**Отчёт по проекту «Калькулятор калорий»**

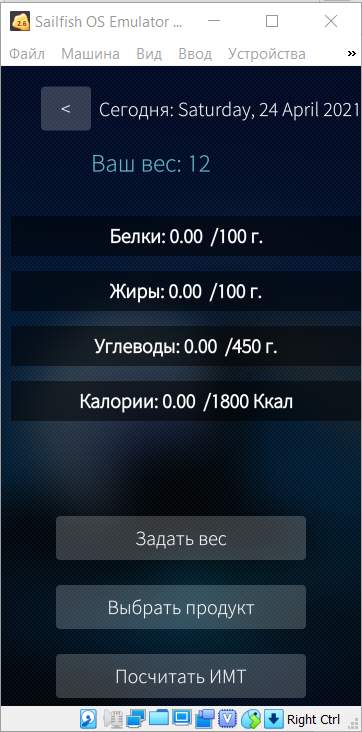
Подготовила:

Студент группы 381808-1

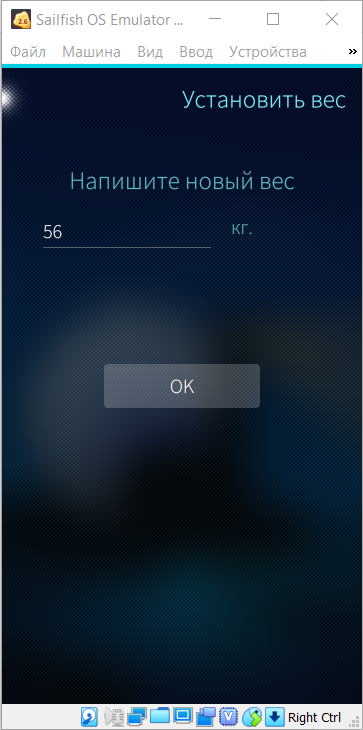
Кустова Анастасия

Данная программа предназначена для того, чтобы человек мог контролировать свой вес: смотреть пищевую ценность для различных продуктов, подсчитывать, сколько белком, жиров, углеводов и калорий он потребил. Информация об этом заносится в базу данных, так что пользователь может посмотреть всю статистику.

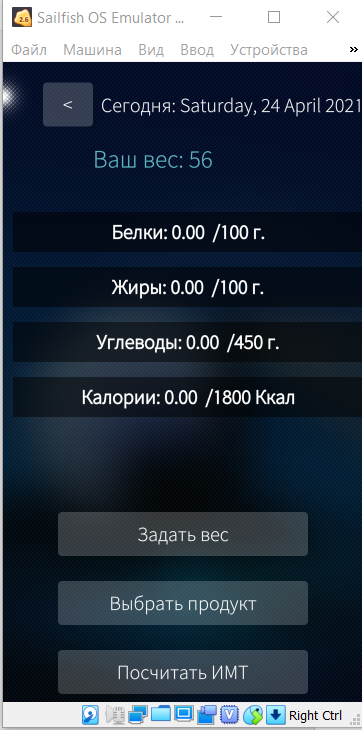
При запуске программы открывается страница с главным меню. На ней отображается сегодняшняя дата, вес, а также количество потреблённых пользователем белков, жиров, углеводов и калорий.



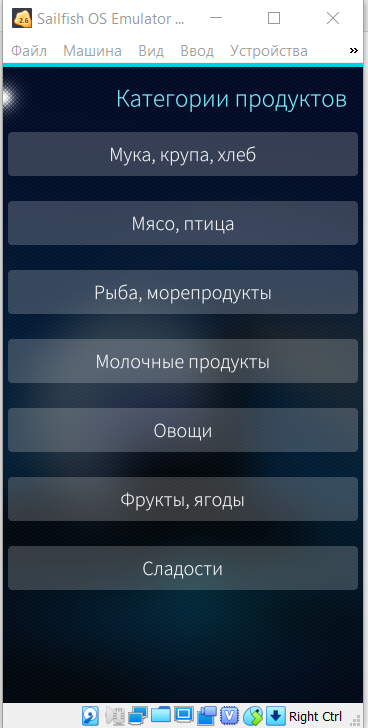
При нажатии на кнопку «Задать вес» открывается страница, на которой можно поменять свой текущий вес.



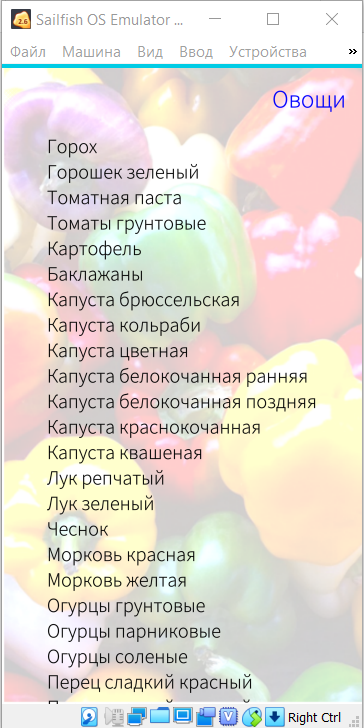
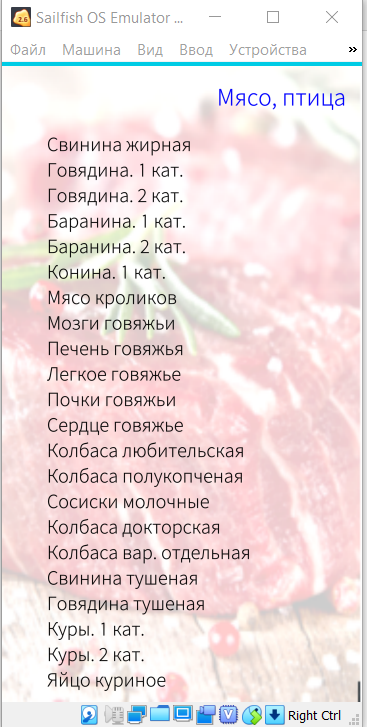
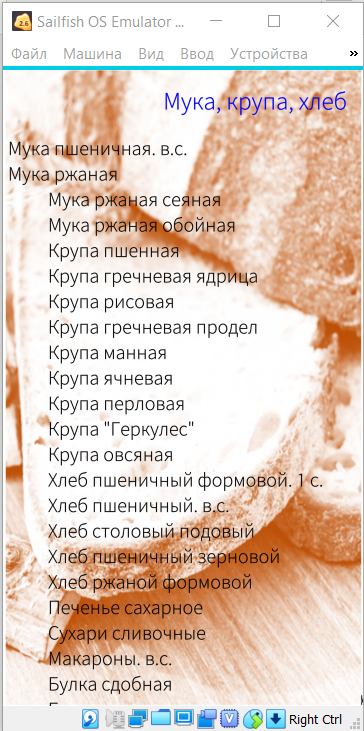
После того, как вы это сделаете, на главной странице вес обновится:

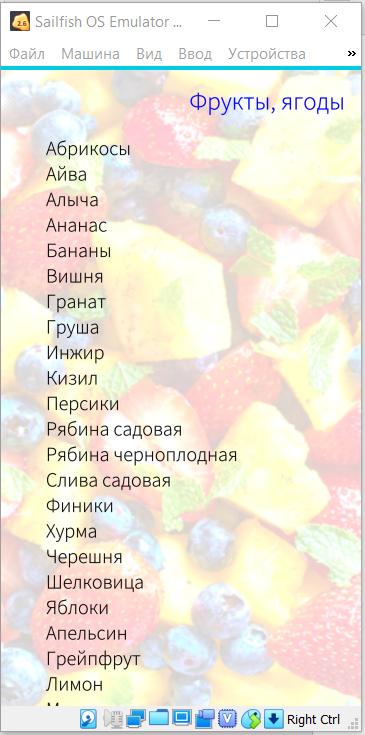
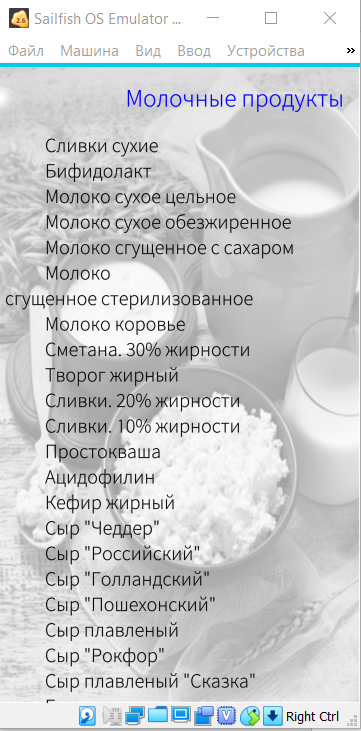
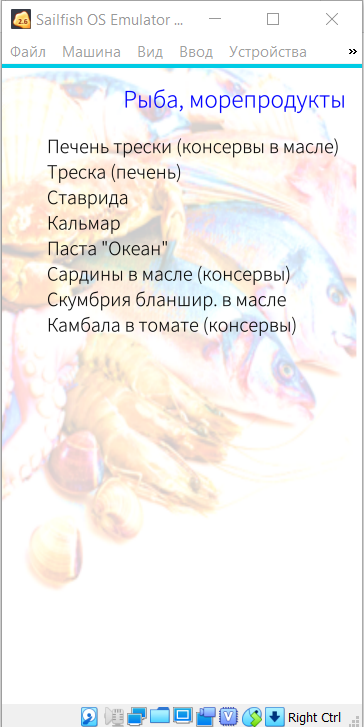


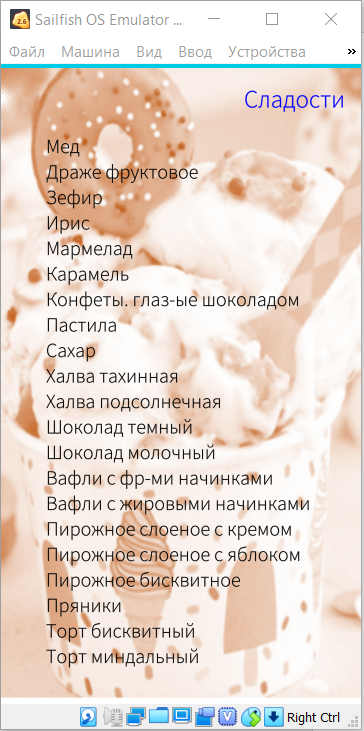
При нажатии на кнопку «Выбрать продукт», открывается страница где написаны различные категории продуктов:



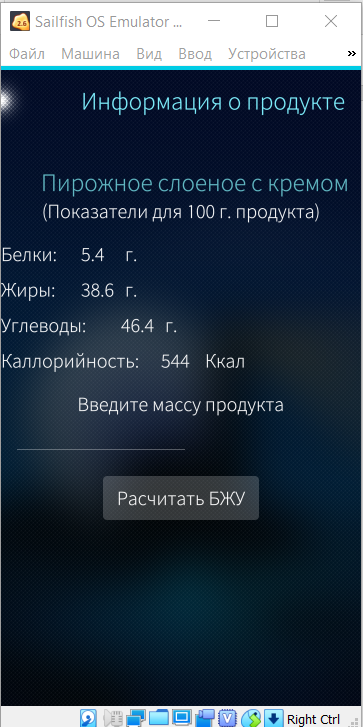
Далее нужно выбрать какую-нибудь категорию, и пользователю покажется список продуктов, попадающих под эту категорию. Эта база категорий, продуктов и их пищевых показателей хранятся в xml файле.



