

**АННОТАЦИЯ (2000 знаков)**

Курсовая работа посвящена созданию приложения, которая ставит перед собой целью расчета калорийности питания пользователя. В ходе работы были использованы знания о пищевых поведениях человека, языках программирования С++ в среде Qt Creator и запросов SQLite, а также управлении базами данных c помощью СУБД SQLite Manager.

В курсовой работе были поставлены и решены следующие вопросы: выбор языка программирования, способы создания связи между Qt Creator и SQLite, принцип хранения входных данных, способы надежного ввода, необходимые методы для расчёта калорийности продуктов питания, формирование PDF-файла и размещение в нем данные и изображений.

**ANNOTATION**

The present coursework is devoted to the creation of an application, which aims to calculate the number of calories present in the user’s food. In the course of this work, it was needed knowledge of human nutritional behavior, programming and query languages such as C++ on Qt Creator and SQLite, and moreover, of databases management through СУБД SQLite Manager.

In this coursework were set and solved the following questions: selection of programming language, methods to create a connection between Qt Creator и SQLite, principles for storing input data, string validation, needed means to calculate the calories of food and creation of a PDF file, in which is possible to allocate date and images.

# ОГЛАВЛЕНИЕ

[ОГЛАВЛЕНИЕ 5](#_Toc513746153)

[ВВЕДЕНИЕ 6](#_Toc513746154)

[1. Описание сред разработки программного обеспечения 7](#_Toc513746155)

[2. Создание Базы Данных 8](#_Toc513746156)

[*a. Выбор источника базы данных 8*](#_Toc513746157)

[*b. Команды на языке запросов SQLite для создания БД 8*](#_Toc513746158)

[3. Реализация приложения 10](#_Toc513746159)

[4. Листинг программы 12](#_Toc513746160)

[ЗАКЛЮЧЕНИЕ 27](#_Toc513746161)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 28](#_Toc513746162)

[ПРИЛОЖЕНИЕ 29](#_Toc513746163)

[*Приложение № 1: 29*](#_Toc513746164)

[*Приложение № 2 30*](#_Toc513746165)

[*Приложение № 3 31*](#_Toc513746166)

[*Приложение № 4 31*](#_Toc513746167)

[*Приложение № 5: 33*](#_Toc513746168)

# ВВЕДЕНИЕ

Данная курсовая работа посвящена разработке программы способной выводить пользователю расчёт калорийности питания в приложении и в виде PDF-файла. Программа создана на языке С++ с помощью среды разработки Qt Creator. Программа получает входные данные как имя, логин, пароль и т.д. После этого, показывается новое окно, где необходимо выбрать или задать название продукта для поиска и его количество (в граммах), дальше заполняется таблица с выбранными данными и рассчитывается общую результирующую калорийность. Пользователь умеет возможность сохранить информацию о калорийности вводимых продуктов в PDF-файле и распечатать его, также он может войти в систему с его логином и паролем столько раз, сколько он хочет.

На каждом этапе ввода данных проверяется на корректность самом приложением. Под корректностью данных подразумевается то, что вводимые данные должны иметь символьный тип, целочисленный тип и не должны быть пустые поля. Если все данные введены корректно, то в качестве вывода пользователь получает результат расчета калорийности питания. Если же данные вводятся некорректно – пользователь получит уведомление об ошибке и ему последует повторить ввод.

Для создания приложение использовался ноутбук, мышью и клавиатурой, процессор с тактовой частотой 1.6 ГГц или выше, 1 Гб ОЗУ, язык программирования – С++, среда разработки – Qt Creator версия ﻿5.9.1, операционная система – MacOS High Sierra 10.13﻿ ﻿Architecture: x86\_64. 2,7 GHz Intel Core i5 версия 10.13.2.

Современное состояние проблемы и актуальность работы: каждый день человек потребляет определенное количество калорий, и его организм расходует их в жизненных процессах. Если он потребляет больше калорий, чем нужно его организму — организм складывает все «про запас» - в жир, и вес человека растет. Если он потребляет меньше калорий, чем нужно организму, он начинает расходовать запасы — то есть, расщепляет жир и пытается подстроиться под новый режим.

Основная проблема, осуществляющая в современном обществе — это эпидемия ожирения, для чего необходимо оздоровить питание и поощрять предпочтение натуральным ингредиентам. Поэтому данное приложение пытается внести изменения в пищевое поведение людей, внимательно следить за составом и выбирать продукты из натуральных ингредиентов, также предоставляет людям удобства и легкости при необходимости расчета калорийности еды и сохранить их полученные результаты в файле для их использования.

Задачи, которые ставятся для достижения цели: исследование различные БД, содержащие информацию о калорийности еды в официальных сайтах; сборка и изучение понятий; связанных с питанием человека, как BMI (Индекс массы тела) и BMR (Базальная Скорость метаболизма); создание БД, которая содержит отношения для данных пользователей и информации о продуктах; сравнение модели приложения с уже существующими аналогами; изучение методов для работы с Qt Creator и SQLite.

# Описание сред разработки программного обеспечения

База данных будет реализована с помощью СУБД (система управления базами данных) SQLitе, так как она позволяет компоновать программу с её библиотеку, которая будет представлять составную часть программы.

Набор инструментов командной строки для управления файлами базы данных SQLite версия 3.19.3, можно скачивать бесплатно на официальном сайте[[1]](#footnote-1), а для управления базами данных будет использоваться расширение SQLite Manager в браузере Mozilla Firefox ESR 52.7.3 (64-bit). Оно дает возможность управлять (в том числе создавать, удалять, обновлять и добавлять) базы данных, таблицы и естественно, данные. Благодаря SQLite Manager, можно посмотреть содержание таблиц сразу, без использования запросов.

Язык запросов SQLite также будет необходим для управления данных при реализации программы на языке C++ в среде Qt Creator версия ﻿5.9.1. Используется документация для работы с запросами через командную строку[[2]](#footnote-2).

1. Создание Базы Данных

## Выбор источника базы данных

В этом работе используется свободная база данных USDA National Nutrient Database for Standard Reference[[3]](#footnote-3). Из этой базы данных необходима следующая информация: название еды, количество энергии в килокалориях (Kcal), жира, белка и углеводов в граммах (g), содержащие в 100 граммов пищи. Выбирается самую популярную еду (всего 190 штук), так как в исходной базе данных существуют фирменные и промышленные продукты. В приложении № 1 показаны исходная и отредактированная таблицы в MS Excel[[4]](#footnote-4).

## Команды на языке запросов SQLite для создания БД

* Создание БД

Имя БД – это “CourseWork2”.

sqlite3 CourseWork2.sqlite

Для открытия и возможной работы с базой данных используются команды:

sqlite> .open CourseWork2.sqlite

sqlite> .quit (для выхода из системы SQLite)

sqlite> .databases (для просмотра бд)

* Создание таблиц

В данной работе употребляются две таблицы.

**Таблица 1:** UserInfo для хранения данных пользователей приложения.

sqlite> CREATE TABLE UserInfo ( -- название таблицы

No INTEGER PRIMARY KEY AUTOINCREMENT, -- порядок (первичный ключ)

Name CHAR(20) NOT NULL, -- имя пользователя

Surname CHAR(20) NOT NULL, -- фамилия пользователя

Age INTEGER(2) NOT NULL, -- возраст

Weight INTEGER(3) NOT NULL, -- вес

Height FLOAT(3) NOT NULL, -- рост

Sex CHAR(1) CHECK (Sex IN(‘’F’’, ‘’M’’) ), -- пол

Username CHAR(20) NOT NULL, -- логин

Password CHAR(20) NOT NULL, -- пароль

CONSTRAINT constraint\_Username UNIQUE (Username) );

В выше созданной таблице элемент No INTEGER является первичным ключом (PRIMARY KEY) что означает, что не может быть повторных элементов, то есть они единственные. Через оператора AUTOINCREMENT данные самостоятельно увеличиваются с единицы. А оператор CONSTRAINT constraint\_Username UNIQUE (Username) – это ограничение, чтобы данные повторялись, то есть не может быть не боле одного пользователя с одинаковым логином. Оператор NOT NULL обозначает, что не может быть пустых полей, то есть пользователь обязан их заполнить (они в первой очереди необходимые для расчётов приложения.)

