МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Нижегородский государственный университет им. Н.И. Лобачевского»
Национальный исследовательский университет

Институт информационных технологий, математики и механики
Кафедра программной инженерии

ОТЧЕТ

по дисциплине «Разработка мобильных приложений»

«Средства организации интерфейса мобильных приложений»

Выполнил:
студент группы 381706-1
Денисов В. Л.

Проверил:
доцент кафедры программной инженерии
Борисов Н. А.

Нижний Новгород
2020.

Содержание

1 Цели

Целью данной лабораторной работы является изучение организации многостраничного приложения, использования контейнеров Silica, вытягиваемого меню и обложки приложения.

2 Постановка задачи

1. 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.
2. 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.
3. Создать приложение с одной кнопкой и текстовым полем. После нажатия на кнопку отображается диалог для ввода текста. После согласия с результатом введённый текст отображается в текстовом поле.
4. 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 текстовом поле.
5. 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
6. Создать приложение со списком SilicaListView, из задач на неделю. Задачи должны содержать дату и описание. В списке задачи группировать по датам.
7. Создать приложение с SilicaWebView для доступа к вашему любимому сайту.
8. Использовать SlideshowView для просмотра и перелистывания задач на неделю. На одном слайде – одна задача.
9. Создать приложение с вытягиваемыми меню сверху и снизу и текстовым полем. После выбора какого-либо элемента меню, его название отобразить в текстовом поле.
10. Создать приложение со списком и контекстным меню. После выбора элемента контекстного меню отобразить в консоли название выбранного элемента меню и индекс элемента списка.
11. Создать приложение с обложной-счётчиком. На обложке отобразить текущий счёт и две кнопки для добавления единицы к счёту и для сброса счётчика.

3 Решение поставленных задач

Создадим проект со стандартной заготовкой приложения, где файлом главной страницы приложения будет являться *FirstPage.qml*, для каждого отдельного задания из постановки задачи тоже будет свой файл.

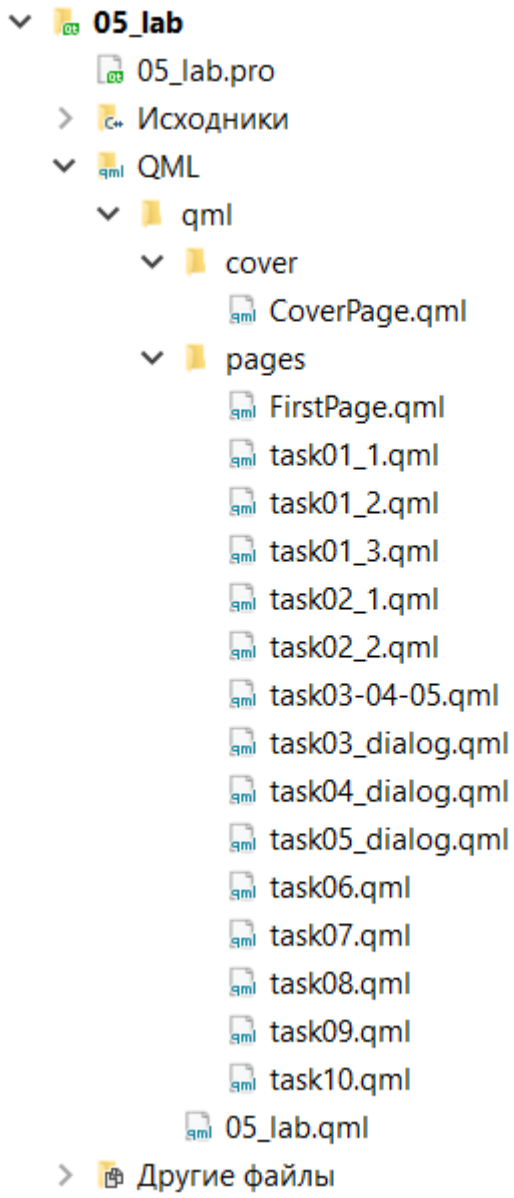

Рисунок 1 Структура проекта.

Вызов демонстрации каждого задания будем производить путем нажатия соответствующей кнопки, расположенной на *FirstPage*. Страница с выбранным заданием будет отправлена в *pageStack.*

1. Добавление страницы в стек выполняется путём вызова метода *pageStack.push(Qt.resolvedUrl("Название_страницы"))*. Извлечение страницы из стека – *pageStack.pop()*. Глубина стека – *pageStack.depth*.

