**Министерство цифрового развития, связи и массовых коммуникаций РФ**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Сибирский государственный университет телекоммуникаций и информатики»**

**(СибГУТИ)**

КУРСОВАЯ РАБОТА

Вариант 6

«Магазин оргтехники»

Выполнил студент группы ИП-115 Шпаков Е.С.

Проверил доцент кафедры ПМиК к.т.н Мерзлякова Е.Ю.

Новосибирск \_\_\_\_\_\_г.

Оглавление

[Введение 3](#_Toc154414061)

[1. Анализ задач и пользователей 4](#_Toc154414062)

[Выбор репрезентативных задач 5](#_Toc154414063)

[Заимствование 7](#_Toc154414064)

[Черновое описание дизайна 8](#_Toc154414065)

[Описание программы 12](#_Toc154414066)

[Скриншоты работы программы 13](#_Toc154414067)

[CWT-анализ 18](#_Toc154414068)

[GOMS-анализ 21](#_Toc154414069)

[Используемые источники 24](#_Toc154414070)

[Листинг 25](#_Toc154414071)

# Введение

В современном мире оргтехника играет важную роль в организации рабочего процесса в различных сферах деятельности. Многочисленные офисы, предприятия и учреждения нуждаются в качественной и надежной технике для эффективного выполнения рабочих задач.

Однако, несмотря на обилие товаров на рынке оргтехники, выбор и приобретение подходящего оборудования может быть сложной задачей. Клиенты сталкиваются с такими вопросами, как определение своих потребностей, сравнение характеристик различных моделей, а также выбор надежного поставщика.

Целью данной курсовой работы является разработка программного решения в виде интернет-магазина оргтехники, которое поможет клиентам упростить процесс выбора и покупки необходимого оборудования. Реализация данного проекта позволит автоматизировать основные процессы взаимодействия между клиентами и магазином, а также повысить эффективность работы самого магазина.

# 1. Анализ задач и пользователей

**Цель:**

Найти двух человек, которые могут быть заинтересованы в решении предложенной задачи. Дайте их краткое описание (возраст, образование, профессия, навыки в выбранной сфере, навыки владения компьютером). Люди должны быть заинтересованы в решении предложенной задачи.

**Решение:**

Магазин оргтехники может понадобится любому: взрослому человеку, подростку, компании, которая нуждается в оптовой закупке техники.

В качестве заинтересованных лиц были выбраны 2 следующих человека:

* Папа: возраст – 46 лет, образование – высшее техническое, профессия – инженер-электрик, навыки в выбранной сфере – иногда покупает новую оргтехнику для дома/офиса, навыки владения компьютером – средние.
* Брат: возраст – 18 лет, образование – среднее общее, профессия – не работает. Постоянной заглядывает на сайты по продаже различной электроники, навыки владения компьютером – высокие.

# Выбор репрезентативных задач

**Цель:**

Определение ключевых функциональных задач и второстепенных действий для обеспечения эффективного использования разрабатываемой программы.

**Решение:**

Репрезентативные задачи:

1. Просмотр записей:

* Разработка интуитивного интерфейса для удобного просмотра подробной информации о товарах, представленных магазином.

1. Использование фильтров:

* Реализация удобного интерфейса, позволяющего пользователям легко находить нужные товары с использованием фильтров.

1. Поиск по записям:

* Предоставление пользователю удобных средств для поиска по имеющимся записям.

1. Добавление корзины:

* Создание интуитивного интерфейса для безопасного и простого просмотра выбранных товаров.

1. Обновление базы данных записей:

* Разработка системы обновление данных в базе данных, обеспечивая их свежесть во время работы с приложением.

Второстепенные задачи:

1. Выход из приложения:

* Обеспечение простого и безопасного завершения работы с приложением.

Такое четкое выделение репрезентативных и второстепенных задач поможет фокусироваться на ключевых аспектах функционала и обеспечит эффективное использование программы.

# Заимствование

**Задача:**

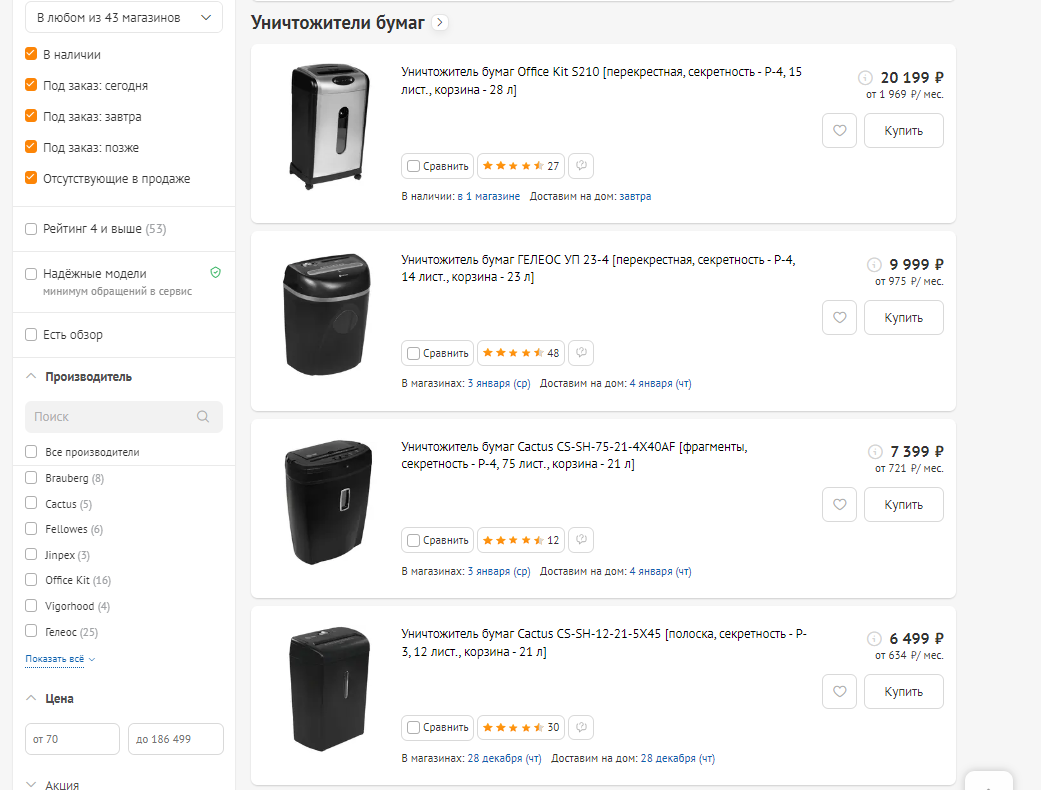
Найти приложения или сайты, с которых можно заимствовать какие-либо решения интерфейса, приведите ссылку на источник. Для обогащения пользовательского интерфейса разрабатываемого приложения, рассматривается возможность вдохновения и заимствования элементов дизайна из существующих приложений и веб-сайтов. Выберите и напишите, что именно Вы будете заимствовать из данных приложений и зачем.

