**Оглавление**

[Аннотация 4](#_Toc8739487)

[Введение 5](#_Toc8739488)

[1 ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММНОМ СРЕДСТВЕ 6](#_Toc8739489)

[1.1 Основное функциональное назначение программного средства 6](#_Toc8739490)

[1.2 Полное наименование программного средства 6](#_Toc8739491)

[1.3 Условное обозначение программного средства 6](#_Toc8739492)

[1.4 Разработчики программного средства 6](#_Toc8739493)

[2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ 7](#_Toc8739494)

[2.1 Основание для разработки 7](#_Toc8739495)

[2.2 Назначение разработки 7](#_Toc8739496)

[2.3 Требование к программному средству 7](#_Toc8739497)

[2.4 Требования к программной документации 8](#_Toc8739498)

[2.5 Требования к эргономике и технической эстетике 8](#_Toc8739499)

[2.6 Стадии и этапы разработки 9](#_Toc8739500)

[2.7 Порядок контроля и приемки 10](#_Toc8739501)

[3 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ПРОГРАММНОМУ ПРОДУКТУ 11](#_Toc8739502)

[3.1 Декомпозиция поставленной задачи 11](#_Toc8739503)

[3.2 Общая архитектура программного средства 12](#_Toc8739504)

[3.3 Разработка алгоритма решения задачи 15](#_Toc8739505)

[3.4 Реализация функционального назначения программного средства 16](#_Toc8739506)

[3.5 Структурная организация данных 16](#_Toc8739507)

[3.6 Разработка интерфейса ПС 20](#_Toc8739508)

[3.7 Описание структуры выходной информации 24](#_Toc8739509)

[4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 25](#_Toc8739510)

[4.1 Назначение программного средства 25](#_Toc8739511)

[4.2 Условия выполнения программного средства 25](#_Toc8739512)

[4.3 Эксплуатация программного средства 25](#_Toc8739513)

[4.4 Сообщения пользователю 30](#_Toc8739514)

[5 РАЗРАБОТКА ТЕСТОВОГО НАБОРА 32](#_Toc8739515)

[5.1 Определение верхней границы количества тестов 32](#_Toc8739516)

[5.2 Разработка тестовых вариантов 33](#_Toc8739517)

[5.3 Составление отчета о тестировании 38](#_Toc8739518)

[ЗАКЛЮЧЕНИЕ 40](#_Toc8739519)

[Оценка качества программного средства с помощью метрик 40](#_Toc8739520)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 43](#_Toc8739521)

[Приложение А Текст кода программного средства 45](#_Toc8739522)

[Приложение Б Спецификация 74](#_Toc8739523)

[Приложение B Диск с программным средством 76](#_Toc8739524)

**Аннотация**

На дипломную работу студента Галимова А.Д. на тему: Автоматизированная информационная система «Кондитерское ателье». Дипломная работа представлена пояснительной запиской на 76 листов машинописного текста. Состоит из введения, 5 глав, заключения и списка литературных источников.

В главе 1 содержится общие сведения о программном средстве Кондитерское ателье. В главе 2 приведено техническое задание на проектирование БД согласно ГОСТ 19.201. Глава 3 содержит пояснительную записку к программному продукту согласно ГОСТ 19.404-79. В главе 4 содержится руководство пользователя согласно ГОСТ 19.504-79. Глава 5 содержит обоснование количества необходимых тестов и тестовые пакеты для оценки работоспособности программного продукта. В главе 6 приведены результаты ручного и автоматического тестирования программного продукта и выполнена оценка качества с помощью метрик.

В работе содержится общее описание программного средства, предназначенного для объекта проектирования – Автоматизированная информационная система «Кондитерское ателье», изложены требования к программному средству, описаны его архитектура, реализация функций, интерфейс, приведено руководство пользователя, обоснованы тестовые пакеты и выполнена оценка качества программного продукта с помощью метрик.

**Введение**

В дипломной работе была поставлена задача разработки базы данных «Кондитерское ателье», который позволяет осуществлять деятельность по продаже тортов кондитерского ателье.

Для достижения поставленной цели был составлен основной алгоритм работы программы, а также разработаны алгоритмы вспомогательных процедур.

Для реализации алгоритмов был выбран язык программирования C#. Выбор обусловлен тем, что возможностей этого языка достаточно для достижения поставленной цели моей дипломной работы.

При разработке интерфейса программы, я учитывал наиболее нужные и важные функциональные возможности, которые должна выполнять данная программа, а также тип пользователей, которые с ней будут работать.

**1 ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММНОМ СРЕДСТВЕ**

**1.1 Основное функциональное назначение программного средства**

Программа предназначена автоматизации деятельности кондитерского ателье. Программа не требует владения современными информационными технологиями для эффективного использования данного программного средства, т.е. подходит для любого пользователя.

**1.2 Полное наименование программного средства**

Разработка интегрированного программного модуля обработки данных Автоматизированная информационная система «Кондитерское ателье».

**1.3 Условное обозначение программного средства**

«CakesStudio 1.01»

**1.4 Разработчики программного средства**

Программное средство разработал студент Зеленодольского Механического колледжа группы 205 Галимов А.Д.

**2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

**2.1 Основание для разработки**

Работа ведётся на основании задания на дипломную работу по специальности 09.02.03 — «Программирование в компьютерных системах».

**2.2 Назначение разработки**

Программа «CakesStudio1.01» работает под управление операционной системы Windows 7, Windows 8, Windows 10. Программа предназначена для автоматизации деятельности кондитерского ателье. Программа позволяет вести учет заявок на изготовление тортов.

**2.3 Требование к программному средству**

«CakesStudio1.01» устанавливается на персональный компьютер, имеющий представленные минимальные системные требования:

Оперативная память: 4 Гб;

Процессор: Intel Core i / Phenom II X4 / AMD Athlon64 X2 4600 + 2.4 ГГц;

Видео карта: nVidia GeForce 8600 / ATI Radeon HD 2400;

Звуковая карта: Совместимая с DirectX;

Программный продукт «CakesStudio1.01» является кроссплатформенным продуктом, что позволяет не ограничиваться в работе с определенной версией ОС.

«CakesStudio1.01» написана на объектно-ориентированном языке программировании C# (си-шарп) в интегрированной среде разработки Visual Studio 2015. В качестве СУБД используется MS SQL. Программа не может подлежать сторонней оптимизации и редактированию программного кода.

**2.4 Требования к программной документации**

Состав программной документации должен включать в себя:

1. техническое задание;

2. сроки выполнения работ;

3. программу и методику испытаний;

4. эксплуатационные инструкции пользователю;

**2.5 Требования к эргономике и технической эстетике**

Система обеспечивает удобный для пользователей системный интерфейс, отвечающий следующим требованиям:

- единый стиль оформления для пользовательских интерфейсов;

- удобная, интуитивно понятная навигация в интерфейсе пользователя;

- взаимодействие пользователя с системой осуществляется на двух языках по выбору (русский и английский), для наибольшего удобства

Пользовательские интерфейсы системы спроектированы и разработаны с применением единых принципов графического представления информации и организации доступа к функциональным возможностям и сервисам. Разработан графический дизайн пользовательских интерфейсов, цветовые, шрифтовые и композиционные решения для отображения текстов, изображений, таблиц, гиперссылок, управляющих и навигационных элементов (меню, кнопок, форм и т.п.).

Экранные формы спроектированы с учетом требований унификации:

- все экранные формы пользовательского интерфейса выполнены в едином графическом дизайне, с одинаковым расположением основных элементов управления и навигации;

- для обозначения сходных операций использованы сходные графические значки, кнопки и другие управляющие (навигационные) элементы. Термины, используемые для обозначения типовых операций (добавление информационной сущности, редактирование поля данных), а также последовательности действий пользователя при их выполнении унифицированы;

- внешнее поведение сходных элементов интерфейса (реакция на наведение указателя «мыши», переключение фокуса, нажатие кнопки) реализованы одинаково для однотипных элементов.

**2.6 Стадии и этапы разработки**

Разработка должна быть проведена в три стадии:

1. разработка технического задания;

2. рабочее проектирование;

3. внедрение.

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

1. разработка программы;

2. разработка программной документации;

3. испытания программы.

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

1. постановка задачи;

2. определение и уточнение требований к техническим средствам;

3. определение требований к программе;

4. определение стадий, этапов и сроков разработки программы и документации на нее;

5. выбор языков программирования;

6. согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по программированию (кодированию) и отладке программы.

На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с пунктом Предварительный состав программной документации настоящего технического задания.

На этапе испытаний программы должны быть выполнены перечисленные ниже виды работ:

1. разработка, согласование и утверждение порядка и методики испытаний;

2. проведение испытаний;

3. корректировка программы и программной документации по результатам испытаний.

**2.7 Порядок контроля и приемки**

Сдача-приёмка работ производится поэтапно, в соответствии с рабочей программой и календарным планом. Сдача-приемка осуществляется комиссией, в состав которой входят представители ГАПОУ и работодатель. По результатам приемки подписывается акт приемочной комиссии.

