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 №6

«Инструменты разработки мобильных приложений»

Выполнил:
студент группы 381908-4
Рябцев М. В.

Нижний Новгород
2022

Оглавление

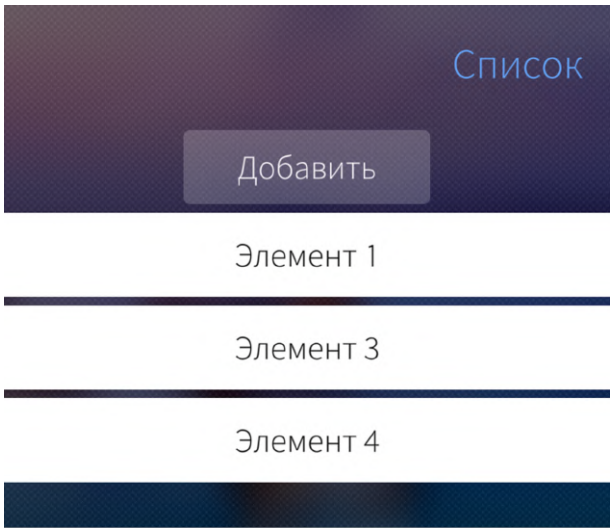
Цель задачи

В данном лабораторной работе требуется научиться использовать различные модели для отображения данных в прокручиваемых списках, взаимодействовать с базой данных и управлять настройками приложения.

Постановка задачи

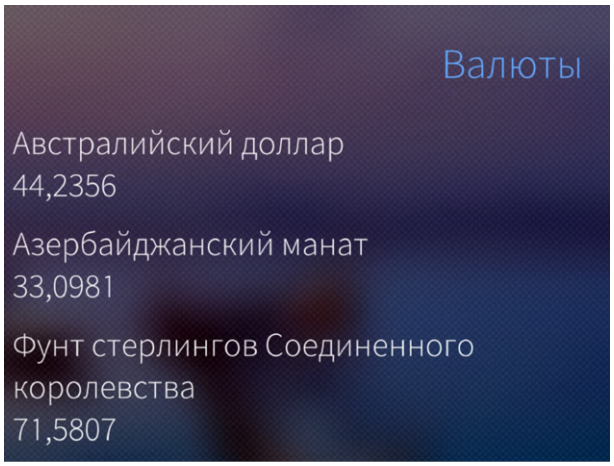
1. Создать приложение, которое позволяет отображать список из прямоугольников с использованием **ListModel**. В модели должны настраиваться цвет фона и текста внутри прямоугольника. Текст содержит название цвета фона прямоугольника.

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.



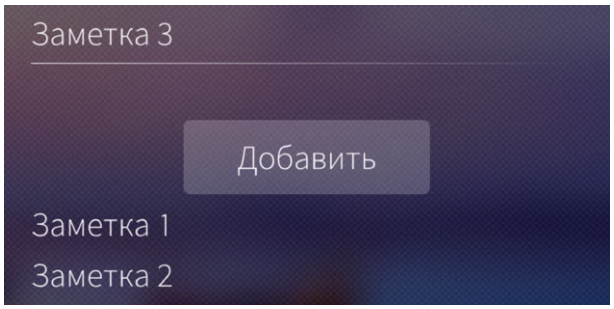
3. Выполнить задание 1 с использованием javascript-модели.

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**.

Описание программной реализации

1. Прямоугольники с использованием ListModel:

```
ListModel {
    id: rectanglesModel
    ListElement { idx: 1; name: "Белый"; bgcolor: "#ffffff"; }
    ListElement { idx: 2; name: "Синий"; bgcolor: "#0000ff"; }
    ListElement { idx: 3; name: "Черный"; bgcolor: "#000000"; }
}
```

   Отображение:

```
SilicaListView {
    anchors.fill: parent
    model: rectanglesModel
    delegate: Rectangle {
        color: bgcolor
        width: parent.width
        height: 200
        Text {
            text: name
            anchors.centerIn: parent
            color: Func.invertColor(bgcolor, 0)
        }
    }
    spacing: 5
}
```

2. Список из прямоугольников, с добавлением и удалением
   Добавление:

```
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 = newColor === undefined ? "#fff" : newColor.bgcolor

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

```
        }
    }
```

Удаление при нажатии:

```
    SilicaListView {
        anchors.fill: parent
        model: rectanglesModel
        delegate: Rectangle {
            color: bgcolor
            width: parent.width
            height: 200
            Text {
                text: idx + " " + name
                anchors.centerIn: parent
                color: Func.invertColor(bgcolor, 0)
            }

