаем значение строковой переменной. В кнопке пишем её название и в методе onClicked создаем переменную, в которую передаем страницу с водом диалога, а также прописываем метод для принятия диалога accepted.connect(function(){}), где в фигурных скобках присваиваем строковой переменной значение аналогичной переменной из файла диалога.

Файл с кодом диалога.

Внутри элемента Dialog создаем элемент TextField для ввода текста и указываем его id. Также создаем переменную строкового типа, куда передаем текст из TextField через его id.

Задание 4.

Внутри Page создаём пустую строковую переменную, элемент Button и элемент Text. В элементе Text прописываем свойство text, куда передаем значение строковой переменной. В кнопке пишем её название и в методе onClicked создаем переменную, в которую методом push() передаем страницу с кодом диалога (в качестве страницы указываем Sailfish.Silica.DatePickerDialog, а вторым параметром {date: new Date()} для начальной даты), а также прописываем метод для принятия диалога accepted.connect(function(){}), где в фигурных скобках присваиваем строковой переменной значение dateText.

Задание 5.

Внутри Page создаём пустую строковую переменную, элемент Button и элемент Text. В элементе Text прописываем свойство text, куда передаем значение строковой переменной. В кнопке пишем её название и в методе onClicked создаем переменную, в которую методом push() передаем страницу с кодом диалога (в качестве страницы указываем Sailfish.Silica.TimePickerDialog, а вторым параметром {hour: 0, minute: 0} для начального времени), а также прописываем метод для принятия диалога accepted.connect(function(){}), где в фигурных скобках присваиваем строковой переменной значение timeText.

Задание 6.

Внутри Page создаём элемент ListModel и указываем его id. В нем создаем нужное нам количество элементов ListElement, для каждого из которых прописываем свойства name для описания задачи и date для указания даты. Все эти элементы будут являться нашей моделью. Также внутри Page создаем элемент SilicaListView, для которого указываем в model id нашей модели, прописываем свойство header (куда передаем PageHeader для заголовка страницы), отделяем секцию с помощью section, в котором указываем свойство property как date и с помощью delegate выделяем заголовки секции. Также с помощью delegate передаем описание задачи в Text.

Задание 7.

Внутри Page создаём элемент TextField, который позиционируем внизу страницы, внутри которого прописываем свойство text, куда передаем url нашего сайта. Также создаем элемент SilicaWebView, который позиционируем так, чтобы он занимал всю страницу от её верха до верха элемента TextField. Внутри прописываем свойство url, куда передаем адрес нашего сайта как строку.

Задание 8.

Внутри Page создаём элемент ListModel и указываем его id. В нем создаем нужное нам количество элементов ListElement, для каждого из которых прописываем свойства name для описания задачи и date для указания даты. Все эти элементы будут являться нашей моделью. Также внутри Page создаем элемент SilicashowView, для которого указываем в model id нашей модели, с помощью delegate передаем квадрат Rectangle (для создания ощущения слайд-шоу), внутри которого создаем элемент Text, где в свойстве text ссылаемся на дату и описание задач из модели.

Задание 9.

Внутри Page создаём пустую строковую переменную, элементы PullDownMenu и PushUpMenu, а также элемент Text, куда в свойство text записываем значение нашей переменной. Для каждого из меню создаем нужное количество MenuItem, для каждого из которых прописываем их название и в методе onClicked указываем присвоение названия элемента меню в переменную.

Задание 10.

Внутри Page создаём элемент ListModel и указываем его id. В нем создаем нужное нам количество элементов ListElement, для каждого из которых прописываем свойство name для названия элемента. Все эти элементы будут являться нашей моделью. Также внутри Page создаем элемент SilicaListView, для которого указываем в model id нашей модели, с помощью delegate передаем ListItem, где прописываем элемент Label для отображения имени элемента списка и создаем контекстное меню с помощью menu: ContextMenu {}, внутри скобок которого создаем MenuLabel для заголовка меню и нужное количество MenuItem для выбора. Внутри каждого MenuItem необходимо прописать его название и в методе onClicked передать в консоль это название и индекс элемента списка через index.

