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 №6

Выполнила:
студентка группы 382006-2
Кулёва Анна Андреевна

Проверил:
Карчков Денис Александрович

Нижний Новгород
2023

Содержание

Содержание 2
1. Цель практического занятия 3
2. Постановка задачи 4
3. Руководство пользователя 6
4. Руководство программиста 11
Заключение 13
Приложение 14

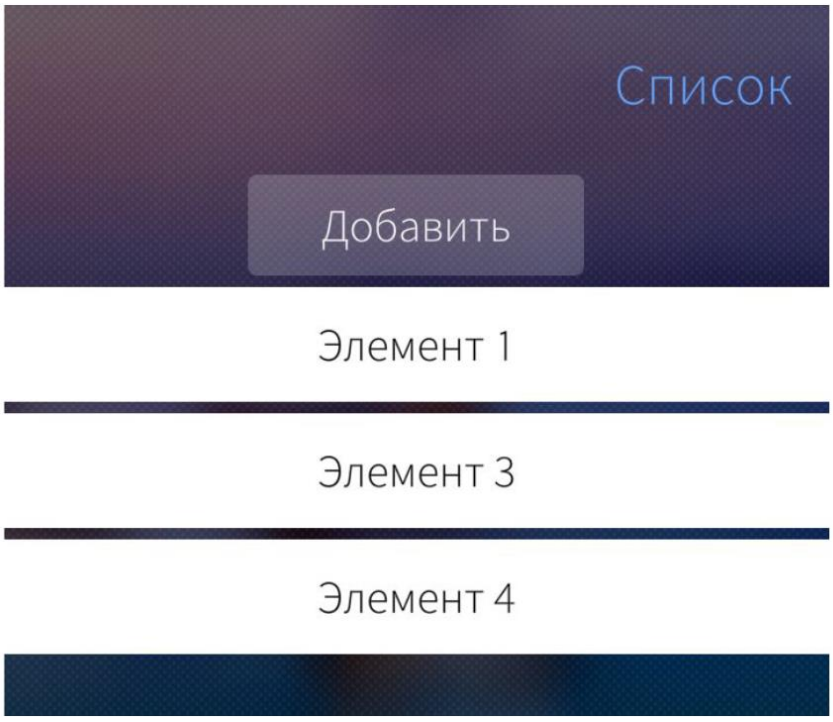
1. Цель практического занятия

Цель данного практического занятия состоит в том, чтобы научиться использовать различные модели для отображения данных в прокручиваемых списках, взаимодействовать с базой данных и управлять настройками приложения.

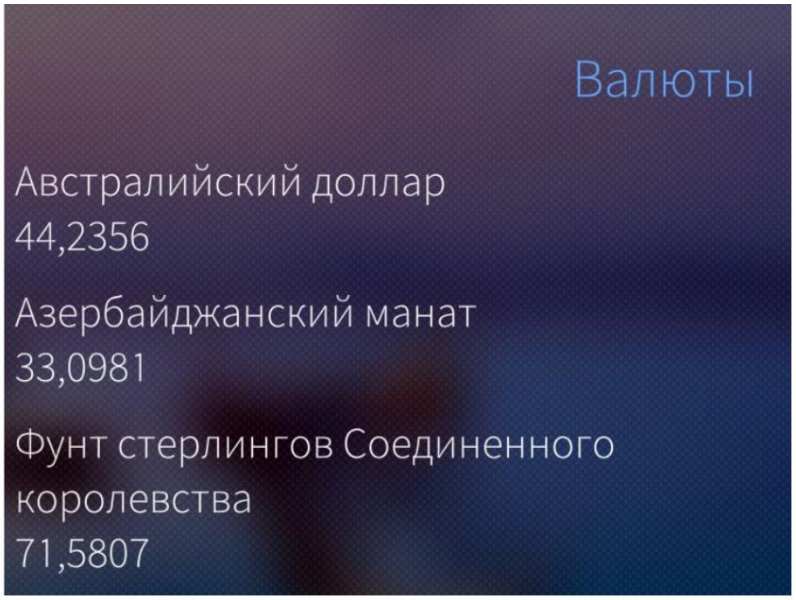
2. Постановка задачи

1. Создать приложение, которое позволяет отображать список из прямоугольников с использованием ListModel. В модели должны настраиваться цвет фона и текста внутри прямоугольника. Текст содержит название цвета фона прямоугольника