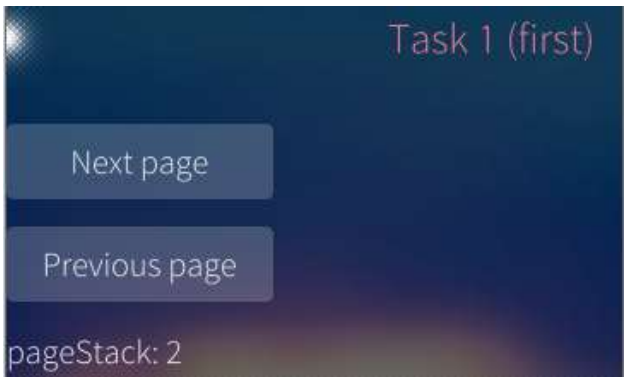

Рисунок 2 В стеке страниц: FirstPage и Task1 (first).

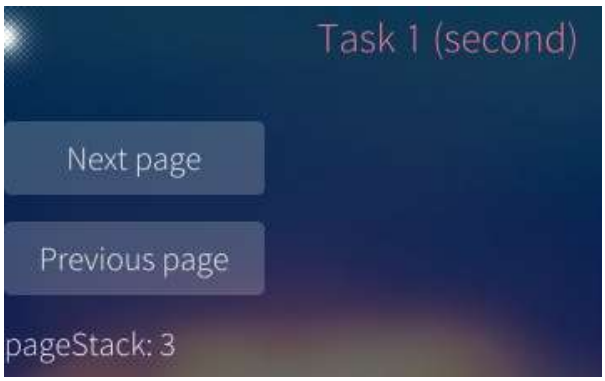

Рисунок 3 В стеке страниц: FirstPage, Task1 (first), Task1 (second)

2. Прикрепление страницы выполняется при помощи функции *pageStack.pushAttached(Qt.resolvedUrl("Название_страницы"))*, откреплние – *pageStack.popAttached()*.

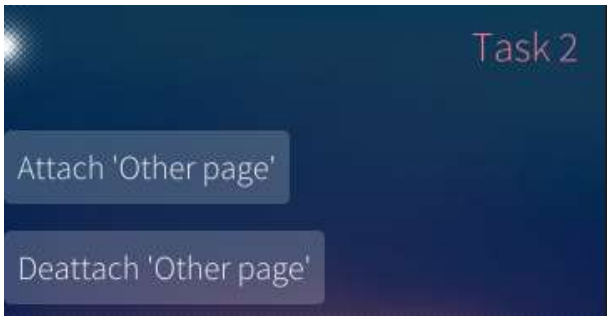

Рисунок 4 Состояние страницы до присоединения к ней другой страницы.

Рисунок 5 Состояние страницы после присоедения другой страницы.

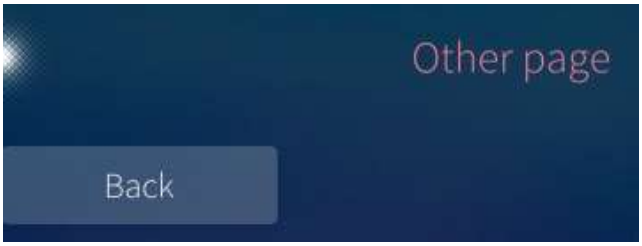

Рисунок 6 Внешний вид присоединенной страницы.

3, 4 ,5. Объединим задания, в которых требуется создать диалог для ввода текста, выбора даты и выбора времени, на одной странице.

Расположим кнопки для вызова соответствующих диалогов внутри контейнера *Column*. Объявим 3 свойства: *property string dialogText, dialogDate, dialogTime*, в которые будем записывать результат, возвращаемый диалогом. Обработчик соответствующего диалога располагается в отдельном файле. Контейнеры для диалогов текста, выбора даты, выбора времени: *Dialog*, *DatePickerDialog*, *TimePickerDialog*.

