

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
Направление подготовки: «Фундаментальная информатика и
информационные технологии»

Отчет по зачетному проекту
«Инструменты разработки мобильных приложений»

Выполнил:
студент группы 381906-2
Андрей Грищенко

Нижний Новгород
2022 г.

Оглавление

Постановка задачи	3
Руководство программиста.....	5
Руководство пользователя.....	7
Вывод.....	10
Приложение.....	11

Постановка задачи

Создать приложение календарь настроения, которое будет использовать базу данных для хранения значений настроения.

Приложение должно содержать меню выбора месяца, дня в месяце и настроения на выбранный день. Также в приложении должен быть отдельный qml-модуль с глобальными переменными.

Руководство программиста

Приложение хранит в базе данных записи, состоящие из даты и настроения на эту дату. В списке дней в месяце отображаются все хранящиеся записи о настроении.

Объект с глобальными переменными:

```
QObject {  
    property var date: new Date()  
    property var db: LocalStorage.openDatabaseSync("QDeclarativeExampleDB", "1.0",  
        "Mood DB", 1000000)  
}
```

Меню выбора месяца состоит из элемента ComboBox:

```
ComboBox {  
    id: comboBox  
    anchors.centerIn: parent  
    label: "Выберите месяц"  
    description: "Описание выпадающего списка"  
    menu: ContextMenu {  
        MenuItem { text: "Январь"; }  
        MenuItem { text: "Февраль"; }  
        MenuItem { text: "Март"; }  
        MenuItem { text: "Апрель"; }  
        MenuItem { text: "Май"; }  
        MenuItem { text: "Июнь"; }  
        MenuItem { text: "Июль"; }  
        MenuItem { text: "Август"; }  
        MenuItem { text: "Сентябрь"; }  
        MenuItem { text: "Октябрь"; }  
        MenuItem { text: "Ноябрь"; }  
        MenuItem { text: "Декабрь"; }  
    }  
    currentIndex: 9  
    onCurrentIndexChanged: {  
        console.log(value, currentIndex)  
        Store.date.setMonth(currentIndex)  
    }  
}
```

Меню выбора дня в месяце состоит из элемента SilicaGridView, которое рисует квадрат на каждый день месяца:

```
SilicaGridView {  
    anchors.fill: parent  
    model: days  
    header: PageHeader { title: "Выберите день" }  
  
    cellwidth: width / 4
```

```

        cellHeight: cellWidth

        id: gridView

        delegate: Item {
            width: GridView.view.cellWidth
            height: GridView.view.cellHeight

            Rectangle {
                width: 120
                height: width
                anchors.centerIn: parent
                border.color: "grey"
                border.width: 2
                radius: 10
                color: "transparent"
                Text {
                    text: name
                    x: 10
                    y: 10
                    font.pixelSize: 20
                }

                IconButton {
                    anchors.centerIn: parent
                    icon.source: mood + ".png"
                    icon.color: undefined
                    onClicked: {
                        console.log(index)
                        Store.date.setDate(index + 1)
                        pageStack.replace(Qt.resolvedUrl("MoodPage.qml"))
                    }
                    opacity: mood === "none" ? 0 : 1
                }
            }
        }
    }
}

```

Для выбора настроения используются три иконки:

```

Row {
    anchors.centerIn: parent

    IconButton {
        icon.source: "good.png"
        icon.color: undefined
        onClicked: {
            setMood("good")
            pageStack.replace(Qt.resolvedUrl("MonthPage.qml"))
        }
    }
}

```

```

    }

    IconButton {
        icon.source: "average.png"
        icon.color: undefined
        onClicked: {
            setMood("average")
            pageStack.replace(Qt.resolvedUrl("MonthPage.qml"))
        }
    }
}

IconButton {
    icon.source: "bad.png"
    icon.color: undefined
    onClicked: {
        setMood("bad")
        pageStack.replace(Qt.resolvedUrl("MonthPage.qml"))
    }
}
}