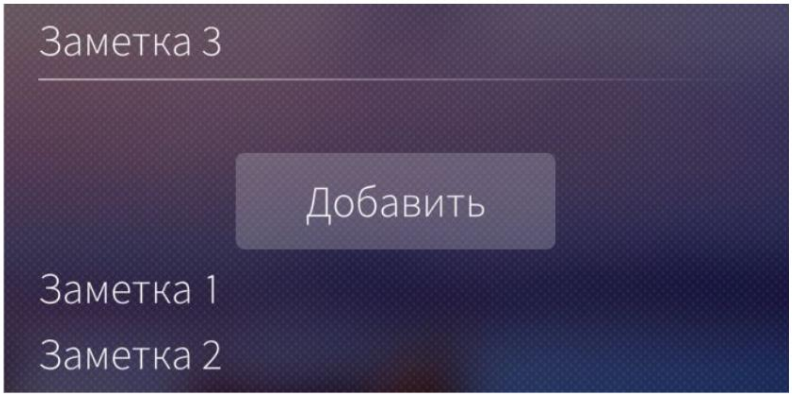
2. Создать приложение, которое позволяет отображать список из прямоугольников. Нажатие на кнопку над списком добавит новый элемент. Нажатие на элемент в списке удалит его из списка. В прямоугольниках должен отображаться порядковый номер, присваиваемый при добавлении в список. При удалении элементов порядковые номера у добавленных прямоугольников остаются неизменными.



3. Выполнить задание 1 с использованием javascript-модели.
4. Получить и отобразить курсы валют из ресурса ЦБ РФ по адресу http://www.cbr.ru/scripts/XML_daily.asp.



5. Выполнить задание 4 с использованием XMLHttpRequest.
6. Создать приложение, позволяющее добавлять и удалять заметки с использованием базы данных и отображать их в списке. Текстовое поле служит для ввода текста, кнопка для добавления заметки, нажатие на заметку удалит её.



7. Создать приложение с текстовым полем и полем с флажком, значение которых сохраняется в настройках приложения с помощью ConfigurationValue.
8. Выполнить задание 7 с помощью ConfigurationGroup

3. Руководство пользователя

При запуске программы пользователь увидит главную страницу с первым заданием: списком прямоугольников с текстом, содержащим название фона прямоугольников (рисунок 1). Третье задание выглядит аналогично. Переключаться между заданиями можно с помощью кнопок «Следующая задача» и «Назад», нажав на которые пользователь перейдёт на следующую страницу или вернётся к предыдущей.