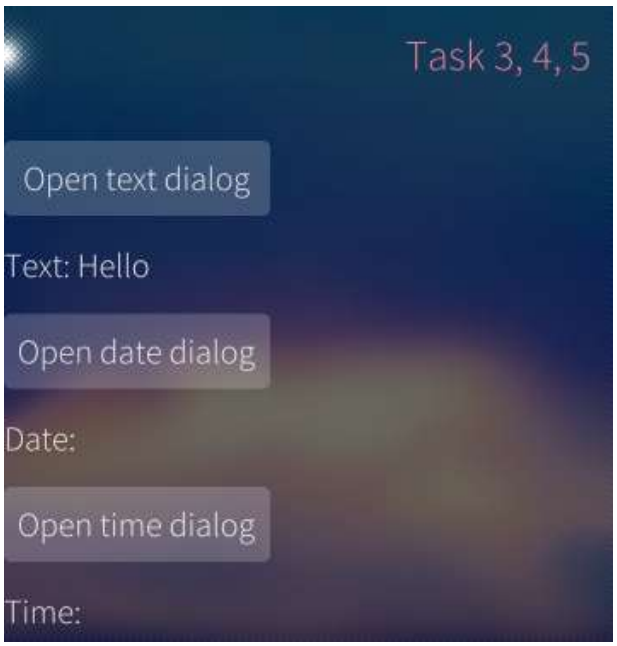

Рисунок 7 Страница, на которой происходит вызов диалогов и отображение результата их работы.

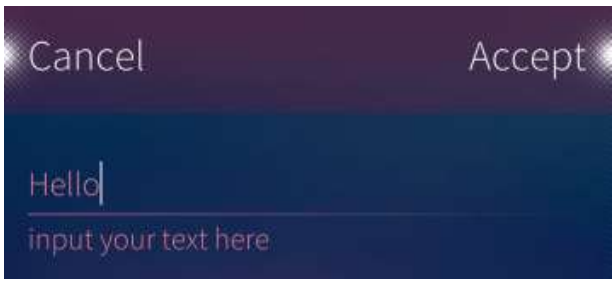

Рисунок 8 Диалог с вводом текста.

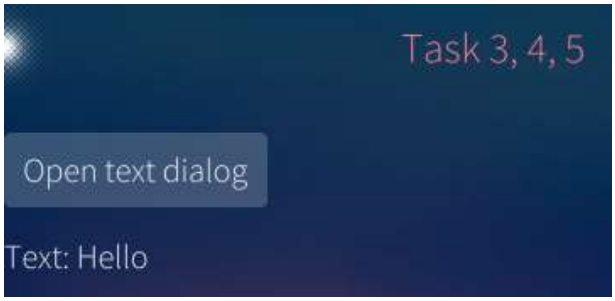

Рисунок 9 Результат работы диалога ввода текста.

6. Создаем элемент *SilicaListView*. Его атрибуту *model* устанавливаем объект *ListModel*, включающий в себя элементы создаваемого списка задач – *ListElement*. Каждый из элементов имеет атрибуты имени и даты: name и date. Требуемая группировка задач по дате в *SilicaListView* достигается путём использования *section* с атрибутом *property: 'date'*, а описание задачи обрабатывается через атрибут *delegate*. Внешний вид для *delegate* описывается вне *section*.

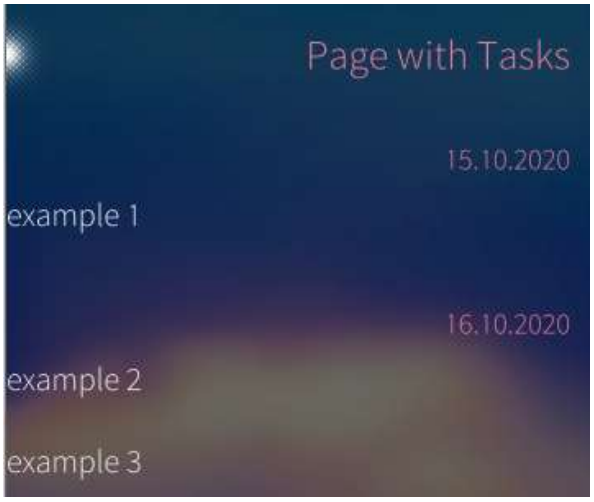

Рисунок 10 Список задач с группировкой по дате.

7. Создаем элемент *SilicaWebView* для доступа к сайтам непосредственно из приложения. Атрибуту *url* указываем адрес сайта с учетом используемого протокола. Изменить его позволяет независимое от *SilicaWebView* текстовое поле *TextField*, которое имеет обработчик сигнала нажатия на клавишу ввода,

изменяющий значение атрибута *url* на введенный текст. Кроме того, подтвердить переход на сайт можно нажатием на кнопку “Go”.