В указанном подразделе, согласно таблице «Сроки выполнения работ» технического задания, будет описаны перечень участвующих организаций, место и сроки проведения работ, согласно п. 2.8 ГОСТ 34.602-89.

Порядок согласования и утверждения приемочной документации должен регламентироваться организационно-распорядительной документацией организации, принимающей участие в создании программного продукта. Согласно разделу «Приемка результатов разработки» ГОСТ 15.001-88 для согласования и утверждения приемочной документации создается приемочная комиссия (приказом).

Статус приемочной комиссии определяется Заказчиком до проведения испытаний.

**3 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К ПРОГРАММНОМУ ПРОДУКТУ**

**3.1 Декомпозиция поставленной задачи**

Для достижения цели работы была поставлена следующая задача:

Создать модуль “CodeDatabase” отвечающий за сохранение БД. Результат работы модуля - сохранение БД.

Поставленные задачи наглядно отображаются на диаграмме деятельности.

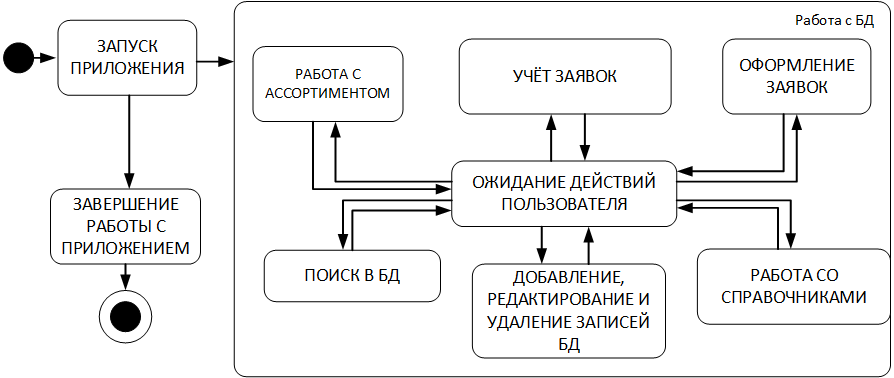


Рис.1 Декомпозиция поставленной задачи

**3.2 Общая архитектура программного средства**

Объектом моделирования является Кондитерское ателье.

Рассматриваемые процессы: Прием и выполнение заявок на торты.

Объекты моделирования представлены на диаграмме классов, рис.2.



Рис.2 Диаграмма классов

По полученной декомпозиции задач была спроектирована следующая архитектура программного средства.

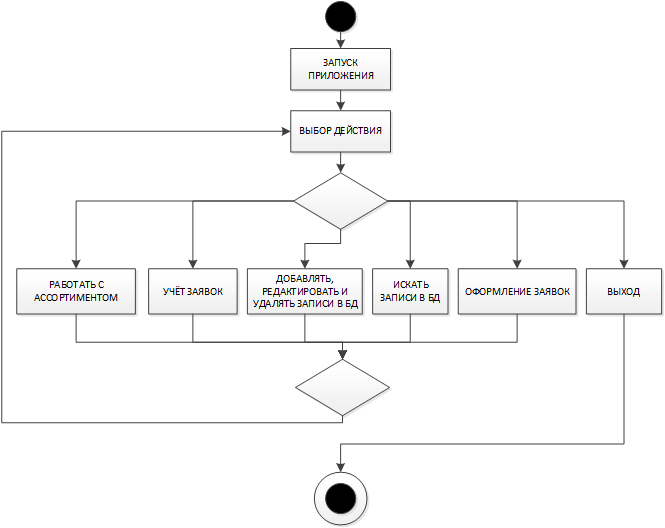


Рис.3 Схема событийно-ориентированной архитектуры

В «CakesStudio1.01» используется событийно-ориентированная архитектура.(Рис.3) Роль агента (источник событий) в программе выполняют: пользователи программы, за роль стоков (потребители событий) отвечают таблицы, входящие в базу данных. Например, когда пользователь выбирает определенное действие: добавить, удалить, редактировать, показать на карте и т.д. система осуществит выбранные действия, и база отреагирует соответствующим образом: запись добавлена, удалена, отредактирована. Системная архитектура пользователя рассматривает это изменение состояния как событие, создаваемое, публикуемое, определяемое и потребляемое различными приложениями в составе архитектуры.



Рис.4 Диаграмма вариантов использования

**3.3 Разработка алгоритма решения задачи**

**Общий алгоритм**

****Рис.5 Алгоритм программы

**3.4 Реализация функционального назначения программного средства**

Программа «CakesStudio1.01» имеет следующий набор входных данных, такие как: сведения о товарах, клиентах, заявки на изготовление торта.

Данные вводятся администратором в соответствующие поля ввода, снабженные всплывающими подсказками.

Выходными данными являются: учет заявок, акты приема заявки и акт о выполнении.

Данные вводятся пользователем в соответствующие поля вывода, снабженные всплывающими подсказками.

Выходные данные редактировать вручную невозможно, т.к. они служат в качестве отчетной информации для пользователя.

**3.5 Структурная организация данных**

Для создания БД необходимо определиться с данными, которые необходимы для полноценного функционирования системы. Все эти данные указаны в реляционной модели «БД Кондитерское ателье» представленная на рисунке. Любая реляционная база данных и называется реляционной, что характеризуется отношениями (relation) между таблицами. На рисунке изображены таблицы моей базы данных. При этом одна таблица является родительской (главной), а вторая – дочерней (подчиненной). Главной таблицей является «Заявка». Реляционная модель автоматизированной системы соответствует всем 12 правилам Кодда. (рис.6)

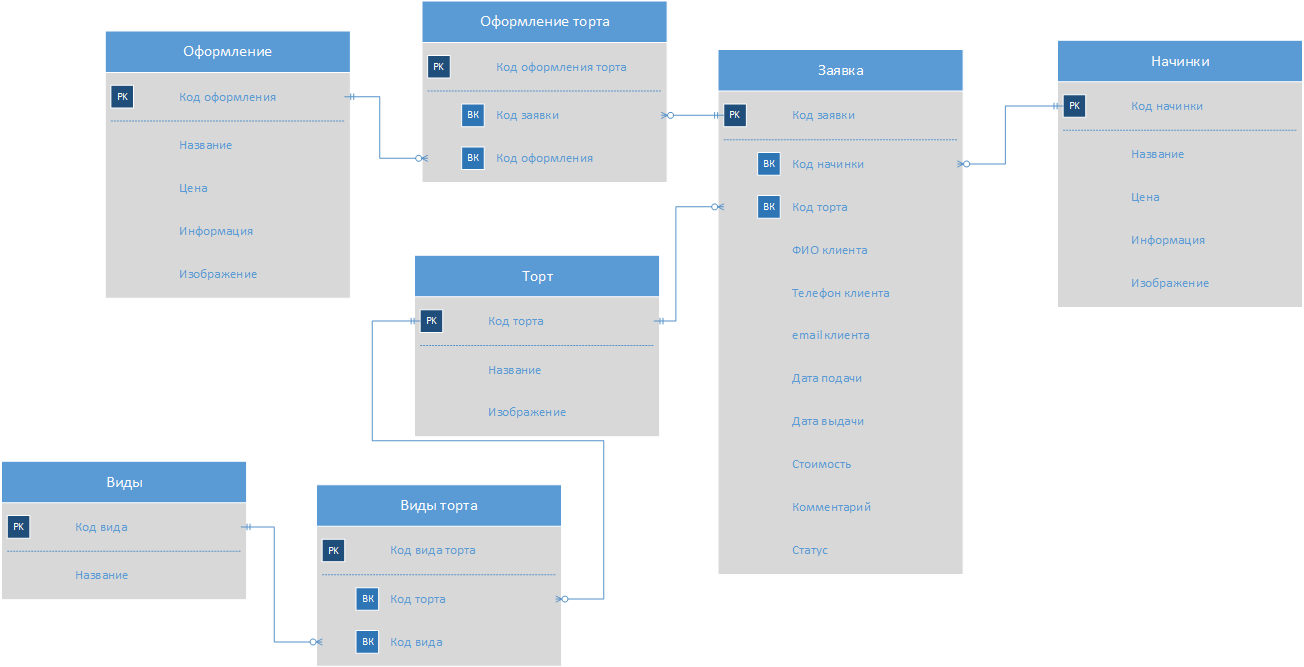


Рис.6 Реляционная модель

Первичный ключ в базе уникален, используется для организации отношений между таблицами, который не может иметь пустых и повторяющихся значений. Первичными ключами в базе являются поля: Код\_вида (таблица Виды), Код\_вида\_торта (Таблица Виды торта), Код\_торта (Таблица Торт), Код\_начинки (таблица Начинки), Код\_оформление (таблица Оформление), Код\_оформление\_торта (Таблица Оформление торта) и Код\_заявки(Таблица Заявки). Остальные ID-ключи являются внешними ключами.

