Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования
Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

Отчет по практическому заданию №3

«Инструменты разработки мобильных приложений»

Выполнил:
студент группы 381906-1
Зайцев М. А.

Нижний Новгород
2022

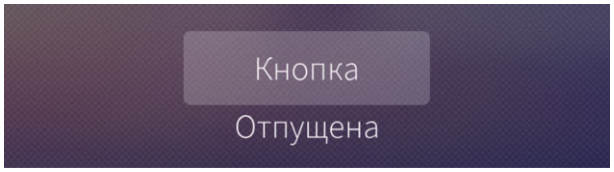
Оглавление

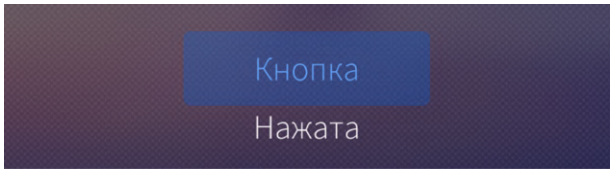
Цель задачи

В данном лабораторной работе требуется научиться применять типовые элементы интерфейса Sailfish OS.

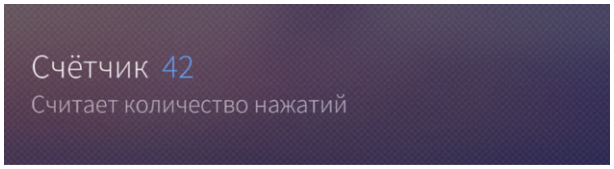
Постановка задачи

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”.



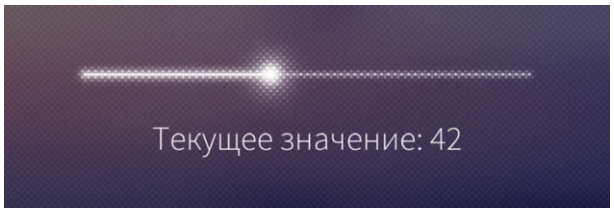


4. Создать кнопку со значением, которая будет отображать количество нажатий на неё.



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.



Описание программной реализации

Задание 1

Создать текстовое поле для ввода числа с заголовком и подсказкой. Для этого задействуем **TextField** с указанием подсказки в *placeholderText* и описанием в *description*.

Задание 2

Создать кнопку, которая будет сохранять визуально нажатое состояние после того, как пользователь нажал на неё один раз. Для этого в элементе **Button** создадим пользовательское свойство *bool toggle*, укажем его в свойство *down*, а в *onClicked* будем делать *toggle = !toggle*.

Задание 3

Создать кнопку и поле с текстом. Поле с текстом должно отображать нажата ли кнопка или нет выводом текста “Нажата” или “Отпущена”. Для этого в элемент **Label** будем передавать свойство *down* у **Button** и воспользуемся условной отрисовкой.

Задание 4

Создать кнопку со значением, которая будет отображать количество нажатий на неё. По аналогии с первой лабораторной, будем увеличивать пользовательское свойство в *onClicked* у **Button** и передавать его в **Label**.

Задание 5

Создать селектор даты, который будет отображать выбранную дату в консоли. Используется **DatePicker** и метод *onDateChanged*.

Задание 6

Создать селектор времени, который будет отображать выбранное время в консоли. Используется **TimePicker** и метод *onTimeChanged*.

Задание 7

Создать поле с выпадающим списком, позволяющее выбрать строку из списка. Результат выбора отобразить в консоли. Используется **ComboBox**. Пункты списка задаются полем *menu*, в котором мы размещаем элементы **MenuItem** со свойством *text*. Вывод результата в консоль в методе *onCurrentIndexChanged*.

Задание 8

Создать переключатель с текстом, в тексте отобразить состояние переключателя “Включен” или “Выключен”. Используется элемент **Switch** и **Label** для вывода состояния. Состояние проверяется свойством *checked* элемента **Switch**.

Задание 9

Создать ползунок и поле с текстом. Поле с текстом должно отображать текущее значение ползунка. Используются **Slider** со свойством *value* и **Label** для вывода результата.