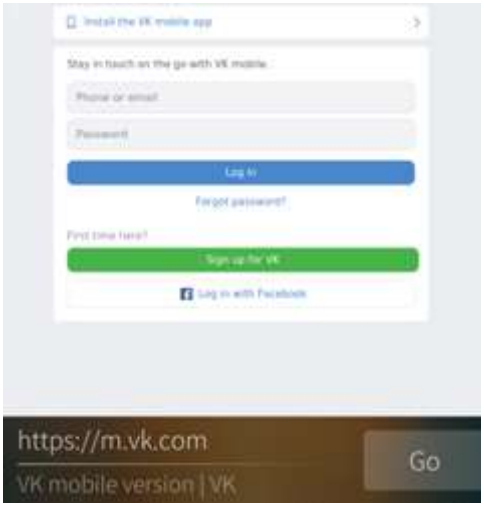

Рисунок 11 Переход на сайт в Интернете из приложения Silica.

8. Для просмотра задач по слайдам используем *SlideshowView*. Необходимое отображение слайда достигается путём назначения атрибуту *delegate* элемента, который будет отвечать за внешний вид. В нашем случае используется *Rectangle*, содержащий пару текстовых блоков *Text* – для даты задачи и её описания.

Рисунок 12 Список задач в виде слайдов.

9. На странице приложения размещаем элемент *SilicaFlickable*, внутри которого поместим элементы *PullDownMenu*, *PullUpMenu* и *Label*, в который будем выводить

текст при клике на выбранный элемент меню. Обработку нажатия на элемент меню выполняем при помощи *onClick*.

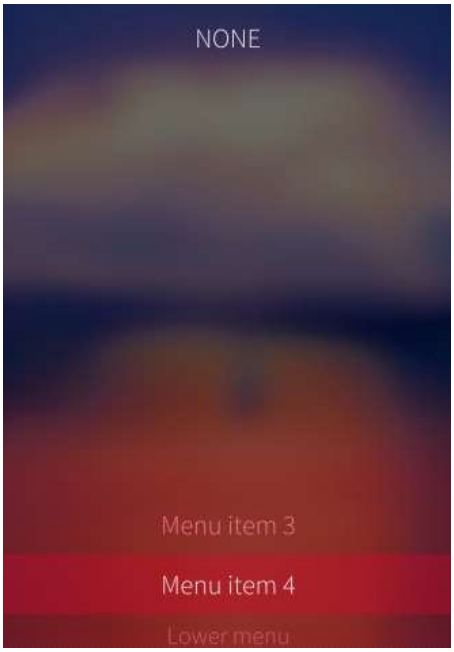

Рисунок 13 Вытягивание нижнего меню.

Рисунок 14 Отображение выбранного элемента меню.

10. Создаем список аналогично пункту 6. Однако в элемент списка *ListItem* добавляем атрибут *menu* и присваиваем ему контейнер *ContexMenu*, содержащий *MenuItem*'ы – пункты контекстного меню.

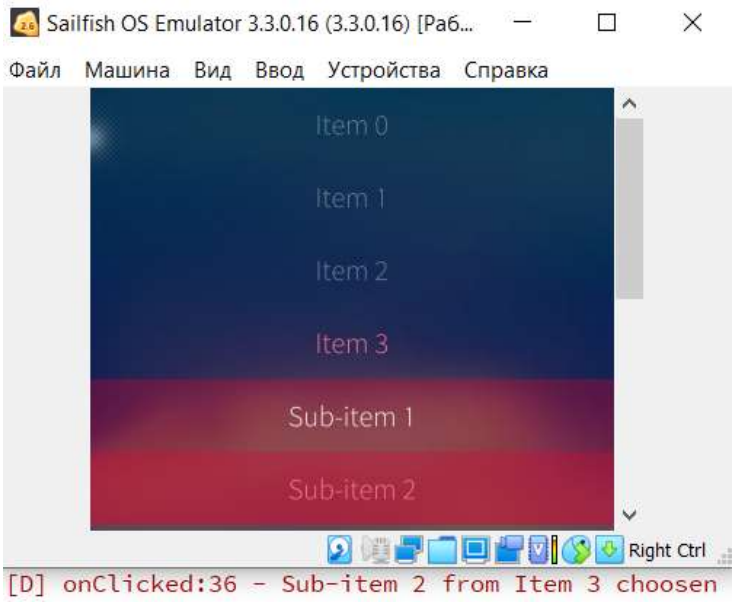

Рисунок 15 Выбор пункта из контекстного меню и вывод информации на консоль.