Для организации более эффективной обработки данных применяется нормализация. Таблицы моей БД находятся в 3НФ:

* БД находится в форме -1НФ потому, что
  + Таблица не имеет повторяющихся записей;
  + Каждый атрибут отношения хранит одно-единственное значение и не является списком, ни множеством значений;
  + Таблица не имеет повторяющихся групп полей.
    - Вторая нормальная форма(2НФ):
      * Устранены атрибуты, зависящие только от части уникального (первичного) идентификатора, т.е. ID.
        + Третья нормальная форма(3НФ):

Отсутствуют атрибуты, зависящие от атрибутов, не входящих в уникальный (первичный) ключ.

На основе реляционной модели базы данных мною в MS SQL была построена следующая база данных. (Рис.7) В ней первичным ключом является поле таблицы “tOrder” – “ID\_order”. Связь таблиц “tOrder” и “tCake” по полю “ID\_Cake” один ко многим (1:М). Таблицы “tOrder” и “tStuffing” связаны по полю “ID\_Stuffing” типом связи 1:М. Таблицы “tOrder” и “tDecorCake” связаны по полю “ID\_order” типом связи 1:М. Таблицы “tDecorCake” и “tDecor” связаны по полю “ID\_decor” типом связи 1:М. Таблицы “tCake” и “tCakeAndCategory” связаны по полю “ID\_cake” типом связи 1:М. Таблицы “tCakeAndCategory” и “tCakeCategory” связаны по полю “ID\_cake\_category” типом связи 1:М. Все связи между таблицами базы данных типа один ко многим.

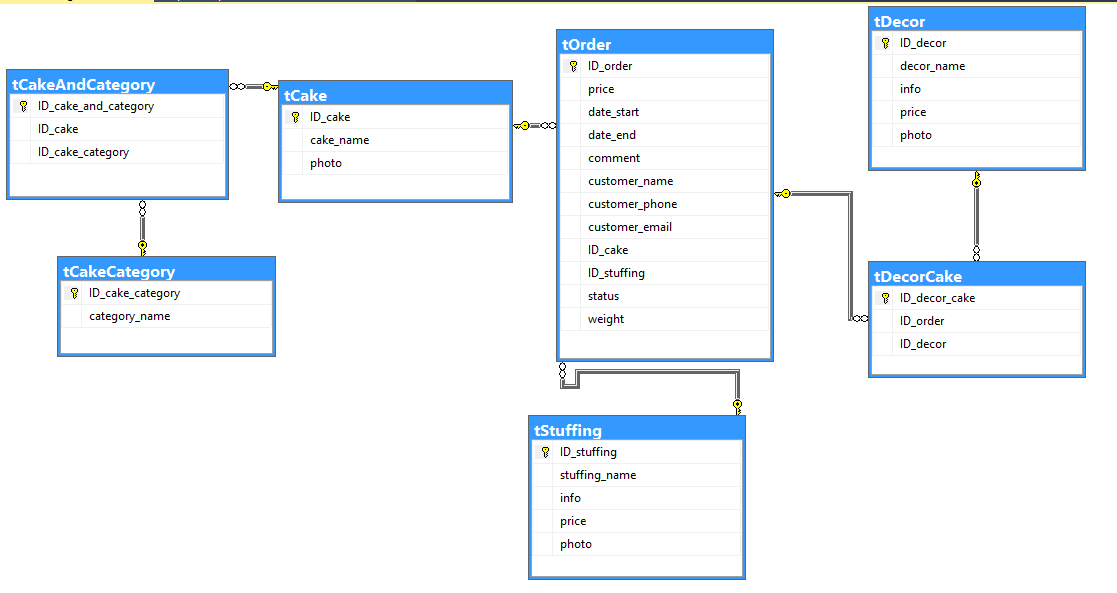


Рис.7 Схема базы данных

Таблица “tCakeCategory”(Виды)

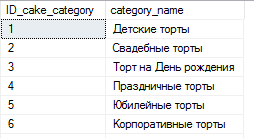


Рис.8 Таблица tCakeCategory

Таблица “tCake”(Торты)

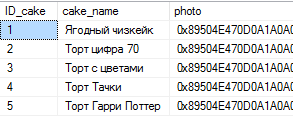


Рис.9 Таблица tCake

Таблица “tCakeAndCategory”(Виды тортов)

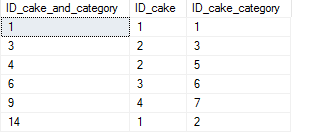


Рис.10 Таблица tCakeAndCategory

Таблица “tDecor”(Оформление)

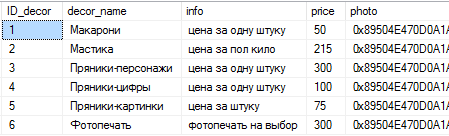


Рис.11 Таблица tDecor

Таблица “tDecorCake”(Оформление торта)

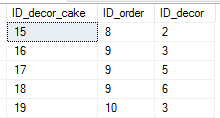


Рис.12 Таблица tDecorCake

Таблица “tStuffing”(Начинки)

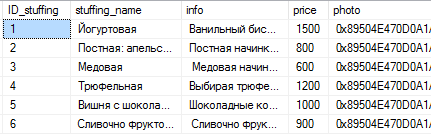


Рис.13 Таблица tStuffing

Таблица “tOrder”(Заявки)

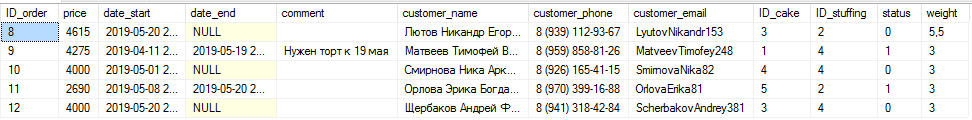


Рис.14 Таблица tOrder

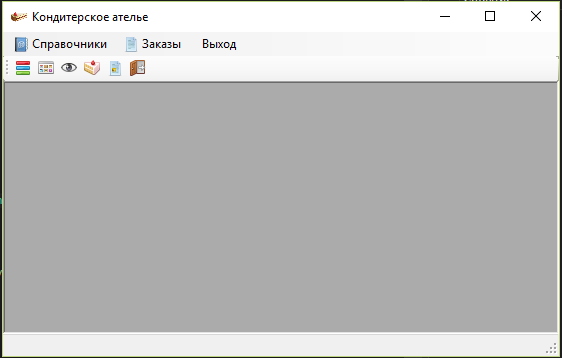
**3.6 Разработка интерфейса ПС**

Приложение однопользовательское, поэтому не предусмотрена авторизация. Для запуска приложения достаточно щелкнуть дважды левой кнопкой мыши по ярлыку.

После запуска программы пользователь приступит к работе с главной формой программы, на которой и будут отображаться все события в базе данных. «CakesStudio1.01» обладает удобным интерфейсом, так что даже неопытному в обращение с пользовательскими компьютерными программами пользователю не составит труда быстро привыкнуть к работе с данным программным средством.

На рабочем экране расположены несколько кнопок для комфортной работы с приложением.

**Форма Кондитерское ателье**. Основная форма справочника. Компоненты формы: menuStrip и toolStrip– для перехода к другим формам приложениия, . (Рис.15)



Выход

Меню

форма Заявки

форма Торты

форма Оформление

форма

Виды тортов

форма Начинки

Рис.15 Кондитерское ателье

**Форма Виды тортов**. Интерфейс данной формы состоит из компонентов: DataGridView, BindingNavigator и ряда кнопок для удаления, сохранения и обновления данных. Также есть возможность поиска категорий. (Рис.16)

Кнопки навигации по записям

Удалить, обновить, сохранить

Поиск

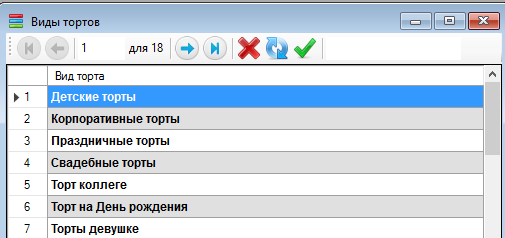


Рис.16 Окно Виды тортов

**Форма Начинки**. Интерфейс данной формы состоит из компонентов: DataGridView, BindingNavigator, pictureBox и ряда кнопок для удаления, сохранения и обновления данных. (Рис.17)

