|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

**ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ**

|  |  |
| --- | --- |
| Студент | Люляев Иван Александрович |
|  |  |

|  |  |
| --- | --- |
| Группа | ИУ6-22Б |

|  |  |
| --- | --- |
| Тип практики | Проектно-технологическая практика |

|  |  |
| --- | --- |
| Название  предприятия | НУК ИУ МГТУ им. Н.Э. Баумана |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент |  | 12.03.2020 | И.А. Люляев |
|  |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |
| Руководитель практики |  |  |  |
|  |  | (Подпись, дата) | (И.О. Фамилия) |

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2020 г.*

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой \_\_\_ИУ6\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Пролетарский

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

**ЗАДАНИЕ**

**на учебную практику**

по теме \_\_\_\_Проектирование и реализация программного обеспечения с использованием структурного и объектного подходов\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент группы ИУ6-22Б

Люляев Иван Александрович .

Тип практики Проектно-технологическая практика

Название предприятия НУК ИУ МГТУ им. Н.Э. Баумана

***Техническое задание:***

**Задание 3. Создание программной системы c Qt интерфейсом на С++**

Выполнить объектную декомпозицию, разработать формы интерфейса, диаграмму состояний интерфейса, диаграммы классов интерфейсной и предметной областей, диаграмму последовательности действий одной из реализуемых операций. Разработать, протестировать и отладить программу в среде Visual Studio или QT Creator.

База данных продуктового магазина содержит сведения о товарах: номер партии, наименование товара, количество (штук), дата поставки, срок годности (в днях). Программа должна в интерактивном режиме формировать файл, добавлять и удалять данные, а также воспринимать каждый из перечисленных запросов и давать на него ответ.

1. Показать все сведения о товарах, поступивших с указанной даты.

2. Определить наименования товаров, срок годности которых истекает сегодня.

3. Определить самый скоропортящийся товар (если их несколько, показать все).

4. Построить график изменения объема поступлений заданного товара по датам.

**Код:**

**main.cpp**

#include "mainwindow.h"

#include <QApplication>

int **main**(int argc, char \*argv[])

{

QApplication a(*argc*, argv);

MainWindow w;

w.show();

return a.exec();}

**mainwindow.h**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QDebug>

#include <fstream>

#include <unistd.h>

#include "chart.h"

QT\_BEGIN\_NAMESPACE

namespace **Ui** { class **MainWindow**; }

QT\_END\_NAMESPACE

class **MainWindow** : public QMainWindow

{

Q\_OBJECT

public:

**MainWindow**(QWidget \*parent = nullptr);

~***MainWindow***();

Ui::MainWindow \*ui;

private:

void **printToTable**(product& buf);

Chart \*chart;

int itemsCount;

// const std::string fileName = "data.bin";

const std::string fileName = "/Users/Ivan/QT/QTProjects/PR3/data.bin";

private slots:

void **showChart**();

void **createFile**();

void **addProduct**();

void **showAll**();

void **on\_btnDeleteAll\_clicked**();

void **on\_btnDeleteProduct\_clicked**();

void **on\_pushButton\_2\_clicked**();

void **on\_btnSearchByDate\_clicked**();

void **on\_btnSearchExpiring\_clicked**();

};

#endif // MAINWINDOW\_H

**mainwindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::**MainWindow**(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

ui->table->setColumnCount(7);

QStringList labels;

labels << "Партия" << "Товар" << "Количество" << "Срок годности" << "Год" << "Месяц" << "День";

ui->table->setHorizontalHeaderLabels(labels);

connect(ui->showChart, SIGNAL(clicked()), this, SLOT(showChart()));

connect(ui->btnCreateFile, SIGNAL(clicked()), this, SLOT(createFile()));

connect(ui->btnAddProduct, SIGNAL(clicked()), this, SLOT(addProduct()));

connect(ui->btnShow, SIGNAL(clicked()), this, SLOT(showAll()));

setWindowTitle("Grocery's database");

}

MainWindow::~***MainWindow***()

{

delete ui;

}

void MainWindow::**printToTable**(product &buf)