11. Обложка приложения настраивается в *CoverPage.qml*. Добавим элементу *CoverBackground* целочисленное свойство *count*, которое будет хранить текущее

значение счетчика. Добавляем 2 возможных действия *CoverAction*: сбросить значение счётчика и увеличить, располагая их в *CoverActionList*. Настраиваем их обработчики *onClick*, выполняющие требуемое действие при нажатии на соответствующий элемент.

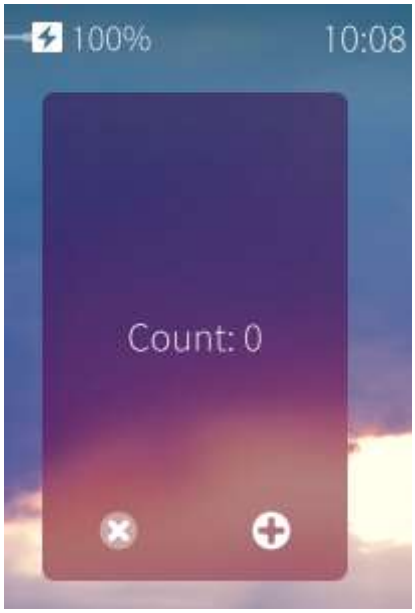

Рисунок 16 Счётчик на обложке до нажатий.

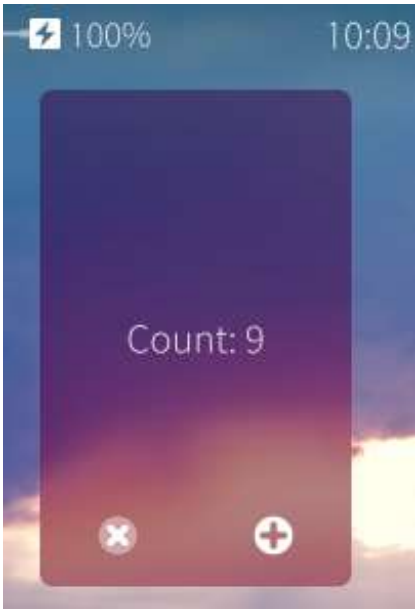

Рисунок 17 Счётчик на обложке после нажатий.

4 Приложение

4.1 Файл FirstPage.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    id: page
    allowedOrientations: Orientation.All
    SilicaFlickable {
        anchors.fill: parent
        contentHeight: column.height
        Column {
            id: column
            width: page.width
            spacing: Theme.paddingLarge
            PageHeader {
                title: qsTr("Main Page")
            }
            Button {
                text: qsTr("Task 1")
                onClicked: pageStack.push(Qt.resolvedUrl("task01_1.qml"))
            }
            Button {
                text: qsTr("Task 2")
                onClicked: pageStack.push(Qt.resolvedUrl("task02_1.qml"))
            }
            Button {
                text: qsTr("Task 3, 4, 5")
                onClicked: pageStack.push(Qt.resolvedUrl("task03-04-05.qml"))
            }
            Button {
                text: qsTr("Task 6")
                onClicked: pageStack.push(Qt.resolvedUrl("task06.qml"))
            }
            Button {
                text: qsTr("Task 7")
                onClicked: pageStack.push(Qt.resolvedUrl("task07.qml"))
            }
            Button {
                text: qsTr("Task 8")
                onClicked: pageStack.push(Qt.resolvedUrl("task08.qml"))
            }
            Button {
                text: qsTr("Task 9")
                onClicked: pageStack.push(Qt.resolvedUrl("task09.qml"))
            }
            Button {
                text: qsTr("Task 10")
                onClicked: pageStack.push(Qt.resolvedUrl("task10.qml"))
            }
        } // Column
    } // SilicaFlickable
} // Page
```

4.2 Файл CoverPage.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0
CoverBackground {
    property  int count : 0
```

```
    Label {
        id: cover_label
        text: "Count: " + count
        anchors.centerIn: parent
    }
    CoverActionList {
        CoverAction {
            iconSource: "image://theme/icon-cover-cancel"
            onTriggered: count = 0
        }
        CoverAction {
            iconSource: "image://theme/icon-cover-new"
            onTriggered: count++
        }
    }
}
```

