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**

# Отчет по практическому заданию для лекции №4

**Выполнила:**
студентка группы 382006-2
Кулёва Анна Андреевна

**Проверил:**
Карчков Денис Александрович

Нижний Новгород
2023

# Содержание

# 1. Цель практического занятия

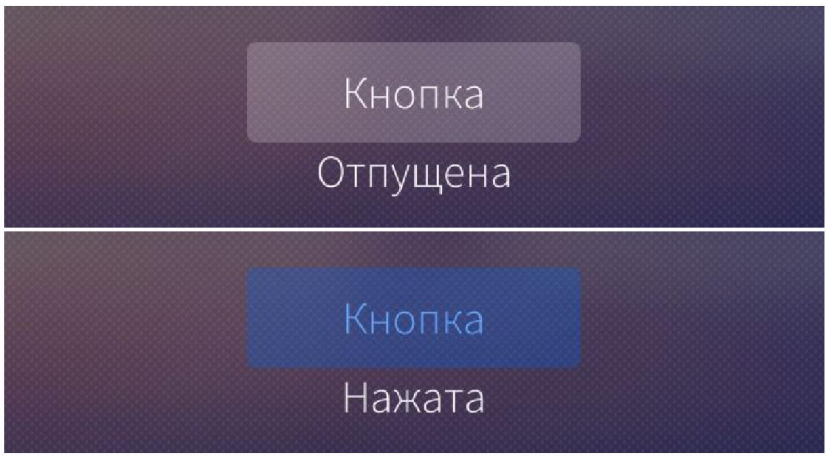
Цель данного практического занятия состоит в том, чтобы научиться применять типовые элементы интерфейса Sailfish OS.

## 2. Постановка задачи

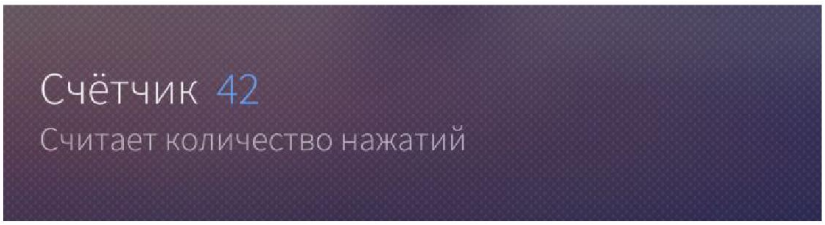
1) Создать текстовое поле для ввода числа с заголовком и подсказкой

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



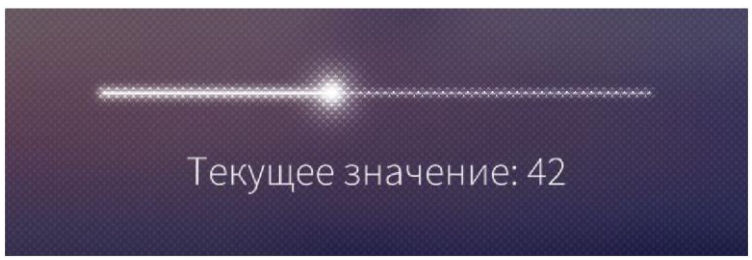

5) Создать селектор даты, который будет отображать выбранную дату в консоли

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. Руководство пользователя

При запуске программы пользователь увидит главную страницу с первым заданием: текстовым полем с заголовком и подсказкой. Переключаться между заданиями можно с помощью кнопок «Вперёд» и «Назад», нажав на которые пользователь перейдёт на следующую страницу или вернётся к предыдущей (рисунок 1).

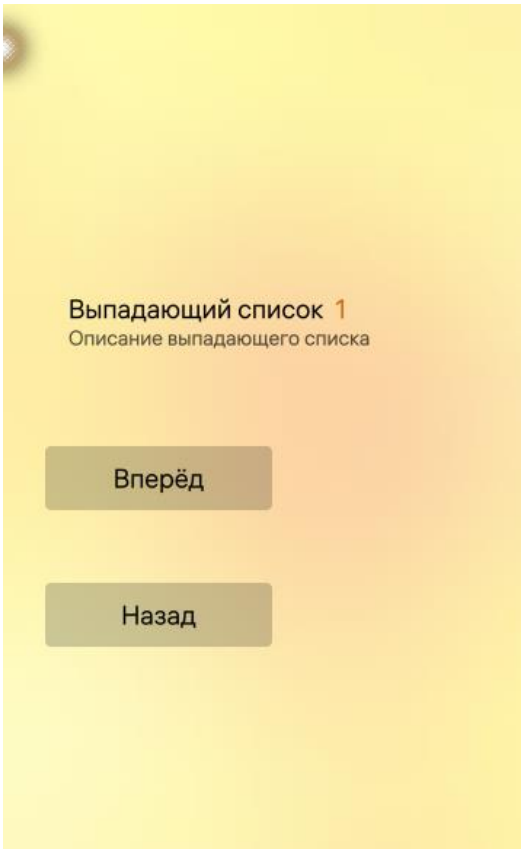



Рисунок 1. Кнопки перемещения между страницами

Для выполнения шагов №5-7 выбранная пользователем информация будет отображаться в консоли (рисунок 2).