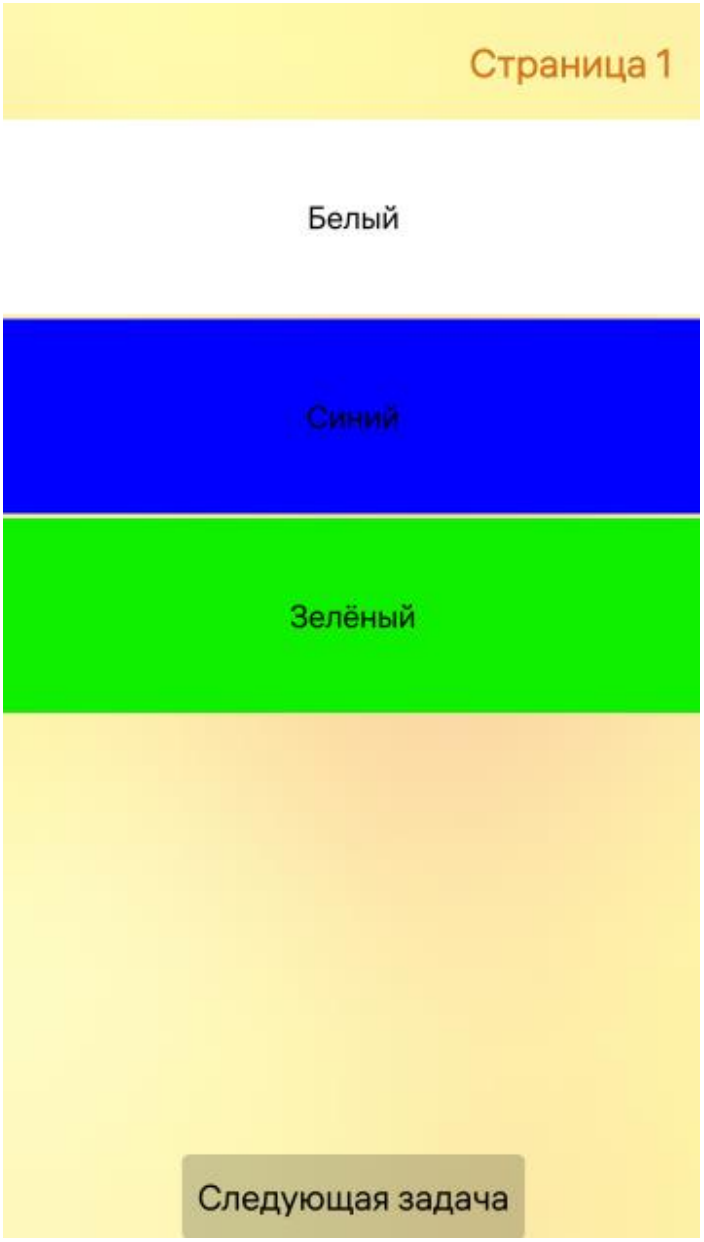


Рисунок 1. Список прямоугольников

При переходе к следующему заданию, пользователь увидит список прямоугольников с порядковыми номерами на них. При нажатии на кнопку «Добавить» в список добавится новый элемент, а при нажатии любой из элементов списка, тот удалится (рисунок 2).

Рисунок 2. Список прямоугольников с возможностью добавления нового элемента и удаления уже существующего

На следующей странице отображены курсы валют из ресурса ЦБ РФ. Список можно прокрутить вниз (рисунок 3). Пятое задание выглядит аналогично.