Руководство пользователя

После запуска программы пользователь видит страницу с первым заданием и кнопку “+”, которая перейдет на следующую страницу.

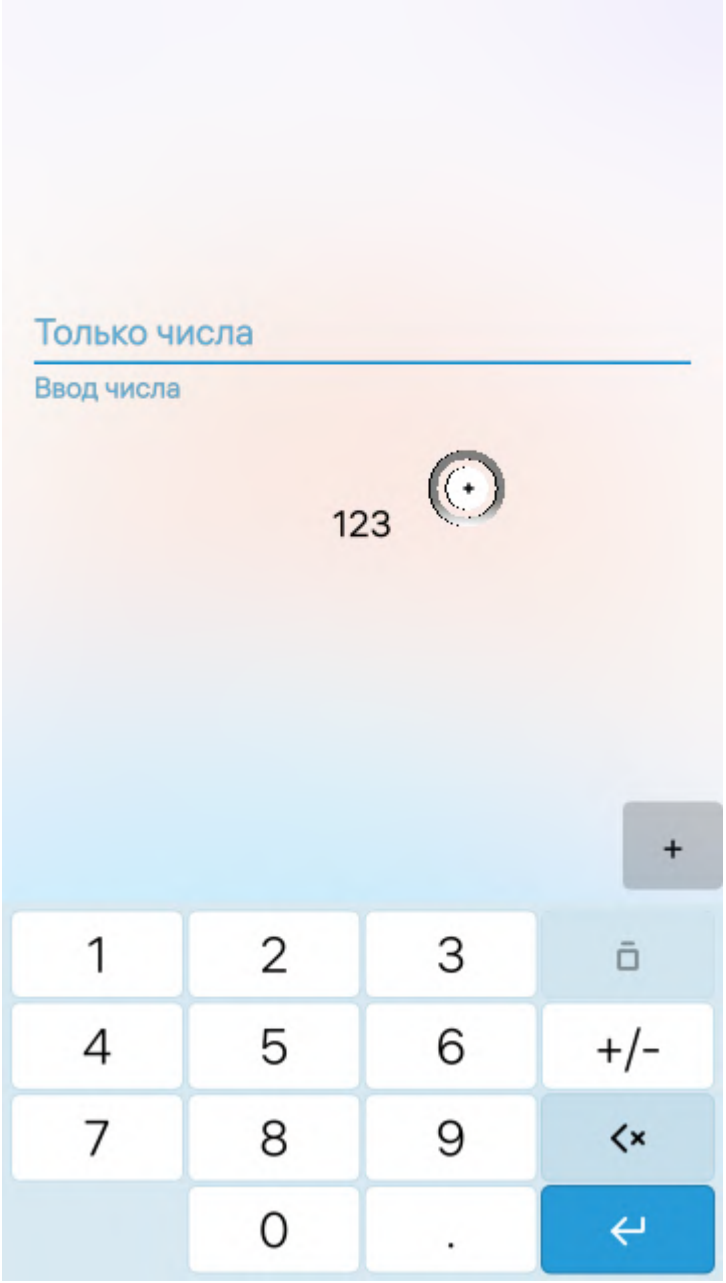


Рис. 1: Текстовое поле для ввода цифр

На следующих страницах появляется кнопка “Back”, возвращающая предыдущую страницу.