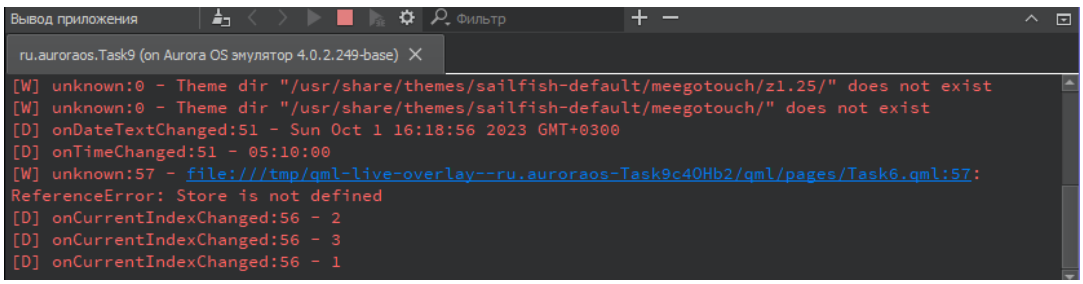



Рисунок 2. Вывод в консоль строки выпадающего списка

# 4. Руководство программиста

Программа реализована на языке программирования QML. Также были созданы 9 новых файлов Task1.qml, …, Task9.qml.

1) Для выполнения шага №1 был реализован элемент TextField. Для создания подзаголовка и подсказки и введения ограничения на ввод были использованы свойства:
   - placeholderText – текст, заполняющий пространство внутри поля до начала ввода
   - inputMethodHints – подсказки по методу ввода
   - description - описание
2) Для выполнения шага №2 был реализован элемент Button. В нём было добавлено пользовательское свойство типа bool: property bool toggle: false. Укажем его в свойство down. При нажатии будем обновлять значение свойства следующим образом, используя свойство onClicked: onClicked: toggle = !toggle.
3) Для выполнения шага №3 был реализован элемент Button со свойством text и текстовое поле Label. Его свойство text принимает соответствующее значение в зависимости от того, нажата ли кнопка или отпущена:
   text: btn.down ? "Нажато" : "Отпущено".
4) Для выполнения шага №4 были реализованы элементы Label и Button. К свойствам кнопки было добавлено пользовательское свойство типа int, которому присваиваем значение 0: property int counter. С помощью функции onClicked реализуем увеличение значения counter на единицу по нажатию кнопки. Значение пользовательского свойства отображается в текстовом поле.
5) Для выполнения шага №5 был реализован элемент DatePicker – селектор даты. Выбранная пользователем дата будет отображаться в консоли при помощи свойства onDateTextChanged.
6) Для выполнения шага №6 были реализованы элементы TimePicker и Timer. В элементе TimePicker заданы значения следующих свойств:
   - id - идентификатор
   - hour – часы (int)
   - minute - минуты (int)

- onTimeChanged: console.log(time.toTimeString()) – во время изменения времени, оно отображается в консоль

В элементе Timer заданы значения следующих свойств:

- id - идентификатор
- interval – значение интервала (int)
- running – состояние работы (bool)
- repeat – повторение (bool)
- onTriggered – действие при срабатывании

В onTriggered реализуем добавление 1 часа при достижении свойства minute значения 60.

7) Для выполнения шага №7 был реализован элемент ComboBox – выпадающий список. В поле menu размещаем элементы MenuItem со свойством text. Эти элементы задают пункты списка. Результат выбора пользователя выводится в консоль с помощью метода onCurrentIndexChanged.
8) Для выполнения шага №8 были реализованы элементы Label и Switch. В текстовое поле выводится состояние переключателя с помощью свойства checked. К переключателю также было добавлено изображение с помощью свойства icon.source.
9) Для выполнения шага №9 были реализованы элементы Label и Slider - ползунок. Slider имеет свойства:
    - id - идентификатор
    - width – ширина
    - label - надпись
    - maximumValue – максимальное значение ползунка
    - minimumValue – минимальное значение ползунка
    - value – стартовое значение ползунка
    - stepSize – значение шага
    - valueText – текст над значением ползунка
    - onValueChanged – действие при изменении значения (передаём значение value в консоль)

    В текстовое поле также передаём значение свойства value.

# Заключение

В данной лабораторной работе я научилась применять типовые элементы интерфейса Sailfish OS. Также были выполнены все шаги практического задания.