Кнопки навигации по записям

Удалить, обновить, сохранить

Поиск

Фото начинки

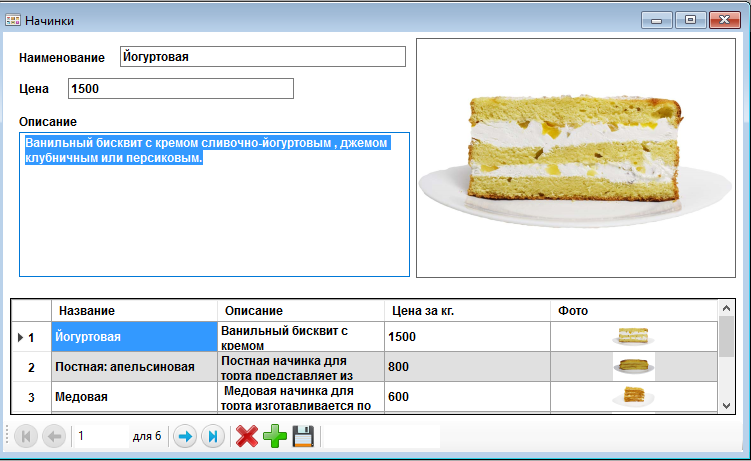
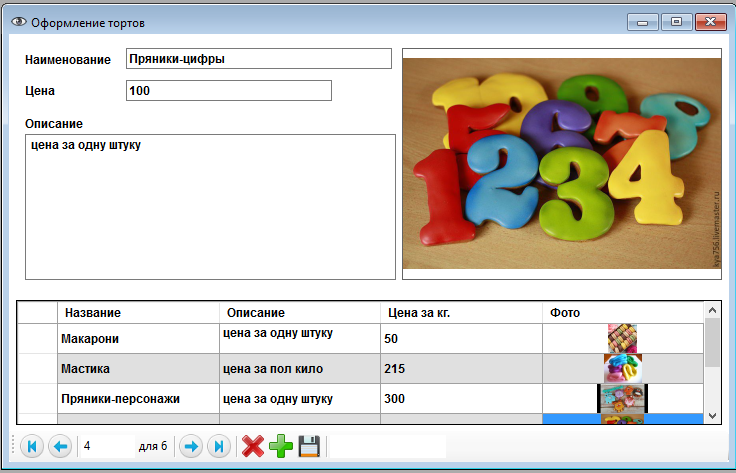


Рис.17 Окно Начинки

**Форма Оформление тортов**. Интерфейс данной формы состоит из компонентов: DataGridView, BindingNavigator, pictureBox и ряда кнопок для удаления, сохранения и обновления данных. (Рис.18)



Кнопки навигации по записям

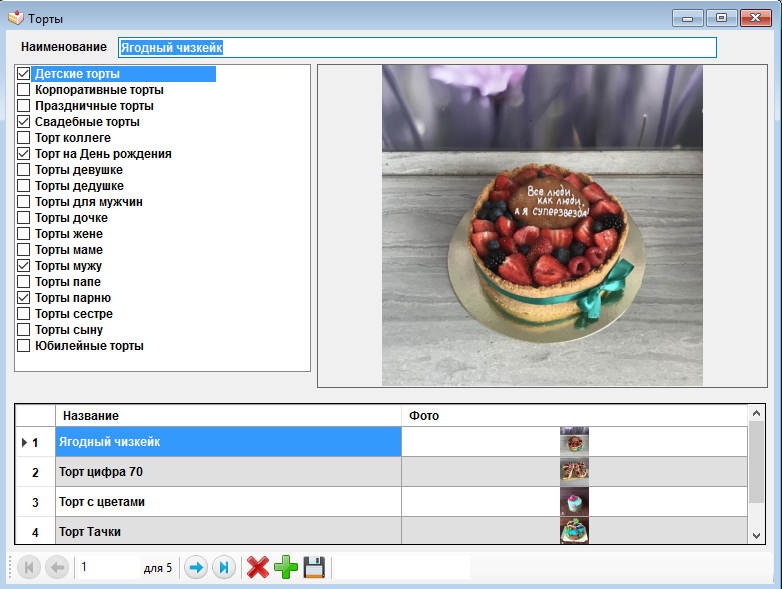
Удалить, обновить, сохранить

Поиск

Фото декора

Рис.18 Окно оформление тортов

**Форма Торт**. Интерфейс данной формы состоит из компонентов: DataGridView,CheckedListBox, BindingNavigator, pictureBox и ряда кнопок для удаления, сохранения и обновления данных. (Рис.19)



Кнопки навигации по записям

Удалить, обновить, сохранить

Поиск

Фото торта

Виды торта

Рис.19 Окно Торты

**Форма Заказы**. Интерфейс данной формы состоит из компонентов: DataGridView, BindingNavigator, Textbox, Combobox, dataTimepicker, checkbox и label и ряда кнопок для удаления и вызова дополнительной формы. (Рис.20)

Кнопки навигации по записям

Удалить

Список заказов

Поиск

вывод в Excel

Открыть доп. форму

фильтрация по статусу и дате

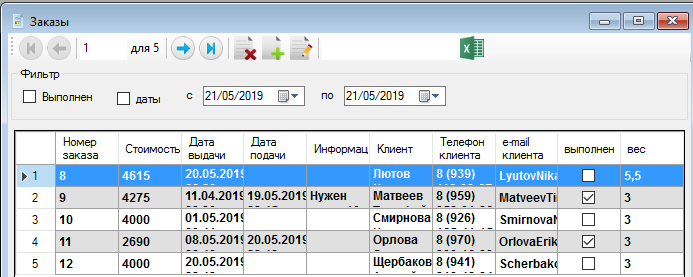
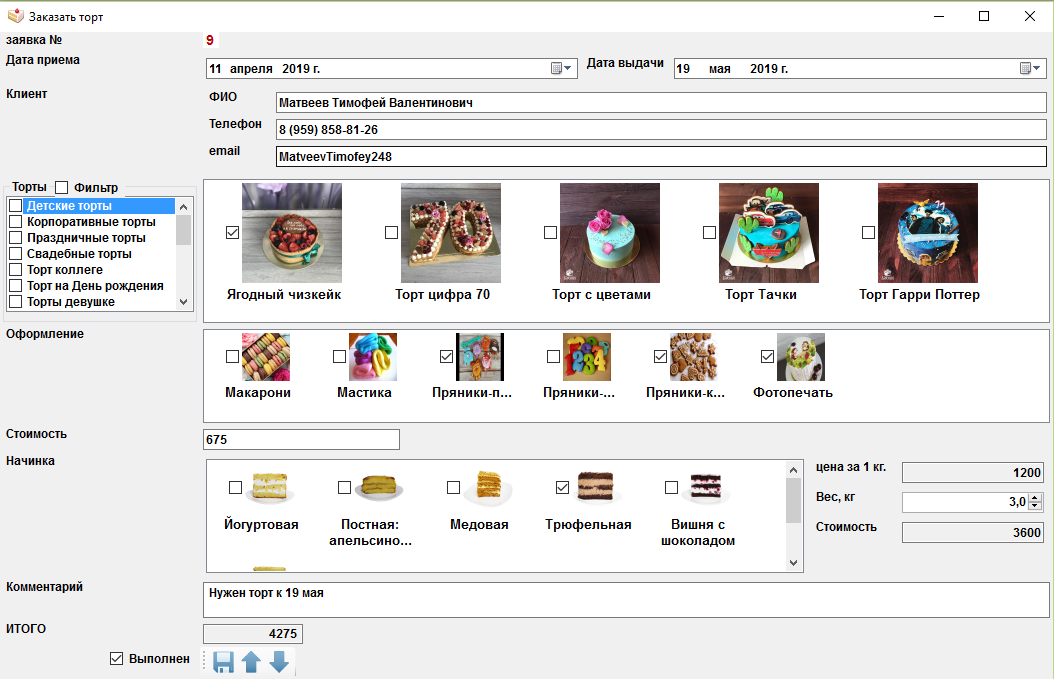


Рис.20 Окно Заказы

**Форма Заказать торт.** Интерфейс данной формы состоит из компонентов: listview – изображения с подписями, textbox – отображают записи из таблицы заявки БД, CheckedListbox – выводит виды тортов, а также кнопки для сохранения и вывода информации в Excel(Рис.22)



Даты заказов: подача и выполнение

CheckedListBox –фильтрация тортов

Вывод данных в Excel

LisView –

списки тортов, оформлений и начинок

Сохранить

Рис.21 Окно Заказать торт

**3.7 Описание структуры выходной информации**

Выходными данными являются бланки о приеме и выполнении работ, список заказов.