**Решение:**

1. **Онлайн магазин DNS**

С данного сервиса была заимствована идея организации просмотренных фильмов в виде списка, а также дата просмотра и оценка.

Также с данного ресурса взято описание товаров, которые будут заранее находиться в базе данных при первом запуске приложения.

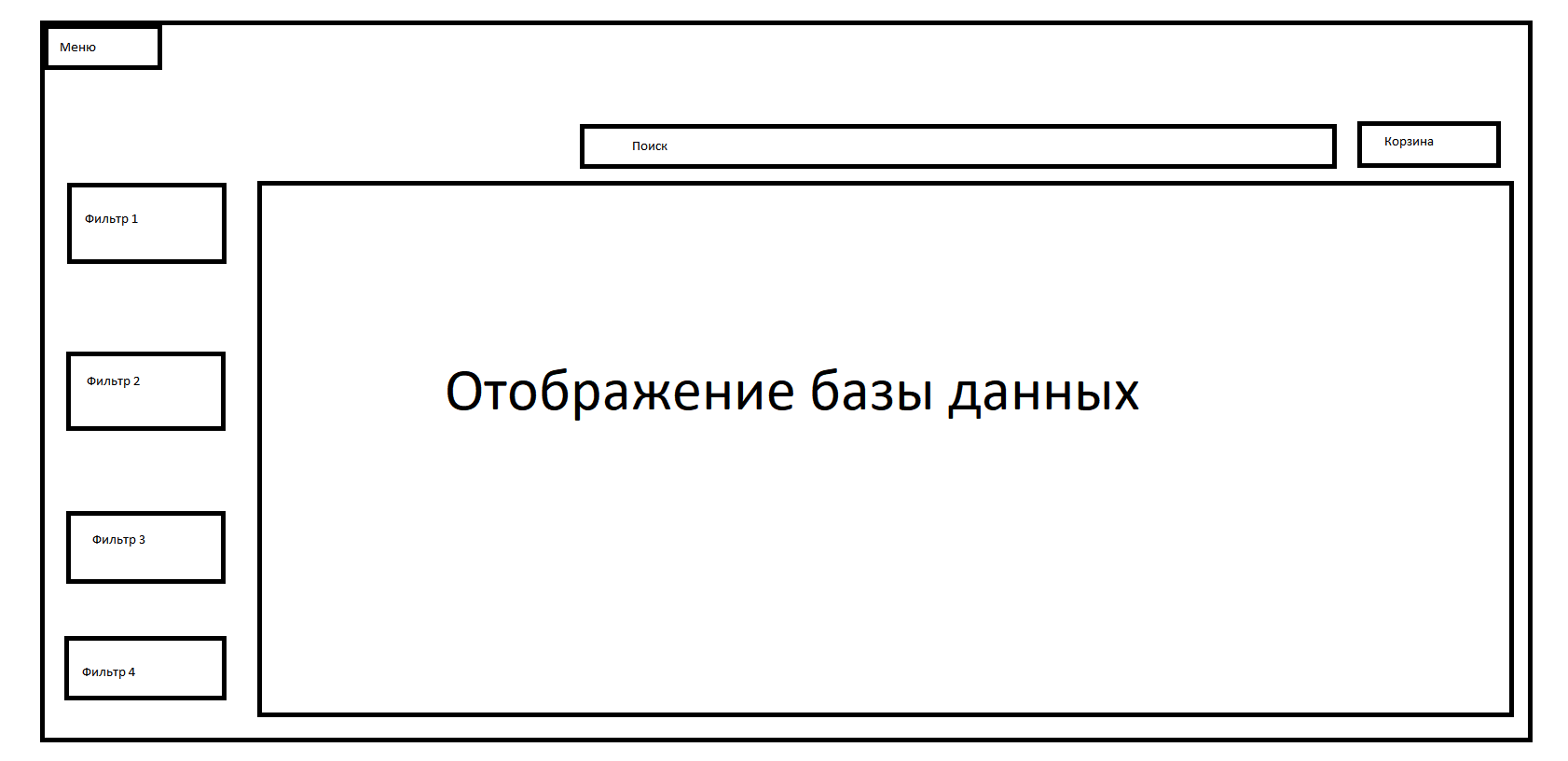


# Черновое описание дизайна

**Задача:**

Описать черновой вариант дизайна словами и графически (иллюстрации с пояснениями). Черновой вариант должен отражать все внешние элементы интерфейса и их назначение.