4.3 Файл task01_1.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    id: page
    allowedOrientations: Orientation.All

    Column {
        id: column
        width: page.width
        spacing: Theme.paddingLarge
        PageHeader {
            title: qsTr("Task 1 (first)")
        }
        Button {
            text: "Next page"
            onClicked: pageStack.push(Qt.resolvedUrl("task01_2.qml"))
        }
        Button {
            text: "Previous page"
            onClicked: pageStack.pop()
        }
        Label {
            text: "pageStack: " + pageStack.depth
        }
    }
}
```

4.4 Файл task01_2.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    id: page
    allowedOrientations: Orientation.All

    Column {
        id: column
        width: page.width
        spacing: Theme.paddingLarge
        PageHeader {
            title: qsTr("Task 1 (second)")
        }
```

```
        Button {
            text: "Next page"
            onClicked: pageStack.push(Qt.resolvedUrl("task01_3.qml"))
        }
        Button {
            text: "Previous page"
            onClicked: pageStack.pop()
        }
        Label {
            text: "pageStack: " + pageStack.depth
        }
    }
}
```

4.5 Файл task01_3.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    id: page
    allowedOrientations: Orientation.All

    Column {
        id: column
        width: page.width
        spacing: Theme.paddingLarge
        PageHeader {
            title: qsTr("Task 1 (third)")
        }
        Button {
            text: "Previous page"
            onClicked: pageStack.pop()
        }
        Label {
            text: "pageStack: " + pageStack.depth
        }
    }
}
```

4.6 Файл task02_1.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    id: page
    allowedOrientations: Orientation.All

    Column {
        id: column
        width: page.width
        spacing: Theme.paddingLarge
        PageHeader {
            title: qsTr("Task 2")
        }
        Button {
            text: "Attach 'Other page'"
            onClicked: pageStack.pushAttached(Qt.resolvedUrl("task02_2.qml"))
        }
        Button {
            text: "Deattach 'Other page'"
            onClicked: pageStack.popAttached()
```

```
        }
    }
}
```

4.7 Файл task02_2.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    id: page
    allowedOrientations: Orientation.All

    Column {
        id: column
        width: page.width
        spacing: Theme.paddingLarge
        PageHeader {
            title: qsTr("Other page")
        }
        Button {
            text: "Back"
            onClicked: pageStack.navigateBack(1)
        }
    }
}
```

4.8 Файл task03-04-05.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    id: page
    allowedOrientations: Orientation.All

    Column {
        id: column
        property string dialogText: ""
        property string dialogDate: ""
        property string dialogTime: ""
        width: page.width
        spacing: Theme.paddingLarge

        PageHeader {
            title: qsTr("Task 3, 4, 5")
        }

        Button {
            text: "Open text dialog"
            onClicked: {
                var dialog =
pageStack.push(Qt.resolvedUrl("task03_dialog.qml"));
                dialog.accepted.connect(function() {
                    column.dialogText = dialog.fieldText;
                });
            }
        }
        Label {
            text: "Text: " + column.dialogText
        }

        Button {
```

```
            text: "Open date dialog"
            onClicked: {
                var dialog =
pageStack.push(Qt.resolvedUrl("task04_dialog.qml"));
                dialog.accepted.connect(function() {
                    column.dialogDate = dialog.dateText;
                });
            }
        }
        Label {
            text: "Date: " + column.dialogDate
        }

        Button {
            text: "Open time dialog"
            onClicked: {
                var dialog =
pageStack.push(Qt.resolvedUrl("task05_dialog.qml"));
                dialog.accepted.connect(function() {
                    column.dialogTime = dialog.timeText;
                });
            }
        }
        Label {
            text: "Time: " + column.dialogTime
        }
    }
}
```

4.8.1 Файл task03_dialog.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Dialog {
    property alias fieldText : field.text
    Column {
        width: parent.width
        spacing: Theme.paddingMedium
        DialogHeader {}
        TextField {
            id: field
            label: "input your text here"
            width: parent.width
        }
    }
}
```

4.8.2 Файл task04_dialog.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

DatePickerDialog {
    Column {
        width: parent.width
        spacing: Theme.paddingMedium
        DialogHeader {}
    }
}
```