```

Первоначальный доступ к базе данных:

```

Store.db.transaction(
    function(tx) {
        // Create the database if it doesn't already exist
        tx.executeSql('DROP TABLE IF EXISTS moods');
        tx.executeSql('CREATE TABLE IF NOT EXISTS moods(date TEXT, mood
TEXT)');

        var someDate = new Date()
        someDate.setTime(0)
        someDate.setFullYear(Store.date.getFullYear())
        someDate.setMonth(Store.date.getMonth())
        someDate.setDate(Store.date.getDate())
        someDate = someDate.getTime().toString()

        // Add (another) greeting row
        tx.executeSql('INSERT INTO moods VALUES(?, ?)', [ someDate, "good" ]);

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

        var r = []
        for (var i = 0; i < rs.rows.length; i++) {
            r.push(rs.rows.item(i))
        }
        console.log(JSON.stringify(r))
        console.log(r[0].date)
        someDate = new Date()
    }
)

```

```

        console.log(someDate)
        someDate.setTime(r[0].date)
        console.log(someDate)
    }
}
)

```

Запись нового настроения в базу данных:

```

function setMood(mood) {
    console.log(mood)

    Store.db.transaction(
        function(tx) {
            // "REPLACE into table (id, name, age) values(1, "A", 19)"
            tx.executeSql('REPLACE INTO moods (date, mood) VALUES(?, ?)', [
Store.date.getTime().toString(), mood ]);
        }
    )
}

```

Руководство пользователя

После запуска программы пользователем, открывается первая страница – выбор месяца.

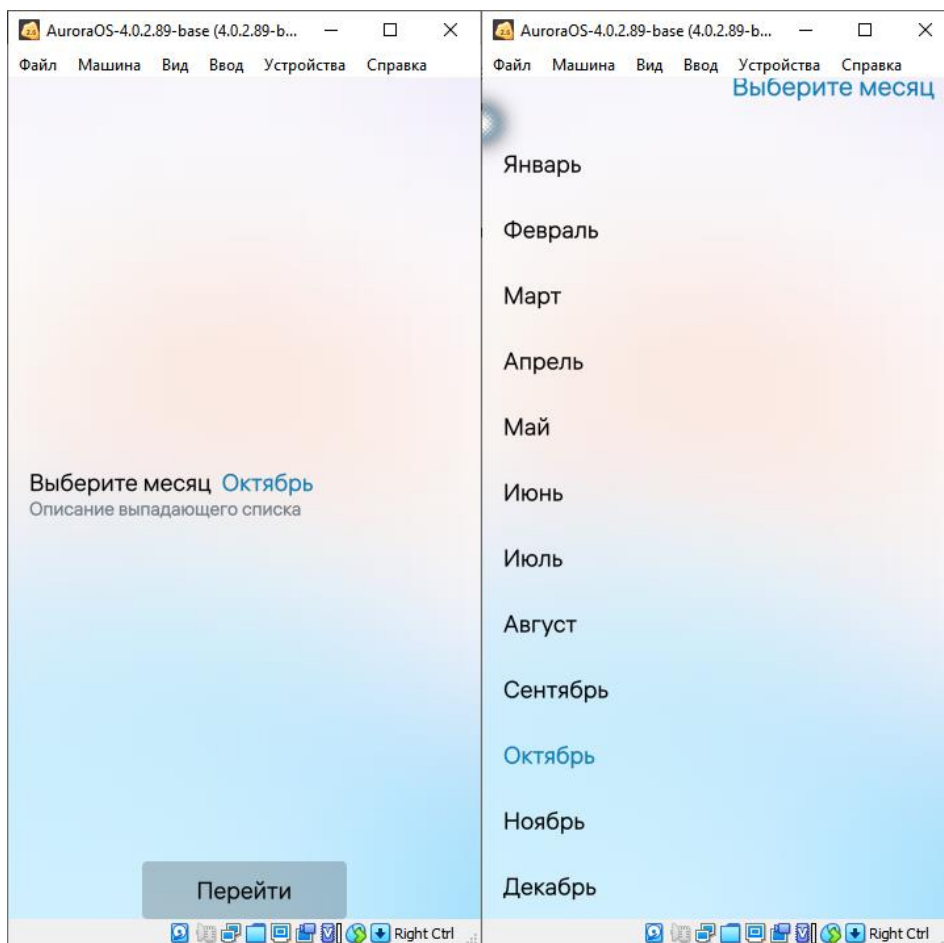


Рис 1. Выбор месяца

При выборе месяца пользователь переходит на следующую страницу выбора дня и просмотра текущих записей.