**Таблица 2:** FoodCalories для хранения информации отредактированной базы данных показана на Рис. 2.

sqlite> CREATE TABLE FoodCalories ( -- название таблицы

No INTEGER PRIMARY KEY AUTOINCREMENT, -- порядок (первичный ключ)

Food CHAR(20) NOT NULL, -- название еды

kCal FLOAT(7) NOT NULL, -- количество калорий

Fat FLOAT(7), -- количество жира

Carbo FLOAT(7), -- количество углеводов

Protein FLOAT(7) ); -- количество белка

В языке запросов SQLite существуют несколько типов данных, которые должны быть указаны после названия нового столбца таблицы, среди них имеются: INTEGER, CHAR, FLOAT, VAR и т.д. От них зависят допустимые данные, введенные пользователем. Например: Food CHAR(20) NOT NULL, где внутри круглых скобок определяется количество разрешенных символов

* Добавление данных

Необходимые команды для добавления данных:

**- для таблицы 2:**

sqlite> INSERT INTO FoodCalories (Food, kCal, Fat, Carbo, Protein)

VALUES ('Almonds', 579, 49.93, 21.55, 21.15);

Где внутри скобок после слова FoodCalories, указываются названия нужных столбцов и после слова VALUES, заполняются этих столбцов с информацией БД.

Для таблицы 1 не используется командная строка SQLite, так как данные будут добавлены пользователем при работе с приложением. В приложении все введенных команд для заполнения таблицы FoodCalories.

* Обновление и удаление данных

При работе с командной строкой постоянно приходится обновлять и удалять уже существующих данных по разным причинам. Например:

- Для обновления

sqlite> ALTER TABLE UserInfo ADD COLUMN Sex CHAR(1); (для добавления столбца в таблицу)

sqlite> ALTER TABLE FoodCalories2 RENAME TO FoodCalories; (для переименования таблицы)

- Для удаления таблиц

sqlite> DROP TABLE FoodCalories2;

В приложении № 2 показываются окончательные таблицы в SQLite Manager - Mozilla Firefox

# Реализация приложения

В данной работе используется объектно-ориентированный язык программирования С++ в среде Qt Creator. Ниже представляются употребляемые формы (всего их 4) с кнопками для перехода к другим формам. Язык интерфейса – это английский, но при заполнении информации пользователь имеет возможность ввести данные латинскими и русскими буквами. Во всех формах, кроме первого проверяется корректность ввода данных (в том числе ограничивается длина строки, создается поля только для чисел и только для букв, и т.д.). Элементы в формах были сгруппированы для удобной работы с ними, то есть в момент, когда пользователь увеличивает размер любой формы, она не исказится, а увеличится вместе с элементами, которые она содержит. Компоновка форм: по сетке. В приложении № 4 показаны созданных форм.

Объектно-ориентированный язык программирования (ООП) С++ основан на работе с объектами. Объест – это сущность, которая связывает в себе поля и методы (функции) программы. Поля содержат данные, а методы представляет себя действия, которые объект выполняет. Боле того, эти методы работают, чтобы никакая другая часть кода не могла получить доступ к этим данным, это и есть цель ООП. (Wikipedia, 2018)

При создании проекта Qt Creator формирует 4 вида файлов для хранения кода каждого класса:

1. файл .pro (который содержит информацию о всех файлах проекта, а также библиотеки), в данном работе добавляются следующие библиотеки:

QT += core gui sql # для работа с БД SQLite

QT += printsupport # для работа с <QPrinter>, который позволяет рисовать изображения в файлах.

1. заголовочные (.h) где сохраняется структура класса,
2. исходники (.cpp) в нем расположен код (содержание) структура класса, и
3. формы (.ui).

Класс – это основа языка программирования С++, определяющий новый тип данных. В нем соединяются код и данные (также функции). Для чего нужны классы? С помощью классов, пользователь имеет возможность определить собственные тиры данных. В качестве примера показывается следующую часть кода:

Класс: MainWindow

class MainWindow : public QMainWindow

{

public:

explicit MainWindow(QWidget \*parent = 0);

~*MainWindow*(); //деструктор

private:

Ui::MainWindow \*ui;

};

Ключевое слово «class» определяет имя класса, в этом примере имя класса это MainWindow. Внутри класса существуют публичные и частные части. По умолчанию все члены класса являются частными. После слова «public» располагаются публичные данные (и функции), а после «private» - частные данные. При этом, пользователь не имеет доступа к частным данным, только к публичным.

# Листинг программы

1. Класс: MainWindow

Заголовочный MainWindow.h

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>// содержит информацию о классе MainWindow

#include <QtSql> // Для работы с БД SQLite

#include <QDebug> // Для проверки статуса кода

#include <QFileInfo> // Предоставляет системно-независимую информацию о файлах. Приложение работает без этих двух последних библиотек, но они были использованы при реализации программы для проверки состояния кода.

#include "signin.h" // для работы с классом Signin

#include "registration.h" // для работы с классом Registration

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~*MainWindow*();

private slots:

void on\_pushButton\_signin\_clicked(); // метод при нажатии на кнопку Sign in

void on\_pushButton\_exit\_clicked(); // метод при нажатии на кнопку Exit

void on\_pushButton\_signup\_clicked(); // метод при нажатии на кнопку Sign up

private:

Ui::MainWindow \*ui;

Signin \*sign\_in; // для перехода в класс Sign in

Registration \*registration\_; // для перехода в класс Registration

};

#endif // MAINWINDOW\_H

Исходник MainWindow.cpp

#include "mainwindow.h" // Для работы с классом MainWindow

#include "ui\_mainwindow.h" // Для работы с формой класса MainWindow

#include <QMessageBox> // Для создания модального диалога для информирования пользователю или для запроса пользователя и получения ответа.

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this); //для запуска формы класса

}

MainWindow::~*MainWindow*() // деструктор

{

delete ui;

}

//Описание методов

void MainWindow::on\_pushButton\_signin\_clicked()// при нажатии на кнопку Sign in

{

//при нажатии на кнопку, осуществляется переход в другую форму

hide(); //скрывается предыдущую форму

sign\_in = new Signin(this); //создание нового объекта

sign\_in->show(); // показывается форма

}

void MainWindow::on\_pushButton\_exit\_clicked()// при нажатии на кнопку exit

{

QMessageBox::StandardButton reply = QMessageBox::question( // создание окно для выхода из приложения. Выбор yes либо no пользователем

this,"EXIT","Are you sure?",QMessageBox::Yes | QMessageBox::No);

if (reply == QMessageBox::Yes) QApplication::quit(); //условие при положительном ответе

}

void MainWindow::on\_pushButton\_signup\_clicked()// при нажатии на кнопку Sign up

{

//при нажатии на кнопку, осуществляется переход в другую форму

hide(); //скрывается предыдущую форму

registration\_ = new Registration(this); // создание нового объекта

registration\_->show();// показывается форма

}

1. Класс: Signin

Заголовочный Signin.h

#ifndef SIGNIN\_H

#define SIGNIN\_H

#include <QDialog> // Базовый класс диалоговых окон. Краткая связь с пользователем.