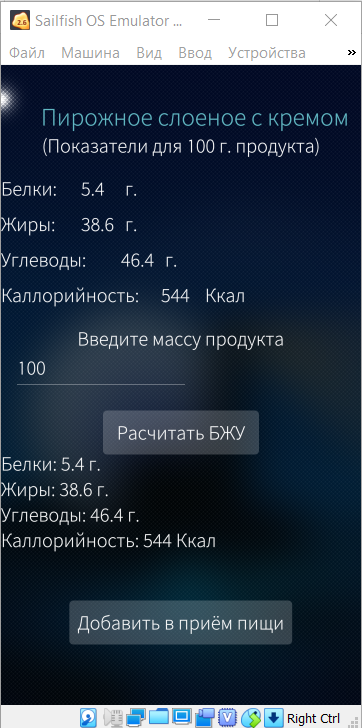




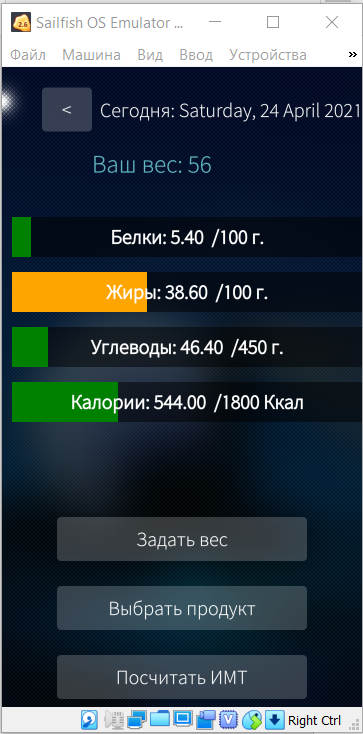
Если нажать на какой-нибудь продукт, то можно увидеть пищевую ценность 100 гр. этого продукта



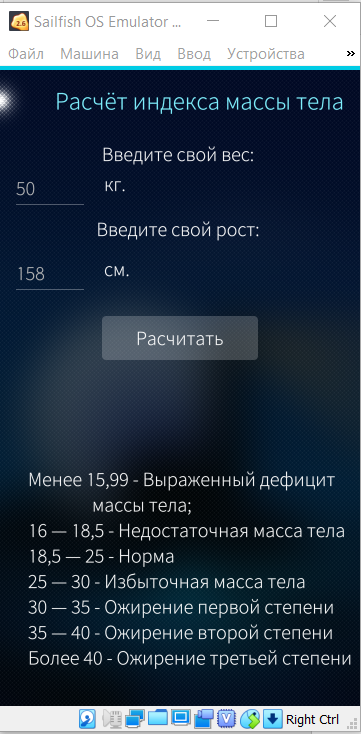
Когда пользователь введёт массу, которую он съел, то программа подсчитает, сколько белков, жиров, углеводов и калорий он потребил.



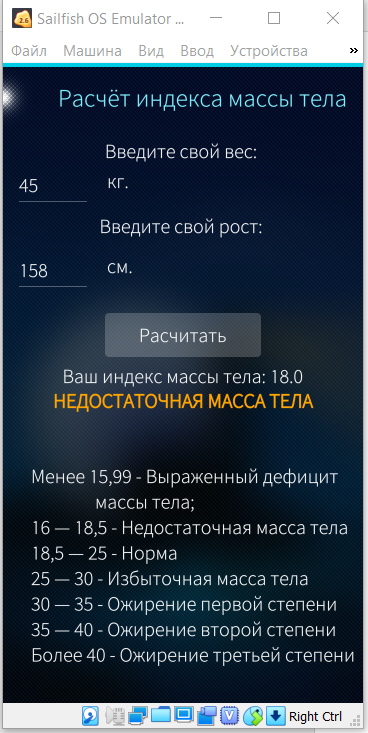
Если нажать на кнопку «Добавить в приём пищи, то можно увидеть, как на главной странице обновятся показатели



Так же на главном экране есть кнопка «Посчитать ИМТ». Если на неё нажать, то пользователь перейдёт на страницу, где расчитывается индекс массы тела.

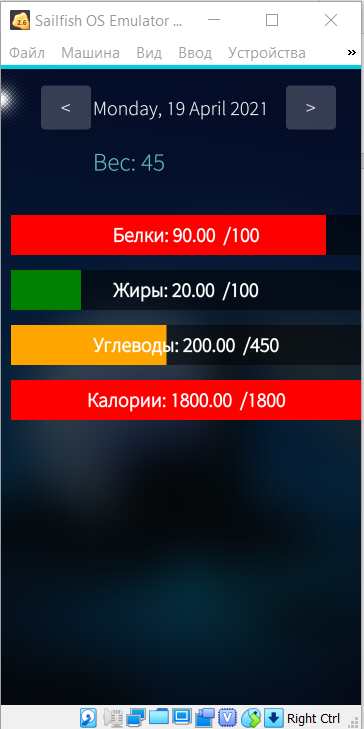


Если ввести свой вес и рост, то можно увидеть свой индекс массы тела с его расшифровкой:



Так же можно просмотреть историю веса и потреблённых единиц по дням, если нажимать на стрелочки, расположенные рядом с датой. Эта информация хранится в базе данных.