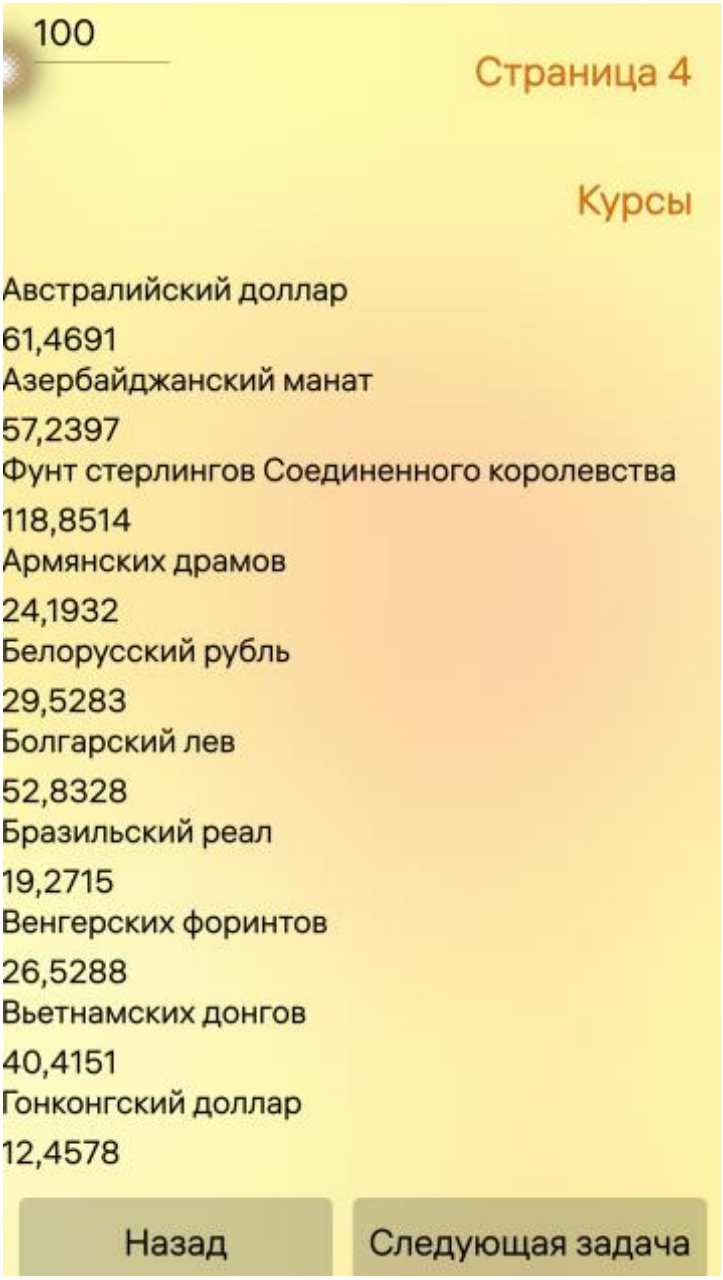


Рисунок 3. Курсы валют

На шестой странице пользователь сможет ввести новую заметку с помощью текстового поля, а затем добавить её при нажатии на кнопку «Добавить» (рисунок 4).

Рисунок 4. Добавление заметок

На 7 и 8 страницах пользователь может ввести текст в текстовое полем и поле с флажком, значение которых сохраняется в настройках приложения (рисунок 5-6).