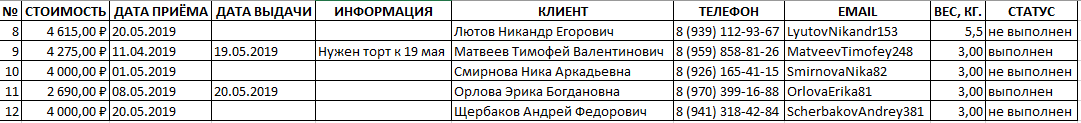


Рис.22 Список всех заказов

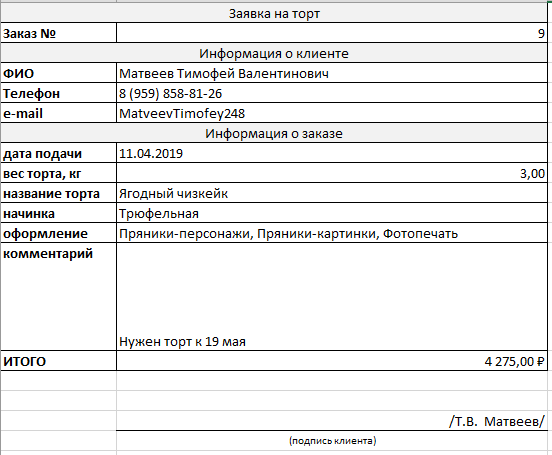


Рис.23 Заказ на торт

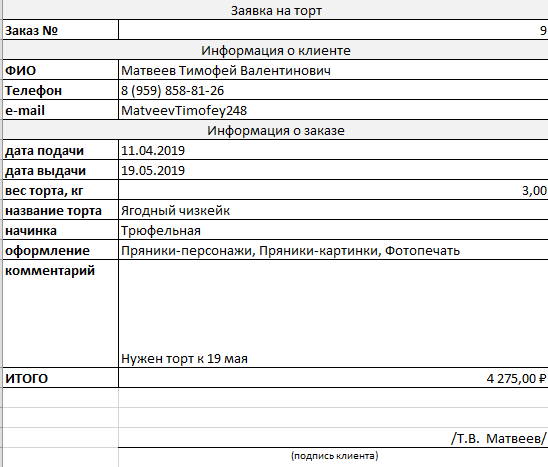


Рис.24 Заказ на торт

**4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

**4.1 Назначение программного средства**

Назначением программного средства является автоматизация учета заявок на гарантийный ремонт товаров.

**4.2 Условия выполнения программного средства**

Условия, при соблюдении которых возможно выполнение работы с программным продуктом:

-Компьютер с установленной на нём «CakesStudio1.01».

Подготовительные действия:

-На иконке «CakesStudio1.01» рабочего стола произвести двойной щелчок левой кнопкой мыши.

**4.3 Эксплуатация программного средства**

Этот раздел содержит обзор приложений «CakesStudio1.01» и ее основных параметров.

***Активация продукта***

Активировать копию программы CakesStudio1.01 можно сразу после её установки, когда появится окно подтверждения подлинности Вашей версии. Ключ активации расположен внутри упаковки программного продукта. Для успешного выполнения активации ключ активации необходимо вводить именно в том виде, в котором он предоставлен.

Вводные сведения об интерфейсе пользователя

В главном окне приложения, расположены 5 кнопок с переходом на основные формы приложения. Дополнительные формы открываются через соответствующие пункты меню. Пункты меню также дублируют функционал кнопок. Пользователь выбирает нужную кнопку и открывается интересующая его форма. (Рис.25)

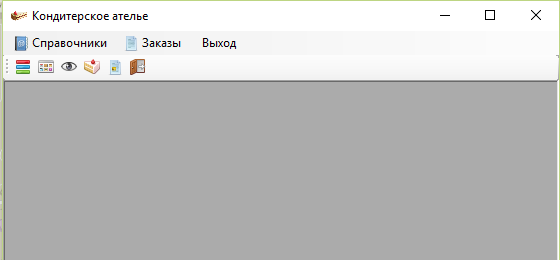


Рис.25 Окно Кондитерское ателье

Кликнув по кнопке «Выход» программа потребует подтверждение. (Рис.26)

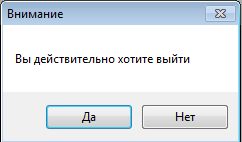


Рис.26 Сообщение программы

Ниже описаны пункты меню управления и информации.

«**Виды тортов**»: эта форма предназначена для ввода, удаления и редактирования данных в таблице Виды. Функционал формы позволяет добавить, удалить, изменить и найти нужную запись. (Рис.27)

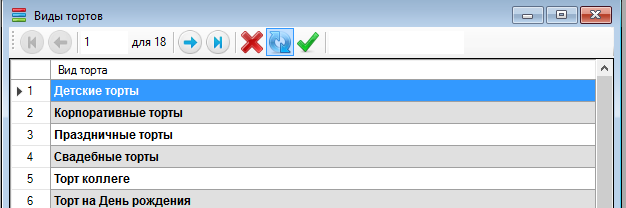


Рис.27 Окно Виды тортов

«Начинки»: эта форма предназначена для ввода, удаления и редактирования данных в таблице Начинки. Функционал формы позволяет добавить, удалить, изменить и найти нужную запись. (Рис.28)

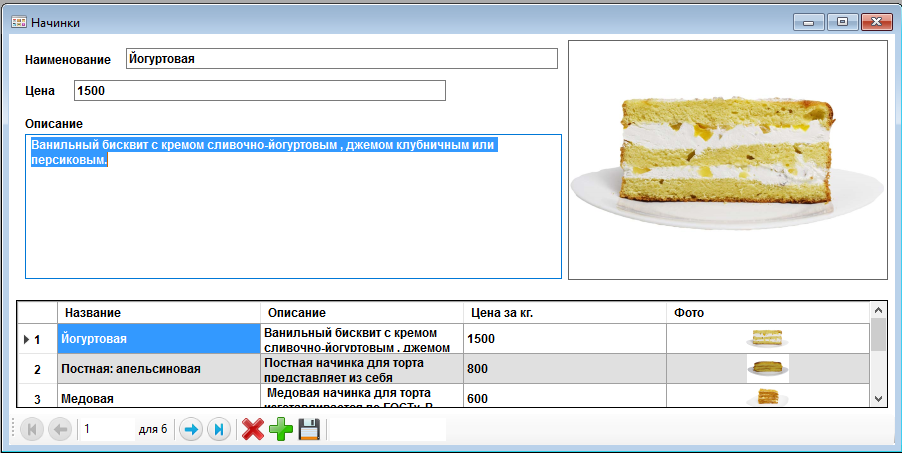


Рис.28 Окно Начинки

«Оформление тортов»: эта форма предназначена для ввода, удаления и редактирования данных в таблице Оформление. Функционал формы позволяет добавить, удалить, изменить и найти нужную запись. (Рис.29)

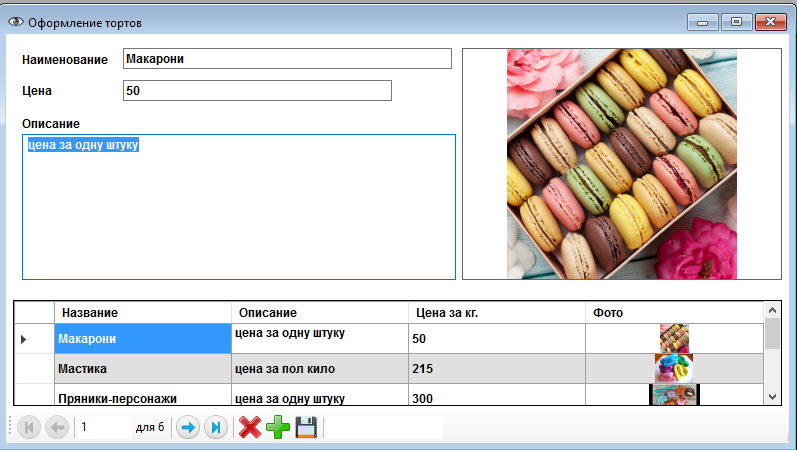


Рис.29 Окно Оформление тортов

«Заказы»: основная форма приложения. Эта форма позволяет вывести список отфильтрованных заказов по двум критериям(статус заказа и дата подачи) в Excel. Кроме этого также реализована возможность удаления заказа. Для того чтобы добавить новый заказ нужно нажать кнопку Добавить. Для того чтобы отредактировать запись нужно либо нажать на кнопку Редактировать, предварительно выбрав нужную запись, либо дважды щелкнуть по выбранной записи.(Рис.30)

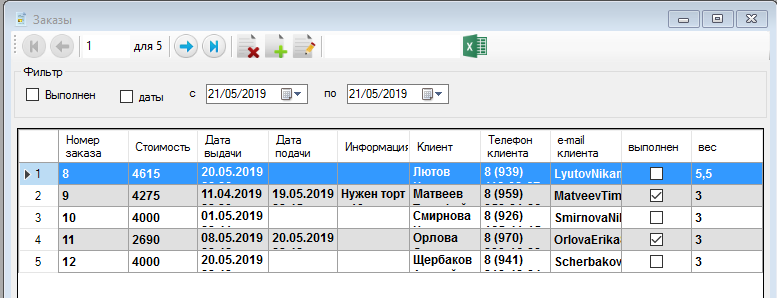


Рис.30 Окно Заказы

«Заказать торт»: форма предназначена для формирования нового заказа или редактирования существующей. Также с неё осуществляется вывод в Excel квитанций(Рис.31)

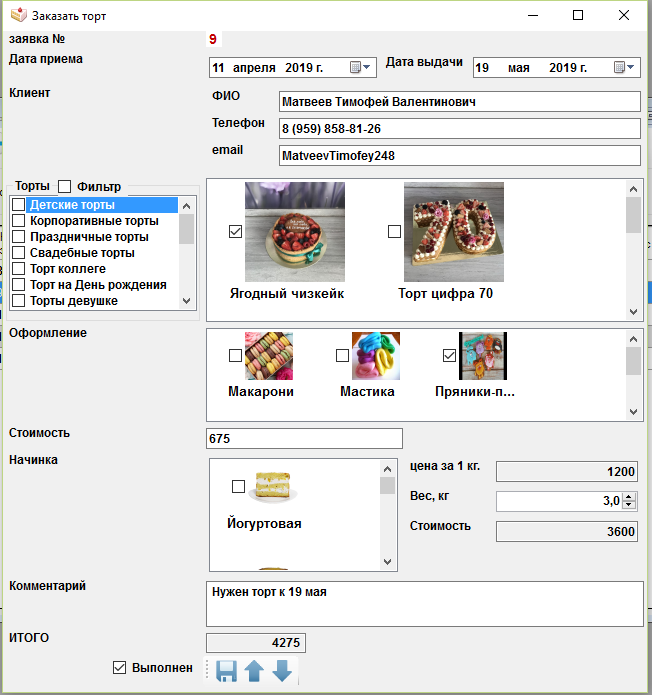


Рис.31 Окно Заказать торт

Работа с CakesStudio1.01

После запуска программы Вы перейдете на главную форму программы. В формах «Виды тортов», «Оформление», «Торты», «Начинки», «Заказы», «Заказать торт» заполняем все необходимые для работы пункты.