**Приложение:**

Ссылка на github: <https://github.com/KustovaAI/Mobile_System-My_project>

Исходный код:

my\_project.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

import Nemo.Configuration 1.0

import "pages"

ApplicationWindow

{

initialPage: Component { MainPage { } }

cover: *Qt*.resolvedUrl("cover/CoverPage.qml")

allowedOrientations: defaultAllowedOrientations

Db {id: *db1*}

ListModel { id: *currentCatModel* }

ListModel { id: *currentItemModel* }

property date todayDate: new *Date*()

property string dateString: *todayDate*.toLocaleDateString()

property variant mDates: []

property string currentDate: *dateString*

property string sourceCatImg: ""

property int delta: 0

property real normProteins: 100

property real normFats: 100

property real normCarbohydrates: 450

property real normCalorie: 1800

property real currentProteins: *setting1*.value[2]

property real currentFats: *setting1*.value[3]

property real currentCarbohydrates: *setting1*.value[4]

property real currentCalorie: *setting1*.value[5]

ConfigurationValue {

id: *setting1*

key: "/apps/app\_name/setting\_name"

defaultValue: ["Monday, 19 April 2021", 48, 0, 0, 0, 0]

}

function *selectAllNotes*() {

*db1*.retrieveNotes(function(notes) {

for (var *i* = 0; *i* < *notes*.length; *i*++) {

var *note* = *notes*.item(*i*);

// console.log(note.id, note.day,note.mass, note.proteins, note.fats, note.calorie);

}

});

}

function *selectAllDates*() {

*mDates* = []

*db1*.selectDates(function(notes) {

for (var *i* = 0; *i* < *notes*.length; *i*++) {

var *note* = *notes*.item(*i*);

// console.log(note.day);

*mDates*.push(*note*.day)

}

});

}

function *getMass*() {

var *my\_mass*;

*db1*.getCurrentValues(function(m) {

for (var *i* = 0; *i* < *m*.length; *i*++) {

*my\_mass* = *m*.item(0).mass;

}

});

return *my\_mass*;

}

function *getCurValues*() {

var *my\_mass*;

*db1*.getCurrentValues(function(m) {

for (var *i* = 0; *i* < *m*.length; *i*++) {

*currentProteins* = *m*.item(0).proteins;

*currentFats* = *m*.item(0).fats;

*currentCarbohydrates* = *m*.item(0).carbohydrates;

*currentCalorie* = *m*.item(0).calorie

}

});

return *my\_mass*;

}

function *getColor*(p) {

if (*p* < 0.33) {

return "green"

} else {

if (*p* < 0.66) {

return "orange"

} else {

return "red"

}

}

}

Timer {

id:*timer2*

interval: 1; running: true; repeat: false

onTriggered: {

if (*setting1*.value[0] !== *dateString*) {

*db1*.insertNote(*dateString*, *setting1*.value[1], 0, 0,0, 0)

*selectAllNotes*()

*console*.log("New Day")

}

}

}

}

MainPage.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page*

allowedOrientations: Orientation.All

SilicaFlickable {

anchors.fill: *parent*

Button {

id: *back*

x: 80

y: 80

width: 100

height: 10

text: " < "

onClicked: {

*selectAllDates*()

if (*mDates*.length >= 1) {

*pageStack*.push(*Qt*.resolvedUrl("HistoryPage.qml"))

*delta* = 1

*currentDate* = *mDates*[*mDates*.length - 1 - *delta*]

*getCurValues*()

}

}

}

contentHeight: *column*.height

Column {

id: *column*

anchors.left: *back*.right

y: 60

spacing: 50

Label {

text:" Сегодня: " + *dateString*

}

Label {

color: Theme.secondaryHighlightColor

font.pixelSize: Theme.fontSizeLarge

text:"Ваш вес: " + *setting1*.value[1]

}

}

Column {

x: 20

y: 300

spacing: 30

Item {

id: *container1*

width: 700

height: 80

property string curColor: "green"

property real progress

progress: (*setting1*.value[2] < *normProteins*)? (*setting1*.value[2] / *normProteins*) : 1

Rectangle { anchors.fill: *parent*; color: "black"; opacity: 0.5 }

Rectangle {

id: *fill1*; color: *getColor*(*container1*.progress); height: *container1*.height

width: *container1*.width \* *container1*.progress

}

Text {

font.pixelSize: 40

font.bold: true

anchors.centerIn: *parent*

text:"Белки: " + (*setting1*.value[2]).toFixed(2) + " /" + *normProteins* + " г."

color: "white"

}

}

Item {

id: *container2*

width: 700

height: 80

property real progress

progress: (*setting1*.value[3] < *normFats*)? (*setting1*.value[3] / *normFats*) : 1

Behavior on opacity { NumberAnimation { duration: 600 } }

Rectangle { anchors.fill: *parent*; color: "black"; opacity: 0.5 }

Rectangle {

id: *fill2*; color: *getColor*(*container2*.progress); height: *container2*.height

width: *container2*.width \* *container2*.progress

}

Text {

font.pixelSize: 40

font.bold: true

anchors.centerIn: *parent*

text:"Жиры: " + (*setting1*.value[3]).toFixed(2) + " /" + *normFats* + " г."

color: "white"

}

}

Item {

id: *container3*

width: 700

height: 80

property real progress

progress: (*setting1*.value[4] < *normCarbohydrates*)? (*setting1*.value[4] / *normCarbohydrates*) : 1

