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»

«Типовые элементы интерфейса на мобильных устройствах»

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 освоение типовых элементов интерфейса Sailfish OS.

2 Постановка задачи

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

3 Решение поставленных задач

Создадим проект со стандартной заготовкой приложения, где файлом главной страницы приложения будет являться *FirstPage.qml*

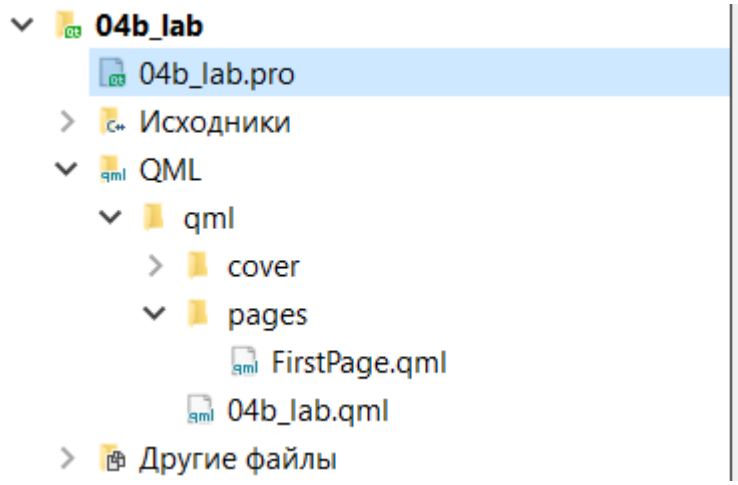


Рисунок 1 Структура проекта.

Все элементы приложения на странице будем располагать внутри контейнера *Column*.

1. Текстовое поле создадим при помощи *TextField*. Установим атрибуту *inputMethodHints* значение *Qt.ImhFormattedNumbersOnly* для ввода только чисел. Текст подсказки содержит атрибут *label*, а размещение самой подсказки внутри пустого поля ввода задаётся при помощи *placeholderText*.

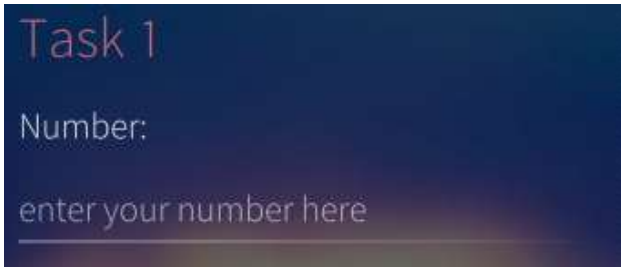


Рисунок 2 Текстовое поле до ввода числа.

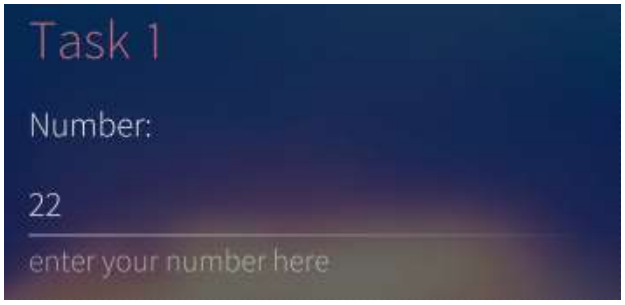


Рисунок 3 Текстовое поле после ввода числа.

2. Реализация кнопки, сохраняющей нажатое положение достигается путем установки атрибута *down* по умолчанию в значение *false*. Затем происходит обработка события *onClicked*, которое изменяет значение атрибута *down* на *true*.
3. Добавляем в обработчик *onClicked* дополнительный шаг по изменению параметра *text*, у соответствующего *Label*, характеризующего положение кнопки.

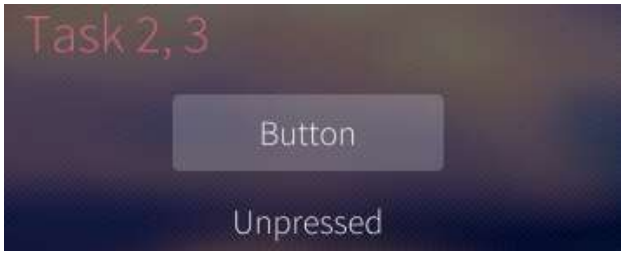


Рисунок 4 Кнопка не нажата.

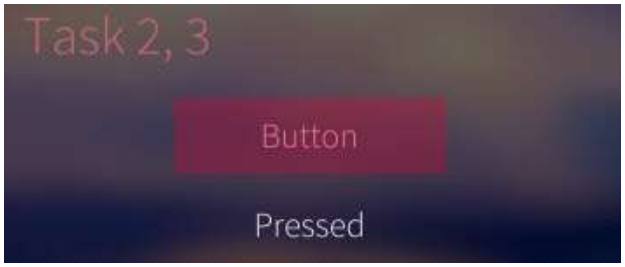


Рисунок 5 Кнопка нажата.