4.8.3 Файл task05_dialog.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

TimePickerDialog {
```

```
    Column {
        width: parent.width
        spacing: Theme.paddingMedium
        DialogHeader {}
    }
}
```

4.9 Файл task06.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    id: page
    allowedOrientations: Orientation.All
    SilicaListView {
        model: ListModel {
            id: tasksModel
            ListElement { name: "example 1"; date: "15.10.2020"; }
            ListElement { name: "example 2"; date: "16.10.2020"; }
            ListElement { name: "example 3"; date: "16.10.2020"; }
            ListElement { name: "example 4"; date: "17.10.2020"; }
            ListElement { name: "example 5"; date: "18.10.2020"; }
            ListElement { name: "example 6"; date: "18.10.2020"; }
        }

        anchors.fill: parent
        header: PageHeader { title: "Page with Tasks"; }
        section {
            property: 'date'
            delegate: SectionHeader { text: section }
        }
        delegate: Item {
            width: ListView.view.width
            height: Theme.itemSizeSmall
            Label { text: name }
        }
    }
}
```

4.10 Файл task07.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    SilicaWebView {
        id: webView
        anchors {
            top: parent.top
            bottom: urlField.top
            left: parent.left
            right: parent.right
        }
        url: "https://m.vk.com/"
    }
    TextField {
        id: urlField
        anchors {
            bottom: parent.bottom
            left: parent.left
            right: goButton.left
        }
```

```
        text: "https://m.vk.com"
        label: webView.title
        EnterKey.onClicked: webView.url = text
    }
    Button {
        id: goButton
        text: "Go"
        onClicked: webView.url = urlField.text
        anchors {
            top: urlField.top
            bottom: parent.bottom
            right: parent.right
        }
        width: parent.width / 4
    }
}
```

4.11 Файл task08.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    ListModel {
                id: taskModel
                ListElement { name: "example 1"; date: "15.10.2020"; }
                ListElement { name: "example 2"; date: "16.10.2020"; }
                ListElement { name: "example 3"; date: "16.10.2020"; }
                ListElement { name: "example 4"; date: "17.10.2020"; }
                ListElement { name: "example 5"; date: "18.10.2020"; }
                ListElement { name: "example 6"; date: "18.10.2020"; }
            }
    SlideshowView {
        id: view
        anchors.centerIn: parent
        height: width
        model: taskModel
        delegate: Rectangle {
            width: view.height
            height: view.width
            Text {
                id: textDate
                anchors.bottom: parent.verticalCenter
                anchors.horizontalCenter: parent.horizontalCenter
                font.pointSize: 36
                text: date
            }
            Text {
                id: textText
                anchors.top: parent.verticalCenter
                anchors.horizontalCenter: parent.horizontalCenter
                font.pointSize: 36
                text: name
            }
        }
    }
}
```

4.12 Файл task09.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0
```

```
Page {
    SilicaFlickable {
        PullDownMenu {
            MenuItem {
                text: "Menu item 1"
                onClicked: label.text = "item 1"
            }
            MenuItem {
                text: "Menu item 2"
                onClicked: label.text = "item 2"
            }
            MenuLabel {
                text: "Upper menu"
            }
        }
        PushUpMenu {
            MenuItem {
                text: "Menu item 3"
                onClicked: label.text = "item 3"
            }
            MenuItem {
                text: "Menu item 4"
                onClicked: label.text = "item 4"
            }
            MenuLabel {
                text: "Lower menu"
            }
        }
        Label {
            id: label
            anchors.centerIn: parent
            text: "NONE"
        }
        anchors.fill: parent
    }
}
```

4.13 Файл task10.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    id: page
    allowedOrientations: Orientation.All

    SilicaListView {
        anchors.fill: parent

        model: ListModel {
            id: listModel
            Component.onCompleted: {
                for (var i = 0; i < 5; i++) {
                    append({"name": "Item " + i})
                }
            }
        }

        delegate: ListItem {
            width: ListView.view.width

            Label {
                id: label
```

```
                text: model.name
                anchors.centerIn: parent
            }

            menu: ContextMenu {
                MenuItem {
                    text: "Sub-item 1"
                    onClicked: console.log(text +" from " + model.name + "
choosen")
                }
                MenuItem {
                    text: "Sub-item 2"
                    onClicked: console.log(text +" from " + model.name + "
choosen")
                }
            }
        }
    }
}
```

5 Используемая литература

1. Документация QT – https://doc.qt.io/qt-5/qmake-project-files.html