#include "calculate.h" //Добавление класса Calculate

#include <QSqlDatabase> //Для работы с БД

class Calculate; //имя класса Calculate – это делается для создания объекта после слова “private”

namespace Ui {

class Signin;

}

class Signin : public QDialog

{

Q\_OBJECT

public:

explicit Signin(QWidget \*parent = 0);

~*Signin*(); // деструктор

public: //Для открытия и закрытия связи с SQLite

QSqlDatabase mydb; //создание связи с БД

void connOpen(); // функция для открытия связи

void connClose(); // функция для закрытия связи

//фунция для передачи QString в другой класс

QString value();

private slots: //методы

void on\_pushButton\_back\_clicked(); // при нажатии на кнопку Back

void on\_pushButton\_login\_clicked(); // при нажатии на кнопку Log in

private:

Ui::Signin \*ui;

Calculate \*calculatew; // cоздание объекта для перехода в класс Calculate

};

#endif // SIGNIN\_H

Исходник Signin.cpp

#include "signin.h" // добавление заголовочного файла класса Signin

#include "ui\_signin.h" // добавление заголовочного файла класса Signin

#include "mainwindow.h" // добавление заголовочного файла класса MainWindow

#include <QMessageBox> // Для создания модального диалога для информирования пользователю или для запроса пользователя и получения ответа.

QString user;// QString для передачи в другой класс (глобальная переменная)

Signin::Signin(QWidget \*parent) :

QDialog(parent),

ui(new Ui::Signin)

{

ui->setupUi(this); // для запуска формы

connOpen(); // открытие связи с SQLite сразу после запуска формы

}

Signin::~*Signin*() // описание конструктора

{

delete ui;

}

void Signin::connOpen() // описание функции для открытия связи с SQLite

{

mydb = QSqlDatabase::addDatabase("QSQLITE"); // создание связи

mydb.setDatabaseName("/Users/a1/CourseWork2.sqlite"); // указание адреса расположения в компьютере файла, содержащего БД

mydb.open(); // открытие связи

}

void Signin::connClose() // описание функции для закрытия связи с SQLite

{

QString connection; // имя связи

connection = mydb.connectionName(); // присваивание

mydb.close(); // закрытие связи

mydb = QSqlDatabase();

mydb.removeDatabase(connection); // удаление связи

}

void Signin::on\_pushButton\_back\_clicked() // описание метода при нажатии на кнопку Back

{

hide(); //скрытие предыдущей формы

MainWindow \*main\_window = new MainWindow(); // создание новой формы

main\_window->show(); // показ текущей формы

connClose(); // закрытие связи - При переходе в другой класс необходимо открыть или закрыть связь с SQLite.

}

void Signin::on\_pushButton\_login\_clicked()// описание метода при нажатии на кнопку Log in

{

QString username,password; // Создание переменных для хранения логина и пароля пользователя

username = ui->lineEdit\_username->text(); //присваивание

password = ui->lineEdit\_password->text(); //присваивание

user = username; //присваивание логина и user (глобальная переменная) для передачи в другую форму

if(!mydb.isOpen()) // условие – если БД не открывается, то выходит сообщение в консоль приложения об ошибке

{

qDebug()<<"Failed to open database!";

return;

}

QSqlQuery qry; // формирование переменной для создания запросов на языке SQLite

qry.prepare("select \* from UserInfo where Username='"+username+"' and Password='"+password+"'"); // Запрос для поиска зарегистрируемого пользователя в БД. Если нет данных о пользователе, то показывается соответствующее сообщение

if(qry.exec()) // запуск запроса

{

int counter = 0;

while(qry.next())

{

counter++;

}

if(counter == 1)

{

QMessageBox::information(this,"OK","Correct username and password");

connClose(); // при положительном ответе от приложения, закрывается связь и сразу осуществляется переход в другой класс

hide(); // скрытие предыдущей формы

calculatew = new Calculate(this); // создание новой формы

calculatew->show(); // показ формы

}

if(counter < 1)

QMessageBox::warning(this,"OK","Wrong username or password!"); // при отрицательном ответе от приложения

}

}

QString Signin::value(){ // функция для передачи данных в другой класс – в этом случае передаётся QString

return user;

}

1. Класс: Registration

Заголовочный Registration.h

#ifndef REGISTRATION\_H

#define REGISTRATION\_H

#include <QDialog>

#include "signin.h"

namespace Ui {

class Registration;

}

class Registration : public QDialog

{

Q\_OBJECT

public:

Signin conn;//For working with Singin class in Registration class

explicit Registration(QWidget \*parent = 0);

~*Registration*();

private slots:

void on\_pushButton\_back\_clicked();

void on\_pushButton\_done\_clicked();

private:

Ui::Registration \*ui;

};

#endif // REGISTRATION\_H

Исходник Registration.cpp

#include "registration.h"

#include "ui\_registration.h"

#include "mainwindow.h"

#include <QMessageBox>

#include <QRegExpValidator>

//String validation

#define ONLYletters\_RX "^[A-zйцукенгшщзхъфывапролджэёячсмитьбю]{15}$"//setting validation for input - letters allowed (15)

#define ONLYnumbers1\_RX "^[0-9]{2}$"

#define ONLYnumbers2\_RX "^[0-9]{3}$"

#define Username\_RX "^[A-z1234567890@-.\_/]{15}$" // 6 - 15

#define Password\_RX "^[A-z1234567890@-.\_/]{32}$"//8 - 32 http://help.pearsoncmg.com/rumba/b2c\_self\_reg/en/Content/b2c\_signin\_guidelines.html

Registration::Registration(QWidget \*parent) :

QDialog(parent),

ui(new Ui::Registration)

{

ui->setupUi(this);

//ComboBox "Sex: Female(F) or Male(M)"

ui->comboBox\_S->addItem("");

ui->comboBox\_S->addItem("F");

ui->comboBox\_S->addItem("M");

//String validation

//validation for name, surname

QRegExp rxLetters(ONLYletters\_RX);

QRegExpValidator \*Vletters = new QRegExpValidator(rxLetters, this);

ui->lineEdit\_name->setValidator(Vletters);

ui->lineEdit\_surname->setValidator(Vletters);

//validation for age

QRegExp rxNumbers1(ONLYnumbers1\_RX);

QRegExpValidator \*VNumbers1 = new QRegExpValidator(rxNumbers1, this);

ui->lineEdit\_age->setValidator(VNumbers1);

//validation for weight, height

QRegExp rxNumbers2(ONLYnumbers2\_RX);

QRegExpValidator \*VNumbers2 = new QRegExpValidator(rxNumbers2, this);

ui->lineEdit\_w->setValidator(VNumbers2);

ui->lineEdit\_h->setValidator(VNumbers2);

//validation for username

QRegExp rxUsername(Username\_RX);

QRegExpValidator \*VUsername = new QRegExpValidator(rxUsername, this);

ui->lineEdit\_username->setValidator(VUsername);

//validation for password

QRegExp rxPassword(Password\_RX);

QRegExpValidator \*VPassword = new QRegExpValidator(rxPassword, this);

ui->lineEdit\_password->setValidator(VPassword);

}

Registration::~*Registration*()

{

delete ui;

}

void Registration::on\_pushButton\_back\_clicked()//back

{

hide();

MainWindow \*main\_window = new MainWindow();

main\_window->show();

conn.connClose();

}

void Registration::on\_pushButton\_done\_clicked() //Done - https://www.youtube.com/watch?v=ENtlGF2iADI&t=59s

{

QString name, surname, age, w; //QStrings for each field in Registration.ui

QString h, s, username, password;

name = ui->lineEdit\_name->text();

surname = ui->lineEdit\_surname->text();

age = ui->lineEdit\_age->text();

w = ui->lineEdit\_w->text();

h = ui->lineEdit\_h->text();

s = ui->comboBox\_S->currentText();

username = ui->lineEdit\_username->text();

password = ui->lineEdit\_password->text();

int ageA = age.toInt();