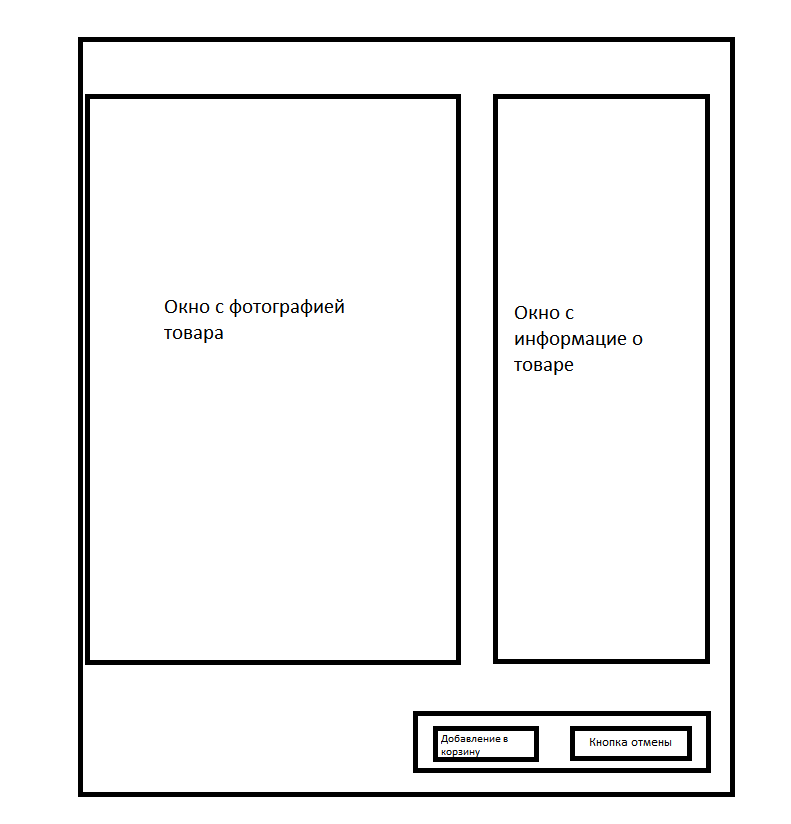
**Решение:**



(рис. 1) Главное окно приложения

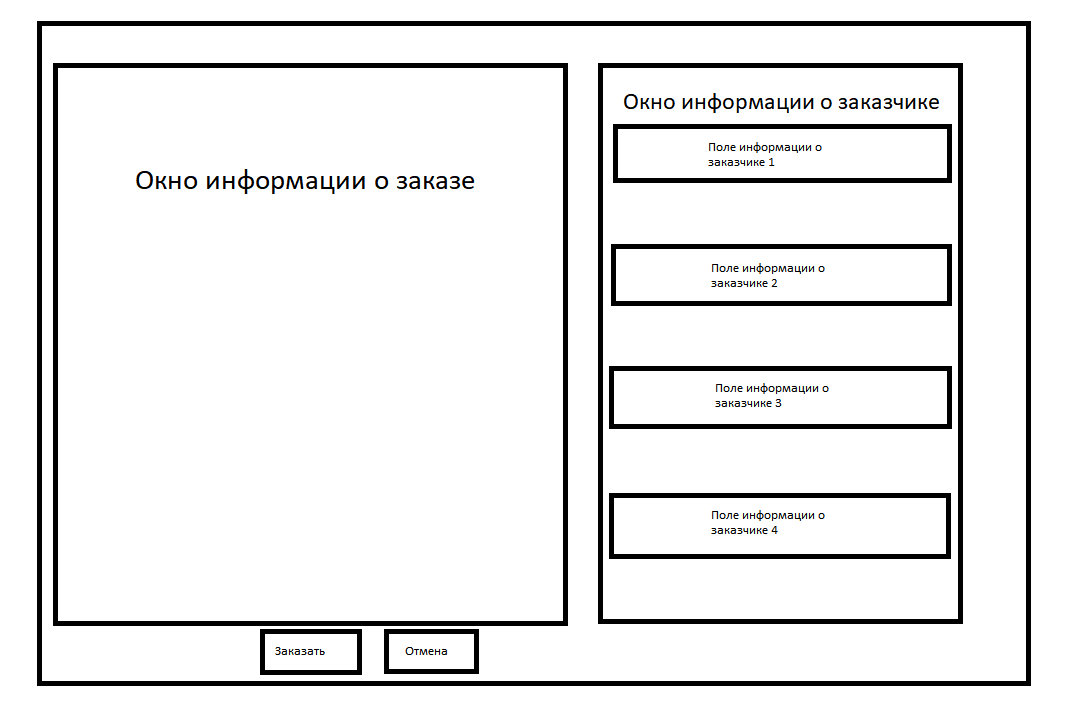
В данном приложении будет производиться отображение базы данных. При нажатии на кнопку одно из фильтров должно изменяться отображение базы данных в зависимости от фильтра. При нажатии на кнопку «Меню» открывается кнопка «Выход» по нажатию на которую происходит выход из приложения. При нажатии на строку в которой написано «Поиск» должна быть возможность вводить информацию для поиска и при нажатии клавиши «Enter» производиться поиск, то есть изменения в отображении базы данных.

По двойному нажатию на поле базы данных, должно открываться окно с расширенной информацией про товар для добавления товара в корзину.



(рис. 2) Окно товара

При открытии товара пользователь может просмотреть информацию о товаре, а также выбрать количество товара, при нажатии на кнопку «Добавление в корзину» товар должен поместиться в корзину с измененным количеством. Кроме того, желаемое количество товара не должно превышать доступное значение. При нажатии на кнопку «Отмена» должно закрывать окно и при этом товар никуда не добавляется.



(рис. 3) Окно корзины

Окно корзины позволяет просмотреть товары уже добавленные в корзину в окне «Информации о заказе». Так же позволяет добавить информацию о себе, в том случае если заказчик больше не будет добавлять товар в заказ. По кнопке «Отмена» будет происходить выход из этого окна и «Информация о заказчике» будет стираться. При нажатии на кнопку «Заказать» данные из этого окна будет помещены в таблицу заказов и на экране появится сообщение об успешном заказе.

# Описание программы

Программа "Магазин оргтехники", разработанная на платформе Qt 4.9.1 (Enterprice), предоставляет пользователю обширный функционал:

1. Взаимодействие с товарами:
   1. Просмотр подробной информации о товарах.
   2. Добавление данных заказов с удобным интерфейсом.
2. Хранение товаров и заказов в базе данных:

Программа структурирована с использованием трёх ключевых классов, каждый из которых отвечает за определенный функционал:

1. Главный класс (MainWindow):

- Основной контроллер, обеспечивающий взаимодействие и координацию работы остальных классов.

- Управление главным окном приложения и основными действиями пользователя.

2. Класс открытия расширенной информации о товаре (extendedProduct):

- Реализация окна для просмотра данных о товаре.

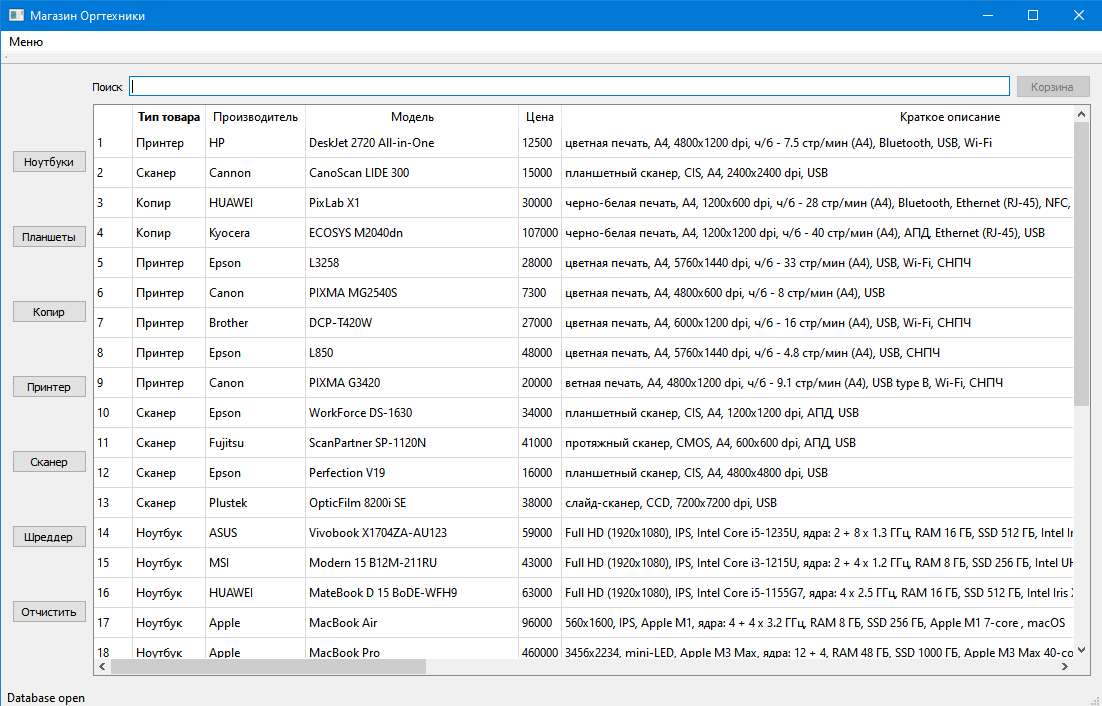
- Предоставление пользователю интерфейса для выбора количества и добавления товара в корзину.