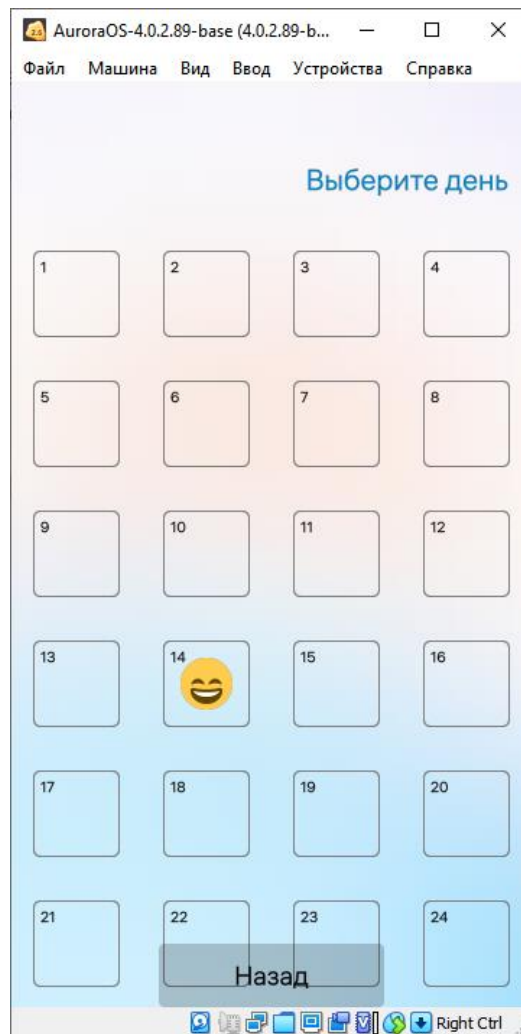


Рис 2. Список дней

При выборе дня открывается страница редактирования записи.

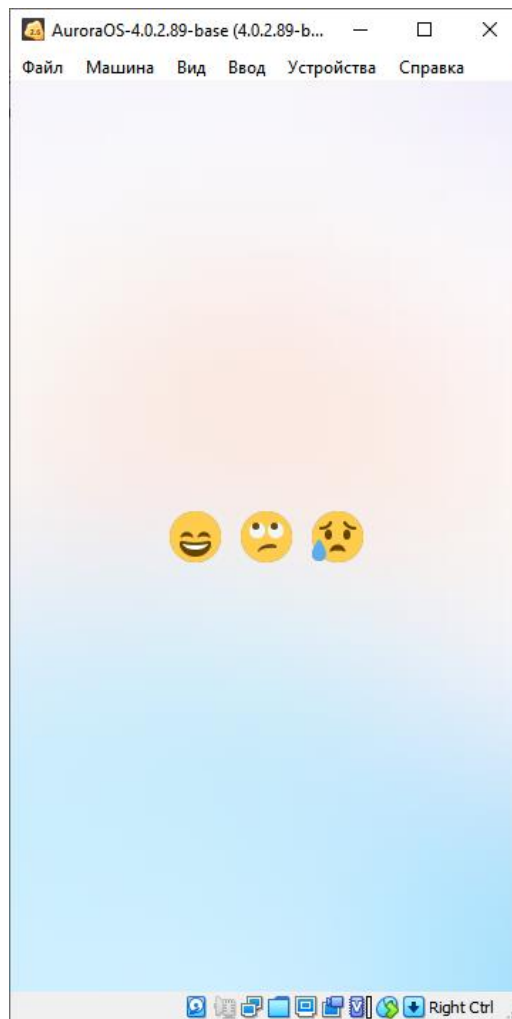


Рис 3. Выбор настроения

После выбора настроения возвращается предыдущий экран, и отображается новая запись.