Behavior on opacity { NumberAnimation { duration: 600 } }

Rectangle { anchors.fill: *parent*; color: "black"; opacity: 0.5 }

Rectangle {

id: *fill3*; color: *getColor*(*container3*.progress); height: *container3*.height

width: *container3*.width \* *container3*.progress

}

Text {

font.pixelSize: 40

font.bold: true

anchors.centerIn: *parent*

text:"Углеводы: " + (*setting1*.value[4]).toFixed(2) + " /" + *normCarbohydrates* + " г."

color: "white"

}

}

Item {

id: *container4*

width: 700

height: 80

property real progress

progress: (*setting1*.value[5] < *normCalorie*)? (*setting1*.value[5] / *normCalorie*) : 1

Behavior on opacity { NumberAnimation { duration: 600 } }

Rectangle { anchors.fill: *parent*; color: "black"; opacity: 0.5 }

Rectangle {

id: *fill4*; color: *getColor*(*container4*.progress); height: *container4*.height

width: *container4*.width \* *container4*.progress

}

Text {

font.pixelSize: 40

font.bold: true

anchors.centerIn: *parent*

text:"Калории: " + (*setting1*.value[5]).toFixed(2) + " /" + *normCalorie* + " Ккал"

color: "white"

}

}

}

Column {

y: 900

anchors.horizontalCenter: *parent*.horizontalCenter

spacing: 50

Button {

text:"Задать вес"

width: 500

onClicked: {

*pageStack*.push(*Qt*.resolvedUrl("SetMassPage.qml"))

}

}

Button {

width: 500

text:"Выбрать продукт"

onClicked: *pageStack*.push(*Qt*.resolvedUrl("CategoriesPage.qml"))

}

Button {

width: 500

text:"Посчитать ИМТ"

onClicked: *pageStack*.push(*Qt*.resolvedUrl("BodyMassIndexPage.qml"))

}

}

}

}

SetMassPage.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page*

allowedOrientations: Orientation.All

SilicaFlickable {

anchors.fill: *parent*

PullDownMenu {

MenuItem {

text: *qsTr*("Show MainPage")

onClicked: pageStack.push(*Qt*.resolvedUrl("MainPage.qml"))

}

}

PageHeader {

title: *qsTr*("Установить вес")

}

Label {

id: *label1*

anchors.horizontalCenter: *parent*.horizontalCenter

y: 200

text: *qsTr*("Напишите новый вес")

color: Theme.secondaryHighlightColor

font.pixelSize: Theme.fontSizeLarge

}

TextField {

x: 50

y: *label1*.y + 100

id: *textField1*

width: 400

}

Label {

id: *label2*

anchors.left: *textField1*.right

y: *textField1*.y

text: *qsTr*(" кг.")

color: Theme.secondaryHighlightColor

}

Button {

anchors.horizontalCenter: *parent*.horizontalCenter

y: 600

text: "OK"

onClicked: {

pageStack.push(*Qt*.resolvedUrl("MainPage.qml"))

db1.setMass(dateString, parseFloat(*textField1*.text))

selectAllNotes()

}

}

}

}

BodyMassIndexPage.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page*

property real imt: 0

allowedOrientations: Orientation.All

SilicaFlickable {

anchors.fill: *parent*

PullDownMenu {

MenuItem {

text: *qsTr*("Show MainPage")

onClicked: pageStack.push(*Qt*.resolvedUrl("MainPage.qml"))

}

}

PageHeader { title: "Расчёт индекса массы тела" }

Label {

id: *label\_1*

y: 150

anchors.horizontalCenter: *parent*.horizontalCenter

text: "Введите свой вес: "

}

TextField {

id: *textField1*

y: *label\_1*.y + 60

placeholderText : "50"

width: 200

}

Label {

id: *label\_1\_2*

y: *textField1*.y

anchors.left: *textField1*.right

text: " кг."

}

Label {

id: *label\_2*

y: 300

anchors.horizontalCenter: *parent*.horizontalCenter

text: "Введите свой рост: "

}

TextField {

id: *textField2*

y: *label\_2*.y + 80

placeholderText : "158"

width: 200

}

Label {

id: *label\_2\_2*

y: *textField2*.y

anchors.left: *textField2*.right

text: " см."

}

Button {

anchors.horizontalCenter: *parent*.horizontalCenter

y: 500

text: "Расчитать"

onClicked: {

*imt* = parseFloat(*textField1*.text) / ((parseFloat(*textField2*.text) \* parseFloat(*textField2*.text) / 10000))

*label\_3*.text = "Ваш индекс массы тела: " + (*imt*).toFixed(1)

if (*imt* < 15.99) {

*label\_4*.text = "ВЫРАЖЕННЫЙ ДЕФИЦИТ МАССЫ ТЕЛА"

*label\_4*.color = "red"

} else if (*imt* < 18.49) {

*label\_4*.text = "НЕДОСТАТОЧНАЯ МАССА ТЕЛА"

*label\_4*.color = "orange"

} else if (*imt* < 24.99) {

*label\_4*.text = "НОРМА"

*label\_4*.color = "green"

} else if (*imt* < 29.99) {

*label\_4*.text = "ИЗБЫТОЧНАЯ МАССА ТЕЛА"

*label\_4*.color = "orange"

} else if (*imt* < 34.99) {

*label\_4*.text = "ОЖИРЕНИЕ 1 СТЕПЕНИ"

*label\_4*.color = "red"

} else if (*imt* < 39.99) {

*label\_4*.text = "ОЖИРЕНИЕ 2 СТЕПЕНИ"

*label\_4*.color = "red"

} else {

*label\_4*.text = "ОЖИРЕНИЕ 3 СТЕПЕНИ"

*label\_4*.color = "red"

}

}

}

Label {

id: *label\_3*

anchors.horizontalCenter: *parent*.horizontalCenter

y: 600

text: ""