3. Класс корзина (Basket):

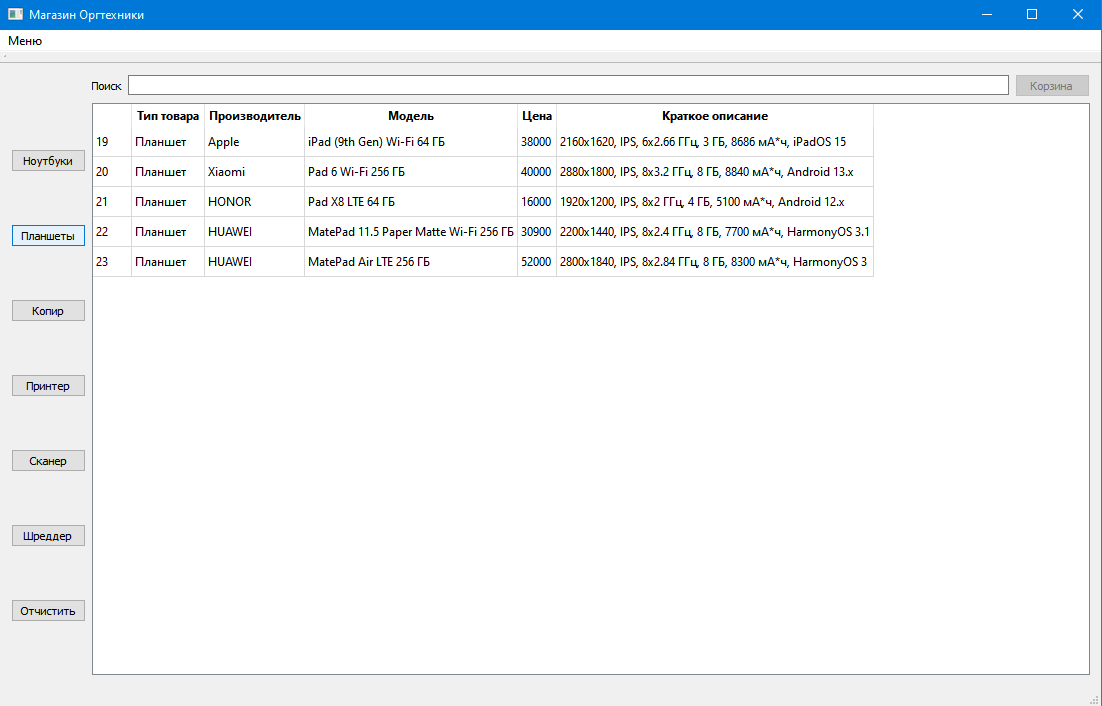
- Разработка окна просмотра выбранных товаров, выбранных пользователем.

- Предоставление возможности заполнения информации о пользователе.