{

ui->table->insertRow(0);

ui->table->setItem(0, 0, new QTableWidgetItem(QString::number(buf.number)));

ui->table->setItem(0, 1, new QTableWidgetItem(buf.name));

ui->table->setItem(0, 2, new QTableWidgetItem(QString::number(buf.quantity)));

ui->table->setItem(0, 3, new QTableWidgetItem(QString::number(buf.expDays)));

ui->table->setItem(0, 4, new QTableWidgetItem(QString::number(buf.date.year)));

ui->table->setItem(0, 5, new QTableWidgetItem(QString::number(buf.date.month)));

ui->table->setItem(0, 6, new QTableWidgetItem(QString::number(buf.date.day)));

}

void MainWindow::**showChart**()

{

chart = new Chart();

chart->setFileName(fileName);

chart->setProduct(ui->lineEditName->text());

chart->show();

}

void MainWindow::**createFile**()

{

std::ofstream file; // output

file.open(fileName, std::ios::binary);

if (file.is\_open()) {

qDebug() << "File created.";

file.close();

itemsCount = 0;

} else {

qDebug() << "Could not create file";

}

itemsCount = 0;

}

void MainWindow::**addProduct**()

{

product buf;

buf.number = ui->lineEditNum->text().toInt();

strcpy(buf.name, ui->lineEditName->text().toUtf8());

buf.quantity = ui->lineEditQuantity->text().toInt();

buf.expDays = ui->lineEditExpDays->text().toInt();

buf.date.year = ui->lineEditYear->text().toInt();

buf.date.month = ui->lineEditMonth->text().toInt();

buf.date.day = ui->lineEditDays->text().toInt();

std::fstream file;

file.open(fileName, std::fstream::out | std::fstream::in | std::fstream::ate | std::fstream::binary);

if (file.is\_open()) {

qDebug() << "File opened.";

// file.seekp(0,std::ios::end);

file.write(reinterpret\_cast<char\*>(&buf), sizeof(product));

file.close();

} else {

qDebug() << "Could not open file";

}

this->showAll();

itemsCount++;

}

void MainWindow::**showAll**()

{

product buf = {};

std::fstream file;

file.open(fileName, std::fstream::out | std::fstream::in | std::fstream::binary);

if (file.is\_open()) {

qDebug() << "File opened.";

ui->table->setRowCount(0);

while(file) {

file.read(reinterpret\_cast<char\*>(&buf), sizeof(product));

if(!file) break;

printToTable(*buf*);

}

file.close();

} else {

qDebug() << "Could not open file";

}

}

void MainWindow::**on\_btnDeleteAll\_clicked**()

{

std::fstream file;

file.open(fileName, std::ios::out | std::ios::binary);

if (file.is\_open()) {

qDebug() << "File created.";

file.close();

} else {

qDebug() << "Could not create file";

}

this->showAll();

itemsCount = 0;

}

void MainWindow::**on\_btnDeleteProduct\_clicked**()

{

FILE \*file;

file = fopen("/Users/Ivan/QT/QTProjects/PR3/data.bin", "a+");

fseek(file, 0, SEEK\_END);

long pos = ftell(file);

int fd = fileno(file);

ftruncate(fd, pos-sizeof(product));

fclose(file);

this->showAll();

itemsCount--;

}

// кнопка "Скоропортящиеся товары"

void MainWindow::**on\_pushButton\_2\_clicked**()

{

product buf = {};

std::fstream file;

file.open(fileName, std::fstream::out | std::fstream::in | std::fstream::binary);

if (file.is\_open()) {

qDebug() << "File opened.";

int minExpDay = INT\_MAX;

// ищем наименьший срок годности

while(file) {

file.read(reinterpret\_cast<char\*>(&buf), sizeof(product));

if (buf.expDays < minExpDay) minExpDay = buf.expDays;

}

qDebug() << minExpDay;

file.close();

// reopen file to start from the beginning

file.open(fileName, std::fstream::out | std::fstream::in | std::fstream::binary);

ui->table->setRowCount(0); // убираем предыдущие строки таблицы

if (file.is\_open()) {

while(file) {

file.read(reinterpret\_cast<char\*>(&buf), sizeof(product));

if (!file) break;

// если у продукта минимальный срок годности среди продуктов, то выводим его в таблицу

if (buf.expDays == minExpDay) {

printToTable(*buf*);

}

}

} else {

qDebug() << "Could not open file";

}

file.close();

} else {

qDebug() << "Could not open file";

}

}

void MainWindow::**on\_btnSearchByDate\_clicked**()

{

product buf = {};

// получаем указанную дату с формы

date inputDate;

inputDate.day = ui->lineEditDays->text().toInt();

inputDate.month = ui->lineEditMonth->text().toInt();

inputDate.year = ui->lineEditYear->text().toInt();