4. Создаем элемент *ValueButton*, задаем новое свойство *count*, которое будет хранить число нажатий, атрибуту *value* присваиваем *count*, в обработчике *onClicked* увеличиваем величину *count*.

Рисунок 6 Кнопка-счётчик при инициализации.

Рисунок 7 Кнопка-счётчик после нескольких нажатий.

5. Селектор даты создаем при помощи элемента *DatePicker*. Атрибуту *date* присваиваем результат вызова конструктора класса *Date*. Включаем отображение

дней и месяцев путём установки значения *true* атрибутам *daysVisible* *monthYearVisible* соответственно. Обработчику *onDateTextChanged* назначаем печать текущего состояния селектора на консоль.

Рисунок 8 Работа селектора даты и её вывод на консоль.

6. Аналогичный подход используем для селектора времени – элемент *TimePicker*. Выполняем инициализацию атрибутов *hour* и *minute* (часы и минуты соответственно) нулём. Обработчик изменения селектора – *onTimeTextChanged*.

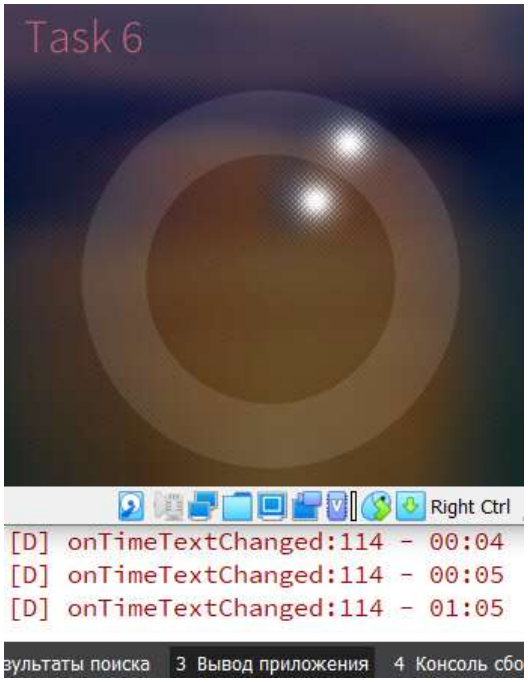
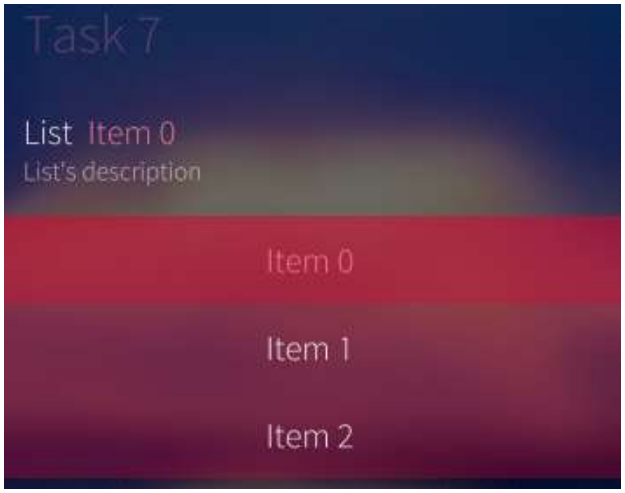


Рисунок 9 Работа селектора времени и его вывод на консоль.

7. Выпадающее меню создаем при помощи элемента *ComboBox*, устанавливая его атрибуту *menu* параметр *ContextMenu*, содержащий элементы меню *MenuItem*.



8. Переключатель задается элементом *Switch*. Для отображения подписи к нему создаем *Label*, с текстом. Текст зависит от положения переключателя, информацию о котором получаем путем обращения к соответствующему атрибуту переключателя – *checked*.

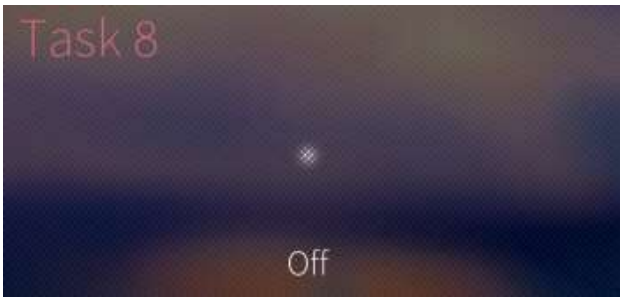


Рисунок 10 Переключатель в положении "Выключен".

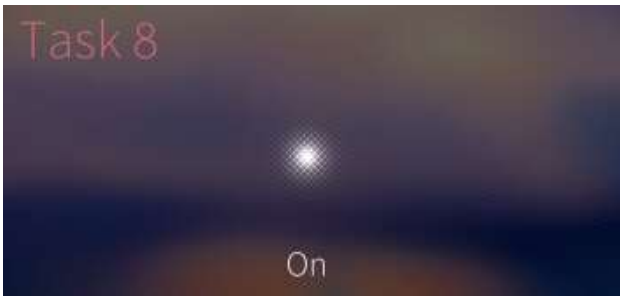


Рисунок 11 Переключатель в положении "Включён".