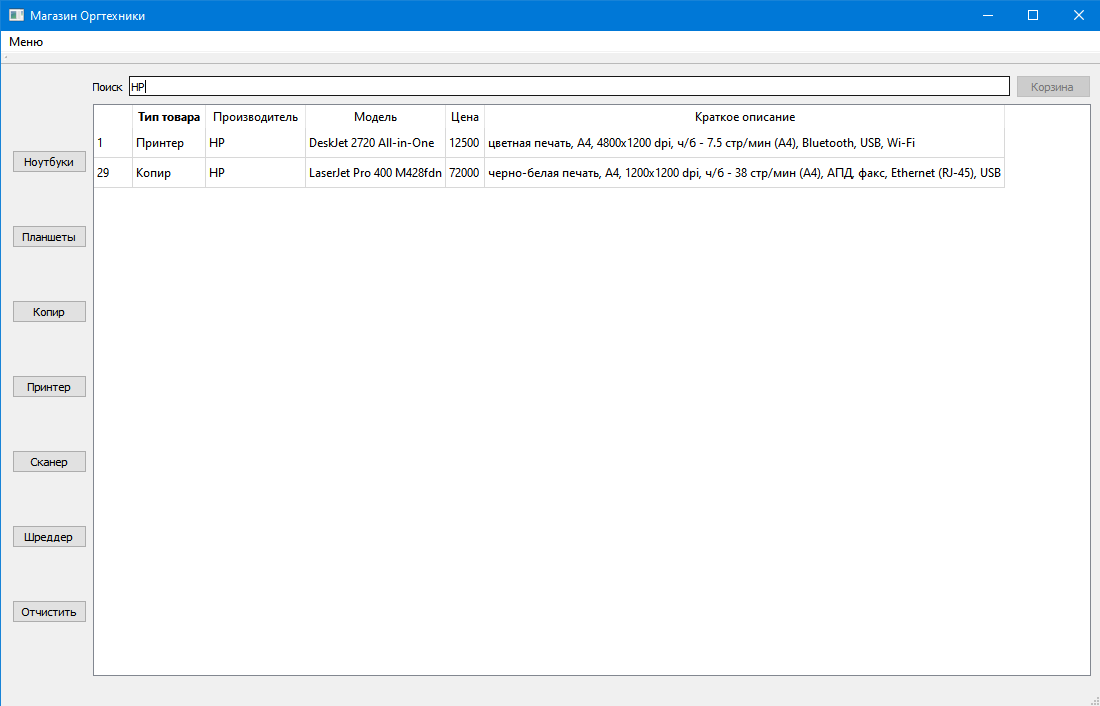
# Скриншоты работы программы



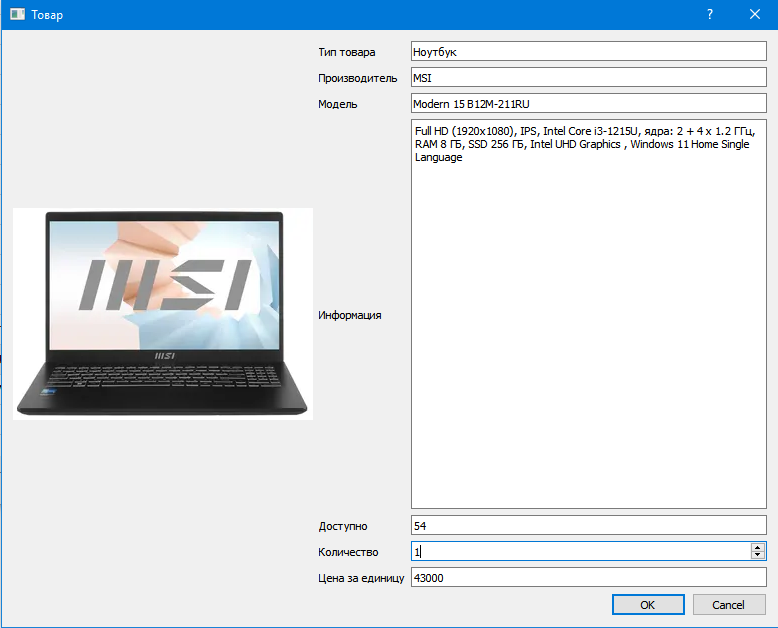
(рис. 4) Основное окно



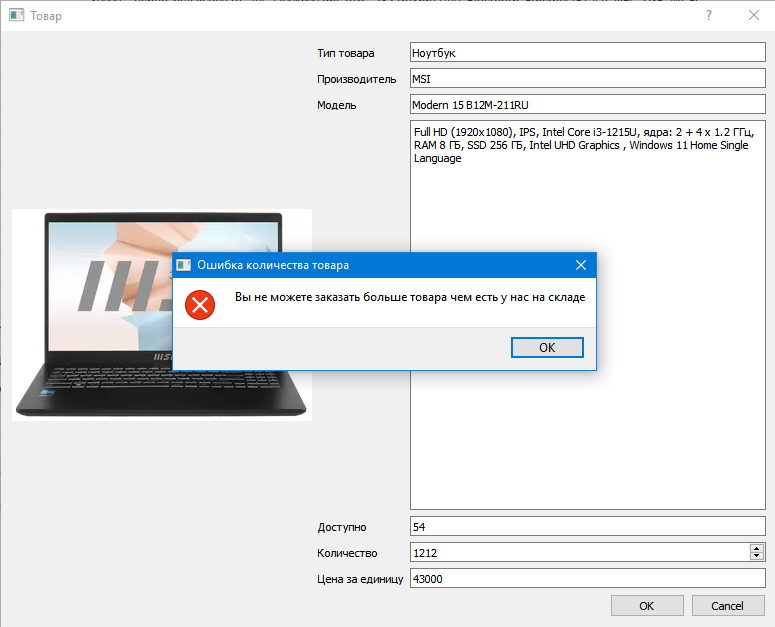
(рис. 5) Работа фильтра «Принтер» в главном окне



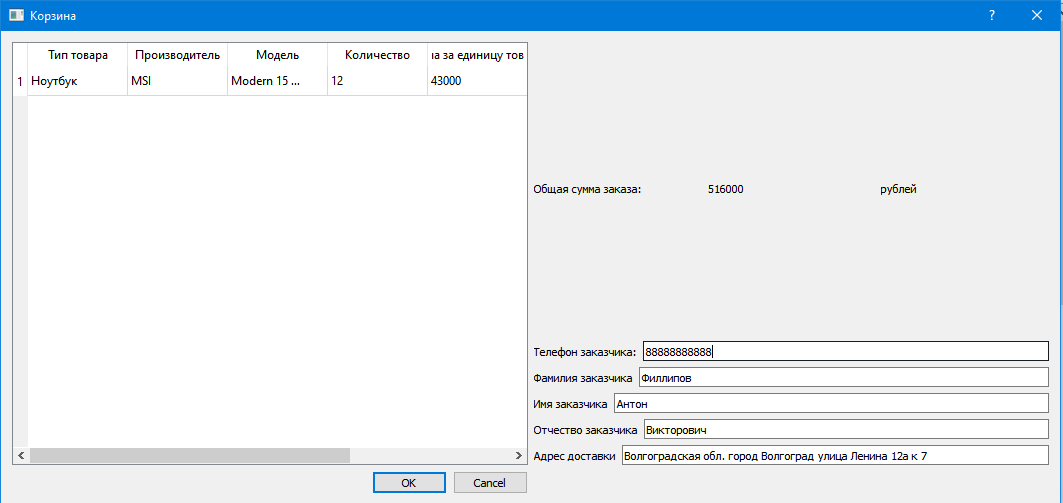
(рис. 6) Работа функции «Поиск» в главном окне



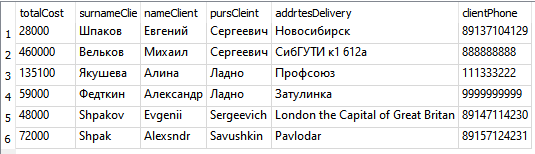
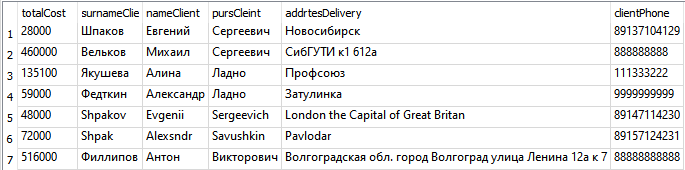
(рис. 7) Окно товара