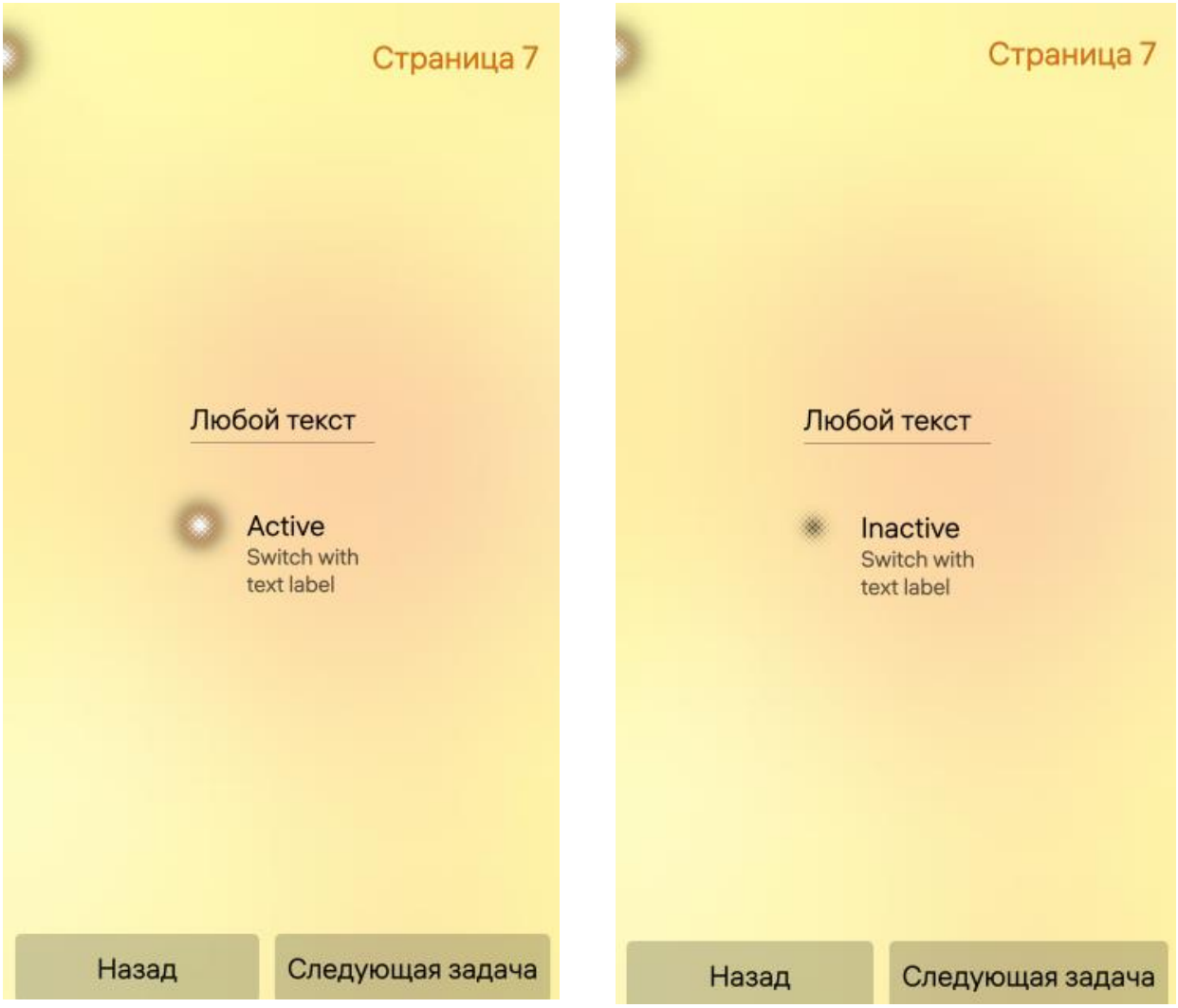


Рисунок 5-6. Текстовое поле с флажком (активным и неактивным)

4. Руководство программиста

Программа реализована на языке программирования QML.

1) Выполнение шага №1.

Начинаем с установки модели данных для элементов в SilicaListView. Передаем модель по id и применяем параметры каждого элемента в свойстве delegate.

2) Выполнение шага №2.

javascript-модель представляет собой массив элементов списка. Отличие от ListModel заключается в использовании свойства modelData при обращении к полям элемента при его отображении.

3) Выполнение шага №3.

Создаем кнопку, которая вызывает метод append модели и передает значение count. Каждый элемент списка представляет собой кнопку, которая позволяет удалять текущий элемент из модели по индексу.

4) Выполнение шага №4.

Для работы с внешними ресурсами используем XmlListModel. Указываем адрес xml-ресурса в свойстве source, путь в xml-схеме до нужных элементов в свойстве query, и преобразуем xml-поля в объекты языка JS с помощью XmlRole, которые будут использованы в модели.

5) Выполнение шага №5.

При использовании XMLHttpRequest метод получения контента описывается с помощью инструментов JS. Создаётся объект XMLHttpRequest, выполняется GET запрос асинхронно. Устанавливается обработчик по событию onreadystatechange, где проверяется, завершен ли запрос, и если да, то текст ответа передается в параметр xml-модели. Запрос отправляется с помощью метода send.

6) Выполнение шага №6.

Работа с БД осуществляется через транзакции на языке SQL. Мы разрабатываем интерфейс для выполнения запросов к базе данных, позволяющий получать, добавлять и удалять записи. После инициализации объекта Item и установки соединения с базой данных, SilicaListView используется для обновления модели данных. Мы также создаём

минимальный пользовательский интерфейс для ввода текста и добавления записок в список. Каждое изменение списка приводит к выполнению запроса к базе данных и последующему обновлению модели списка.

7) Выполнение шага №7.

Для задания глобальных параметров приложения используем ConfigurationValue. Указываем идентификатор параметра в свойстве key и значение по умолчанию в свойстве defaultValue.

8) Выполнение шага №8.

Для хранения набора параметров используем ConfigurationGroup, где настраиваем нужные параметры как свойства элемента. Идентификатор группы задается в свойстве path.