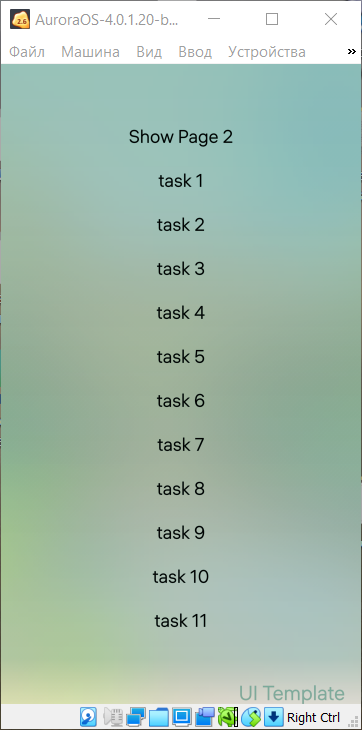
Задание 11.

Для этого задания изменим уже существующий файл CoverPage.qml .Здесь внутри элемента CoverBackground создаем целочисленную переменную для счетчика с каким-то начальным значением. Также для каждого CoverAction из CoverActionList изменяем иконки и с помощью метода onTriggered передаем действия: увеличение значения счетчика и сброс его до начального значения.

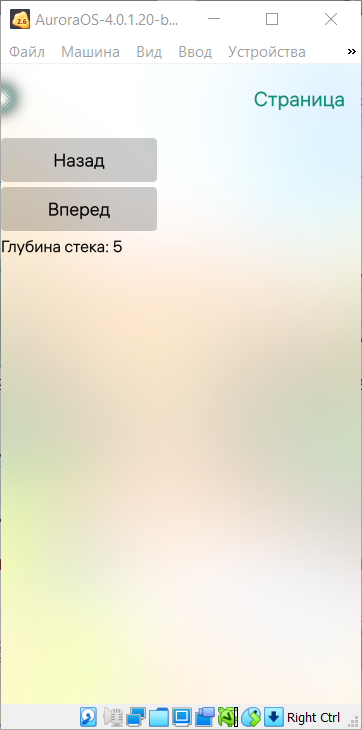
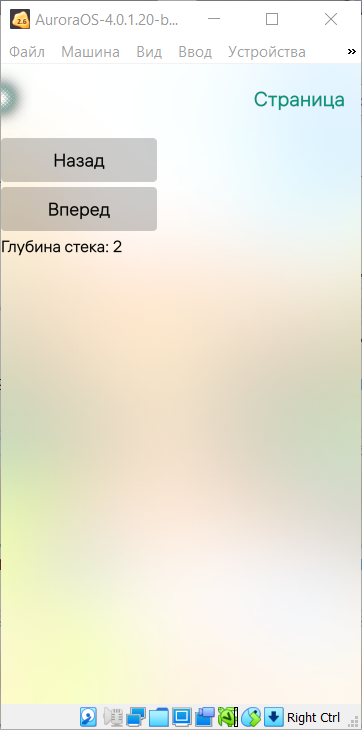
Осталось только сохранить все изменения и запустить эмулятор.

1. Руководство пользователя

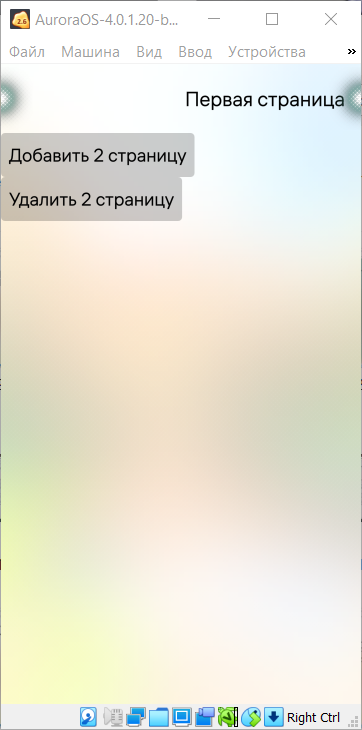
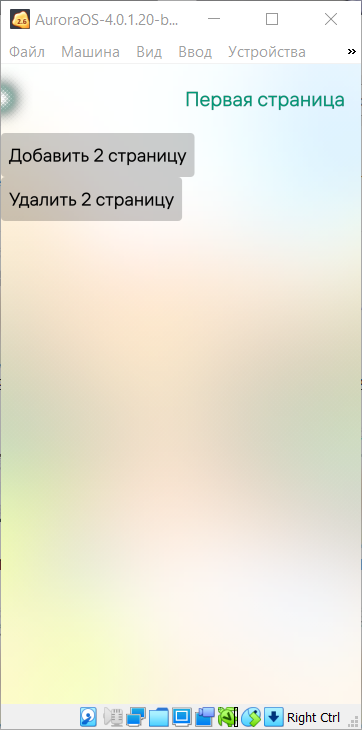
При запуске приложения пользователь попадает на страницу FirstPage. Потянув вниз, открывается меню с нашими страницами.