//checking the size of each string

if(name.size() > 0 && surname.size() > 0 && age.size() > 0 &&

ageA > 0 && w.size() > 0 && h.size() > 0 && s.size() > 0

&& username.size() > 0 && password.size() > 0)

{

if(!conn.mydb.isOpen())//opening database

{

qDebug() << "Failed to open the database!";

return;

}

QSqlQuery qry;

qry.prepare("insert into UserInfo (Name,Surname,Age,Weight,Height,Sex,Username,Password) values ('"

+name+"','"+surname+"','"+age+"','"+w+"','"+h+"','"+s+"','"+username+"','"+password+"')");

if(qry.exec())//executing database

{

QMessageBox::information(this, "OK", "Dada saved!");

conn.connClose();

hide();

MainWindow \*mainWindow = new MainWindow;

mainWindow->show();

}

else

QMessageBox::critical(this,tr("error::"),qry.lastError().text());

}

else QMessageBox::warning(this, "Warning1", "Invalid data entered!");

}

1. Класс: Calculate

Заголовочный Calculate.h

#ifndef CALCULATE\_H

#define CALCULATE\_H

#include <QDialog>

#include <QtSql>

#include <QFileDialog>

#include <QPrinter> // PDF

#include <QPainter> // PDF

#include <QPixmap> // to load an image

#include "signin.h"

namespace Ui {

class Calculate;

}

class Calculate : public QDialog

{

Q\_OBJECT

public:

explicit Calculate(QWidget \*parent = 0);

~*Calculate*();

private slots:

void on\_pushButton\_back\_clicked();

void on\_pushButton\_lt\_clicked();

void on\_pushButton\_searchf\_clicked();

void on\_tableView\_activated(const QModelIndex &index);

void on\_pushButton\_add\_clicked();

void on\_pushButton\_cc\_clicked();

void on\_pushButton\_del\_clicked();

void on\_pushButton\_PDF\_clicked();

private:

Ui::Calculate \*ui;

};

#endif // CALCULATE\_H

Исходник Calculate.cpp

#include "calculate.h"

#include "ui\_calculate.h"

#include <QMessageBox>

//String Validation

#define ONLYletters\_RX "^[A-zйцукенгшщзхъфывапролджэёячсмитьбю]{10}$"//setting validation for input - 10 letters allowed

#define ONLYnumbers1\_RX "^[0-9]{5}$"

int flag = 0;

Calculate::Calculate(QWidget \*parent) :

QDialog(parent),

ui(new Ui::Calculate)

{

ui->setupUi(this);

setWindowTitle("Calculation Window");

//String validation

//validation Enter food

QRegExp rxLetters(ONLYletters\_RX);

QRegExpValidator \*Vletters = new QRegExpValidator(rxLetters, this);

ui->lineEdit\_food->setValidator(Vletters);

//validation for enter grams

QRegExp rxNumbers1(ONLYnumbers1\_RX);

QRegExpValidator \*VNumbers1 = new QRegExpValidator(rxNumbers1, this);

ui->lineEdit\_grams->setValidator(VNumbers1);

//setting features for Table Widget

QStringList myList;

ui->tableWidget->setColumnCount(3);

myList << "Food" << "Calories (kCal)" << "Amount (g)";

ui->tableWidget->setHorizontalHeaderLabels(myList);

//Welcome [...]!

QString welcome = "Welcome ";

QString exc\_mark = "!";

Signin nameR;

QString usernameR = nameR.value();//saving username from Signin class

QSqlQuery qry;

qry.prepare("select \* from UserInfo where Username = '"+usernameR+"'");

if(qry.exec())

{

while(qry.next())

{

QString greeting = welcome + qry.value(1).toString() + exc\_mark;

ui->label\_hello->setText(greeting);//1 is the column number

}

nameR.connClose();//close connection

}

else

QMessageBox::critical(this,tr("error::"),qry.lastError().text());

}

Calculate::~*Calculate*()

{

delete ui;

}

void Calculate::on\_pushButton\_back\_clicked()//back

{

hide();

Signin \*sign\_in = new Signin();

sign\_in->show();

}

void Calculate::on\_pushButton\_lt\_clicked()//load table - https://www.youtube.com/watch?v=OrHitqwJu8Q&t=76s

{

Signin conn;

QSqlQueryModel \*myModel = new QSqlQueryModel();

QSqlQuery \*qry = new QSqlQuery(conn.mydb);

qry->prepare("select Food, kCal, Fat, Carbo, Protein from FoodCalories");

qry->exec();

myModel->setQuery(\*qry);

ui->tableView->*setModel*(myModel);

ui->tableView->resizeColumnsToContents();

conn.connClose();

}

void Calculate::on\_pushButton\_searchf\_clicked()//search food - https://ru-clip.com/video/WpU9aU\_h8nQ/simple-library-management-application-using-qt.html

{

QString foodN = ui->lineEdit\_food->text();

if(!foodN.size()) QMessageBox::warning(this, "OK", "Insert data!");

else{

QString foodBind = foodN + "%"; //to find any word started by the given letter

Signin conn;

QSqlQueryModel \*myModel = new QSqlQueryModel();

QSqlQuery \*qry = new QSqlQuery(conn.mydb);

qry->prepare("select Food, kCal, Fat, Carbo, Protein from FoodCalories where Food LIKE :food COLLATE NOCASE;"); //COLLATE NOCASE - case insensitive

qry->bindValue(":food", foodBind);

qry->exec();

myModel->setQuery(\*qry);

ui->tableView->*setModel*(myModel);

conn.connClose();

}

}

void Calculate::on\_tableView\_activated(const QModelIndex &index)// - https://www.youtube.com/watch?v=6GeOUAV0TaA&t=22s

{

QString val = ui->tableView->model()->*data*(index).toString();//we create a String for the selected item in the table

Signin conn;

QSqlQuery qry;

qry.prepare("select \* from FoodCalories where Food = '"+val+"'");

if(qry.exec())

{

while(qry.next())

{

ui->lineEdit\_food->setText(qry.value(1).toString());//1 is the column number

ui->lineEdit\_ca->setText(qry.value(2).toString());

}

conn.connClose();

}

else

QMessageBox::critical(this,tr("error::"),qry.lastError().text());

}

void Calculate::on\_pushButton\_add\_clicked()//add button - http://minuvahard10.com/ - https://www.youtube.com/watch?v=qXDyK6kTSZk

{

QString F,G,C;

F = ui->lineEdit\_food->text();

G = ui->lineEdit\_grams->text();

C = ui->lineEdit\_ca->text();

if(F.size() > 0 && G.size() > 0 && C.size() > 0)

{

ui->tableWidget->insertRow(ui->tableWidget->rowCount());//to insert the number of raws needed

ui->tableWidget->setItem(ui->tableWidget->rowCount()-1,0,new QTableWidgetItem(ui->lineEdit\_food->text()));

ui->tableWidget->setItem(ui->tableWidget->rowCount()-1,1,new QTableWidgetItem(ui->lineEdit\_ca->text()));

ui->tableWidget->setItem(ui->tableWidget->rowCount()-1,2,new QTableWidgetItem(ui->lineEdit\_grams->text()));

ui->tableWidget->resizeColumnsToContents();

}

else

if(F.size() > 0 && C.size() > 0 && G.size() == 0)

QMessageBox::warning(this, "OK", "Enter number of grams!");

else

QMessageBox::warning(this, "OK", "Select item form database!");

}

void Calculate::on\_pushButton\_cc\_clicked()//calculate calories

{

float cal,amount,xcal,totalc = 0;

if(ui->tableWidget->rowCount() == 0)

QMessageBox::warning(this,"OK","Insert data!");

else

{

for(int i = 0 ; i < ui->tableWidget->rowCount() ; i++)

{

cal = ui->tableWidget->item(i,1)->text().toFloat(); //calories

amount = ui->tableWidget->item(i,2)->text().toFloat(); //amount

xcal = (cal\*amount)/100;//calculation

totalc = totalc + xcal;//add to current value

ui->label\_tcr->setText(QString::number(totalc));

