МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ  
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**

Направление подготовки: «Программная инженерия»

**ОТЧЁТ**

по лабораторной работе №5

**РАБОТА СО СТРАНИЦАМИ И ЭКРАНОМ**

**Выполнил:** студент группы

382008-1

\_\_\_А.Д. Кандрин \_\_\_\_\_\_\_

Подпись

**Проверил:** младший научный сотрудник

\_\_\_\_Д.А. Карчков\_\_\_\_\_\_\_\_

Подпись

Нижний Новгород  
2023 г.

## Содержание

[1. Постановка задачи 3](#_Toc27873363)

[2. ХОД РАБОТЫ](#_Toc27873364) 5

[3. ЗАКЛЮЧЕНИЕ](#_Toc27873365) 9

[4. ЛИТЕРАТУРА](#_Toc27873374) 10

## Постановка задачи.

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

## Ход работы.

1. Создадим 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.

Column {

id: column

width: parent.width

spacing: Theme.paddingLarge

PageHeader {

title: "Страница"

}

Button

{

text: "Назад"

onClicked: pageStack.pop()

}

Button

{

text: "Вперед"

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

}

Label

{

text: "Глубина стека: " + pageStack.depth

}

}

1. Создадим 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 будет кнопка для возврата на первую страницу без закрытия второй.

// Код для первой страницы

Column {

id: column

width: parent.width

spacing: Theme.paddingLarge

PageHeader {

title: "Первая страница"

}

Button

{

text: "Добавить страницу"

onClicked: pageStack.pushAttached(Qt.resolvedUrl("Page2\_2.qml"))

}

Button

{

text: "Убрать страницу"

onClicked: {

pageStack.popAttached()

}

}

}

// Код для второй страницы

Column {

id: column

width: parent.width

spacing: Theme.paddingLarge

PageHeader {

title: "Вторая страница"

}

Button

{

text: "Назад"

onClicked: pageStack.navigateBack()

}

}

1. Создадим приложение с одной кнопкой и текстовом поле. После нажатия на кнопку отобразим диалог для ввода текста. После согласия с результатом введённый текст отобразим в текстовое поле.

Column {

id: column

width: parent.width

spacing: Theme.paddingLarge

PageHeader {

title: "Третья страница"

}

Button

{

text: "Открыть диалог"

onClicked: {

var dialog = pageStack.push(Qt.resolvedUrl("Page3\_2.qml"));

dialog.accepted.connect(function() {

label.text = dialog.textFieldValue;

});

}

}

Label

{

id: label

text: ""

}

}

// Код для страницы с диалогом

property alias textFieldValue: textField.text

Column {

width: parent.width

spacing: Theme.paddingMedium

DialogHeader { }

TextField {

id: textField

width: parent.width

}

}

1. Создадим приложение с одной кнопкой и текстовым полем. После нажатия на кнопку отобразим диалог для выбора даты. После согласия с результатом ввода выбранную дату отобразим в текстовое поле.

Column {

id: column

width: parent.width

spacing: Theme.paddingLarge

PageHeader {

title: "Четвертая страница"

}

Button

{

text: "Открыть диалог"

onClicked: {

var dialog = pageStack.push("Sailfish.Silica.DatePickerDialog");

dialog.accepted.connect(function() {

label.text = dialog.dateText

});

}

}

Label

{

id: label

text: ""

}

}

// Код для страницы с диалогом

property alias result: datePicker.dateText

Column {

width: parent.width

spacing: Theme.paddingMedium

DialogHeader { }

// 5) Создать селектор даты, который будет отображать выбранную дату в консоли

DatePicker

{

id: datePicker

daysVisible: true

weeksVisible: true

}

}

1. Создадим приложение с одной кнопкой и текстовым полем. После нажатия на кнопку отобразим диалог для выбора времени. После согласия с результатом ввода выбранное время отобразим в текстовом поле.

Column {

id: column

width: parent.width

spacing: Theme.paddingLarge

PageHeader {

title: "Пятая страница"

}

Button

{

text: "Открыть диалог"

onClicked: {

var dialog = pageStack.push("Sailfish.Silica.TimePickerDialog");

dialog.accepted.connect(function() {

label.text = dialog.timeText

});

}

}

Label

{

id: label

text: ""

}

}

// Код для страницы с диалогом

property alias result: timePicker.timeText

Column {

width: parent.width

spacing: Theme.paddingMedium

DialogHeader { }

// 5) Создать селектор даты, который будет отображать выбранную дату в консоли

Item {

height: timePicker.height

width: parent.width

TimePicker

{

id: timePicker

anchors.horizontalCenter: parent.horizontalCenter

hourMode: DateTime.TwentyFourHours

}

Label

{

anchors.centerIn: timePicker

text: timePicker.timeText

font.pixelSize: Theme.fontSizeExtraLarge

}

}

}

1. Создадим приложение со списком SilicaListView, из задач на неделю. Задачи будут содержать дату и описание. В списке задачи сгруппируем по датам.