// открываем файл и ищем подходящие товары

std::fstream file;

file.open(fileName, std::fstream::out | std::fstream::in | std::fstream::binary);

if (file.is\_open()) {

qDebug() << "File opened.";

ui->table->setRowCount(0); // убираем предыдущие строки таблицы

while(file) {

file.read(reinterpret\_cast<char\*>(&buf), sizeof(product));

if (!file) break;

if (buf.date.year > inputDate.year) {

printToTable(*buf*);

} else if (buf.date.year == inputDate.year) {

if (buf.date.month > inputDate.month) {

printToTable(*buf*);

} else if (buf.date.month == inputDate.month) {

if (buf.date.day >= inputDate.day) {

printToTable(*buf*);

}

}

}

}

file.close();

} else {

qDebug() << "Could not open file";

}

}

// кнопка "Товары с истекающим сроком годности" (истекает сегодня - today)

void MainWindow::**on\_btnSearchExpiring\_clicked**()

{

product buf = {};

const date today = {9, 4, 2020};

std::fstream file;

file.open(fileName, std::fstream::out | std::fstream::in | std::fstream::binary);

if (file.is\_open()) {

qDebug() << "File opened.";

ui->table->setRowCount(0); // убираем предыдущие строки таблицы

date exp; // дата, когда срок годности истечет, формируемая для каждого товара

while(file) {

file.read(reinterpret\_cast<char\*>(&buf), sizeof(product));

if (!file) break;

// определение даты, когда срок годности товара истечет

exp.year = buf.date.year;

exp.month = buf.date.month;

if ((buf.date.day + buf.expDays) > 31) {

exp.month = exp.month + ((buf.date.day + buf.expDays) / 31);

if (exp.month > 12) {

exp.year = exp.year + (exp.month / 12);

exp.month = exp.month % 12;

}

exp.day = (buf.date.day + buf.expDays) % 31;

} else {

exp.day = (buf.date.day + buf.expDays);

}

// проверка является ли эта дата сегоднешней, если да, то выводим в таблицу

if ((exp.year == today.year) and (exp.month == today.month) and (exp.day == today.day)) {

printToTable(*buf*);

}

}

file.close();

} else {

qDebug() << "Could not open file";

}

}

**chart.h**

#ifndef CHART\_H

#define CHART\_H

#include <QWidget>

#include <QPainter>

#include <fstream>

#include <QDebug>

namespace **Ui** {

class **Chart**;

}

#pragma pack(push, 1)

typedef struct **date** {

int day, month, year;

} date;

#pragma pack(pop)

#pragma pack(push, 1)

typedef struct **product** {

int number, quantity, expDays;

char name[50];

date date;

} product;

#pragma pack(pop)

class **Chart** : public QWidget

{

Q\_OBJECT

public:

explicit **Chart**(QWidget \*parent = nullptr);

~***Chart***();

void **setFileName**(const std::string fileName);

void **setProduct**(const QString name);

protected:

void ***paintEvent***(QPaintEvent \*event); // override();

private:

std::string fileName;

QString name;

};

#endif // CHART\_H

**chart.cpp**

#include "chart.h"

#include <math.h>

#define XMAX 520

#define DELX 63

#define MAX\_AXIS\_X 248

Chart::**Chart**(QWidget \*parent) :

QWidget(parent) {

setFixedSize(590, 285);

setWindowTitle("Chart");

}

Chart::~***Chart***()

{

}

void Chart::**setFileName**(const std::string fileName) {

this->fileName = fileName;

}

void Chart::**setProduct**(const QString name)

{

this->name = name;

}

void Chart::***paintEvent***(QPaintEvent \*event)

{

QPainter\* painter = new QPainter(this);

// рисуем осевые линии

painter->drawLine(31, 12, 25, 6);

painter->drawLine(19, 12, 25, 6);

painter->drawLine(25, 6, 25, 260);

painter->drawLine(25, 260, 535, 260);

painter->drawLine(535, 260, 529, 254);

painter->drawLine(535, 260, 529, 266);

int day = 9; // сегодняшний день

QString s0 = "20/04/";

int dx = DELX; // начальный сдвиг для отрисовки дат

// отрисовка дат на графике

for (int i = 0; i < 7; i++) {

QString s1 = QString::number(day);

painter->drawText(XMAX - dx - 23, 280, (s0 + s1));

painter->drawLine(XMAX - dx, 260, XMAX - dx, 255);

dx += DELX;

day--;