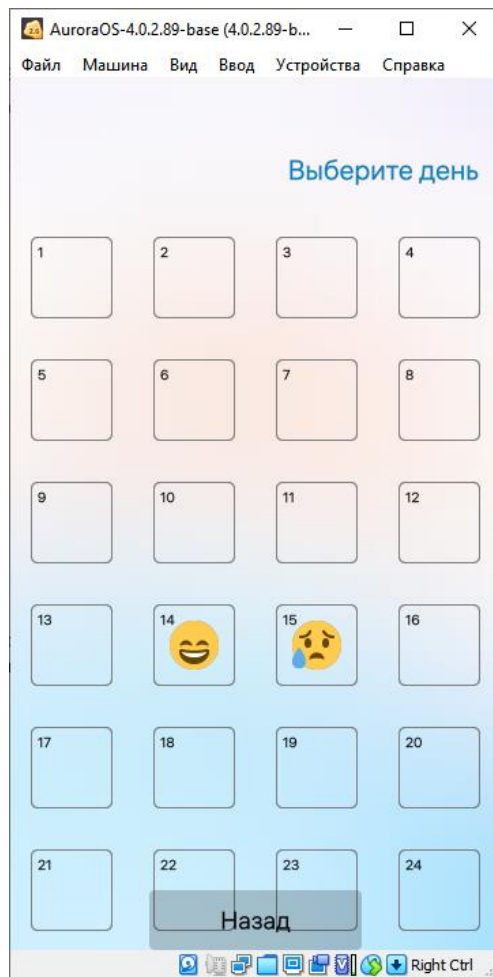


Рис 4. Обновленный список дней

Вывод

В процессе выполнения данной лабораторной работы я создал собственное приложение, использующие различные технологии, такие как база данных, и имеющее полезные функции.

Приложение

MainPage

/*****

**

**

** Copyright (C) 2022 Open Mobile Platform LLC.

** Contact: <https://community.omprussia.ru/open-source>

**

** This file is part of the Aurora OS Application Template project.

**

** Redistribution and use in source and binary forms,

** with or without modification, are permitted provided

** that the following conditions are met:

**

** * Redistributions of source code must retain the above copyright notice,

** this list of conditions and the following disclaimer.

** * Redistributions in binary form must reproduce the above copyright notice,

** this list of conditions and the following disclaimer

** in the documentation and/or other materials provided with the distribution.

** * Neither the name of the copyright holder nor the names of its contributors

** may be used to endorse or promote products derived from this software

** without specific prior written permission.

**

** THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND

CONTRIBUTORS "AS IS"

** AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
TO,

** THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS

** FOR A PARTICULAR PURPOSE ARE DISCLAIMED.

** IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE

** FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY,

** OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,

** PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;

** LOSS OF USE, DATA, OR PROFITS;

```

** OR BUSINESS INTERRUPTION)
** HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
** WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
** (INCLUDING NEGLIGENCE OR OTHERWISE)
** ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
** EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
**
*****
**/

```

```

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

```

```

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

```

```

    Item {
        width: parent.width
        anchors.centerIn: parent
        height: parent.height * 0.7

```

```

    ComboBox {
        id: comboBox
        anchors.centerIn: parent
        label: "Выберите месяц"
        description: "Описание выпадающего списка"
        menu: ContextMenu {
            MenuItem { text: "Январь"; }
            MenuItem { text: "Февраль"; }
            MenuItem { text: "Март"; }
            MenuItem { text: "Апрель"; }
            MenuItem { text: "Май"; }
            MenuItem { text: "Июнь"; }

```

```

MenuItem { text: "Июль"; }
MenuItem { text: "Август"; }
MenuItem { text: "Сентябрь"; }
MenuItem { text: "Октябрь"; }
MenuItem { text: "Ноябрь"; }
MenuItem { text: "Декабрь"; }
}
currentIndex: 9
onCurrentIndexChanged: {
    console.log(value, currentIndex)
    Store.date.setMonth(currentIndex)
}
}
}

```

```

Button {
    text: "Перейти"
    onClicked: pageStack.replace(Qt.resolvedUrl("MonthPage.qml"))
    anchors.bottom: parent.bottom
    anchors.horizontalCenter: parent.horizontalCenter
}

```

```

Component.onCompleted: {
    Store.date.setTime(0)
    Store.date.setFullYear(new Date().getFullYear())
    Store.date.setMonth(new Date().getMonth())
    Store.date.setDate(new Date().getDate())

    comboBox.currentIndex = Store.date.getMonth()
}