ListModel {

id: animalsModel

ListElement { description: "Сделать дело"; dateText: "17.02.2017"; }

ListElement { description: "Гулять смело"; dateText: "17.02.2017"; }

ListElement { description: "Прочитать книгу"; dateText: "16.02.2017"; }

ListElement { description: "Сходить на учебу"; dateText: "15.02.2017"; }

}

SilicaListView {

anchors.fill: parent

model: animalsModel

header: PageHeader { title: "Задачи" }

section {

property: 'dateText'

delegate: SectionHeader { text: section }

}

delegate: Text {

text: description

font.pointSize: Theme.fontSizeSmall

color: "white"

}

}

1. Создадим приложение с SilicaWebView для доступа к вашему любимому сайту.

SilicaWebView {

id: webView

anchors {

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

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

}

url: "http://www.google.ru"

}

TextField {

id: urlField

anchors {

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

bottom: parent.bottom

}

text: "http://www.google.ru"

label: webView.title

EnterKey.onClicked: webView.url = text

}

1. Использовуем SlideshowView для просмотра и перелистывания задач на неделю. На одном слайде – одна задача.

ListModel {

id: taskModel

ListElement { description: "Сделать дело"; dateText: "17.02.2017"; }

ListElement { description: "Гулять смело"; dateText: "17.02.2017"; }

ListElement { description: "Прочитать книгу"; dateText: "16.02.2017"; }

ListElement { description: "Сходить на учебу"; dateText: "15.02.2017"; }

}

SlideshowView {

id: view

anchors.centerIn: parent

height: width

itemHeight: width; itemWidth: width;

model: taskModel

delegate: Column {

width: view.itemWidth;

height: view.itemHeight;

PageHeader

{

title: dateText

}

Label {

text: description

}

}

}

1. Создадим приложение с вытягиваемыми меню сверху и снизу и текстовым полем. После выбора какого-либо элемента меню, его название отобразим в текстовом поле.

SilicaFlickable {

anchors.fill: parent

contentHeight: column.height

PullDownMenu {

MenuItem {

text: "Menu Item 1"

onClicked: label.text = text

}

MenuItem {

text: "Menu Item 2"

onClicked: label.text = text

}

}

PushUpMenu {

MenuItem {

text: "Menu Item 3"

onClicked: label.text = text

}

MenuItem {

text: "Menu Item 4"

onClicked: label.text = text

}

}

Column

{

id: column

width: page.width

PageHeader

{

title: "Страница с меню"

}

Label

{

id: label

text: ""

}

}

}

1. Создадим приложение со списком и контекстным меню. После выбора элемента контекстного меню отобразим в консоли название выбранного элемента меню и индекс элемента списка.

SilicaListView {

anchors.fill: parent

model: 10

delegate: ListItem {

id: delegate

Label {

x: Theme.paddingLarge

text: "List Item #" + index

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

}

menu: ContextMenu {

MenuLabel { text: "Context Menu" }

MenuItem {

text: "Menu Item 1"

onClicked: console.log(text + " (ListItem index = " + index + ")")

}

MenuItem {

text: "Menu Item 2"

onClicked: console.log(text + " (ListItem index = " + index + ")")

}

}

}

}

1. Создадим приложение с обложной-счётчиком. На обложке отобразим текущий счёт и две кнопки для добавления единицы к счёту и для сброса счётчика.

// Для этого добавим Cover укажем его в свойство ApplicationWindow

Cover {

Label {

id: label

property int counter: 0

text: counter

anchors.centerIn: parent

font.pixelSize: Theme.fontSizeExtraLarge

}

CoverActionList {

CoverAction {

iconSource: "image://theme/icon-cover-cancel"

onTriggered: label.counter = 0

}

CoverAction {

iconSource: "image://theme/icon-cover-new"

onTriggered: label.counter++

}

}

}

ApplicationWindow

{

initialPage: Component { StartPage { } }

cover: Qt.resolvedUrl("cover/CoverPage.qml")

allowedOrientations: defaultAllowedOrientations

}

1. **Заключение.**

Мы научились организовывать многостраничное приложение, использовать контейнеры Silica, вытягиваемые меню и обложку приложения.

**7. Литература.**

1. Документация Qt: сайт. — URL: https://doc.qt.io (дата обращения: 01.03.2022). — Текст: электронный.