После этого кликаем по кнопке «Добавить запись».

Если Вы желаете удалить любую введенную ранее информацию, Вам необходимо кликнуть по нужной кнопке.

Для того чтобы изменить запись, введите в редактируемые поля данные и нажмите кнопку сохранить. Данные формы редактированию и управлению со стороны пользователя не подлежат.

На главной форме приложения реализован функционал для фильтрации и поиска заказов по различным критериям

После того, как Вы закончите сеанс работы с программой, если Вы хотите выйти из CakesStudio1.01 кликните на соответствующей кнопке «Выход».

Основные кнопки CakesStudio1.01

Кнопки управления «Добавить запись», «Удалить запись», «Сохранить» расположены на каждой конкретной форме в базе данных.

Действия этих кнопок приведены в таблице 1:

Таблица-1 Действие кнопок

|  |  |
| --- | --- |
| Добавить запись | Прежде чем добавить запись в таблицу необходимо нажать на данную кнопку. Поля формы будут очищены, пользователю требуется ввести необходимые данные. Если пользователь нажмет кнопку «Сохранить данные», то введенная информация добавится в БД. Если нажать кнопку «Отменить», то введенная информация пользователем удалится. |
| Удалить запись/отменить  , | Кнопка «Удалить запись» позволяет отменить действия при вводе, либо удаляет конкретную запись. |
| Сохранить данные/обновить запись  , | Кнопка «Сохранить данные» добавляет запись введенную пользователем в БД, либо обновляет существующую. |
| Редактировать | Открывает форму в режиме редактирования информации |
|  | «Поиск» позволяет найти нужную информацию. |

**4.4 Сообщения пользователю**

Для уверенной и комфортной работы пользователю важно понимать поведение программы, а программе, в свою очередь, поведение пользователя. Для информирования и уточнения намерений пользователей в программах используются сообщения.

При попытке удалить запись, пользователю выводится сообщение с запросом на подтверждение или отмену действия. (Рис.32)



Рис.32 Сообщение Подтверждение удаления

После добавления новой записи, пользователю выводится сообщение. (Рис.33)



Рис.33 Сообщение Запись добавлена

После обновления записи, пользователю выводится сообщение. (Рис.34)



Рис.34 Сообщение Запись обновлена

При попытке удалить связанные записи, программа выводит пользователю сообщение о невозможности действия. (Рис.35)



Рис.35 Сообщение Ошибка удаления

**5 РАЗРАБОТКА ТЕСТОВОГО НАБОРА**

**5.1 Определение верхней границы количества тестов**

Для разработанного алгоритма был построен граф.

****

Рис.36 Потоковый граф

Необходимое количество тестов определяется по формуле V(G) = E-N+2, где E – это число ребер, а N–это число вершин данного графа. Согласно представленному графу E=34, N=25. Далее по формуле рассчитаем необходимое количество тестовых сценариев (тест-кейсов).

V(G)=34-25+2=11.

Для данного программного средства необходимо разработать не менее 11 тестовых сценариев.

Чтобы определить количество тестовых прогонов для программного продукта CakesStudio1.01 необходимо выделить требования. Из требований к приложению выделим поддержку 2-х операционных систем с одним основным языком локализации и выполнения 6-ти основных функций. Кроме того, приложение позволяет выполнять 5 функций по взаимодействию с системой (запуск приложения, выход из приложения, сохранение результатов в файл, работа с буфером и т.п.).

Таким образом, полное покрытие требований задаёт набор из 2\*1\*(6+5) =22 тестовых прогонов.

**5.2 Разработка тестовых вариантов**

Таблица 2 – Тестирование корректного добавления информации в Виды тортов

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного добавления информации в таблицу Виды тортов | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Виды тортов | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Вид = Детские | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Виды тортов открыта | | Отображается форма Виды тортов |  |

Таблица 3 – Тестирование некорректного добавления информации в таблицу Виды тортов

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного добавления информации в таблицу Виды тортов | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Виды тортов | | Форма открыта |  |
| **Шаги теста:** | |  |  |

|  |  |  |
| --- | --- | --- |
| Нажмите кнопку добавить  Заполните форму  Вид = | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | Кнопка «Сохранить» заблокирована, так как поле пустое | **Тест пройден** |
| **Постусловие:** |  |  |
| Форма Виды тортов открыта | Отображается форма Виды тортов |  |

Таблица 4 – Тестирование корректного добавления информации в таблицу Начинки

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного добавления информации в таблицу Начинки | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Начинки | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Наименование = Йогуртовая  Цена = 1500  Изображение = загружено | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Начинки открыта | | Отображается форма Начинки |  |

Таблица 5 – Тестирование некорректного добавления информации в таблицу Начинки

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного добавления информации в таблицу Начинки | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Начинки | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Наименование =  Цена = 1500  Изображение = загружено | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, поле пустое | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Начинки открыта | | Отображается форма Начинки |  |

Таблица 6 – Тестирование корректного добавления информации в таблицу Оформление

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного добавления информации в таблицу Оформление тортов | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Оформление тортов | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Наименование = макарони  Цена = 1500  Изображение = загружено | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Оформление тортов  открыта | | Отображается форма Оформление тортов |  |

Таблица 7 – Тестирование некорректного добавления информации в таблицу Оформление тортов

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного добавления информации в таблицу Оформление тортов | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Оформление тортов | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  Наименование =  Цена = 1500  Изображение = загружено | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, поле пустое | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Оформление тортов  открыта | | Отображается форма Оформление тортов |  |

Таблица 8 – Тестирование корректного добавления информации в таблицу Заказ

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного добавления информации в таблицу Заказ | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Заказ | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  ФИО = Смирнова Ника  Телефон = 8961624148  Email = smirnovaNika  Торт = Торт Тачки  Оформление = Пряники-цифры  Начинка = Трюфельная  Вес = 3 кг  Дата приема заказа = 1 мая 2019 г. | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Заказ открыта | | отображается форма Заказ |  |

Таблица 9 – Тестирование некорректного добавления информации в таблицу Заказ

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного добавления информации в таблицу Заказ | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Заказ | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Нажмите кнопку добавить  Заполните форму  ФИО =  Телефон = 8961624148  Email = smirnovaNika  Торт = Торт Тачки  Оформление = Пряники-цифры  Начинка = Трюфельная  Вес = 3 кг  Дата приема заказа = 1 мая 2019 г. | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, так как ключевое поле пустое | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Заказ открыта | | отображается форма Заказ |  |

Таблица 10 – Тестирование корректного редактирования информации в таблице Заказ

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование корректного редактирования информации в таблице Заказ | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Заказ | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные, заполните форму  ФИО = Смирнова Ника  Телефон = 8961624148  Email = smirnovaNika  Торт = Торт Тачки  Оформление = Пряники-цифры  Начинка = Трюфельная  Вес = 3 кг  Дата приема заказа = 1 мая 2019 г. | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Данные успешно сохранены | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Заказ открыта | | отображается форма Заказ |  |

Таблица 11 – Тестирование некорректного редактирования информации в таблице Заказ

|  |  |  |  |
| --- | --- | --- | --- |
| **Название:** | Тестирование некорректного редактирования информации в таблице Заказ | | |
| **Функция:** | Циклы | | |
| **Действие** | | **Ожидаемый результат** | **Результат теста:**   * пройден * провален * заблокирован |
| **Предусловие:** | |  |  |
| Открываем форму Заказ | | Форма открыта |  |
| **Шаги теста:** | |  |  |
| Выберите запись для редактирования и измените данные, заполните форму  ФИО =  Телефон = 8961624148  Email = smirnovaNika  Торт = Торт Тачки  Оформление = Пряники-цифры  Начинка = Трюфельная  Вес = 3 кг  Дата приема заказа = 1 мая 2019 г. | | Данные успешно введены |  |
| Нажмите кнопку «Сохранить» | | Кнопка «Сохранить» заблокирована, поле пустое | **Тест пройден** |
| **Постусловие:** | |  |  |
| Форма Заказ открыта | | отображается форма Заказ |  |

**5.3 Составление отчета о тестировании**

Для форм проекта были созданы и проведены тесты, результаты сведены в таблицу:

Таблица 12 – Результаты тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Прохождение | | |
| Форма | Число ТП | пройдено | | не пройдено |
| Главная (Кондитерское ателье) | 4 | 4 | | 0 |
| Виды тортов | 2 | 2 | | 0 |
| Начинки | 2 | 2 | | 0 |
| Оформление тортов | 2 | 2 | | 0 |
| Заказ | 4 | 4 | | 0 |
| Торты | 4 | 4 | | 0 |
| ИТОГО | 18 | 18 | 0 | |

По результатам тестирования были выявлены и устранены следующие проблемы:

При выполнении теста таблица №8. Тестирование формы «Заказ», Некорректный запрос

**Ошибка**



Рис.37 Ошибка теста №8

**Причина**

В поле Накопления было введено вещественное число. В программном коде при сохранении предусматривалось, что сумма будет целочисленная.