}

Label {

id: *label\_4*

anchors.horizontalCenter: *parent*.horizontalCenter

y: 650

font.bold: true

text: ""

}

Label {

id: *label\_5*

anchors.horizontalCenter: *parent*.horizontalCenter

y: 800

font.pixelSize: 40

text: " Менее 15,99 - Выраженный дефицит

массы тела;

16 — 18,5 - Недостаточная масса тела

18,5 — 25 - Норма

25 — 30 - Избыточная масса тела

30 — 35 - Ожирение первой степени

35 — 40 - Ожирение второй степени

Более 40 - Ожирение третьей степени"

}

}

}

CategoriesPage.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

import QtQuick.XmlListModel 2.0

Page {

id: *page*

allowedOrientations: Orientation.All

SilicaFlickable {

anchors.fill: *parent*

PullDownMenu {

MenuItem {

text: *qsTr*("Show MainPage")

onClicked: pageStack.push(*Qt*.resolvedUrl("MainPage.qml"))

}

}

XmlListModel {

id: *modelCat*

query: "/Table/Category"

source: "../pages/table.xml"

XmlRole { name: "nameCat"; query: "nameCat/string()"; }

XmlRole { name: "imgCat"; query: "imgCat/string()"; }

}

XmlListModel {

id: *modelItems*

query: "/Table/Category/Item"

source: "../pages/table.xml"

XmlRole { name: "name"; query: "name/string()"; }

XmlRole { name: "proteins"; query: "proteins/string()"; }

XmlRole { name: "fats"; query: "fats/string()"; }

XmlRole { name: "carbohydrates"; query: "carbohydrates/string()"; }

XmlRole { name: "calorie"; query: "calorie/string()"; }

XmlRole { name: "cat"; query: "cat/string()"; }

}

SilicaListView {

anchors.fill: *parent*

header: PageHeader { title: "Категории продуктов" }

model: *modelCat*

spacing: 50

delegate: Column {

x: 10; width: *parent*.width - 2 \* *x*

Button {

width: *parent*.width

text: nameCat

onClicked: {

sourceCatImg = imgCat

model: *modelItems*

currentCatModel.clear()

for (var *i* = 0; *i* < *modelItems*.count; *i*++ ) {

if (*modelItems*.get(*i*).cat === nameCat) {

currentCatModel.append({

name: *modelItems*.get(*i*).name,

proteins: *modelItems*.get(*i*).proteins,

fats: *modelItems*.get(*i*).fats,

carbohydrates: *modelItems*.get(*i*).carbohydrates,

calorie: *modelItems*.get(*i*).calorie,

cat: *modelItems*.get(*i*).cat});

}

}

pageStack.push(*Qt*.resolvedUrl("CurentCategoryPage.qml"))

}

}

}

}

}

}

CurentCategoryPage.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

import QtQuick.XmlListModel 2.0

Page {

id: *page*

allowedOrientations: Orientation.All

Image {

source: sourceCatImg

width: *parent*.width

height: *parent*.height

}

SilicaFlickable {

anchors.fill: *parent*

PullDownMenu {

MenuItem {

text: *qsTr*("Show MainPage")

onClicked: pageStack.push(*Qt*.resolvedUrl("MainPage.qml"))

}

}

SilicaListView {

id: *listView*

anchors.fill: *parent*

VerticalScrollDecorator { flickable: *listView*; color: "black"; width: 15 }

header: PageHeader { title: currentCatModel.get(0).cat; titleColor: "blue" }

model: currentCatModel

delegate: Column {

x: 10; width: *parent*.width - 2 \* *x*

Label {

width: *parent*.width

color: "black"

wrapMode: Text.WordWrap

text:name

MouseArea {

anchors.fill: *parent*

onClicked: {

currentItemModel.clear()

currentItemModel.append({

name: name,

proteins: proteins,

fats: fats,

carbohydrates: carbohydrates,

calorie: calorie});

pageStack.push(*Qt*.resolvedUrl("ItemPage.qml"))

}

}

}

}

}

}

}

ItemPage.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

import QtQuick.XmlListModel 2.0

import QtQuick.LocalStorage 2.0

Page {

id: *page*

property real \_prot: 0

property real \_fats: 0

property real \_carb: 0

property real \_cal: 0

allowedOrientations: Orientation.All

SilicaFlickable {

anchors.fill: *parent*

PullDownMenu {

MenuItem {

text: *qsTr*("Show MainPage")

onClicked: pageStack.push(*Qt*.resolvedUrl("MainPage.qml"))

}

}

PageHeader { title: "Информация о продукте" }

Label {

id: *label\_name*

y: 200

color: Theme.secondaryHighlightColor

font.pixelSize: Theme.fontSizeLarge

anchors.horizontalCenter: *parent*.horizontalCenter

text: currentItemModel.get(0).name

}

Label {

anchors.top : *label\_name*.bottom

anchors.horizontalCenter: *parent*.horizontalCenter

text: "(Показатели для 100 г. продукта)"

}

Column {

id: *col1*

y: 350

spacing: 20

Label {

id: *label\_prot1*

text: "Белки: " + currentItemModel.get(0).proteins + " г."

}

Label {

id: *label\_fats1*

text: "Жиры: " + currentItemModel.get(0).fats + " г."

}

Label {

id: *label\_carb1*

text: "Углеводы: " + currentItemModel.get(0).carbohydrates + " г."