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

3. Прямоугольники с использованием ListModel и JavaScript

```
    property var rectanglesModel: [
        { idx: 1, name: "Белый", bgcolor: "#ffffff" },
        { idx: 2, name: "Синий", bgcolor: "#0000ff" },
        { idx: 3, name: "Черный", bgcolor: "#000000" },
        { idx: 4, name: "Красный", bgcolor: "#ff0000" },
    ]
```

Отображение:

```
    SilicaListView {
        anchors.fill: parent
        model: container.rectanglesModel
        delegate: Rectangle {
            color: modelData.bgcolor
            width: parent.width
            height: 200
            Text {
                text: modelData.name
                anchors.centerIn: parent
                color: Func.invertColor(modelData.bgcolor, 0)
```

```
        }
    }
    spacing: 5
}
```

4. Отображение курсов валют через XmlListModel

```
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" }
    section {
        property: 'Name'
        delegate: SectionHeader { text: section }
    }
    delegate: Text { text: Value; }
}
```

5. Отображение курсов валют через XMLHttpRequest

```
function loadNews() {
    var xhr = new XMLHttpRequest();
    xhr.open('GET', 'http://www.cbr.ru/scripts/XML_daily.asp', true);
    // xhr.setRequestHeader("Content-Type", "text/plain;charset=UTF-8");
    xhr.onreadystatechange = function() {
        if (xhr.readyState === XMLHttpRequest.DONE) {
            xmlListModel.xml = xhr.responseText;
            // console.log(decodeURIComponent(escape(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()
        }
    }
```

6. Заметки с использованием LocalStorage

```
    property var db: LocalStorage.openDatabaseSync("QDeclarativeExampleDB", "1.0"

    Button {
        text: "Добавить"
        onClicked: {
            db.transaction(function(tx) {
                tx.executeSql("INSERT INTO notes (note_text) VALUES(?);", [txtfie

                // Show all added greetings
                var rs = tx.executeSql('SELECT * FROM notes');

                var r = []
                for (var i = 0; i < rs.rows.length; i++) {
                    r.push(rs.rows.item(i))
                }
                console.log(r)
                container.notesModel = r
            });
        }
    }

    SilicaListView {
        anchors.fill: parent
        model: container.notesModel
        delegate: Label {
            width: parent.width
            height: 100
            Text {
                text: modelData.note_text
                anchors.centerIn: parent
            }
        }
        spacing: 5
    }

    function findGreetings() {
        db.transaction(
                    function(tx) {
                        // Create the database if it doesn't already exist
```

```
                    tx.executeSql('CREATE TABLE IF NOT EXISTS notes(note_text

                    // Add (another) greeting row
                    tx.executeSql('INSERT INTO notes VALUES(?)', [ 'hello' ])

                    // Show all added greetings
                    var rs = tx.executeSql('SELECT * FROM notes');

                    var r = []
                    for (var i = 0; i < rs.rows.length; i++) {
                        r.push(rs.rows.item(i))
                    }
                    console.log(r)
                    container.notesModel = r
                }
                )
    }
    Component.onCompleted: findGreetings()
```

7. ConfigurationValue

```
    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
    }
```

8. ConfigurationGroup

```
    ConfigurationGroup {
        id: settings
        path: "/apps/app_name/settings"
        property var tf: "empty"
        property bool sw: false
    }
```