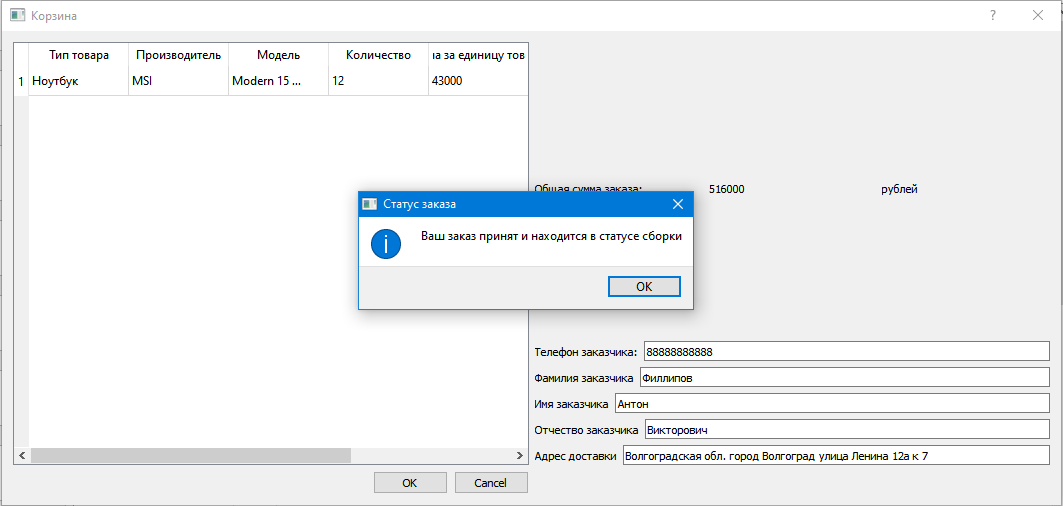
(рис. 8) Ошибка при вводе неверного количества товара



(рис. 9) Заполненное окно «Корзина»

(рис. 10) Сохранение информации о заказах до и после сохранения



(рис. 11) Успешное заверешение работы корзины

# **CWT-анал**из

**Задача №1:**

Просмотр товаров в главном окне.

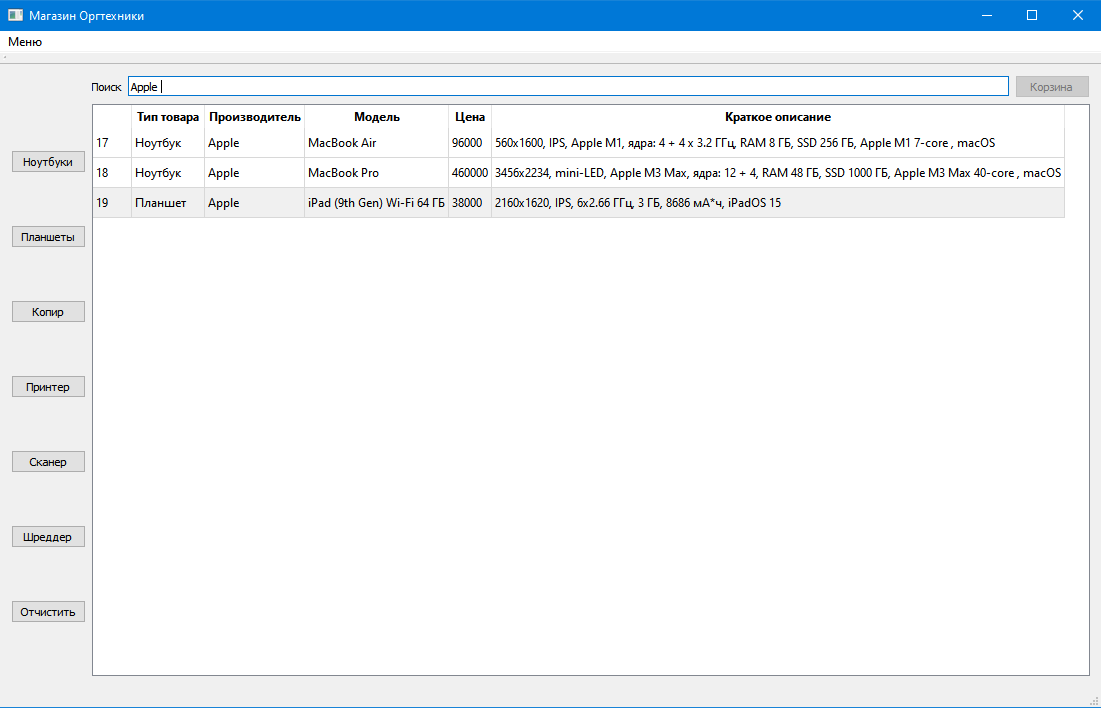
История действий пользователя, впервые столкнувшегося с данной задачей:

1. Запуск приложения

Пользователь двойным кликом запускает приложение и попадает в основное окно программы

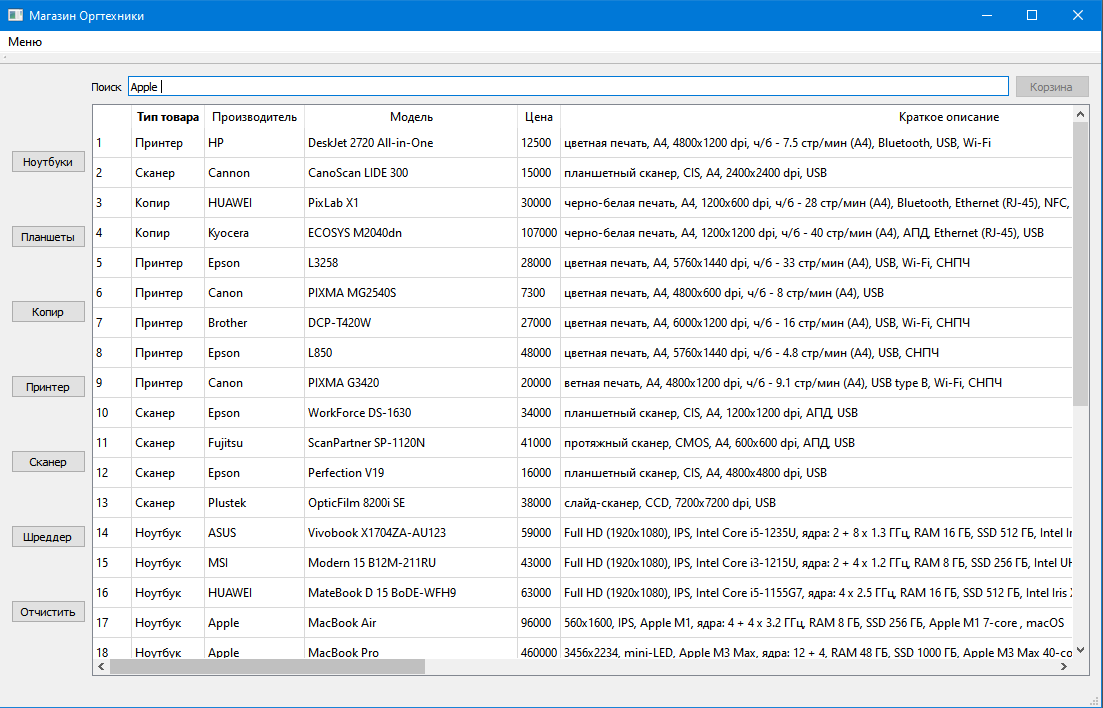
1. Нажатие различный фильтров или работа с использованием поиска

Пользователь поочередно нажимает кнопки фильтров и просматривает изменения состояния отображения окна



Найденные проблемы:

1. В какой-то момент просмотра таблицы товаров она полностью заполняется без участия пользователя;



Возможные способы решения:

1. Добавление информационного окна для сообщения, что база данных обновится в ближайшее время.
2. Блокировать обновление, пока пользователь находится на главное странице;

**Задача №2:**

Создание заказа.