}

}

}

void Calculate::on\_pushButton\_del\_clicked() //Delete item

{

if(ui->tableWidget->rowCount() == 0)

QMessageBox::information(this, "0", "Empty table!");

else

{

if(ui->tableWidget->currentRow() > -1)

ui->tableWidget->removeRow(ui->tableWidget->currentRow());

else

QMessageBox::information(this, "0", "Select a raw to delete");

}

}

void Calculate::on\_pushButton\_PDF\_clicked() //Create text file and PDF

{

//txt. file creation

if(ui->tableWidget->rowCount() == 0 || ui->label\_tcr->text().size() == 0)

QMessageBox::warning(this, "OK", "Insert data!");

else

{

flag++; //flag sirve para contar las veces que se dio cick al icono "create PDF"

//debug

QString order = QString::number(flag); //conversion int to QString

QString file\_name1 = "/Users/a1/cursavaya2/myResults" + order + ".txt"; qDebug() << "flag = " << flag;

QFile file(file\_name1);

if(!file.*open*(QIODevice::WriteOnly)) //https://forum.qt.io/topic/23764/save-qtablewidget/4

QMessageBox::warning(this, "file1", "Error: File not opened"); //if the fail could not be opened

else

{

QTextStream out(&file);

QStringList strList;//create a List

out << "MY NUTRITIONAL TABLE" << "\n";

out << endl;

// - // - // find name and surname in database // - // - //

Signin nameR;

QString usernameR = nameR.value(); //saving username from Signin class

QString Wname; //whole name

QString age, weight, height, sex;

QSqlQuery qry;

qry.prepare("select \* from UserInfo where Username = '"+usernameR+"'");

if(qry.exec())

{

while(qry.next())

{

Wname = qry.value(1).toString() + " " + qry.value(2).toString();

age = qry.value(3).toString();

weight = qry.value(4).toString();

height = qry.value(5).toString();

sex = qry.value(6).toString();

}

nameR.connClose(); //close connection

}

else

QMessageBox::critical(this,tr("error::"),qry.lastError().text());

// - // - //

out << "- User: " << Wname << "\n";

out << "- Age: " << age << " years old." << "\t- Sex: " << sex << "\n";

out << "- Weight: " << weight << " kg." << "\t- Height: " << height << " cm." << "\n" << endl;

for(int i = 0 ; i < ui->tableWidget->columnCount() ; i++)

{

strList <<

"" +

ui->tableWidget->horizontalHeaderItem(i)->*data*(Qt::DisplayRole).toString() +

"";

}

out << strList.join(" | ") << "\n";

for( int r = 0 ; r < ui->tableWidget->rowCount() ; r++ )

{

strList.clear();

for( int c = 0 ; c < ui->tableWidget->columnCount() ; c++ )

{

QTableWidgetItem \*item = ui->tableWidget->item(r, c);//Load items

if (!item || item->text().isEmpty()) //Test if there is something at item(r,c)

ui->tableWidget->setItem(r, c, new QTableWidgetItem("0"));//IF there is nothing write 0

strList << "" + ui->tableWidget->item(r, c)->text() + "";

}

out << strList.join( " | " ) + "\n";

}

out << endl;

out << "Total Calories: " << ui->label\_tcr->text() << " kCal." << endl;

//Calculate BMI Body mass index - http://www.calculator.net/bmi-calculator.html?ctype=metric&cage=20&csex=m&cheightfeet=5&cheightinch=10&cpound=160&cheightmeter=172&ckg=62&printit=0&x=81&y=21

//BMI is a measurement of a person's leanness or corpulence based on their height and weight, and is intended to quantify tissue mass.

float BMI = 0;

float weightF = weight.toFloat(); //mass(kg)

float heightF = height.toFloat(); //height(cm)

BMI = weightF / pow(heightF \* 0.01, 2); //height(m)

out << "BMI (Body mass index): " << BMI << " kg/m2 - ";

//BMI Table for Adults

if (BMI < 16)

out << "Severe Thinness :(" << endl;

else if (BMI >= 16 || BMI < 17)

out << "Moderate Thinness :(" << endl;

else if (BMI >= 17 || BMI < 18.5)

out << "Mild Thinness :(" << endl;

else if (BMI >= 18.5 || BMI < 25)

out << "Normal (:" << endl;

else if (BMI >= 25 || BMI < 30)

out << "Overweight :(" << endl;

else if (BMI >= 30 || BMI < 35)

out << "Obese Class I :(" << endl;

else if (BMI >= 35 || BMI < 40)

out << "Obese Class II :(" << endl;

else if (BMI >= 40)

out << "Obese Class III :(" << endl;

//BMR (Basal Metabolic Rate) - http://www.calculator.net/calorie-calculator.html?ctype=metric&cage=20&csex=m&cheightfeet=5&cheightinch=10&cpound=160&cheightmeter=172&ckg=62&cactivity=1&printit=0&x=91&y=21

// - Your BMR is the amount of calories your body requires just to perform daily, life sustaining functions -

//analogue https://www.freedieting.com/calorie-calculator

int ageF = age.toInt();

long BMR = 0;

if (sex == 'F') {

//For women: 10 × weight(kg) + 6.25 × height(cm) - 5 × age(y) - 161

BMR = 10 \* weightF + 6.25 \* heightF - 5 \* ageF - 161;

out << "BMR (Basal Metabolic Rate): " << BMR << " Cal/day" << endl;

} else {

//For men: 10 × weight(kg) + 6.25 × height(cm) - 5 × age(y) + 5

BMR = 10 \* weightF + 6.25 \* heightF - 5 \* ageF + 5;

out << "BMR (Basal Metabolic Rate): " << BMR << " Cal/day\n" << endl;

}

out << "More info:\n";

out << "BMI is a measurement of a person's leanness intended\nto quantify tissue mass.\n";

out << "BMR is the amount of calories your body requires to\nperform life sustaining functions.";

//QMessageBox::information(this, "file1", "Data successfully saved!");

file.*close*();

}

//PDF

//1.Read .txt file already created

if(!file.*open*(QIODevice::ReadOnly))

QMessageBox::warning(this, "error1", "Error: failed to open file"); //si el archivo no se abre, muestra un mensaje de error

else

{

//convetir un archivo a stream

QTextStream in(&file); int k = 0;

QString textL[20];

while (!in.atEnd())

{

textL[k] = in.readLine();

k++;

}

file.*close*();

//2.Choose name for PDF file

//сохранить данные в выбранном пользователем файле

QString filter = "Text File (\*.pdf)";

QString file\_name2 = QFileDialog::getSaveFileName

(this, "Save file", "/Users/a1/cursavaya2/",filter);

QPrinter printer;

printer.setOutputFormat(QPrinter::PdfFormat);

printer.setOutputFileName(file\_name2);

QPainter painter;

//To load a resource (image)

QPixmap pixmap("/Users/a1/cursavaya2/resourcePDF1.png");

if (! painter.begin(&printer)) // failed to open file

{

qWarning("failed to open file, is it writable?");

QMessageBox::warning(this,"pdfFile","Error: failed to open file");

flag--;//then the counter is 0 again, because the file was not chosen

}

else

{

//print image in PDF file, set target and source - http://doc.qt.io/qt-5/qpainter.html#drawPixmap

QRectF target(-10, -10, 595, 842); //PDF size 595X842

QRectF source(0, 0, 1560, 2208); // image size 1560X2208

painter.drawPixmap(target, pixmap, source);

//print text in PDF file

painter.setPen(Qt::white); //white letters

painter.setFont(QFont("courier", 9)); // change font size - century

int y = 410; // coordinates for "y" will increase while writing lines in the pdf file

for (int i = 0; i < k; i++)

{

painter.drawText(45, y, textL[i]); // x = 75, y = 450

painter.drawText(45, y+= 10, "");

}

/\*

for printing a new page

if (! printer.newPage())

{

qWarning("failed in flushing page to disk, disk full?");

QMessageBox::warning(this,"pdfFile2","Error: failed in flushing page to disk");

}

painter.drawText(20, 20, "Test 2"); // x = 20, y = 20

\*/

painter.end();

QMessageBox::information(this, "fileC", "PDF successfully created!");

}

}

}

//To create a background for our PDF file, a new image resource is created

}

Пример PDF-файла с результатами можно посмотреть в приложении № 5.