}

// открываем файл и считаем поставки за каждый день

int supplyPerDay[] = {0, 0, 0, 0, 0, 0, 0}; // 7

product buf = {};

std::fstream file;

file.open(fileName, std::fstream::out | std::fstream::in | std::fstream::binary);

if (file.is\_open()) {

qDebug() << "File opened.";

while(file) {

file.read(reinterpret\_cast<char\*>(&buf), sizeof(product));

if(!file) break;

day = 3; // начальный день, с которого начинается отрисовка графика

// strcpy(buf.name, ui->lineEditName->text().toUtf8());

if (buf.name == this->name) {

qDebug() << "Match!";

for (int i = 0; i < 7; i++) {

if ((buf.date.day == day) && (buf.date.month == 4) && (buf.date.year == 2020)) {

supplyPerDay[i] += buf.quantity;

}

day++;

}

}

}

file.close();

} else {

qDebug() << "Could not open file";

}

// находим максимум среди поставок

float maxSupply = -1.0; // !!!!!!

for (int i = 0; i < 7; i++) {

if (supplyPerDay[i] > maxSupply)

maxSupply = supplyPerDay[i];

}

if (maxSupply == 0) maxSupply = 1.0; // защита от ошибки деления на ноль

// вывод линии графика

painter->drawText(1, 260, "O");

dx = 0;

int dy = 0;

int x, y;

QString quantity;

int prev\_x = 25;

int prev\_y = 260;

for (int i = 0; i < 7; i++) {

dx += DELX;

dy = (int)(trunc((MAX\_AXIS\_X\*(supplyPerDay[i]/maxSupply))));

x = 15+dx;

y = 260-dy;

painter->drawLine(prev\_x, prev\_y, x, y);

QPen pen;

pen.setWidth(3);

painter->setPen(pen);

painter->drawEllipse(x-1, y, 3, 3);

pen.setWidth(1);

painter->setPen(pen);

prev\_x = x;

prev\_y = y;

quantity = QString::number(supplyPerDay[i]);

painter->drawText(1, y, quantity);

}

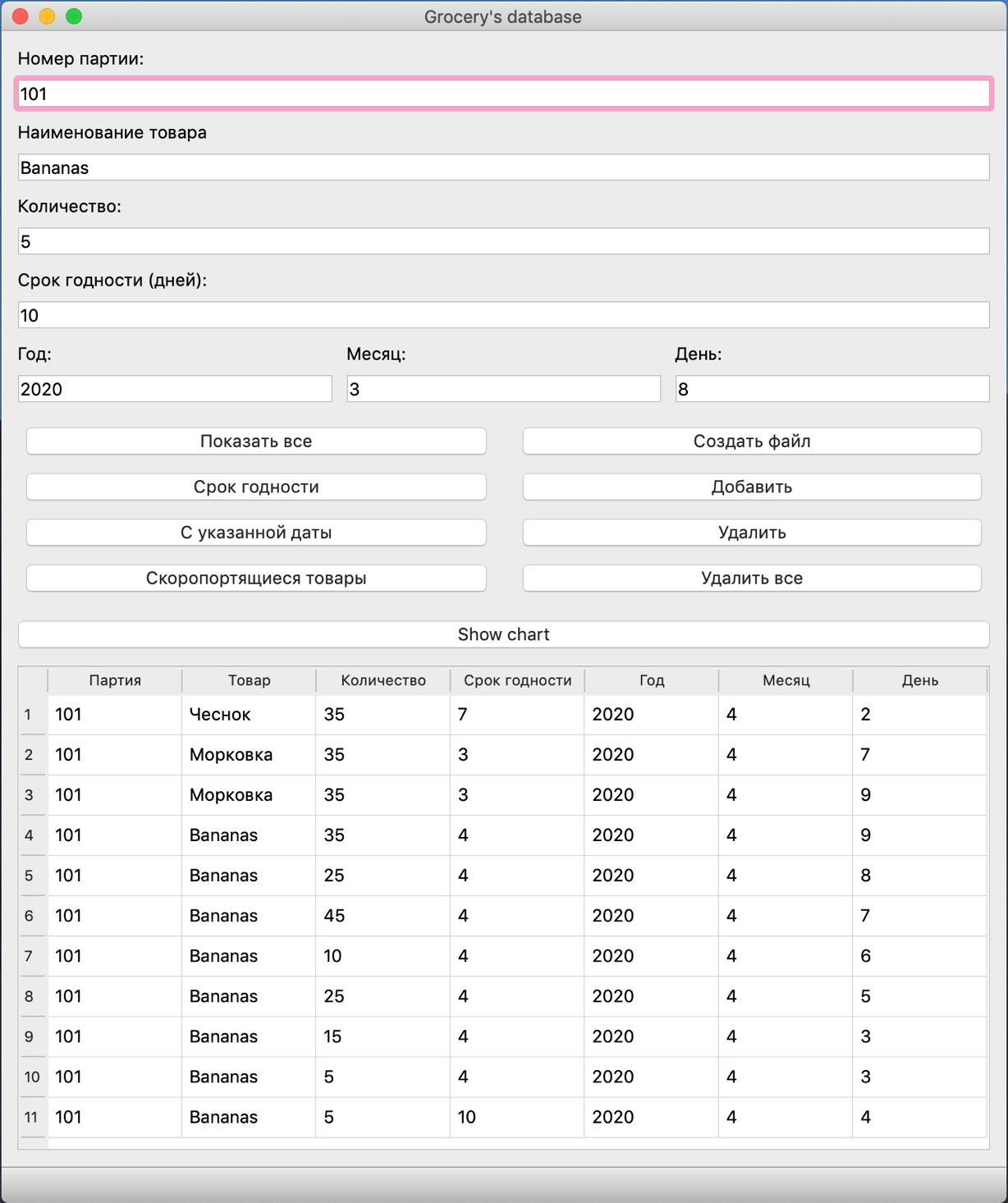
painter->drawText(35, 15, "Количество");