9. Ползунок создаем, используя элемент *Slider*. Его атрибуты *minimumValue* и *maximimValue* задают минимальное и максимальное значения соответственно. Величина шага при перемещении ползунка – *stepSize*. Значение по умолчанию – *value*. Для отображения текущего значения ползунка вместо отдельного поля с текстом будем изменять параметр атрибута *label* – подпись.

Рисунок 12 Элемент "Ползунок"("Слайдер").

4 Приложение

4.1 Файл 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
    SilicaFlickable {
        anchors.fill: parent
        contentHeight: column.height
        // Place our content in a Column.
        Column {
           id: column
           width: page.width
           spacing: Theme.paddingLarge
           PageHeader {
               title: qsTr("Lab 4")
           }

           // Task 1
           Label {
               x: Theme.horizontalPageMargin
               text: qsTr("Task 1")
               color: Theme.secondaryHighlightColor
               font.pixelSize: Theme.fontSizeExtraLarge
           }
           Label {
               x: Theme.horizontalPageMargin
               text: qsTr("Number:")
           }
            TextField {
                id: numberTextField;
                width: parent.width;
                inputMethodHints: Qt.ImhFormattedNumbersOnly;
                label: "enter your number here"
                placeholderText: label
            }

            // Task 2, 3
            Label {
                x: Theme.horizontalPageMargin
                text: qsTr("Task 2, 3")
                color: Theme.secondaryHighlightColor
                font.pixelSize: Theme.fontSizeExtraLarge
            }
            Button {
                id: pressedButton
                anchors.horizontalCenter: parent.horizontalCenter;
                text: "Button"
                down: false
                onClicked: {
                    if (down == true) {
                        down = false
                        pressedLabel.text = "Unpressed"
                    } else {
                        down = true
```

```
            pressedLabel.text = "Pressed"
        }
    }
}
Label {
    id: pressedLabel
    text: "Unpressed"
    anchors.horizontalCenter: parent.horizontalCenter;
}

// Task 4
Label {
    x: Theme.horizontalPageMargin
    text: qsTr("Task 4")
    color: Theme.secondaryHighlightColor
    font.pixelSize: Theme.fontSizeExtraLarge
}
ValueButton {
    id: counterButton
    anchors.horizontalCenter: parent.horizontalCenter;
    property int count : 0
    value: count
    description: "Tap to increase"
    onClicked: {
        count++;
    }
}

// Task 5
Label {
    x: Theme.horizontalPageMargin
    text: qsTr("Task 5")
    color: Theme.secondaryHighlightColor
    font.pixelSize: Theme.fontSizeExtraLarge
}
DatePicker {
    date: new Date();
    daysVisible: true
    monthYearVisible: true
    onDateTextChanged: {
        console.log(dateText);
    }
}

// Task 6
Label {
    x: Theme.horizontalPageMargin
    text: qsTr("Task 6")
    color: Theme.secondaryHighlightColor
    font.pixelSize: Theme.fontSizeExtraLarge
}
TimePicker {
    anchors.horizontalCenter: parent.horizontalCenter;
    hour: 0
    minute: 0
    onTimeTextChanged: {
        console.log(timeText);
    }
}

// Task 7
Label {
    x: Theme.horizontalPageMargin
```

```
                text: qsTr("Task 7")
                color: Theme.secondaryHighlightColor
                font.pixelSize: Theme.fontSizeExtraLarge
            }
            ComboBox {
                label: "List"
                description: "List's description"
                menu: ContextMenu {
                    id: menu
                    MenuItem { text: "Item 0" }
                    MenuItem { text: "Item 1" }
                    MenuItem { text: "Item 2" }
                }
                onCurrentIndexChanged: { console.log("Current menu item: " +
menu.children[currentIndex].text) }
            }

            // Task 8
            Label {
                x: Theme.horizontalPageMargin
                text: qsTr("Task 8")
                color: Theme.secondaryHighlightColor
                font.pixelSize: Theme.fontSizeExtraLarge
            }
            Switch {
                anchors.horizontalCenter: parent.horizontalCenter;
                id: switchItem
            }
            Label {
                anchors.horizontalCenter: parent.horizontalCenter;
                x: Theme.horizontalPageMargin
                text: qsTr(switchItem.checked ? "On" : "Off")
            }

            // Task 9
            Label {
                x: Theme.horizontalPageMargin
                text: qsTr("Task 9")
                color: Theme.secondaryHighlightColor
                font.pixelSize: Theme.fontSizeExtraLarge
            }
            Slider {
                width: parent.width
                minimumValue: 0
                maximumValue: 1
                stepSize: 0.01
                value: 0.5
                label: "Current value: " + value
            }

            // For an empty space at the end of the page
            Rectangle {
                color: "transparent"
                width: 50
                height: 500
            }
        } // Column
    } // SiilicaFlickable
} // Page
```

5 Используемая литература

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