Заключение

В данной лабораторной работе я научилась использовать различные модели для отображения данных в прокручиваемых списках, взаимодействовать с базой данных и управлять настройками приложения. Также были выполнены все шаги практического задания.

Приложение

Page1.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    PageHeader {
        objectName: "pageHeader"
        title: "Страница 1"
    }

    ListModel {
        id: rectanglesModel
        ListElement { idx: 1; name: "Белый"; bgcolor: "#ffffff"; }
        ListElement { idx: 2; name: "Синий"; bgcolor: "#0000ff"; }
        ListElement { idx: 3; name: "Зелёный"; bgcolor: "#0ff000"; }
    }

    Item {
        anchors {
            left: parent.left; right: parent.right;
            verticalCenter: parent.verticalCenter;
        }
        height: parent.height * 0.8

        SilicaListView {
            anchors.fill: parent
            model: rectanglesModel
            delegate: Rectangle {
                color: bgcolor
                width: parent.width
                height: 200
                Text {
                    text: name
                    anchors.centerIn: parent
                }
            }
            spacing: 5
        }
    }


    Row {
            anchors.horizontalCenter: parent.horizontalCenter
            anchors.bottom: parent.bottom
            spacing: 20

            Button {
                text: "Следующая задача"
                onClicked: pageStack.push(Qt.resolvedUrl(qsTr("Page2.qml")))
            }
        }

}
```

Page2.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    PageHeader {
        objectName: "pageHeader"
        title: "Страница 2"    }


    ListModel {
        id: rectanglesModel
        ListElement { idx: 1; name: "Белый"; bgcolor: "#ffffff"; }
        ListElement { idx: 2; name: "Белый"; bgcolor: "#ffffff"; }
        ListElement { idx: 3; name: "Белый"; bgcolor: "#ffffff"; }
    }

    Item {
        anchors {
            left: parent.left; right: parent.right;
            verticalCenter: parent.verticalCenter;
        }
        height: parent.height * 0.8


        SilicaListView {
            anchors.fill: parent
            model: rectanglesModel
            delegate: Rectangle {
                color: bgcolor
                width: parent.width
                height: 200
                Text {
                    text: "Элемент " + idx
                    anchors.centerIn: parent
                }

                MouseArea {
                    anchors.fill: parent
                    onClicked: {
                        for (var i = 0; i < rectanglesModel.rowCount(); i++)
{
                            if (rectanglesModel.get(i).idx === idx) {
                                rectanglesModel.remove(i)
                            }
                        }
                    }
                }
            }
            spacing: 5
        }

        Button {
            text: "Добавить"
            anchors {
                bottom: parent.bottom;
                horizontalCenter: parent.horizontalCenter;
            }
            onClicked: {
                var prev = rectanglesModel.rowCount() - 3
                prev = prev < 0 ? 0 : prev
```

```
                var newName = rectanglesModel.get(prev)
                newName = newName === undefined ? "Белый" : newName.name
                var newColor = rectanglesModel.get(prev)
                newColor = "#ffffff"

                rectanglesModel.append({
                                           idx: rectanglesModel.rowCount() +
1,
                                           name: newName,
                                           bgcolor: newColor
                                       })
            }
        }
    }

    Row {
            anchors.horizontalCenter: parent.horizontalCenter
            anchors.bottom: parent.bottom
            spacing: 20
            Button {
                text: "Назад"
                onClicked: pageStack.pop()
            }
            Button {
                text: "Следующая задача"
                onClicked: pageStack.push(Qt.resolvedUrl(qsTr("Page3.qml")))
            }
        }

}
```

Page3.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0

Page {
    PageHeader {
        objectName: "pageHeader"
        title: "Страница 3"
    }


    Item {
        id: container
        anchors {
            left: parent.left; right: parent.right;
            verticalCenter: parent.verticalCenter;
        }
        height: parent.height * 0.8

        property var rectanglesModel: [
            { idx: 1, name: "Белый", bgcolor: "#ffffff" },
            { idx: 2, name: "Синий", bgcolor: "#0000ff" },
            { idx: 3, name: "Зелёный", bgcolor: "#0ff000" },
        ]