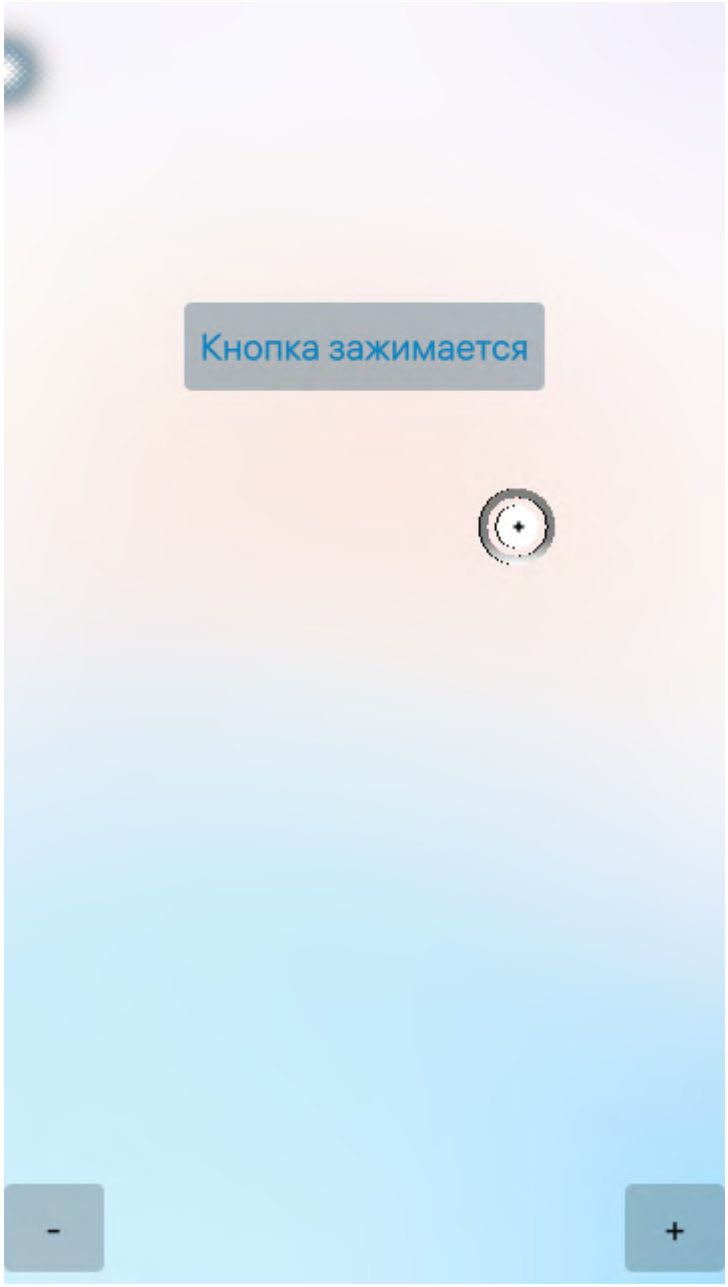


Рис. 2: Кнопка с залипанием

При изменении выбора элемента, в консоль делается вывод.