# ЗАКЛЮЧЕНИЕ

В результате выполнения данной курсовой работы были освоены понятия о языке программирования С++, язык запросов SQLite и использование визуальные компоненты, из которых строится графический интерфейс в среде Qt Creator. Также были рассмотрены способы работы с Qt Creator и SQLite одновременно. С помощью созданного приложения имеется возможность рассчитать калорийности питания разных продуктов, выбранных пользователем в зависимости от их количества. Более того, в приложении можно создать личный аккаунт, пользователь заполняет необходимые поля и для входа в систему от него требуется писать логин и пароль. Это делает, что приложение было удобным инструментом для работы с личными данными, которые доступны только определенному пользователю.

Работа с базой данных делает возможно управление больших объемов информации, которую легко манипулируется с помощью запросов SQLite. Благодаря этому свойству, приложение работает более оптимально и выдает только требуемые результаты. Также информация в БД является независимой от программы, то есть возможно в любой момент добавить или удалить данные не изменяя состояние кода приложения.

Помимо того, при реализации работы осуществлялось исследование разные варианты БД и уже существующих аналогов приложения, содержащие информацию о калорийности еды, необходимые методы для расчётов, и т.д. Это помогло при создании новой БД для данного приложения. Также были изучены понятия, связанные с питанием человека, как BMI (Индекс массы тела) и BMR (Базальная скорость метаболизма).

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

**Food Calculator** [В Интернете] / авт. WebMD LLC // Diet & weight management - Food Calculator. - 2005-2018 г.. - 20 Ноябрь 2017 г.. - https://www.webmd.com/diet/healthtool-food-calorie-counter.

**ProgrammingKnowledge** [В Интернете] / авт. Codebind // ProgrammingKnowledge. - 2018 г.. - 04 Ноябрь 2017 г.. - http://www.codebind.com/.

**ProgrammingKnowledge** [В Интернете] / авт. ProgrammingKnowledge // ProgrammingKnowledge. - 2012 г.. - 25 Сентябрь 2017 г.. - https://www.youtube.com/user/ProgrammingKnowledge/about.

**Qt Documentation** [В Интернете] / авт. The Qt Company // Qt. - 2018 г.. - 20 Ноябрь 2017 г.. - http://doc.qt.io/.

**SQLite** [В Интернете] / авт. SQLite // SQLite. - D. Richard Hipp, 2018 г.. - 10 Сентябрь 2017 г.. - https://www.sqlite.org/download.html.

**Stack Overflow** [В Интернете] / авт. Stack Overflow // Stack Overflow. - Stack Exchange Inc, 2008 г.. - 10 Сентябр 2017 г.. - https://stackoverflow.com/company.

**Tutorialspoint** [В Интернете] / авт. Tutorialspoint // Learn SQLITE. - 2018 г.. - 10 Декабрь 2017 г.. - https://www.tutorialspoint.com/sqlite/index.htm.

**USDA** [В Интернете] / авт. U.S. Department of agriculture // U. S. DEPARTMENT OF AGRICULTURE. - United States government. - 05 Ноябрь 2018 г.. - https://www.usda.gov/.

**Объектно-ориентированное программирование** [В Интернете] / авт. Wikipedia // Wikipedia The Free Encyclopedia. - 2018 г.. - 12 Февраль 2018 г.. - https://ru.wikipedia.org/wiki/Объектно-ориентированное\_программирование.