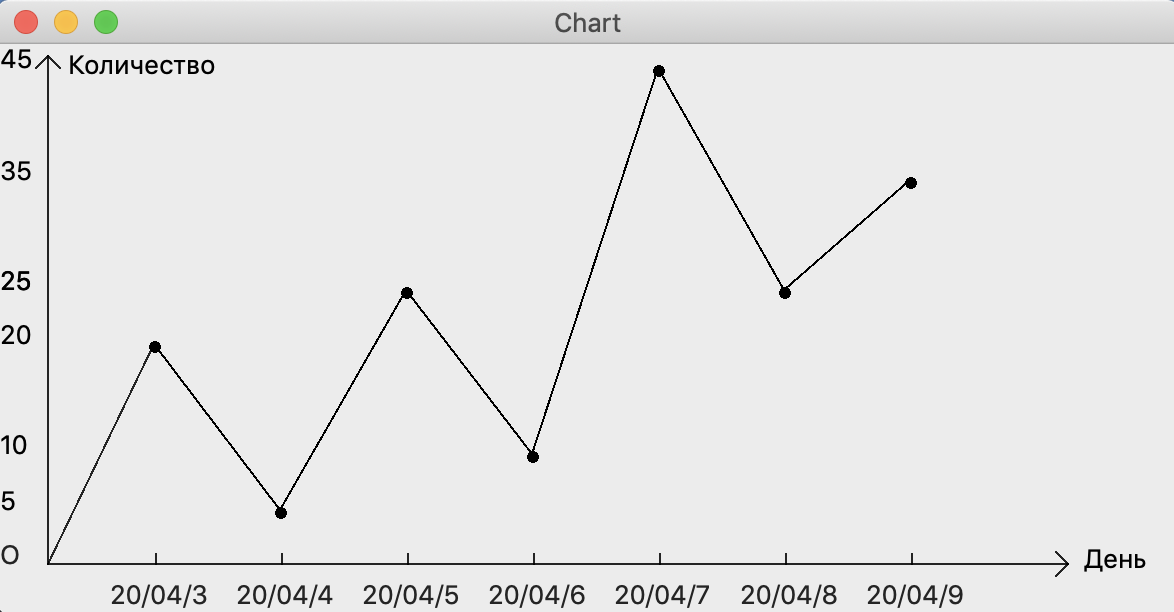
painter->drawText(543, 262, "День");

painter->end();