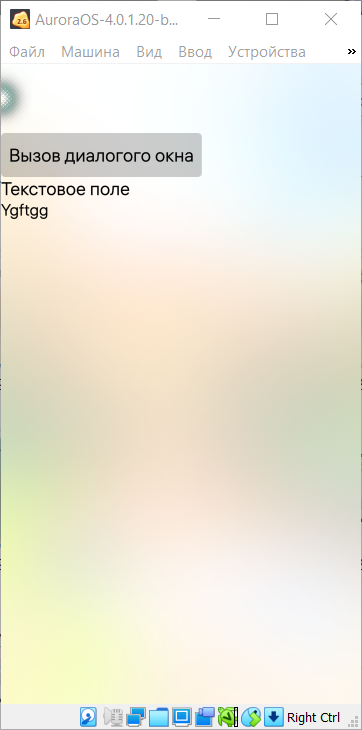
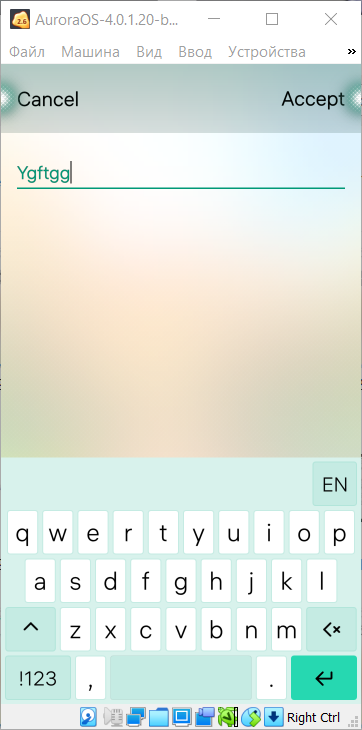
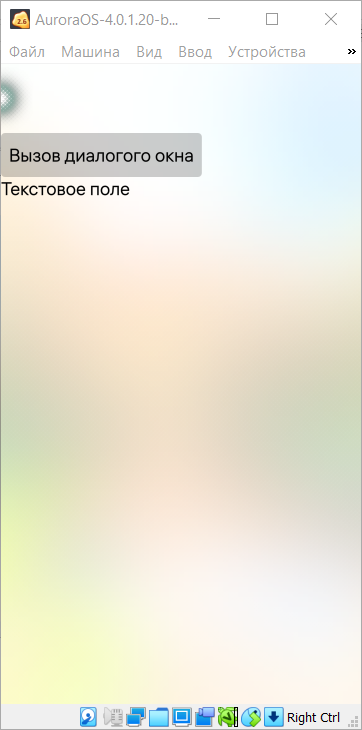
При нажатии на каждую из страниц, она загрузит нам наше задание. Страница 1:



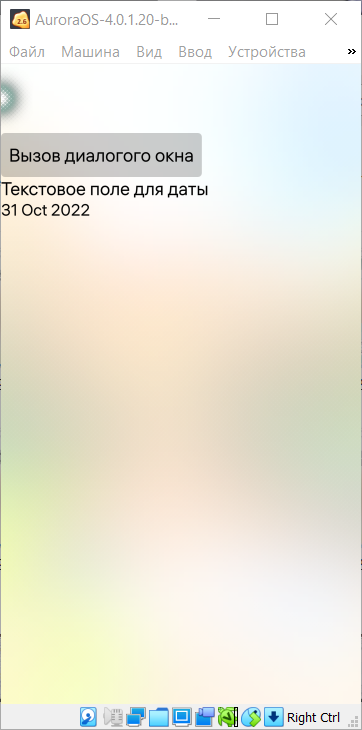
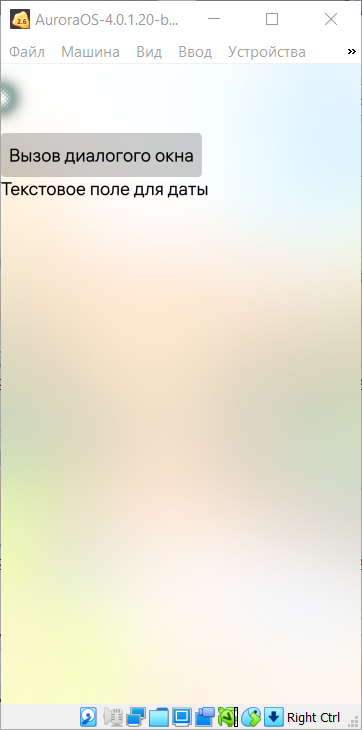
Страница 2:



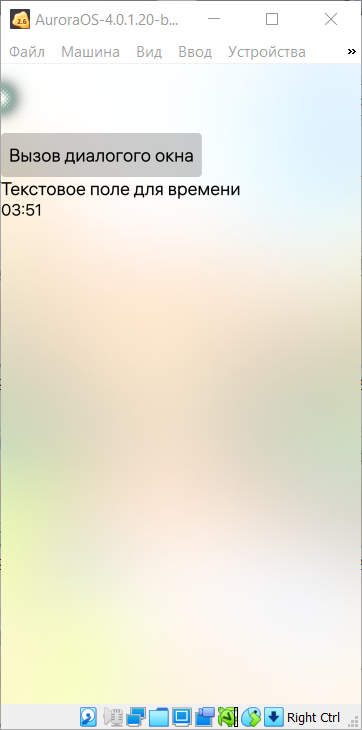
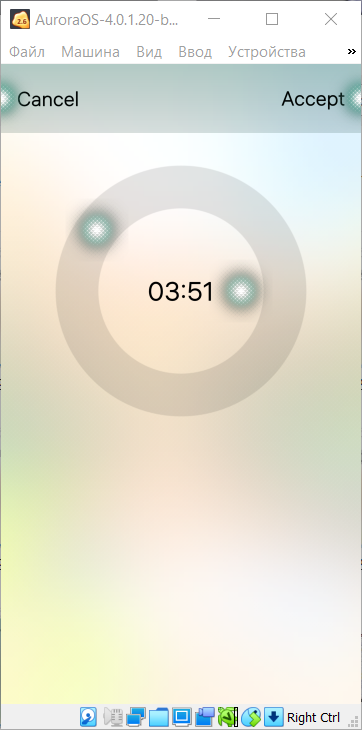
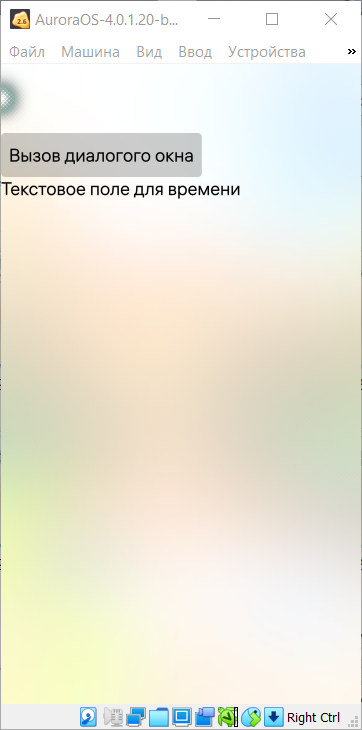
Страница 3:



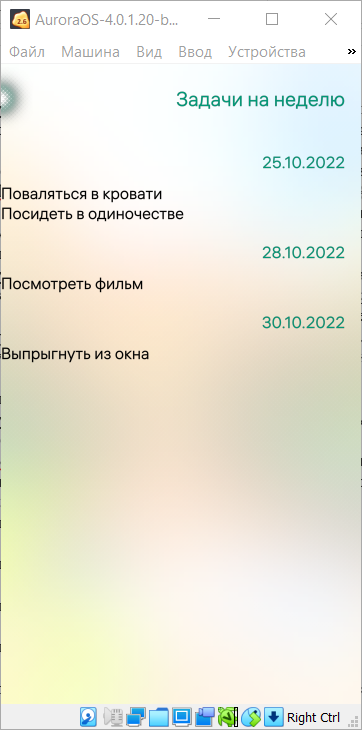
Страница 4:



Страница 5:



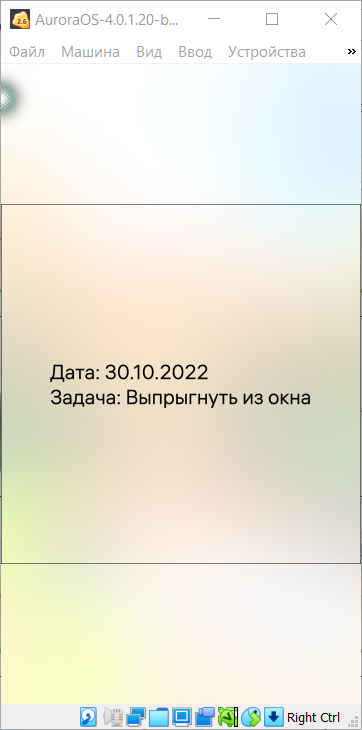
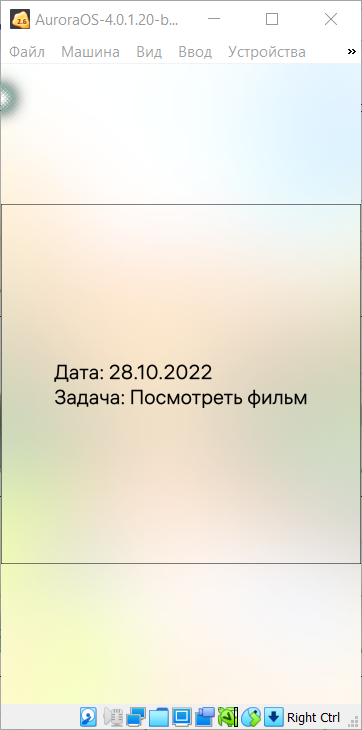
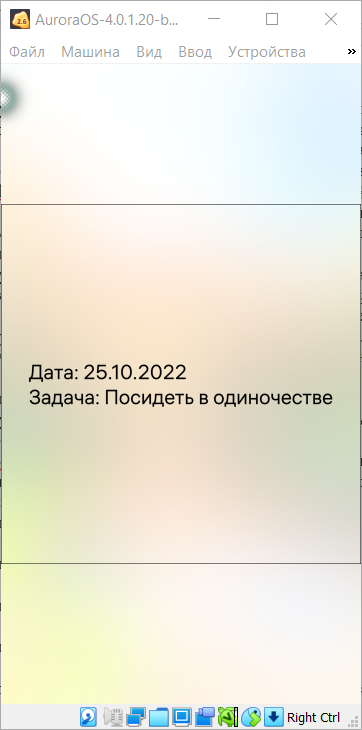
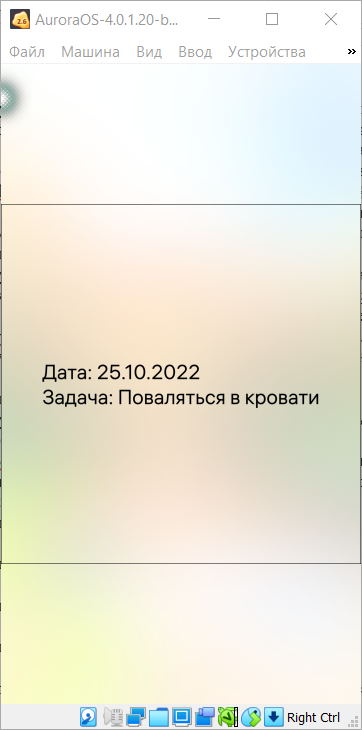
Страница 6:



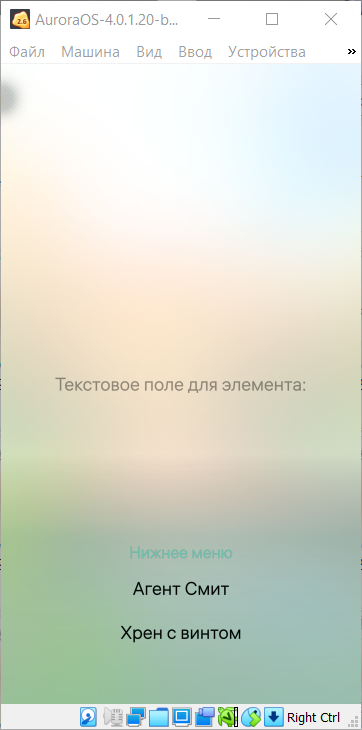
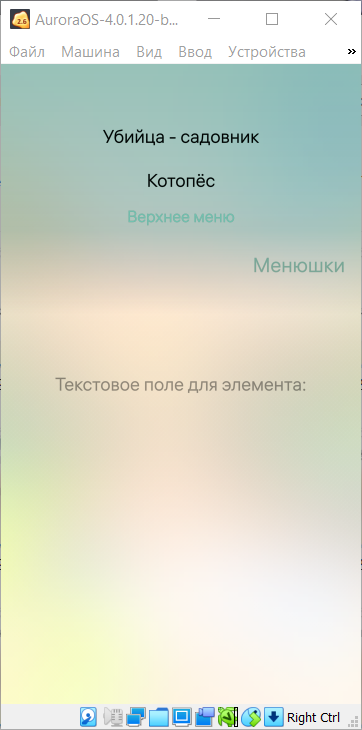
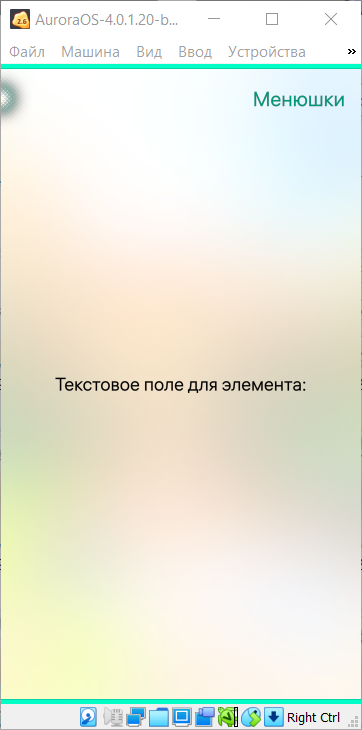
Страница 7:



Страница 8:



Страница 9:



Страница 10:

C:\Users\1\YandexDisk\Скриншоты\2022-10-28_01-01-05.png

Страница 11;



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 10")

onClicked: *pageStack*.push(*Qt*.resolvedUrl("task\_10.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

Column {

anchors.fill: *parent*

spacing: 10

PageHeader {

id: *head*

title: "Страница"

}

Button{

text: "Назад"

onClicked: {

pageStack.pop()

}

}

Button {

text: "Вперед"

onClicked: {

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

}

}

Text{

text: "Глубина стека: " + pageStack.depth

}

}

}

//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

Column {

anchors.fill: *parent*

PageHeader {

title: "Первая страница"

}

Button {

text: "Добавить 2 страницу"

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

}

Button {

text: "Удалить 2 страницу"

onClicked: pageStack.popAttached()

}

}

}

//task\_2\_2.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

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

allowedOrientations: Orientation.All

Column {

anchors.fill: *parent*

PageHeader{title: "Вторая страница"}

Button {

text: "Назад"

onClicked: pageStack.navigateBack()

}

}

}

//task\_3.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page3*

property string txt1 : ""

property string full\_text: "" + *txt1*

Column {

PageHeader{}

Button {

text: "Вызов диалогого окна"

onClicked: {

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

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

*txt1* = *dialog\_text*.txt

})

}

}

Label {text: "Текстовое поле"}

Text {

id: *tx*

text: "" + *txt1*

}

}

}

//dialog\_text.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Dialog {

property string txt: *txField*.text

property string saved\_text: ""

Column {

width: *parent*.width

spacing: Theme.paddingMedium

DialogHeader { }

TextField {

id: *txField*

width: *parent*.width

text: "" + *saved\_text*

}

}

}

//task\_4.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page4*

property string txt1 : ""

Column {

PageHeader{}

Button {

text: "Вызов диалогого окна"

onClicked: {

var *dialog\_date* = pageStack.push("Sailfish.Silica.DatePickerDialog",{date: new *Date*()})

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

*txt1* = *dialog\_date*.dateText

})

}

}

Label {text: "Текстовое поле для даты"}

Text {

id: *tx*

text: "" + *txt1*

}

}

}

//task\_5.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page5*

property string txt1 : ""

Column {

PageHeader{}

Button {

text: "Вызов диалогого окна"

onClicked: {

var *dialog\_time* = pageStack.push("Sailfish.Silica.TimePickerDialog",{hour: 0, minute: 0})

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

*txt1* = *dialog\_time*.timeText

})