# ПРИЛОЖЕНИЕ

## Приложение № 1:

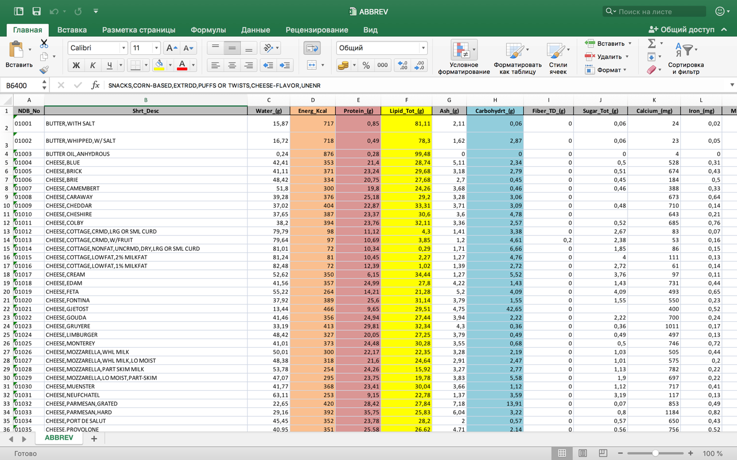


Рис.1. База данных USDA в MS Excel

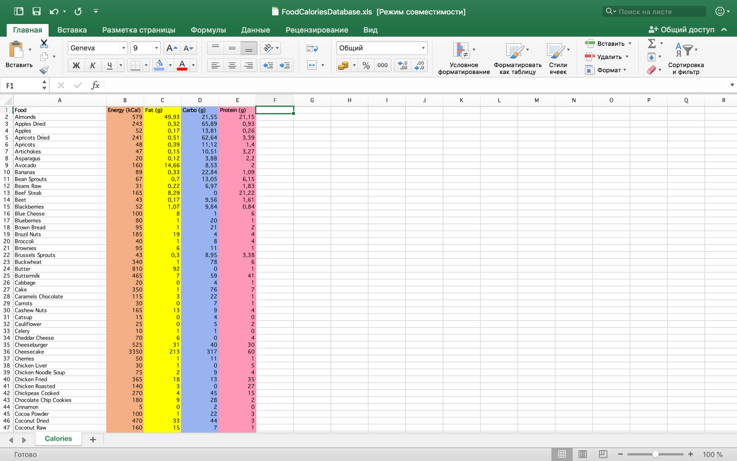


Рис.2. Отредактированная бд в MS Excel

## Приложение № 2

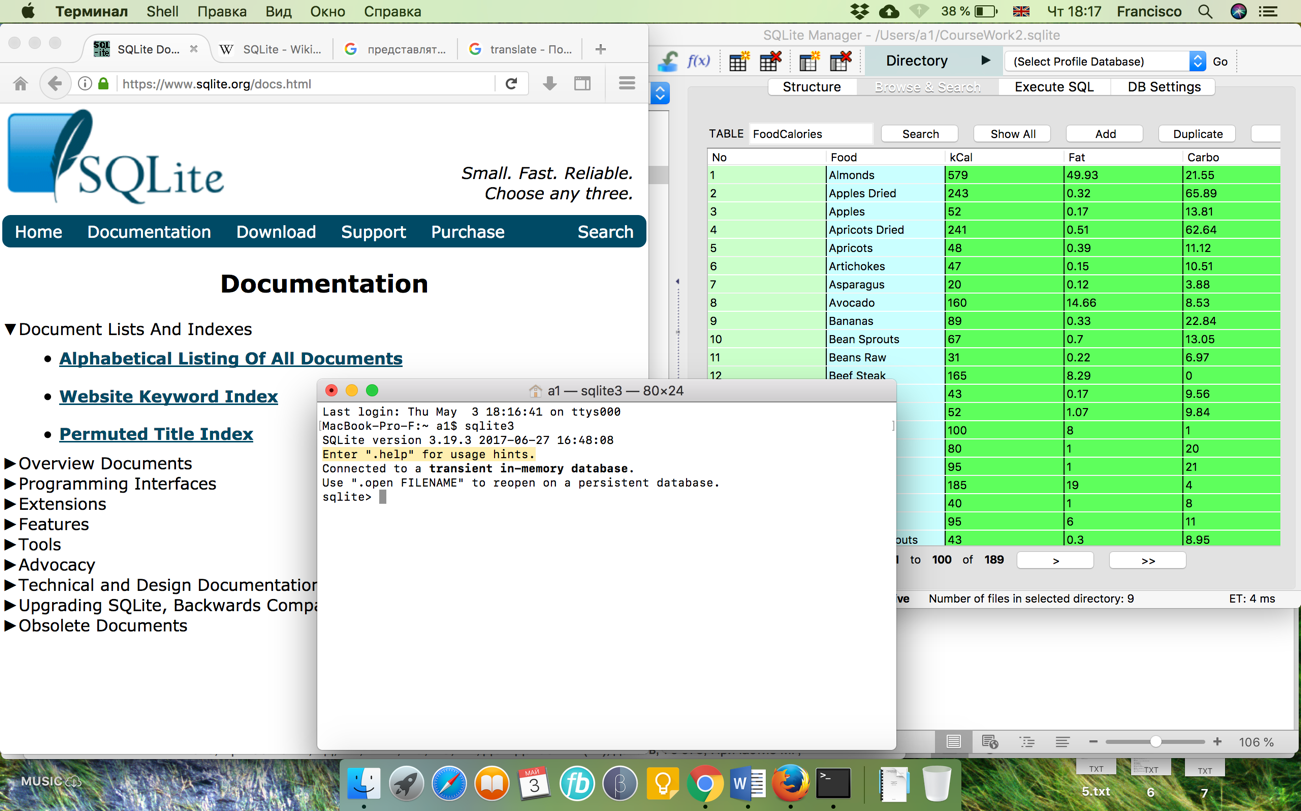


Рис.3. SQLite version 3.19.3 командная строка

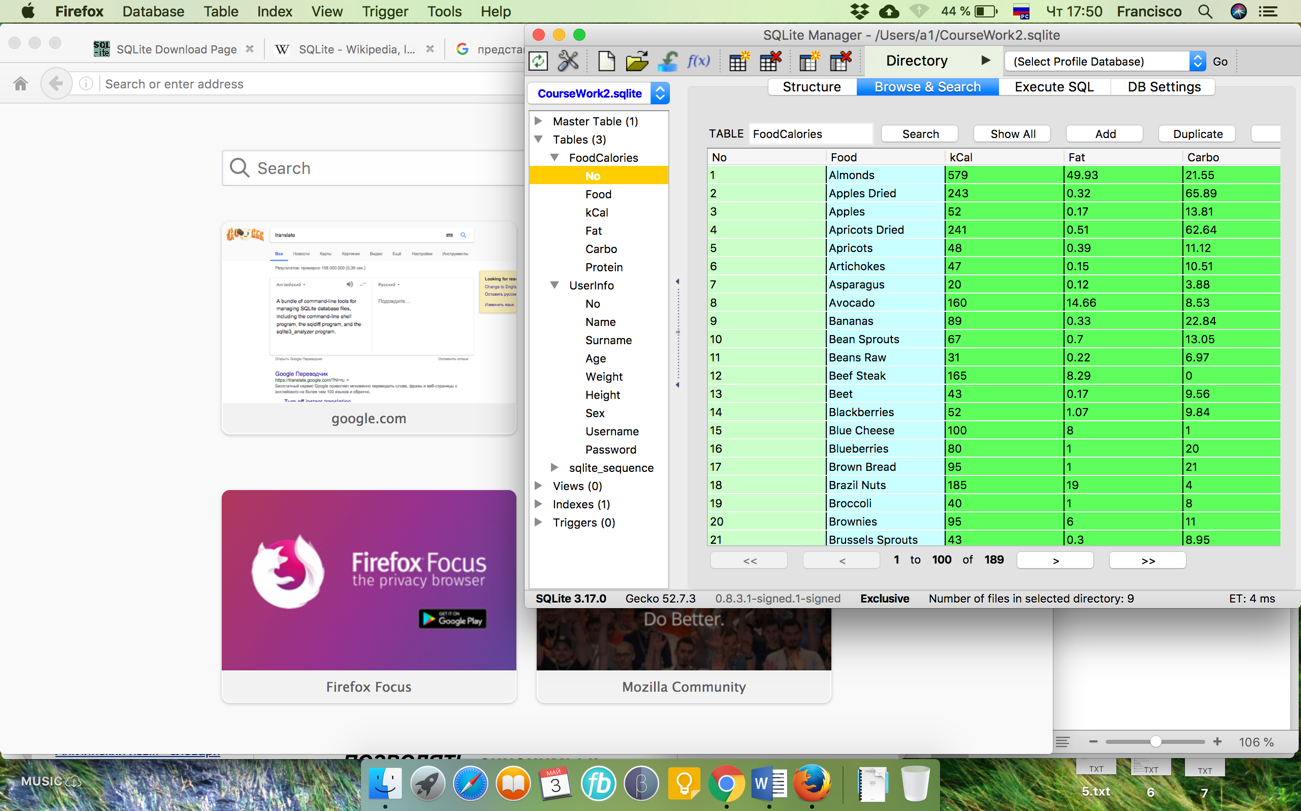


Рис.4. SQLite Manager - Mozilla Firefox

## Приложение № 3



Рис.5. Команды для заполнения таблицы FoodCalories

## Приложение № 4

**Форма 1:** MainWindow.ui

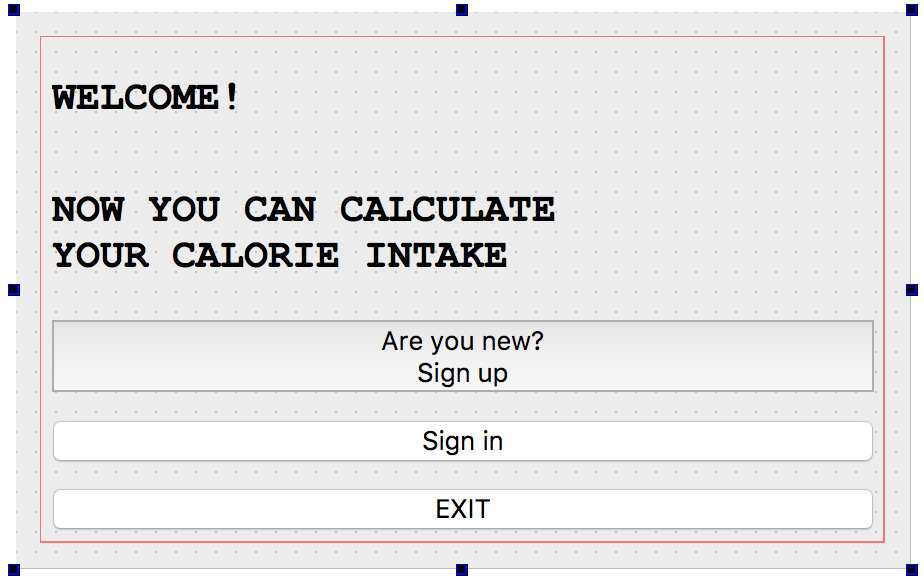


Рис.6. Форма MainWindow.ui

**Форма 2:** Signin.ui

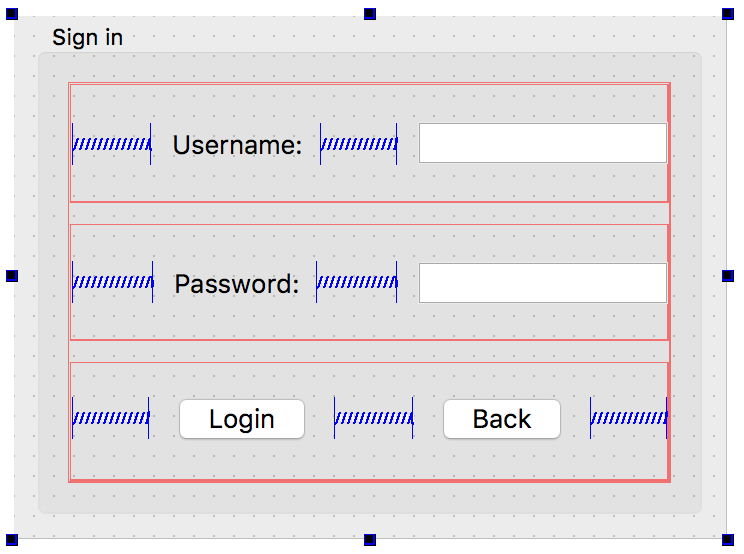


Рис.7. Форма Signin.ui

**Форма 3:** Registration.ui

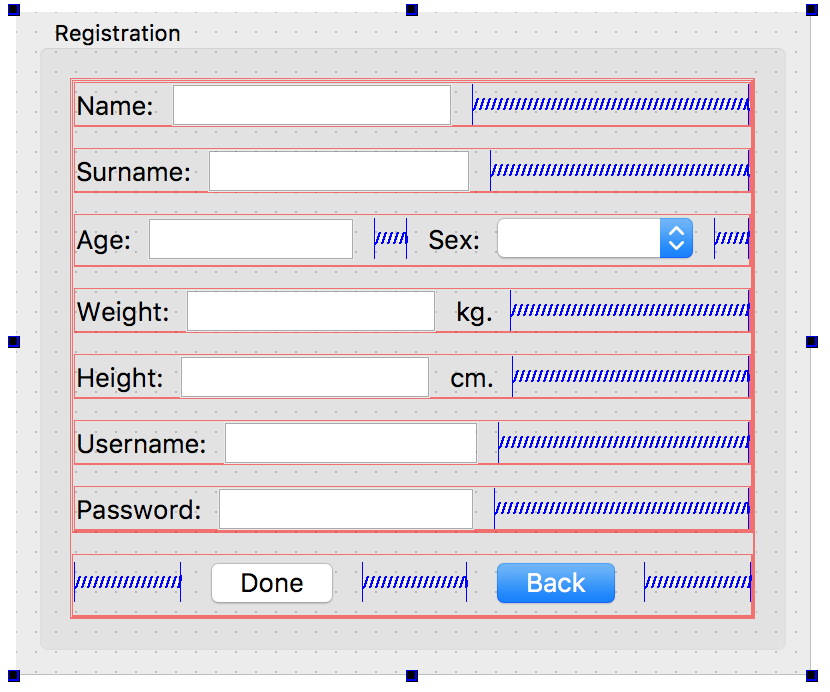