        SilicaListView {
            anchors.fill: parent
            model: container.rectanglesModel
            delegate: Rectangle {
                color: modelData.bgcolor
                width: parent.width
```

```
                height: 200
                Text {
                    text: modelData.name
                    anchors.centerIn: parent
                }
            }
            spacing: 5
        }
    }

    Row {
            anchors.horizontalCenter: parent.horizontalCenter
            anchors.bottom: parent.bottom
            spacing: 20
            Button {
                text: "Назад"
                onClicked: pageStack.pop()
            }
            Button {
                text: "Следующая задача"
                onClicked: pageStack.push(Qt.resolvedUrl(qsTr("Page4.qml")))
            }
        }
}
```

Page4.qml

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

Page {
    PageHeader {
        objectName: "pageHeader"
        title: "Страница 4"
    }


    Item {
        id: container
        anchors {
            left: parent.left; right: parent.right;
            verticalCenter: parent.verticalCenter;
        }
        height: parent.height * 0.8

        XmlListModel {
            id: xmlListModel
            source: "http://www.cbr.ru/scripts/XML_daily.asp"
            query: "//Valute"
            XmlRole { name: "Name"; query: "Name/string()" }
            XmlRole { name: "Value"; query: "Value/string()" }
        }

        SilicaListView {
            anchors.fill: parent
            model: xmlListModel
            header: PageHeader { title: "Курсы" }

            delegate: Column {
                spacing: 10
                Text { text: Name}
                Text { text: Value}
```

```
            }
        }
    }

    TextField {
        anchors.left: parent.left
        anchors.top: parent.top
        width: 200
        id: textField
        validator: DoubleValidator{}
        text: "100"
        onTextChanged: {
            console.log(text - 0)
        }
    }

    Row {
            anchors.horizontalCenter: parent.horizontalCenter
            anchors.bottom: parent.bottom
            spacing: 20
            Button {
                text: "Назад"
                onClicked: pageStack.pop()
            }
            Button {
                text: "Следующая задача"
                onClicked: pageStack.push(Qt.resolvedUrl(qsTr("Page5.qml")))
            }
        }
}
```

Page5.qml

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

Page {
    PageHeader {
        objectName: "pageHeader"
        title: "Страница 5"
    }

    Item {
        id: container
        anchors {
            left: parent.left; right: parent.right;
            verticalCenter: parent.verticalCenter;
        }
        height: parent.height * 0.8

        function loadNews() {
            var xhr = new XMLHttpRequest();
            xhr.open('GET', 'http://www.cbr.ru/scripts/XML_daily.asp', true);
            xhr.setRequestHeader('Content-type', 'application/x-www-form-
urlencoded; charset=ISO-8859-1')
            xhr.onreadystatechange = function() {
                if (xhr.readyState === XMLHttpRequest.DONE) {
                    xmlListModel.xml = xhr.responseText;
                    console.log(xhr.responseText)
                }
            }
            xhr.send();
```

```
        }

        XmlListModel {
            id: xmlListModel
            query: "/ValCurs/Valute"
            XmlRole { name: "Name"; query: "Name/string()"; }
            XmlRole { name: "Value"; query: "Value/string()"; }
        }

        SilicaListView {
            anchors.fill: parent
            model: xmlListModel
            header: PageHeader { title: "Курсы" }
            section {
                property: 'Name'
                delegate: SectionHeader { text: section }
            }
            delegate: Text { text: Value; }
            Component.onCompleted: {
                container.loadNews()
            }
        }
    }