}

}

Label {text: "Текстовое поле для времени"}

Text {

id: *tx*

text: "" + *txt1*

}

}

}

//task\_6.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page6*

ListModel {

id: *task\_list*

ListElement { name: "Поваляться в кровати"; date:"25.10.2022";}

ListElement { name: "Посидеть в одиночестве";date:"25.10.2022";}

ListElement { name: "Посмотреть фильм"; date:"28.10.2022";}

ListElement { name: "Выпрыгнуть из окна"; date:"30.10.2022";}

}

SilicaListView {

anchors.fill: *parent*

model: *task\_list*

header: PageHeader { title: "Задачи на неделю" }

section {

property: 'date'

delegate: SectionHeader { text: section }

}

delegate: Text { text: name }

}

}

// task\_7.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page7*

SilicaWebView {

id: *webView*

anchors {

top: *parent*.top; bottom: *urlField*.top;

left: *parent*.left; right: *parent*.right;

}

url: "https://www.tv-asahi.co.jp/goodmorning/uranai/#mizugame"

}

TextField {

id: *urlField*

anchors {

left: *parent*.left; right: *parent*.right;

bottom: *parent*.bottom

}

text: *webView*.url

label: "Прогресс: " + *webView*.loadProgress + "\nЗаголовок: " + *webView*.title

EnterKey.onClicked: *webView*.url = *text*

}

}

// task\_8.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page8*

ListModel {

id: *task\_list*

ListElement { name: "Поваляться в кровати"; date:"25.10.2022";}

ListElement { name: "Посидеть в одиночестве";date:"25.10.2022";}

ListElement { name: "Посмотреть фильм"; date:"28.10.2022";}

ListElement { name: "Выпрыгнуть из окна"; date:"30.10.2022";}

}

SlideshowView {

id: *view*

anchors.centerIn: *parent*

height: *width*

itemHeight: *width*; itemWidth: *width*;

model: *task\_list*

delegate: Rectangle {

width: *view*.itemWidth;

height: *view*.itemHeight;

color: "transparent"

border.color: "black"

Text {

anchors.centerIn: *parent*

font.pixelSize: 40

text: "Дата: " + date + "\nЗадача: " + name

}

}

}

}

// task\_9.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page9*

property string txt1 : ""

SilicaListView {

PullDownMenu {

MenuItem {

text: "Убийца - садовник"

onClicked: {*txt1* = *text*}

}

MenuItem {

text: "Котопёс"

onClicked: {*txt1* = *text*}

}

MenuLabel { text: *qsTr*("Верхнее меню") }

}

PushUpMenu {

MenuLabel { text: *qsTr*("Нижнее меню") }

MenuItem {

text: "Агент Смит"

onClicked: {*txt1* = *text*}

}

MenuItem {

text: "Хрен с винтом"

onClicked: {*txt1* = *text*}

}

}

anchors.fill: *parent*

header: PageHeader { title: "Менюшки" }

}

Label {

id: *lb*

anchors.centerIn: *parent*

text: "Текстовое поле для элемента:"

}

Text {

anchors.top: *lb*.bottom

anchors.horizontalCenter: *lb*.horizontalCenter

id: *tx*

text: "" + *txt1*

}

}

// task\_10.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page10*

ListModel {

id: *task\_model*

ListElement { name: "Ноль" }

ListElement { name: "Целковый" }

ListElement { name: "Подружка" }

ListElement { name: "Ватрушка" }

ListElement { name: "Квизи" }

ListElement { name: "Ёрик" }

}

SilicaListView {

anchors.fill: *parent*

model: *task\_model*

delegate: ListItem {

id: *delegate*

Label {

x: Theme.paddingLarge

anchors.horizontalCenter: *parent*.horizontalCenter

text: name + " #" + index

color: *delegate*.highlighted ? Theme.highlightColor : Theme.primaryColor

}

menu: ContextMenu {

id: *contmenu*

MenuLabel { text: "Context Menu" }

MenuItem {

text: "Ким-за-за"

onClicked: *console*.log("выбран элемент " + *text* + " с индексом элемента списка " + index)

}

MenuItem {

text: "Киш-Мыш"

onClicked: *console*.log("выбран элемент " + *text* + " с индексом элемента списка " + index)

}

}

}

VerticalScrollDecorator {}

}

}

// CoverPage.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

CoverBackground {

Label {

id: *label*

anchors.centerIn: *parent*

text: "Текущий счёт:\n" + *coverAction*.count

}

CoverActionList {

property int count: 0

id: *coverAction*

CoverAction {

iconSource: "image://theme/icon-m-add"

onTriggered: *coverAction*.count++

}

CoverAction {

iconSource: "image://theme/icon-m-clear"

onTriggered: *coverAction*.count = 0

}

}

}