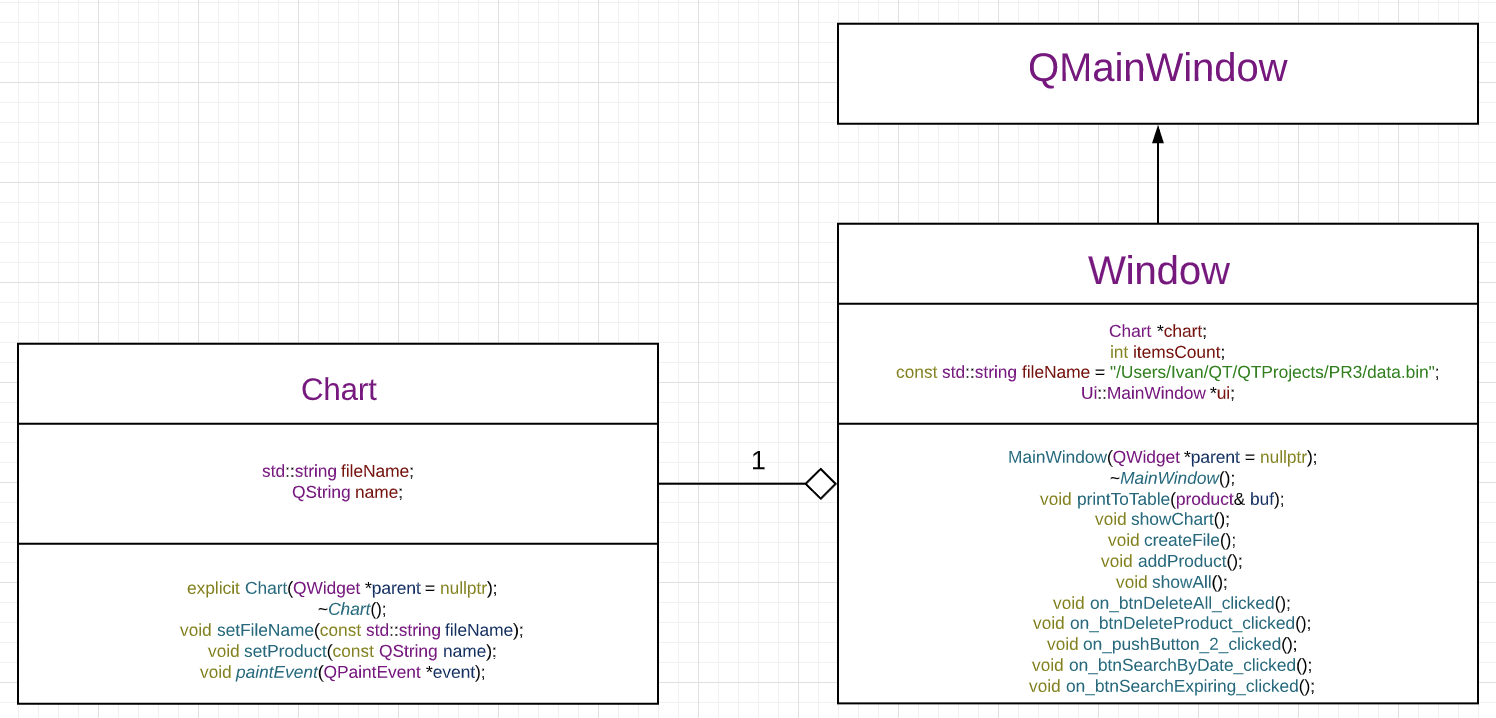
}

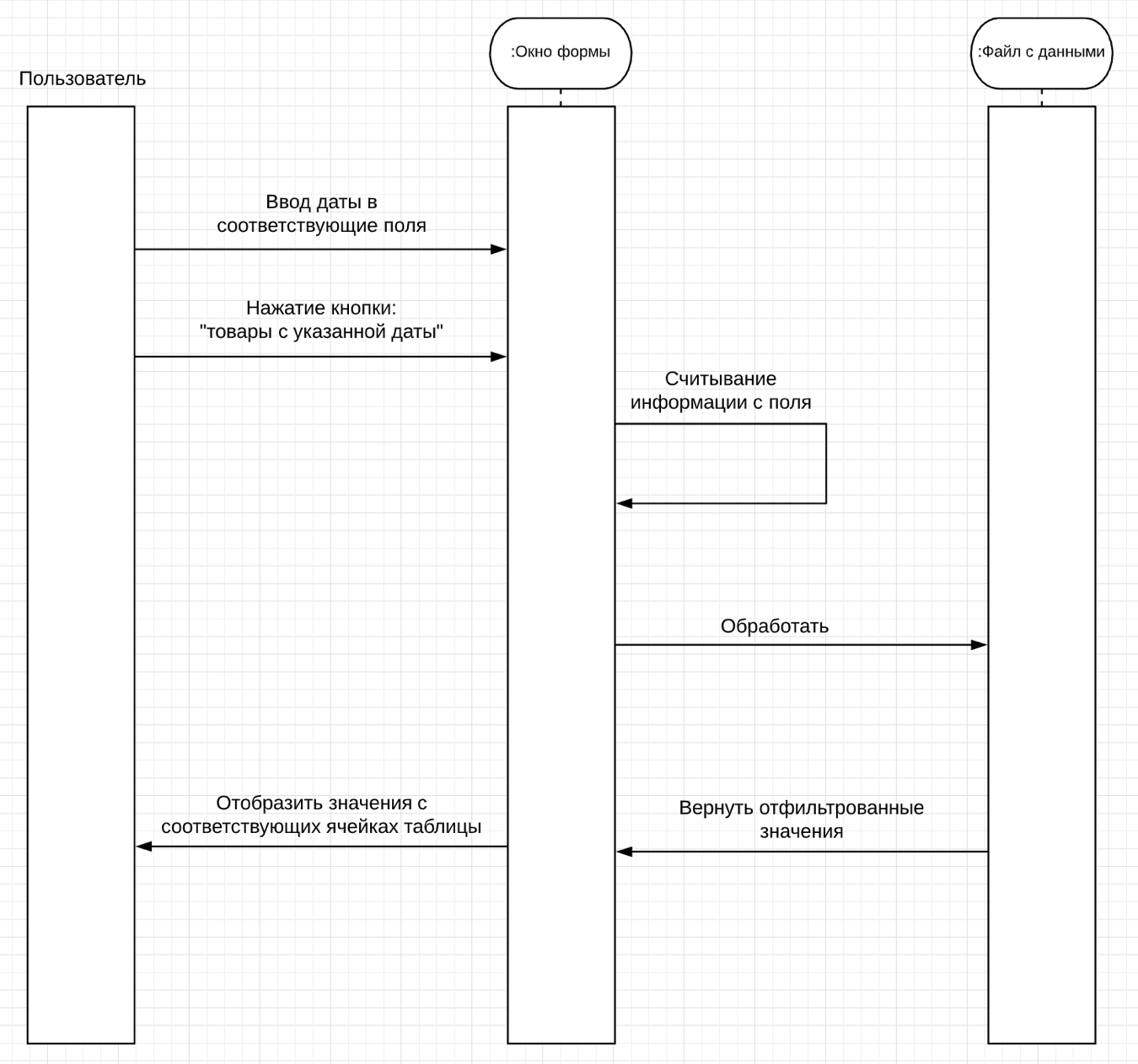
**Интерфейс:**

****

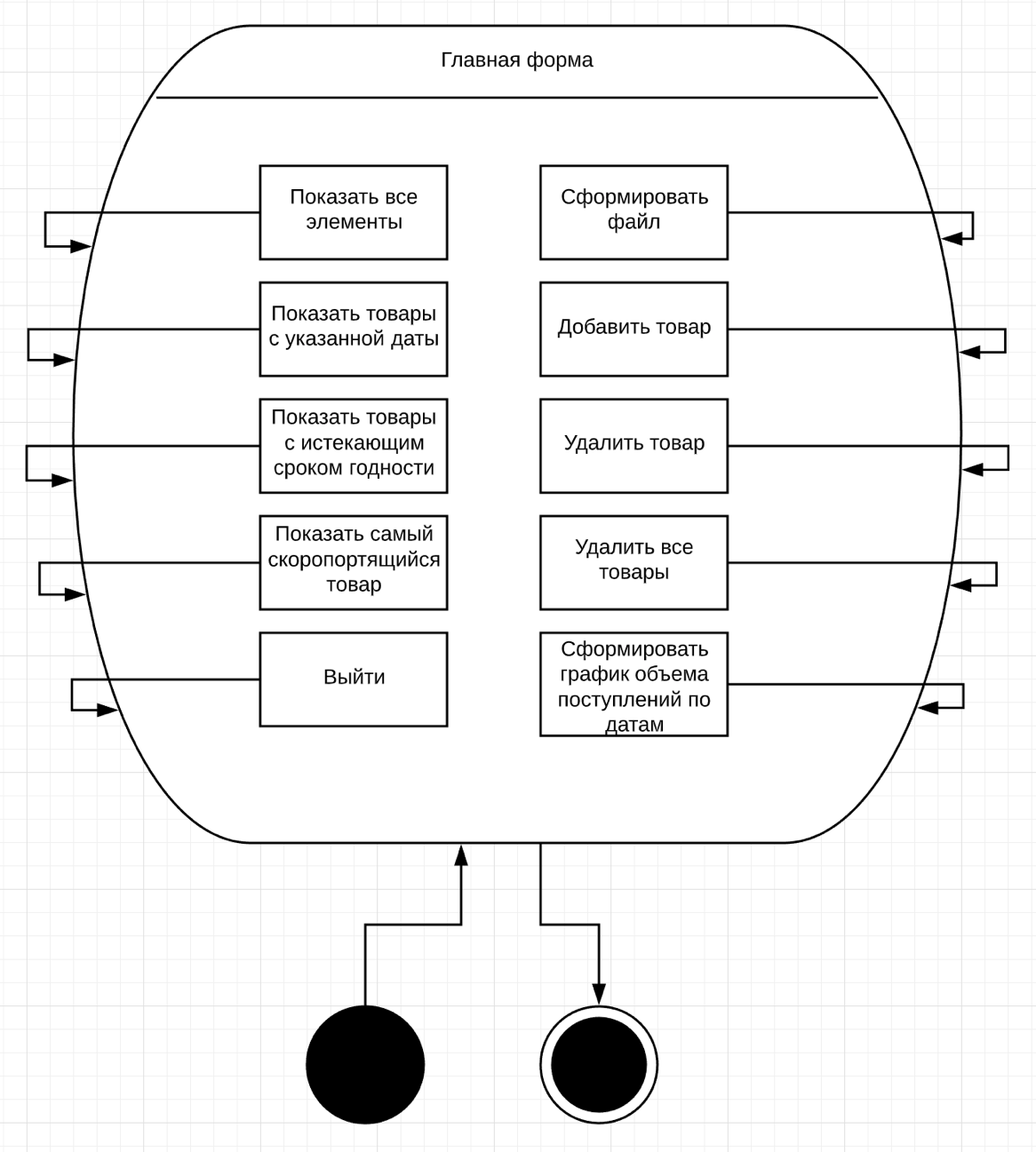
****

**Диаграмма классов:**

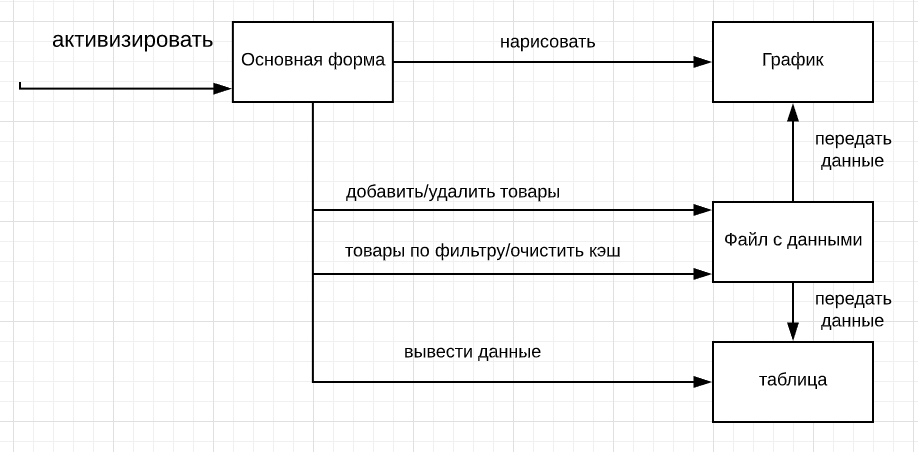


**Диаграмма последовательности определения наименования товаров, срок годности которых истекает сегодня: **

**Диаграмма состояний интерфейса:**

****

**Объектная декомпозиция:**

****

**Вывод:**

Среда разработки QT позволяет нам реализовывать событийное программирование при работе с формами, мы можем подключать различные классы, в т.ч. реализующие построение таблиц и вручную рисовать адаптивные графики для наглядного отображения нужных данных из файла с помощью QPainter. Но, по сравнению с Delphi pascal, имеет расширенное количество возможностей.