История действий пользователя, впервые столкнувшегося с данной задачей:

1. Запуск приложения

Пользователь двойным кликом запускает приложение и попадает в основное окно программы

1. Выбор товаров

Пользователь выбирает нужные товары, тем самым заполняя корзину.

1. Переход в корзину и заполнение реквизитов.

Пользователь переходит в корзину и заполняет информацию о себе, после чего решает добавить в ещё товаров в корзину.

1. Выход из корзины.

Пользователь выходит из корзины и добавляет еще несколько товаров в корзину.

1. Переход обратно в корзину.

Найденные проблемы:

1. Удаление информации о пользователе при выходе из корзины, что неприятно для пользователя.

*Возможные способы решения:*

1. Добавление информационного окна о отчистке данных в этом окне;

# GOMS-анализ

**Описание анализа**

Практически все интерфейсные взаимодействия можно описать следующими операциями:

K – нажатие клавиши;

B – клик кнопкой мыши;

P – наведение указателя мыши;

R – ожидание ответной реакции компьютера;

H – перенос руки с клавиатуры на мышь или наоборот

D – проведение с помощью мыши прямой линии (например, выделение или прокрутка текста);

M – мыслительная подготовка (к осуществлению одной из перечисленных операций).

Разные пользователи выполняют указанные операции за разное время. Однако, GOMS исследует работу опытного пользователя. Многочисленные исследования выявили средние значения времени операций, выполняемых опытными пользователями.

* K 0.2 с
* B 0.2 с
* P 1.1 с
* H 0.4 с
* M 1.35 с

**Цель №1:** поиск записи в таблице с использованием функции поисковой строки.

Для выполнения цели сформулируем подцели:

1. Навести курсор мыши на поле поиска. **M + P + B = 1.35 + 1.1 + 0.2 =** **2.65**
2. Ввести интересующий запрос, состоящий из 5 символов. **M + H + 5 \* K = 1.35 + 0.4 + 5 \* 0.2 = 2.75**
3. Нажать на кнопку поиска. **M + H + P + B = 1.35 + 0.4 + 1.1 + 0.2 = 3.05**

Итог: **2.65 + 2.75 + 3.05 = 8.45с**

Возможное решение: можно убрать кнопку поиска и заменять её функционал нажатием на клавишу «Enter».

В данном случае сократиться время лишь последнего пункта до:

Нажать на клавишу «Enter». **M + K = 1.35 + 0.2 = 1.55**

Итог: **2.65 + 2.75 + 1.55 = 6.95с**

**Цель №2:** выход из приложения

Для выполнения цели сформулируем подцели:

1. Навестись на кнопку «Меню» **M + P + B = 1.35 + 1.1 + 0.2 = 2.65**
2. Закрыть программу выбором из контекстного меню пункта «Выход»

**M + P + B = 1.35 + 1.1 + 0.2 = 2.65**

Итог: **(2.65 + 2.65) = 5.3 с**

Возможное решение: можно добавить сочетание клавиш “Ctrl + Q” при нажатии на которые будет совершаться выход их приложения*.*

Закрытие программы **M + K + K = 1.35 + 0.2 + 0.2 = 1.75**

# Используемые источники

1. SQLiteStudio (3.4.4) – использовался для просмотра баз данных и составления их каркаса процессе разработки проекта;

URL:  [https://xn--90abhbolvbbfgb9aje4m.xn--p1ai/sqlitestudio-svobodnyj-menedzher-sqlite-baz-dannyh/](https://sqlitebrowser.org/)

1. Qt Documentation – официальная документация по разработке в среде Qt;

URL: <https://doc.qt.io/>

# Листинг

Главная страница интерфейса:

|  |  |
| --- | --- |
| #include "mainwindow.h"  #include "ui\_mainwindow.h"  MainWindow::**MainWindow**(QWidget \*parent) :  QMainWindow(parent),  ui(new Ui::MainWindow)  {  ui->setupUi(this);  // добавление базы данных  db = QSqlDatabase::addDatabase("QSQLITE", "db");  db.setDatabaseName("C:/Users/esspa/Desktop/VUZ/5sem/VPandPMM/Corse/Course/OrgTechnik.db");  if(db.open()) {  ui->statusBar->showMessage("Database open", 3000);  }  else {  ui->statusBar->showMessage("Database not open", 3000);  }  resezeVisualDB();  ui->mainTableOfProduct->setEditTriggers(QAbstractItemView::NoEditTriggers);  //загрузка базы данных  variant = QSqlQuery(db);  loadDataBase();  //отключение кнопки «корзина»  ui->pushButtonBasket->setEnabled(false);  //запуск таймера для обновления базы данных  timer = new QTimer();  connect(timer, SIGNAL(timeout()), this, SLOT(timerUpdate()));  connect(ui->leaveButton, SIGNAL(triggered()), qApp, SLOT(quit()));  timer->start(60000);  }  MainWindow::~***MainWindow***()  {  delete ui;  }  void MainWindow::**timerUpdate**() {  loadDataBase();  }  //загрузка базы данных в объект типа QSqlQuery  void MainWindow::**loadDataBase**() {  ui -> mainTableOfProduct->setRowCount(0);  if (!variant.exec("SELECT ID, Type, Producer, Model, Price, Info, Picture, Count FROM dataBaseForShop")){  ui->statusBar->showMessage("Database table 'DataBaseForShop' not open", 3000);  return;  }  //заполнение окна отображения базы данных  while (variant.next()) {  ui->mainTableOfProduct->insertRow(ui->mainTableOfProduct->rowCount());  for (int i = 0; i < 6; i++) ui->mainTableOfProduct->setItem(ui->mainTableOfProduct->rowCount()-1, i, new QTableWidgetItem(variant.value(i).toString()));  }  }  //открытие окна товара двойным нажатием  void MainWindow::**on\_mainTableOfProduct\_itemDoubleClicked**()  {  if (ui->mainTableOfProduct->selectedItems().isEmpty()) return;  QString ID = ui->mainTableOfProduct->selectedItems()[0]->text();  if (!variant.exec("SELECT \* FROM DataBaseForShop WHERE ID = " +ID)){  ui->statusBar->showMessage("Database table 'DataBaseForShop' not open", 3000);  return;  }  else {  ui->statusBar->showMessage("Database table 'DataBaseForShop' is open", 3000);  }  values = new QString[7];  while(variant.next()){  for (int i = 0; i < 6; i++) values[i] = variant.value(i+1).toString();  values[6] = variant.value(7).toString();  byteArray = variant.value(6).toByteArray();  }  product = new extendedProduct(values, byteArray, variant, this);  if (product->*exec*() == QDialog::Accepted) {  ui->pushButtonBasket->setEnabled(true);  ui->statusBar->showMessage(values[6], 10000);  for(int i = 0; i < 3; i++) {  result.push\_back(values[i]);  }  result.push\_back(values[6]);  result.push\_back(values[3]);  int t = values[6].toInt() \* values[3].toInt();  result.push\_back(QString::number(t));  }  }  //обработка нажатия кнопки фильтра Сканнер  void MainWindow::**on\_pushButtonScaner\_clicked**()  {  resezeVisualDB();  const QString data = "Сканер";  QTableWidget \*table = ui->mainTableOfProduct;;  for(int i = 0; i < table->rowCount(); i++) {  if (table->item(i, 1)->text() != data) {  table->setRowHidden(i, true);  }  else {  table->setRowHidden(i, false);  }  }  }  //обработка нажатия кнопки фильтра Принтер  void MainWindow::**on\_pushButtonPrinter\_clicked**()  {  resezeVisualDB();  const QString data = "Принтер";  QTableWidget \*table = ui->mainTableOfProduct;;  for(int i = 0; i < table->rowCount(); i++) {  if (table->item(i, 1)->text() != data) {  table->setRowHidden(i, true);  }  else {  table->setRowHidden(i, false);  }  }  }  //обработка нажатия кнопки фильтра Коопир  void MainWindow::**on\_pushButtonCopy\_clicked**()  {  resezeVisualDB();  const QString data = "Копир";  QTableWidget \*table = ui->mainTableOfProduct;;  for(int i = 0; i < table->rowCount(); i++) {  if (table->item(i, 1)->text() != data) {  table->setRowHidden(i, true); | }  else {  table->setRowHidden(i, false);  }  }  }  //обработка нажатия кнопки фильтра Ноутбук  void MainWindow::**on\_pushButtonNotebook\_clicked**()  {  resezeVisualDB();  const QString data = "Ноутбук";  QTableWidget \*table = ui->mainTableOfProduct;;  for(int i = 0; i < table->rowCount(); i++) {  if (table->item(i, 1)->text() != data) {  table->setRowHidden(i, true);  }  else {  table->setRowHidden(i, false);  }  }  }  //обработка нажатия кнопки фильтра Шреддер  void MainWindow::**on\_pushButtonShredder\_clicked**()  {  resezeVisualDB();  const QString data = "Шреддер";  QTableWidget \*table = ui->mainTableOfProduct;;  for(int i = 0; i < table->rowCount(); i++) {  if (table->item(i, 1)->text() != data) {  table->setRowHidden(i, true);  }  else {  table->setRowHidden(i, false);  }  }  }  //обработка нажатия кнопки фильтра Планшет  void MainWindow::**on\_pushButtonTablet\_clicked**()  {  resezeVisualDB();  const QString data = "Планшет";  QTableWidget \*table = ui->mainTableOfProduct;;  for(int i = 0; i < table->rowCount(); i++) {  if (table->item(i, 1)->text() != data) {  table->setRowHidden(i, true);  }  else {  table->setRowHidden(i, false);  }  }  }  //обработка нажатия кнопки фильтра отчистки фильтров  void MainWindow::**on\_pushButtonClear\_clicked**()  {  resezeVisualDB();  QTableWidget \*table = ui->mainTableOfProduct;;  for(int i = 0; i < table->rowCount(); i++) {  table->setRowHidden(i, false);  }  }  //задание настроек таблици  void MainWindow::**resezeVisualDB**() {  ui->mainTableOfProduct->horizontalHeader()->setSectionResizeMode(QHeaderView::ResizeToContents);  ui->mainTableOfProduct->setSelectionBehavior(QAbstractItemView::SelectRows);  }  //обработка нажатия на клавишу «Корзина»  void MainWindow::**on\_pushButtonBasket\_clicked**()  {  basket = new Basket(result, this);  basket->setWindowModality(Qt::ApplicationModal);  connect(basket, SIGNAL(sendToWidget(QString, QString, QString, QString, QString, QString)),  this, SLOT(addBayer(QString, QString, QString, QString, QString, QString)));  if (basket->*exec*() == QDialog::Accepted) {  }  }  // обработка сигнала для заполнения базы данных заказов  void MainWindow::**addBayer**(QString summ, QString number, QString Surname, QString Name, QString Patronomic, QString addres) {  variant.prepare( "INSERT INTO dataBaseOfPuarchase (totalCost, surnameClient, nameClient, pursCleint, addrtesDelivery, clientPhone) "  "VALUES (?,?,?,?,?,?)");  variant.addBindValue(summ);  variant.addBindValue(Surname);  variant.addBindValue(Name);  variant.addBindValue(Patronomic);  variant.addBindValue(addres);  variant.addBindValue(number);  if(!variant.exec()){  ui->statusBar->showMessage("Can't insert new order to database", 3000);  return;  }  QMessageBox::information(this, "Статус заказа", "Ваш заказ принят и находится в статусе сборки", QMessageBox::Ok);  result.clear();  ui->pushButtonBasket->setEnabled(false);  }  //реализация функции поиска  void MainWindow::**on\_lineEditSearch\_returnPressed**()  {  resezeVisualDB();  const QString data = ui->lineEditSearch->text();  QTableWidget \*table = ui->mainTableOfProduct;  for(int i = 0; i < table->rowCount(); i++) {  for(int j = 0; j < 6; j++) {  if (table->item(i, j)->text() != data) {  table->setRowHidden(i, true);  }  else {  table->setRowHidden(i, false);  break;  }  }  }  } |

**Исходный код проекта со всеми материалами опубликован в GitHub по ссылке: https://github.com/Kagava/VizualCourse**