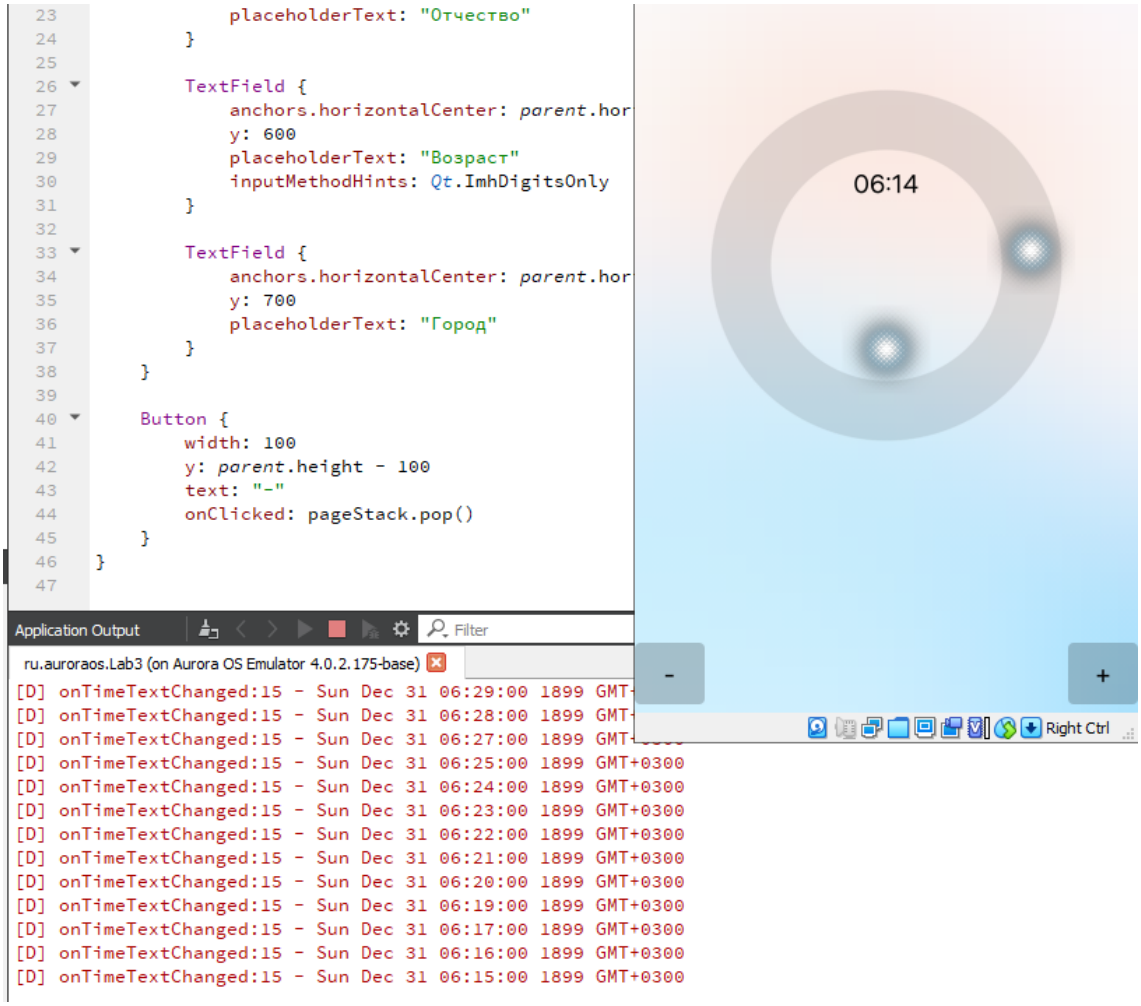


Рис. 3: Выбор времени и вывод в консоли

Заключение

В процессе выполнения данной лабораторной работы мы научились использовать типовые элементы интерфейса.

Приложение

Page1.qml

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

Page {

    TextField {
        anchors.horizontalCenter: parent.horizontalCenter
        y: 300
        placeholderText: "Только числа"
        inputMethodHints: Qt.ImhDigitsOnly
        description: "Ввод числа"
    }

    Button {
        width: 100
        x: parent.width - 100
        y: parent.height - 100
        text: "+"
        onClicked: pageStack.push(Qt.resolvedUrl("Page2.qml"))
    }

    Label {
        anchors.horizontalCenter: parent.horizontalCenter
        y: 500
        text: Store.time
    }
}
```

Page2.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {


    Button {
        anchors.horizontalCenter: parent.horizontalCenter
        y: 300
        text: "Кнопка зажимается"
        onClicked: down = !down
    }
```

```
    Button {
        width: 100
        x: parent.width - 100
        y: parent.height - 100
        text: "+"
        onClicked: pageStack.push(Qt.resolvedUrl("Page3.qml"))
    }

    Button {
        width: 100
        y: parent.height - 100
        text: "-"
        onClicked: pageStack.pop()
    }
}
```

Page3.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {

    Button {
        y: 300
        id: btn
        anchors.horizontalCenter: parent.horizontalCenter
        text: "Кнопка"
    }
    Label {
        y: 400
        anchors.horizontalCenter: parent.horizontalCenter
        text: btn.down ? "Нажато" : "Не нажато"
    }

    Button {
        width: 100
        x: parent.width - 100
        y: parent.height - 100
        text: "+"
        onClicked: pageStack.push(Qt.resolvedUrl("Page4.qml"))
    }