# Приложение

## Task1.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {

    Column {
        spacing: 100
        anchors.centerIn: parent

        TextField {
            placeholderText: "Число"

            inputMethodHints: Qt.ImhDigitsOnly
            description: "Число"
        }

        Button {
            text: "Вперёд"
            onClicked: pageStack.push(Qt.resolvedUrl("Task2.qml"))
        }
    }
}
```

## Task2.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {

    Column {
        spacing: 100
        anchors.centerIn: parent

        Button {
            property bool toggle: false
            text: "Залипание"

            down: toggle
            onClicked: toggle = !toggle
        }

        Button {
            text: "Вперёд"
            onClicked: pageStack.push(Qt.resolvedUrl("Task3.qml"))
        }
        Button {
            text: "Назад"
            onClicked: pageStack.pop()
        }
    }
}
```

## Task3.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {

    Column {
        spacing: 100
        anchors.centerIn: parent

        Button {
            id: btn
            text: "Нажать!"
        }
        Label {
            text: btn.down ? "Нажато" : "Отпущено"
        }

        Button {
            text: "Вперёд"
            onClicked: pageStack.push(Qt.resolvedUrl("Task4.qml"))
        }
        Button {
            text: "Назад"
            onClicked: pageStack.pop()
        }
    }
}
```

## Task4.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {

    Column {
        spacing: 100
        anchors.centerIn: parent

        Button {

            property int counter: 0

            id: btn
            text: "Счетчик"
            onClicked: counter++
        }
        Label {
            text: btn.counter
        }

        Button {
            text: "Вперёд"
            onClicked: pageStack.push(Qt.resolvedUrl("Task5.qml"))
        }
        Button {
            text: "Назад"
            onClicked: pageStack.pop()
        }
    }
}
```

## Task5.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {

    Column {
        spacing: 100
        anchors.centerIn: parent

        DatePicker {
            date: new Date()
            daysVisible: true
            weeksVisible: true
            onDateTextChanged: console.log(this.date)
        }

        Button {
            text: "Вперёд"
            onClicked: pageStack.push(Qt.resolvedUrl("Task6.qml"))
        }
        Button {
            text: "Назад"
            onClicked: pageStack.pop()
        }
    }
}
```

## Task6.qml

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

Page {
    Column {
        spacing: 100
        anchors.centerIn: parent

        TimePicker {
            id: timePicker
            hour: 5
            minute: 10
            onTimeChanged: console.log(time.toTimeString())
        }

        Timer {
            id: timer
            interval: 1000
            running: Store.run
            repeat: true
            onTriggered: {
                if (timePicker.minute === 59) {
                    timePicker.minute = 0
                    timePicker.hour++
                }
                timePicker.minute++

            }
        }

        Button {
```

```
            text: "Вперёд"
            onClicked: pageStack.push(Qt.resolvedUrl("Task7.qml"))
        }
        Button {
            text: "Назад"
            onClicked: pageStack.pop()
        }
    }
}
```

## Task7.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {

    Column {
        spacing: 100
        anchors.centerIn: parent
        width: 600

        ComboBox {
            label: "Выпадающий список"
            description: "Описание выпадающего списка"
            menu: ContextMenu {
                MenuItem { text: "1" }
                MenuItem { text: "2" }
                MenuItem { text: "3" }
            }
            onCurrentIndexChanged: console.log(value)
        }

        Button {
            text: "Вперёд"
            onClicked: pageStack.push(Qt.resolvedUrl("Task8.qml"))
        }
        Button {
            text: "Назад"
            onClicked: pageStack.pop()
        }
    }
}
```

## Task8.qml

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

Page {

    Column {
        spacing: 100
        anchors.centerIn: parent

        Switch {
            id: mute
            icon.source: "image://theme/icon-m-speaker-mute?"
                         + (checked ? Theme.highlightColor
                                    : Theme.primaryColor)
            onCheckedChanged: Store.run = !Store.run
        }
```

```
        Label {
            text: "Звук " + (mute.checked ? "включён" : "выключен")
        }

        Button {
            text: "Вперёд"
            onClicked: pageStack.push(Qt.resolvedUrl("Task9.qml"))
        }
        Button {
            text: "Назад"
            onClicked: pageStack.pop()
        }
    }
}
```

## Task9.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {

    Column {
        spacing: 100
        anchors.centerIn: parent

        Slider {
            width: 500
            label: "Слайдер"
            maximumValue: 50
            minimumValue: 0
            value: 10
            stepSize: 2
            valueText: value
            onValueChanged: console.log(value)
            id: slider
        }

        Label {
            text: + slider.value
        }

        Button {
            text: "Назад"
            onClicked: pageStack.pop()
        }
    }
}
```