**Принятые меры**

Ошибка устранена. Программный код был адаптирован под вещественные данные.

При выполнении теста таблица 6. Тестирование формы «Торты», Некорректный запрос

**Ошибка**

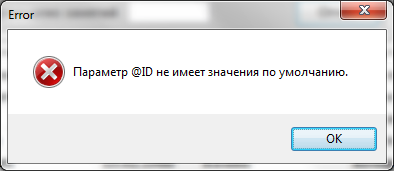


Рис.38 Ошибка теста №6

**Причина**

Ввод данных поиска символами. Посылался запрос в таблицу Торты на поиск торта с = «!М». Но такого вклада нет.

**Принятые меры**

Код изменен. В поле поиска можно вводить любые символы и поиск корректно работает.

**ЗАКЛЮЧЕНИЕ**

**Оценка качества программного средства с помощью метрик**

Оценка качества программного средства проводится исходя из стандарта оценки качества ISО 9126.

На компьютере следующей конфигурации:

Установленная операционная система – Windows 7 Ultimate 32бит.

Процессор: Intel Pentium G2030 3.0 GHz.

ОЗУ: 2 Gb

Таблица 13 – Требования к количественным характеристикам качества программного средства

|  |  |  |  |
| --- | --- | --- | --- |
| **Характеристики качества** | **Мера** | **Требуемое значение** | **Реальное значение** |
| **Надежность** |  | | |
| *Завершенность:*  *-* наработка на отказ при отсутствии рестарта; | Часы | 10 | 10 |
| *Устойчивость:*  - относительные ресурсы на обеспечение надежности и рестарта. | % | 0,5 | 0,5 |
| *Восстанавливаемость:*  -длительность восстановления. | Минуты | 2 | 2 |
| *Доступность-готовность:*  -относительное время работоспособного функционирования. | Вероятность | 0,998 | 0,998 |
| **Эффективность** |  | | |
| *Временная эффективность:*  *-*время отклика - получение результатов на типовое задание;  - пропускная способность число типовых заданий, исполняемых в единицу времени. | Секунды  Число в минуту | 3  60 | 3  60 |
| *Используемость ресурсов:*  *-* относительная величина использования ресурсов ЭВМ при нормальном функционировании программного средства. |  | 0,001 | 0,001 |

Таблица 14 – Требования к качественным характеристикам программного средства.

|  |  |  |  |
| --- | --- | --- | --- |
| **Характеристики качества** | **Мера** | **Требуемое значение** | **Реальное значение** |
| **Практичность** |  | | |
| *Простота использования:*  **-**среднее время ввода заданий;  -среднее время отклика на задание. | Секунды  Секунды | 10  3 | 10  3 |
| *Изучаемость :*  - трудоемкость изучения применения ПС;  -продолжительность изучения;  -объем эксплуатационной документации; | Чел.-часы  Часы  Страницы | 5  2  15 | 5  2  15 |
| **Сопровождаемость** |  | | |
| *Изменяемость:*  -трудоемкость подготовки изменений;  -длительность подготовки изменений. | Чел.-часы  Часы | 5  5 | 5  5 |
| *Тестируемость:*  *-трудоемкость тестирования изменений;*  *- длительность тестирования изменений.* | Чел.-часы  Часы | 3  3 | 3  3 |
| **Мобильность** |  | | |
| *Адаптируемость:*  - трудоемкость адаптации;  -длительность адаптации. | Чел.-часы  Часы | 0,2  1 | 0,2  1 |
| *Простота установки:*  -трудоемкость инсталляции;  -длительность инсталляции. | Чел.-часы  Часы | 0,2  1 | 0,2  1 |
| *Замещаемость:*  -трудоемкость замены компонентов;  - длительность замены компонентов. | Чел.-часы  Часы | 0,2  0,2 | 0,2  0,2 |

Таблица 15 – Сложность

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Тип объекта | Количество | Вес | | | Итого |
|  |  | Простой | Средний | Сложный |  |
| Экран | 10 | x1 | x2 | **x3** | =30 |
| Отчет | 3 | x2 | **x5** | x8 | =15 |

Таблица 16 – Сложность экрана

|  |  |  |  |
| --- | --- | --- | --- |
| Экраны | Количество клиентских (клт) таблиц данных | | |
| Количество представлений | Всего <4 | **Всего <8** | Всего >8 |
|  | (<2 срв, <3 клт) | **(2-3 срв, 3-5 клт)** | (>3 срв, >5 клт) |
| <3 | Простой | Простой | Средний |
| 3-7 | Простой | Средний | Сложный |
| **>8** | Средний | **Сложный** | Сложный |

Таблица 17 – Сложность отчета

|  |  |  |  |
| --- | --- | --- | --- |
| Отчеты | Количество клиентских (клт) таблиц данных | | |
| Количество представлений | Всего <4 | Всего <8 | Всего >8 |
|  | (<2 срв, <3 клт) | **(2-3 срв, 3-5 клт)** | (>3 срв, >5 клт) |
| 0 или 1 | Простой | Простой | Средний |
| **2 или 3** | Простой | **Средний** | Сложный |
| >4 | Средний | Сложный | Сложный |

Производительность = 4 тысяч строк кода/2 месяца= 2 тысяч строк кода/месяц

Качество = 2 ошибки /4 тысяч строк кода=0,5 ошибок/тысяч строк кода

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Г. Шилдт C#: Учебный курс. - Спб.: Питер, 2021. - 512с

2. Либерти, Д. Программирование на Си Шарп. - М.: Символ-плюс, 2022. - 684 с.

3. Ватсон К. Си Шарп/К.Ватсон.-М.:Лори, 2021.-862 с

4. Балена Ф., Димауро Д. Современная практика программирования на Visual Basic и Visual C#.-М.:Русская редакция, 2022,604с.

5. Енин А., Енин Н. Локальная СУБД своими руками. Учимся на примерах.-М.:СОЛОН-ПРЕСС, 2022,464с.,СД

6. Лабор В.В. Си Шарп: Создание приложений для Windows/-Мн.: Харвест, 2022-384с.

7. Фролов А.В., Фролов Г.В. Визуальное проектирование приложений C#.-М.:КУДИЦ-ОБРАЗ, 2022-512с.

8. ГОСТ 19.201-78.ЕСПД. Техническое задание. Требования к содержанию и оформлению

9. ГОСТ 19.102-77.ЕСПД. Стадии разработки

10. ГОСТ 19.505-79.ЕСПД. Руководство оператора. Требования к содержанию и оформлению

11. ГОСТ 19.002-80.ЕСПД. Схемы алгоритмов и программ. Правила выполнения

12. ГОСТ 19.404-79.ЕСПД. Пояснительная записка. Требования к содержанию и оформлению

13. Владимир Биллинг. Основы программирования на С#. 2022 г.

[Электронный ресурс http://mexalib.com/view/42671]

14. Никита Культин. Основы программирования в MicrosoftVisualC# 2022г.

[Электронный ресурс http://www.litres.ru/nikita-kultin/osnovy-programmirovaniya-v-microsoft-visual-c-2010-2]

15. Фролов А.В., Фролов Г.В. Визуальное проектирование приложений C#.-М.:КУДИЦ-ОБРАЗ, 2022.

[Электронный ресурс http://www.twirpx.com/file/27852]

16. Йен Гриффитс. Программирование на С#. 2022 г.