    Row {
            anchors.horizontalCenter: parent.horizontalCenter
            anchors.bottom: parent.bottom
            spacing: 20
            Button {
                text: "Назад"
                onClicked: pageStack.pop()
            }
            Button {
                text: "Следующая задача"
                onClicked: pageStack.push(Qt.resolvedUrl(qsTr("Page6.qml")))
            }
        }
}
```

Page6.qml

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

Page {
    PageHeader {
        objectName: "pageHeader"
        title: "Страница 5"
    }

    Item {
        id: container
        anchors {
            left: parent.left; right: parent.right;
            verticalCenter: parent.verticalCenter;
        }
        height: parent.height * 0.8

        function loadNews() {
            var xhr = new XMLHttpRequest();
            xhr.open('GET', 'http://www.cbr.ru/scripts/XML_daily.asp', true);
```

```
            xhr.setRequestHeader('Content-type', 'application/x-www-form-
urlencoded; charset=ISO-8859-1')
            xhr.onreadystatechange = function() {
                if (xhr.readyState === XMLHttpRequest.DONE) {
                    xmlListModel.xml = xhr.responseText;
                    console.log(xhr.responseText)
                }
            }
            xhr.send();
        }

        XmlListModel {
            id: xmlListModel
            query: "/ValCurs/Valute"
            XmlRole { name: "Name"; query: "Name/string()"; }
            XmlRole { name: "Value"; query: "Value/string()"; }
        }

        SilicaListView {
            anchors.fill: parent
            model: xmlListModel
            header: PageHeader { title: "Курсы" }
            section {
                property: 'Name'
                delegate: SectionHeader { text: section }
            }
            delegate: Text { text: Value; }
            Component.onCompleted: {
                container.loadNews()
            }
        }
    }

    Row {
            anchors.horizontalCenter: parent.horizontalCenter
            anchors.bottom: parent.bottom
            spacing: 20
            Button {
                text: "Назад"
                onClicked: pageStack.pop()
            }
            Button {
                text: "Следующая задача"
                onClicked: pageStack.push(Qt.resolvedUrl(qsTr("Page6.qml")))
            }
        }
}
```

Page7.qml

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

Page {
    PageHeader {
        objectName: "pageHeader"
        title: "Страница 7"
    }

    ConfigurationValue {
        id: setting_1
```

```
        key: "/apps/app_name/setting_1"
        defaultValue: "Menu Default"
    }

    ConfigurationValue {
        id: setting_2
        key: "/apps/app_name/setting_2"
        defaultValue: false
    }


    Column {
                anchors.centerIn: parent
                TextField {
                    width: 300
                    text: "Текст"
                    onTextChanged: {
                        setting_1.value = text
                        console.log(setting_1.value)
                    }
                }

                TextSwitch {
                    text: checked ? qsTr("Active") : qsTr("Inactive")
                    description: qsTr("Switch with text label")
                    onCheckedChanged: {
                        setting_2.value = checked
                        console.log(setting_2.value)
                    }
                }
            }

    Row {
            anchors.horizontalCenter: parent.horizontalCenter
            anchors.bottom: parent.bottom
            spacing: 20
            Button {
                text: "Назад"
                onClicked: pageStack.pop()
            }
            Button {
                text: "Следующая задача"
                onClicked: pageStack.push(Qt.resolvedUrl(qsTr("Page8.qml")))
            }
        }
}
```

Page8.qml

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

Page {
    PageHeader {
        objectName: "pageHeader"
        title: "Страница 8"
    }

    ConfigurationGroup {
        id: settings
        path: "/apps/app_name/settings"
```

```
        property string tf: "empty"
        property bool sw: false
    }

    Column {
                        anchors.centerIn: parent
                        TextField {
                            width: 300
                            text: "Текст"
                            onTextChanged: {
                                settings.tf = text
                                //console.log(settings.tf)
                            }
                        }

                        TextSwitch {
                            text: checked ? qsTr("Active") : qsTr("Inactive")
                            description: qsTr("Switch with text label")
                            onCheckedChanged: {
                                settings.sw = checked
                                console.log(settings.sw)
                            }
                        }
                    }
    Row {
            anchors.horizontalCenter: parent.horizontalCenter
            anchors.bottom: parent.bottom
            spacing: 20
            Button {
                text: "Назад"
                onClicked: pageStack.pop()
            }

        }
}
```