Руководство пользователя

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

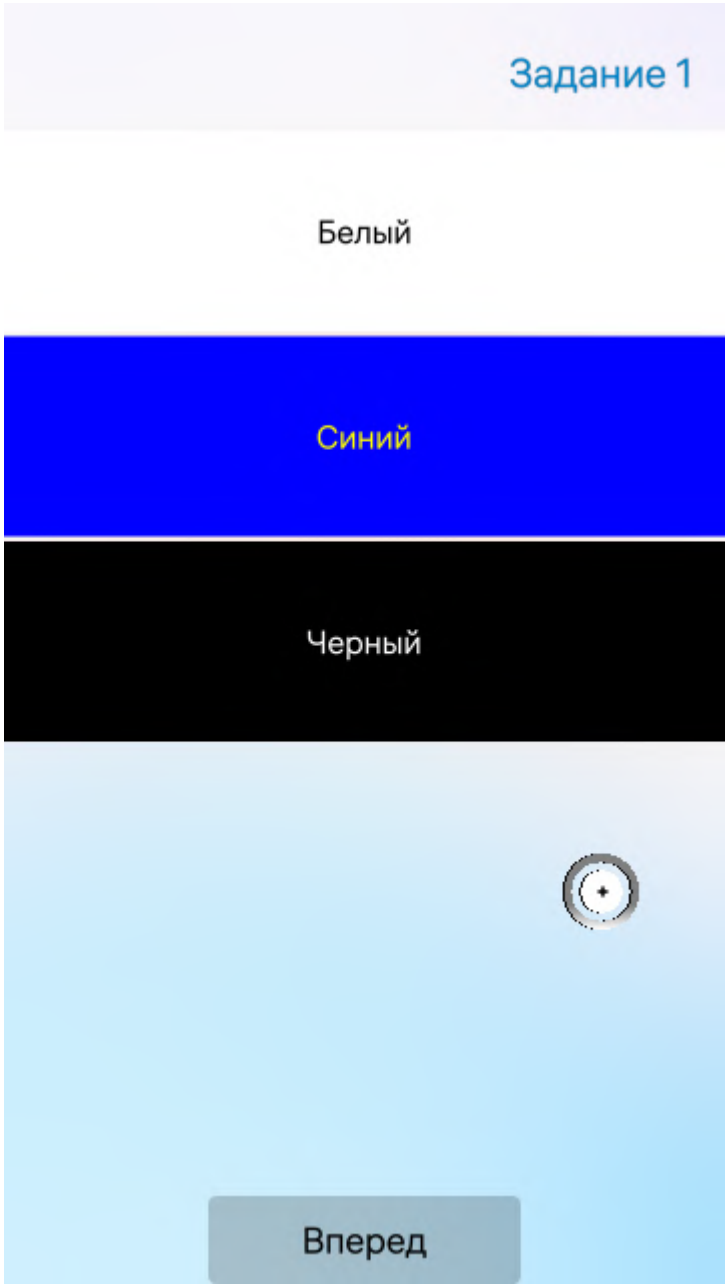


Рис. 1: Страница с первым заданием

На второй странице по нажатии на кнопку “Добавить” появляются новые прямоугольники, при нажатии на сами прямоугольники, они удаляются..

Рис. 2: Страница со вторым заданием

Четвертое задание представляет из себя список курсов валют, загружаемых из интернета.

Рис. 3: Курсы валют

В шестом задании пользователь вводит текст и добавляет новые заметки.

Рис. 4: Заметки

Заключение

В процессе выполнения данной лабораторной работы мы научились использовать различные модели для отображения данных в прокручиваемых списках, взаимодействовать с базой данных и управлять настройками приложения.

Приложение

Task1.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0
import "func.js" as Func

Page {
    objectName: "mainPage"
    allowedOrientations: Orientation.All

    PageHeader {
        objectName: "pageHeader"
        title: qsTr("Задание 1")
    }

    ListModel {
        id: rectanglesModel
        ListElement { idx: 1; name: "Белый"; bgcolor: "#ffffff"; }
        ListElement { idx: 2; name: "Синий"; bgcolor: "#0000ff"; }
        ListElement { idx: 3; name: "Черный"; bgcolor: "#000000"; }
    }

    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
                    color: Func.invertColor(bgcolor, 0)
                }
            }
            spacing: 5
        }
    }
```

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

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

func.js

```
function padZero(str, len) {
    len = len || 2;
    var zeros = new Array(len).join('0');
    return (zeros + str).slice(-len);
}

function invertColor(hex, bw) {
    if (hex.indexOf('#') === 0) {
        hex = hex.slice(1);
    }
    // convert 3-digit hex to 6-digits.
    if (hex.length === 3) {
        hex = hex[0] + hex[0] + hex[1] + hex[1] + hex[2] + hex[2];
    }
    if (hex.length !== 6) {
        throw new Error('Invalid HEX color.');
    }
    var r = parseInt(hex.slice(0, 2), 16),
        g = parseInt(hex.slice(2, 4), 16),
        b = parseInt(hex.slice(4, 6), 16);
    if (bw) {
        // https://stackoverflow.com/a/3943023/112731
        return (r * 0.299 + g * 0.587 + b * 0.114) > 186
            ? '#000000'
            : '#FFFFFF';
    }
    // invert color components
    r = (255 - r).toString(16);
    g = (255 - g).toString(16);
    b = (255 - b).toString(16);
    // pad each with zeros and return
    return "#" + padZero(r) + padZero(g) + padZero(b);
}
```

Task2.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0
import "func.js" as Func

Page {
    objectName: "mainPage"
    allowedOrientations: Orientation.All

    PageHeader {
        objectName: "pageHeader"
        title: qsTr("Задание 2")
    }

    ListModel {
        id: rectanglesModel
        ListElement { idx: 1; name: "Белый"; bgcolor: "#ffffff"; }
        ListElement { idx: 2; name: "Синий"; bgcolor: "#0000ff"; }
        ListElement { idx: 3; name: "Черный"; bgcolor: "#000000"; }
    }

    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: idx + " " + name
                    anchors.centerIn: parent
                    color: Func.invertColor(bgcolor, 0)
                }

                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 = newColor === undefined ? "#fff" : newColor.bgcolor

                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: "Вперед"
            onClicked: pageStack.push(Qt.resolvedUrl(qsTr("Task3.qml")))
        }
    }
}
```