}

Label {

id: *label\_calorie1*

text: "Каллорийность: " + currentItemModel.get(0).calorie + " Ккал"

}

}

Label {

id: *l1*

y: 650

anchors.horizontalCenter: *parent*.horizontalCenter

text: "Введите массу продукта"

}

TextField {

id: *textField1*

y: *l1*.y + 50

width: 400

}

Button {

y: *textField1*.y + 120

anchors.horizontalCenter: *parent*.horizontalCenter

text: "Расчитать БЖУ"

onClicked: {

*b2*.visible = true

*\_prot* = (parseFloat(currentItemModel.get(0).proteins) / 100 \* parseInt(*textField1*.text)).toFixed(2)

*\_fats* = (parseFloat(currentItemModel.get(0).fats) / 100 \* parseInt(*textField1*.text)).toFixed(2)

*\_carb* = (parseFloat(currentItemModel.get(0).carbohydrates) / 100 \* parseInt(*textField1*.text)).toFixed(2)

*\_cal* = (parseFloat(currentItemModel.get(0).calorie) / 100 \* parseInt(*textField1*.text)).toFixed(2)

*label\_prot2*.text = "Белки: " + *\_prot* + " г."

*label\_fats2*.text = "Жиры: " + *\_fats* + " г."

*label\_carb2*.text = "Углеводы: " + *\_carb* + " г."

*label\_calorie2*.text = "Каллорийность: " + *\_cal* + " Ккал"

}

}

Column {

y: 900

Label {

id: *label\_prot2*

text: ""

}

Label {

id: *label\_fats2*

text: ""

}

Label {

id: *label\_carb2*

text: ""

}

Label {

id: *label\_calorie2*

text: ""

}

}

Button {

id: *b2*

y: 1200

anchors.horizontalCenter: *parent*.horizontalCenter

text: "Добавить в приём пищи"

visible: false

onClicked: {

db1.insertNote(dateString, setting1.value[1], *\_prot*, *\_fats*, *\_carb*, *\_cal*)

pageStack.push(*Qt*.resolvedUrl("CategoriesPage.qml"))

selectAllNotes()

}

}

}

}

Db.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

import QtQuick.LocalStorage 2.0

Item {

property var db;

Component.onCompleted: {

*db* = LocalStorage.openDatabaseSync("diary", "1.0");

*createNotesTable*();

}

function *createNotesTable*() {

*db*.transaction(function(tx) {

*tx*.executeSql("CREATE TABLE IF NOT EXISTS diary (

id INTEGER PRIMARY KEY AUTOINCREMENT,"

+ "day TEXT NOT NULL, mass INT, proteins INT, fats INT, carbohydrates INT, calorie INT);");

});

}

func*tion insertNot*e(\_date, \_mass, \_proteins, \_fats, \_carbohydrates, \_calorie) {

retrieveNotes*(*functi*o*n(n*otes)* {

for (var *i =* 0; *i <* notes.*l*ength; i++) {

var *n*ote = *note*s.item(i);

if (i ==*= note*s.l*ength* -1) {

var *l\_date* = *notes*.item(*i*).day;

var *l\_mass = n*ote*s.ite*m(i).m*a*ss;

var *l\_prot*ein*s = n*otes.i*t*em(i).proteins;

var *l\_fats = notes.*ite*m(i).*fats;

var *l\_carbohy*dra*tes =* notes*.*item(i).carbohydrates;

var *l\_calo*rie = *note*s.item(i).calorie;

if (*l\_date ===* *\_date*) *{*

*updateNote(*\_da*te,* *\_mass,* *l\_proteins + \_proteins, l\_fat*s *+ \_fats,* *l\_carbohydrates + \_car*boh*ydrates, l\_cal*or*ie + \_cal*ori*e)*

setting1.value = [\_date, \_mass, l\_proteins + \_proteins, l\_fats + \_fats, l\_carbohydrates + \_carbohydrates, l\_calorie + \_calorie]

} el*se* {

db.t*ra*nsaction(function(tx) {

t*x.exe*cu*teSql(*"INSERT INTO diary (day, mass, proteins, fats, carbohydrates, calorie)

*VAL*UES(?, ?, *?, ?,* ?*, ?);*", *[\_date,* \_m*ass,* \_p*roteins, \_fats*, *\_carbohy*drates, \_calorie]);

});

setting1.value = [\_date, \_mass, \_proteins, \_fats, \_carbohydrates, \_calorie]

}

}

}

});

}

function *setMa*ss(*\_*date, *\_*mas*s) {*

getDa*t*e(func*tion(l*ast*\_date) {*

for (var *i = 0*; i *< last\_d*ate.le*n*gth; i++) {

var *l\_date* = la*st\_da*te.item(i).day;

var *l\_*mass = last\_date.item(i).mass;

if (l\_da*te* === \_date) {

db.transaction(function*(tx)* {

*tx.e*xecuteSql(*"UPDA*TE *diar*y *SET mass* = ? WHERE *day = ?;*", [\_mass, *\_date]);*

});

*se*tting1.value = [\_date, \_mass, setting1.value[2], setting1.value[3], setting1.value[4], setting1.value[5]]

} else {

db.transa*ction*(function(tx) {

*tx.e*xecuteSql(*"INSE*RT *INTO* diary (day, mass, proteins, fats, carbohydrates, calorie)

VALUES(?, ?, ?, ?, ?, ?);", [\_date, \_mass, 0, 0, 0, 0]);

});

setting1.value = [\_date, \_mass, 0, 0, 0, 0]

}

}

*})*;

}

function updateNote(\_*date,* \_*mass,* *\_pr*ot*eins,* \_*fats,* *\_carbohy*dr*ates,* *\_c*al*orie)* {

db.transaction(function(tx) {

t*x.executeS*ql("UPDATE diary SET mass = ?, proteins = ?, fats = ?, carbohydrates = ?,

calorie *=* ? WHERE day = ?;", [\_mass, \_proteins, \_fats, \_carbohydrates, \_calorie, \_date]);