    Button {
        width: 100
        y: parent.height - 100
        text: "-"
```

```
        onClicked: pageStack.pop()
    }
}
```

Page4.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    Button {

        property int counter: 0

        id: btn
        y: 300
        anchors.horizontalCenter: parent.horizontalCenter

        text: "Счетчик"
        onClicked: counter++
    }
    Label {
        y: 400
        anchors.horizontalCenter: parent.horizontalCenter
        text: btn.counter
    }

    Button {
        width: 100
        x: parent.width - 100
        y: parent.height - 100
        text: "+"
        onClicked: pageStack.push(Qt.resolvedUrl("Page5.qml"))
    }

    Button {
        width: 100
        y: parent.height - 100
        text: "-"
        onClicked: pageStack.pop()
    }
}
```

Page5.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {

    DatePicker {
        anchors.centerIn: parent
        date: new Date()
        onDateTextChanged: console.log(this.date)
    }


    Button {
        width: 100
        x: parent.width - 100
        y: parent.height - 100
        text: "+"
        onClicked: pageStack.push(Qt.resolvedUrl("Page6.qml"))
    }

    Button {
        width: 100
        y: parent.height - 100
        text: "-"
        onClicked: pageStack.pop()
    }
}
```

Page6.qml

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

Page {

    TimePicker {
        anchors.centerIn: parent

        id: timePicker
        hour: 6
        minute: 30

        onTimeTextChanged: {
            console.log(this.time)
            Store.time = hour + " : " + minute
```

```
        }
    }
    Label {
        anchors.horizontalCenter: parent.horizontalCenter
        y: 500
        text: timePicker.timeText
    }


    Button {
        width: 100
        x: parent.width - 100
        y: parent.height - 100
        text: "+"
        onClicked: pageStack.push(Qt.resolvedUrl("Page7.qml"))
    }

    Button {
        width: 100
        y: parent.height - 100
        text: "-"
        onClicked: pageStack.pop()
    }
}
```

Page7.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    ComboBox {
        anchors.centerIn: parent
        label: "ComboBox"
        description: "Описание"

        menu: ContextMenu {
            MenuItem { text: "1" }
            MenuItem { text: "2" }
            MenuItem { text: "3" }
            MenuItem { text: "4" }
        }
        onCurrentIndexChanged: console.log(value)
    }

    Button {
        width: 100
        x: parent.width - 100
```

```
        y: parent.height - 100
        text: "+"
        onClicked: pageStack.push(Qt.resolvedUrl("Page8.qml"))
    }

    Button {
        width: 100
        y: parent.height - 100
        text: "-"
        onClicked: pageStack.pop()
    }
}
```

Page8.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    Column {
        anchors.centerIn: parent

        Switch {
            anchors.horizontalCenter: parent.horizontalCenter
            id: mute
            icon.source: "image://theme/icon-m-speaker-mute?"
                         + (checked ? Theme.highlightColor
                                    : Theme.primaryColor)
        }
        Label {
            text: "Звук " + (mute.checked ? "включен" : "выключен")
        }
    }

    Button {
        width: 100
        x: parent.width - 100
        y: parent.height - 100
        text: "+"
        onClicked: pageStack.push(Qt.resolvedUrl("Page9.qml"))
    }

    Button {
        width: 100
        y: parent.height - 100
        text: "-"
        onClicked: pageStack.pop()
    }
```

```
}
```

Page9.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    Slider {
        anchors.horizontalCenter: parent.horizontalCenter
        y: 300

        width: 500

        maximumValue: 10
        minimumValue: 0
        value: 5
        stepSize: 1

        onValueChanged: console.log(value)
        id: slider
    }

    Label {
        anchors.horizontalCenter: parent.horizontalCenter
        y: 400
        text: slider.value
    }

    Button {
        width: 100
        x: parent.width - 100
        y: parent.height - 100
        text: "+"
        onClicked: pageStack.push(Qt.resolvedUrl("Page10.qml"))
    }

    Button {
        width: 100
        y: parent.height - 100
        text: "-"
        onClicked: pageStack.pop()
    }
}
```