Task3.qml

```
import QtQuick 2.0
import Sailfish.Silica 1.0
import "func.js" as Func

Page {
    objectName: "mainPage"
    allowedOrientations: Orientation.All

    PageHeader {
        objectName: "pageHeader"
        title: qsTr("Задание 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: "Черный", bgcolor: "#000000" },
            { idx: 4, name: "Красный", bgcolor: "#ff0000" },
        ]

        SilicaListView {
            anchors.fill: parent
            model: container.rectanglesModel
            delegate: Rectangle {
                color: modelData.bgcolor
                width: parent.width
                height: 200
                Text {
                    text: modelData.name
                    anchors.centerIn: parent
                    color: Func.invertColor(modelData.bgcolor, 0)
                }
            }
            spacing: 5
        }
    }

    Row {
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.bottom: parent.bottom
```

```
        spacing: 20

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

Task4.qml

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

Page {
    objectName: "mainPage"
    allowedOrientations: Orientation.All

    PageHeader {
        objectName: "pageHeader"
        title: qsTr("Задание 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" }
```

```
            section {
                property: 'Name'
                delegate: SectionHeader { text: section }
            }
            delegate: Text { text: Value; }
        }
    }

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

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

Task5.qml

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

Page {
    objectName: "mainPage"
    allowedOrientations: Orientation.All

    PageHeader {
        objectName: "pageHeader"
        title: qsTr("Задание 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", "text/plain;charset=UTF-8");
            xhr.onreadystatechange = function() {
                if (xhr.readyState === XMLHttpRequest.DONE) {
                    xmlListModel.xml = xhr.responseText;
                    // console.log(decodeURIComponent(escape(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: "Вперед"
            onClicked: pageStack.push(Qt.resolvedUrl(qsTr("Task6.qml")))
        }
    }
}
```

Task6.qml

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

Page {
    objectName: "mainPage"
    allowedOrientations: Orientation.All

    PageHeader {
        objectName: "pageHeader"
        title: qsTr("Задание 6")
    }

    property var db: LocalStorage.openDatabaseSync("QDeclarativeExampleDB", "1.0", "Tl

    Column {
        y: 100
        width: parent.width

        TextField {
            id: txtfield
            placeholderText: "Текст заметка"
        }

        Button {
            text: "Добавить"
            onClicked: {
                db.transaction(function(tx) {
                    tx.executeSql("INSERT INTO notes (note_text) VALUES(?);", [txtfiel

                    // Show all added greetings
                    var rs = tx.executeSql('SELECT * FROM notes');

                    var r = []
                    for (var i = 0; i < rs.rows.length; i++) {
                        r.push(rs.rows.item(i))
                    }
                    console.log(r)
                    container.notesModel = r
                });

            }
        }
    }

    Item {
        id: container
        anchors {
```

```
        left: parent.left; right: parent.right;
        verticalCenter: parent.verticalCenter;
    }
    height: parent.height * 0.5

    property var notesModel: []

    SilicaListView {
        anchors.fill: parent
        model: container.notesModel
        delegate: Label {
            width: parent.width
            height: 100
            Text {
                text: modelData.note_text
                anchors.centerIn: parent
            }
        }
        spacing: 5
    }

    function findGreetings() {
        db.transaction(
                    function(tx) {
                        // Create the database if it doesn't already exist
                        tx.executeSql('CREATE TABLE IF NOT EXISTS notes(note_text

                        // Add (another) greeting row
                        tx.executeSql('INSERT INTO notes VALUES(?)', [ 'hello' ])

                        // Show all added greetings
                        var rs = tx.executeSql('SELECT * FROM notes');

                        var r = []
                        for (var i = 0; i < rs.rows.length; i++) {
                            r.push(rs.rows.item(i))
                        }
                        console.log(r)
                        container.notesModel = r
                    }
                    )
    }
    Component.onCompleted: findGreetings()
}

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

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

Task7.qml

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

Page {
    objectName: "mainPage"
    allowedOrientations: Orientation.All

    PageHeader {
        objectName: "pageHeader"
        title: qsTr("Задание 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 {
        y: 200
        TextField {
            width: 300
            text: "Text"
            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: "Вперед"
            onClicked: pageStack.push(Qt.resolvedUrl(qsTr("Task8.qml")))
        }
    }
}
```

Task8.qml

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

Page {
    objectName: "mainPage"
    allowedOrientations: Orientation.All

    PageHeader {
        objectName: "pageHeader"
        title: qsTr("Задание 8")
    }

    ConfigurationGroup {
        id: settings
        path: "/apps/app_name/settings"
        property var tf: "empty"
        property bool sw: false
```

```
    }

    Column {
        y: 200
        TextField {
            width: 300
            text: "Text"
            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()
        }
    }
}
```