});

}

function deleteNote(id) {

db.tr*ansaction(fun*ction(tx) {

tx.executeSql("DELETE FROM diary WHERE id > 0;");

});

*}*

function *r*e*trieve*Notes(callback) {

db.readTransaction(function(tx) {

v*ar result =* tx.executeSql("SELECT \* FROM diary;");

callback(result.rows);

});

*}*

function *s*e*lectDa*tes(callback) {

db.readTransaction(function(tx) {

v*ar resu*lt = tx.executeSql("SELECT day FROM diary;");

callback(result.rows);

});

*}*

function *g*e*tDate(*callback) {

db.readTransaction(function(tx) {

v*ar result = tx.e*xecuteSql("SELECT \* FROM diary WHERE id=(SELECT MAX(id) FROM diary);");

callback(result.rows);

});

*}*

function *g*e*tCurre*ntValues(callback) {

db.readTra*nsaction(*function(tx) {

var result = tx.executeSql("SELECT \* FROM diary WHERE day = ?;", [currentDate]);

callback(result.rows);

});

}

}

HistoryPage.qml

import QtQuick 2.0

import Sailfish.Silica 1.0

Page {

id: *page*

allowedOrientations: Orientation.All

SilicaFlickable {

anchors.fill: *parent*

PullDownMenu {

MenuItem {

text: *qsTr*("Show MainPage")

onClicked: {

delta = 0

pageStack.push(*Qt*.resolvedUrl("MainPage.qml"))

}

}

}

Button {

id: *back*

x: 80

y: 80

width: 100

height: 10

text: " < "

onClicked: {

selectAllDates()

if (mDates.length - 2 - delta >= 0) {

pageStack.push(*Qt*.resolvedUrl("HistoryPage.qml"))

delta += 1

currentDate = mDates[mDates.length - 1 - delta]

getCurValues()

*l\_d*.text = currentDate

*l\_m*.text = "Вес: " + getMass()

}

}

}

Button {

id: *forward*

x: *parent*.width - 150

y: 80

width: 100

height: 10

text: " > "

onClicked: {

selectAllDates()

if (delta === 1) {

pageStack.push(*Qt*.resolvedUrl("MainPage.qml"))

delta -= 1

currentDate = mDates[mDates.length - 1 - delta]

} else {

pageStack.push(*Qt*.resolvedUrl("HistoryPage.qml"))

delta -= 1

currentDate = mDates[mDates.length - 1 - delta]

getCurValues()

*l\_d*.text = currentDate

*l\_m*.text = "Вес: " + getMass()

}

}

}

contentHeight: *column*.height

Column {

id: *column*

anchors.horizontalCenter: *parent*.horizontalCenter

y: 60

spacing: 50

Label {

id: *l\_d*

text:currentDate

}

Label {

id: *l\_m*

color: Theme.secondaryHighlightColor

font.pixelSize: Theme.fontSizeLarge

text:"Вес: " + getMass()

}

}

Column {

x: 20

y: 300

spacing: 30

Item {

id: *container1*

width: 700

height: 80

property string curColor: "green"

property real progress

progress: (currentProteins < normProteins)? (currentProteins / normProteins) : 1

Rectangle { anchors.fill: *parent*; color: "black"; opacity: 0.5 }

Rectangle {

id: *fill1*; color: getColor(*container1*.progress); height: *container1*.height

width: *container1*.width \* *container1*.progress

}

Text {

font.pixelSize: 40

font.bold: true

anchors.centerIn: *parent*

text:"Белки: " + (currentProteins).toFixed(2) + " /" + normProteins

color: "white"

}

}

Item {

id: *container2*

width: 700

height: 80

property real progress

progress: (currentFats < normFats)? (currentFats / normFats) : 1

Behavior on opacity { NumberAnimation { duration: 600 } }

Rectangle { anchors.fill: *parent*; color: "black"; opacity: 0.5 }

Rectangle {

id: *fill2*; color: getColor(*container2*.progress); height: *container2*.height

width: *container2*.width \* *container2*.progress

}

Text {

font.pixelSize: 40

font.bold: true

anchors.centerIn: *parent*

text:"Жиры: " + (currentFats).toFixed(2) + " /" + normFats

color: "white"

}

}

Item {

id: *container3*

width: 700

height: 80

property real progress

progress: (currentCarbohydrates < normCarbohydrates)? (currentCarbohydrates / normCarbohydrates) : 1

Behavior on opacity { NumberAnimation { duration: 600 } }

Rectangle { anchors.fill: *parent*; color: "black"; opacity: 0.5 }

Rectangle {

id: *fill3*; color: getColor(*container3*.progress); height: *container3*.height

width: *container3*.width \* *container3*.progress

}

Text {

font.pixelSize: 40

font.bold: true

anchors.centerIn: *parent*

text:"Углеводы: " + (currentCarbohydrates).toFixed(2) + " /" + normCarbohydrates

color: "white"

}

}

Item {

id: *container4*

width: 700

height: 80

property real progress

progress: (currentCalorie < normCalorie)? (currentCalorie / normCalorie) : 1

Behavior on opacity { NumberAnimation { duration: 600 } }

Rectangle { anchors.fill: *parent*; color: "black"; opacity: 0.5 }

Rectangle {

id: *fill4*; color: getColor(*container4*.progress); height: *container4*.height

width: *container4*.width \* *container4*.progress

}

Text {

font.pixelSize: 40

font.bold: true

anchors.centerIn: *parent*

text:"Калории: " + (currentCalorie).toFixed(2) + " /" + normCalorie

color: "white"

}

}

}

}

}