Рис.8. Форма Registration.ui

**Форма 4:** Calculate.ui

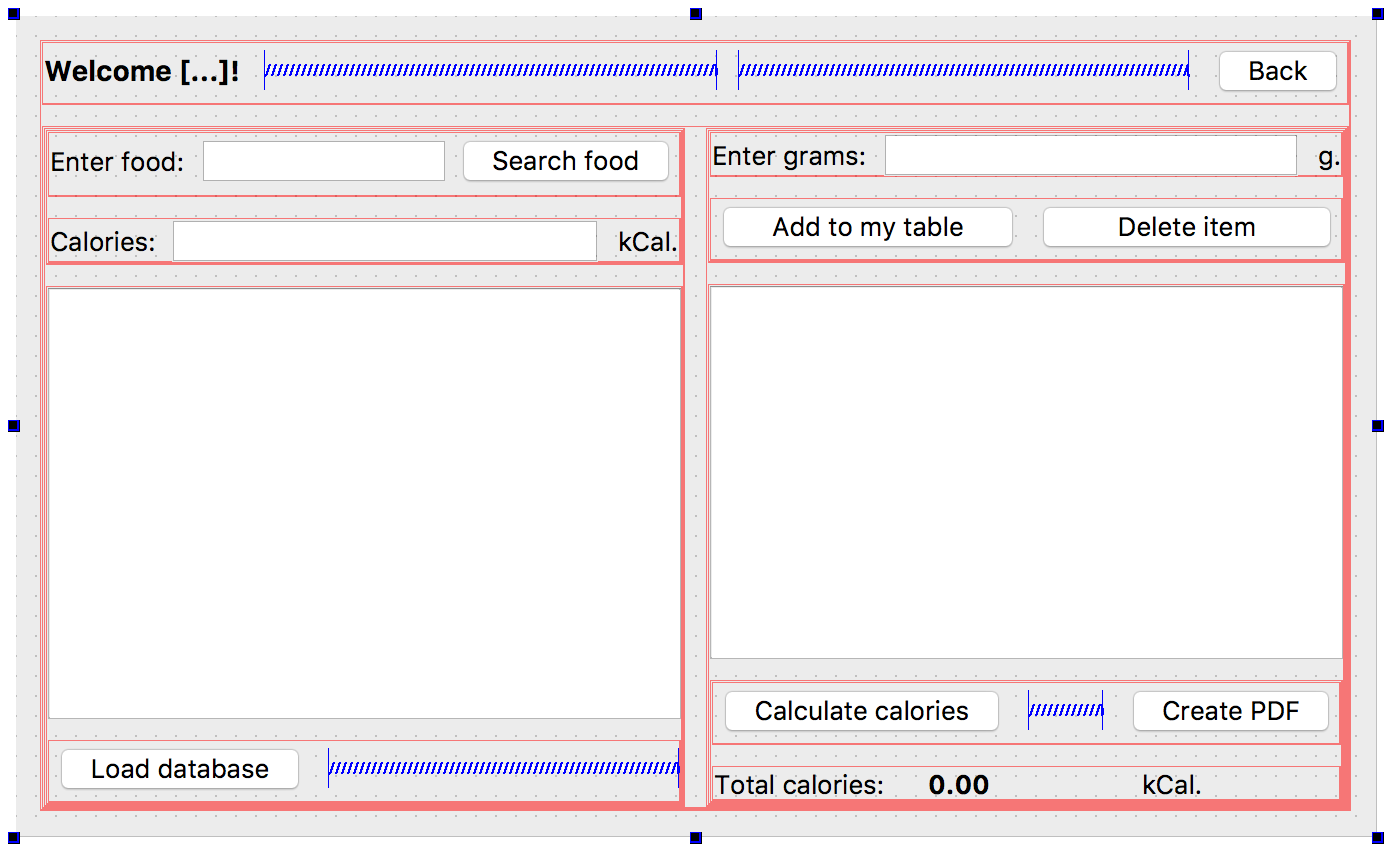


Рис.9. Форма Calculate.ui

## Приложение № 5:



Рис.10. Пример PDF-файла с результатами.

1. https://www.sqlite.org/download.html [↑](#footnote-ref-1)
2. https://www.sqlite.org/docs.html, https://www.tutorialspoint.com/sqlite/index.htm [↑](#footnote-ref-2)
3. https://www.usda.gov/ - https://www.ars.usda.gov/northeast-area/beltsville-md/beltsville-human-nutrition-research-center/nutrient-data-laboratory/docs/sr28-download-files/ [↑](#footnote-ref-3)
4. https://www.ars.usda.gov/northeast-area/beltsville-md-bhnrc/beltsville-human-nutrition-research-center/nutrient-data-laboratory/docs/usda-national-nutrient-database-for-standard-reference/ [↑](#footnote-ref-4)