```

```

Store.db.transaction(
    function(tx) {
        // Create the database if it doesn't already exist
        tx.executeSql('DROP TABLE IF EXISTS moods');
        tx.executeSql('CREATE TABLE IF NOT EXISTS moods(date TEXT, mood TEXT)');
    }
)

```

```

    var someDate = new Date()
    someDate.setTime(0)
    someDate.setFullYear(Store.date.getFullYear())
    someDate.setMonth(Store.date.getMonth())
    someDate.setDate(Store.date.getDate())
    someDate = someDate.getTime().toString()

    // Add (another) greeting row
    tx.executeSql('INSERT INTO moods VALUES(?, ?)', [ someDate, "good" ]);

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

    var r = []
    for (var i = 0; i < rs.rows.length; i++) {
        r.push(rs.rows.item(i))
    }
    console.log(JSON.stringify(r))
    console.log(r[0].date)
    someDate = new Date()
    console.log(someDate)
    someDate.setTime(r[0].date)
    console.log(someDate)
    }
    )
    }
}

MonthPage
import QtQuick 2.0
import Sailfish.Silica 1.0
import "."
import QtQuick.LocalStorage 2.0

Page {

```



```
ListModel {  
    id: days  
}
```

```
Item {  
    width: parent.width  
    height: parent.height * 0.9  
    anchors.centerIn: parent  
    SilicaGridView {  
        anchors.fill: parent  
        model: days  
        header: PageHeader { title: "Выберите день" }  
  
        cellWidth: width / 4  
        cellHeight: cellWidth  
  
        id: gridView  
  
        delegate: Item {  
            width: GridView.view.cellWidth  
            height: GridView.view.cellHeight  
  
            Rectangle {  
                width: 120  
                height: width  
                anchors.centerIn: parent  
                border.color: "grey"  
                border.width: 2  
                radius: 10  
                color: "transparent"  
                Text {  
                    text: name  
                    x: 10  
                    y: 10
```



```

Store.db.transaction(
  function(tx) {
    // Show all added greetings
    var rs = tx.executeSql('SELECT * FROM moods');

    var r = []
    for (var i = 0; i < rs.rows.length; i++) {
      r.push(rs.rows.item(i))
    }
    data = r.filter(function(x) {
      var oldTime = x.date
      x.date = new Date()
      x.date.setTime(oldTime)

      return x
    })
  }
)

console.log(JSON.stringify(data))

var daysNum = daysInMonth(Store.date.getMonth() + 1, Store.date.getFullYear())

for (var i = 1; i <= daysNum; i++) {
  var oldDate = Store.date.getTime().toString()
  var someDate = new Date()
  someDate.setTime(oldDate)

  // console.log(someDate)
  someDate.setDate(i)
  // console.log(someDate, i - 1)

  var newRecord = { name: i, mood: "none" }
  data.forEach(function(x) {

```

```

        if (x.date.getMonth() === someDate.getMonth() && x.date.getDate() ===
someDate.getDate()) {
            newRecord.mood = x.mood
        }
    })
    days.append(newRecord)
}
}
}

```

MoodPage

```

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

```

Page {

Row {

anchors.centerIn: parent

IconButton {

icon.source: "good.png"

icon.color: undefined

onClicked: {

setMood("good")

pageStack.replace(Qt.resolvedUrl("MonthPage.qml"))

}

}

IconButton {

icon.source: "average.png"

icon.color: undefined

onClicked: {

setMood("average")

```

        pageStack.replace(Qt.resolvedUrl("MonthPage.qml"))
    }
}

IconButton {
    icon.source: "bad.png"
    icon.color: undefined
    onClicked: {
        setMood("bad")
        pageStack.replace(Qt.resolvedUrl("MonthPage.qml"))
    }
}

function setMood(mood) {
    console.log(mood)

    Store.db.transaction(
        function(tx) {
            // "REPLACE into table (id, name, age) values(1, "A", 19)"
            tx.executeSql('REPLACE INTO moods (date, mood) VALUES(?, ?)', [
Store.date.getTime().toString(), mood ]);
        }
    )
}

}

Store
pragma Singleton
import QtQuick 2.0
import QtQuick.LocalStorage 2.0

QObject {
    property var date: new Date()

```

```
property var db: LocalStorage.openDatabaseSync("QDeclarativeExampleDB", "1.0", "Mood
DB", 1000000)
}
```