[Электронный ресурс http://rutracker.org/forum/viewtopic.php?t=4772853]

**Приложение А**

Автоматизированная информационная система «Кондитерское ателье»

Текст кода программного средства

460.ЗМК.00005-01 12 01

**Текст кода программного средства**

**Форма Главная**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Konditer

{

public partial class MainForm : Form

{

private CategoryForm categoryForm;

private DecorForm decorForm;

private StuffingForm stuffingForm;

private CakeForm cakeForm;

private OrdersForm ordersForm;

public MainForm()

{

InitializeComponent();

}

private void mnuExit\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void MainForm\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

private void MainForm\_FormClosing(object sender, FormClosingEventArgs e)

{

try

{

DialogResult result = MessageBox.Show("Вы действительно хотите выйти из приложения", "Внимание",

MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (result == DialogResult.No)

{

e.Cancel = true;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

private void видыТортовToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (categoryForm == null || categoryForm.IsDisposed)

{

categoryForm = new CategoryForm();

categoryForm.MdiParent = this;

categoryForm.Show();

}

else

{

categoryForm.Activate();

}

}

private void mnuStuffing\_Click(object sender, EventArgs e)

{

if (stuffingForm == null || stuffingForm.IsDisposed)

{

stuffingForm = new StuffingForm();

stuffingForm.MdiParent = this;

stuffingForm.Show();

}

else

{

stuffingForm.Activate();

}

}

private void mnuDecoration\_Click(object sender, EventArgs e)

{

if (decorForm == null || decorForm.IsDisposed)

{

decorForm = new DecorForm();

decorForm.MdiParent = this;

decorForm.Show();

}

else

{

decorForm.Activate();

}

}

private void mnuCake\_Click(object sender, EventArgs e)

{

if (cakeForm == null || cakeForm.IsDisposed)

{

cakeForm = new CakeForm();

cakeForm.MdiParent = this;

cakeForm.Show();

}

else

{

cakeForm.Activate();

}

}

private void mnuOrders\_Click(object sender, EventArgs e)

{

if (ordersForm == null || ordersForm.IsDisposed)

{

ordersForm = new OrdersForm();

ordersForm.MdiParent = this;

ordersForm.Show();

}

else

{

ordersForm.Activate();

} } }}

**Форма Виды тортов**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Configuration;

using System.Data.SqlClient;

namespace Konditer

{

public partial class CategoryForm : Form

{

string connectionString = ConfigurationManager.ConnectionStrings["SqlCon"].ConnectionString;

SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["SqlCon"].ConnectionString);

DataSet ds;

SqlDataAdapter dataAdapter;

BindingSource bs1;

public CategoryForm()

{

InitializeComponent();

LoadDataFromTable();

}

private void saveToolStripButton\_Click(object sender, EventArgs e)

{

try

{

SqlCommandBuilder CmbSAve = new SqlCommandBuilder(dataAdapter);

dataAdapter.Update(ds);

saveToolStripButton.Enabled = false;

}

catch (Exception ex)

{

if (ex.HResult == -2146232060)

MessageBox.Show("Ошибка удаления, есть связанные записи");

else MessageBox.Show(ex.Message);

LoadDataFromTable();

}

}

void LoadDataFromTable()

{

try

{

ds = new DataSet();

dataAdapter = new SqlDataAdapter();

dataAdapter.SelectCommand = new SqlCommand("Select \* from tCakeCategory Order by category\_name", connection);

//dt1 = new DataTable();

ds.Clear();

dataAdapter.Fill(ds);

ds.AcceptChanges();

bs1 = new BindingSource();

bs1.DataSource = ds.Tables[0];

bindingNavigator1.BindingSource = bs1;

dgvTypeTO.DataSource = bs1;

dgvTypeTO.Columns[0].Visible = false;

dgvTypeTO.Columns[1].HeaderText = "Вид торта";

dgvTypeTO.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

private void deleteStripButton2\_Click(object sender, EventArgs e)

{

try

{

int i = dgvTypeTO.CurrentRow.Index;

dgvTypeTO.Rows.RemoveAt(i);

saveToolStripButton.Enabled = true;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void toolStripButton1\_Click(object sender, EventArgs e)

{

LoadDataFromTable();

}

private void dgvTypeTO\_UserAddedRow(object sender, DataGridViewRowEventArgs e)

{

saveToolStripButton.Enabled = true;

int i = dgvTypeTO.CurrentRow.Index;

}

private void dgvTypeTO\_UserDeletedRow(object sender, DataGridViewRowEventArgs e)

{

saveToolStripButton.Enabled = true;

}

private void dgvTypeTO\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

saveToolStripButton.Enabled = true;

}

private void toolStripTextBox1\_TextChanged(object sender, EventArgs e)

{

bs1.Filter = "category\_name LIKE '%" + toolStripTextBox1.Text + "%'";

}

private void dgvTypeTO\_RowPostPaint(object sender, DataGridViewRowPostPaintEventArgs e)

{

var grid = sender as DataGridView;

var rowIdx = (e.RowIndex + 1).ToString();

var centerFormat = new StringFormat()

{

// right alignment might actually make more sense for numbers

Alignment = StringAlignment.Center,

LineAlignment = StringAlignment.Center

};

var headerBounds = new Rectangle(e.RowBounds.Left, e.RowBounds.Top, grid.RowHeadersWidth, e.RowBounds.Height);

e.Graphics.DrawString(rowIdx, this.Font, SystemBrushes.ControlText, headerBounds, centerFormat); } }}

**Форма Начинки**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Configuration;

using System.Data.SqlClient;

using System.Drawing.Imaging;

using System.IO;

using System.Windows.Forms.VisualStyles;

namespace Konditer

{

public partial class StuffingForm : Form

{

string connectionString = ConfigurationManager.ConnectionStrings["SqlCon"].ConnectionString;

SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["SqlCon"].ConnectionString);

DataTable dtStuff;

SqlDataAdapter dataStuffAdapter;

BindingSource bsStuff;

bool add\_items;

public StuffingForm()

{

InitializeComponent();

LoadDataFromTable();

add\_items = false;

}

void LoadDataFromTable()

{

try

{

dtStuff = new DataTable();

dataStuffAdapter = new SqlDataAdapter();

dataStuffAdapter.SelectCommand = new SqlCommand("SELECT \* " +

"FROM tStuffing", connection);

dataStuffAdapter.Fill(dtStuff);

bsStuff = new BindingSource();

bsStuff.DataSource = dtStuff;

bindingNavigator1.BindingSource = bsStuff;

dgvTypeTO.DataSource = bsStuff;

dgvTypeTO.Columns[0].Visible = false;

dgvTypeTO.Columns[1].HeaderText = "Название";

dgvTypeTO.Columns[2].HeaderText = "Описание";

dgvTypeTO.Columns[3].HeaderText = "Цена за кг.";

dgvTypeTO.Columns[4].HeaderText = "Фото";

((DataGridViewImageColumn)dgvTypeTO.Columns[4]).ImageLayout = DataGridViewImageCellLayout.Zoom;

dgvTypeTO.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

tbName.DataBindings.Clear();

tbName.DataBindings.Add(new Binding("Text", bsStuff, "stuffing\_name"));

tbPrice.DataBindings.Clear();

tbPrice.DataBindings.Add(new Binding("Text", bsStuff, "price"));

tbInfo.DataBindings.Clear();

tbInfo.DataBindings.Add(new Binding("Text", bsStuff, "info"));

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

private void tbPrice\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) &&

(e.KeyChar != '.'))

{

e.Handled = true;

}

}

Image LoadImage()

{

if (bsStuff.Count > 0)

{

int i = bsStuff.Position;

if ((((DataRowView)this.bsStuff.Current).Row["photo"] != System.DBNull.Value))

{

byte[] byteArrayIn = (byte[])(((DataRowView)this.bsStuff.Current).Row["photo"]);

using (var ms = new MemoryStream(byteArrayIn))

{

return Image.FromStream(ms);

}

}

return null;

}

else return null;

}

private void deleteStripButton2\_Click(object sender, EventArgs e)

{

if (add\_items)

{

LoadDataFromTable();

add\_items = false;

return;

}

DeleteData();

}

/// <summary>

/// преобразует картинку в биты

/// </summary>

/// <param name="img"></param>

/// <returns></returns>

byte[] ConvertInBytes(Image img)

{

byte[] bytes;

using (MemoryStream ms = new MemoryStream())

{

img.Save(ms, ImageFormat.Png);

bytes = ms.ToArray();

}

return bytes;

}

private void saveToolStripButton\_Click(object sender, EventArgs e)

{

if ((tbName.Text == "") || (tbPrice.Text == "") || (tbInfo.Text == "") )

return;

if (add\_items)

{

SaveData();

}

else

{

UpdateData();

}

}

private void dgvTypeTO\_SelectionChanged(object sender, EventArgs e)

{

LoadCombo();

}

private void pictureBox1\_DoubleClick(object sender, EventArgs e)

{

if (openFileDialog1.ShowDialog() == DialogResult.Cancel)

return;

// получаем выбранный файл

try

{

string filename = openFileDialog1.FileName;

pictureBox1.Image = Image.FromFile(filename);// читаем файл в строку

}

catch

{

MessageBox.Show("Ошибка загрузки файла", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void btnAdd\_Click(object sender, EventArgs e)

{

ClearItems();

add\_items = true;

}

private void toolStripTextBox1\_TextChanged(object sender, EventArgs e)

{

// bsGoods.Filter = "ID\_good LIKE '%" + toolStripTextBox1.Text + "%'";

bsStuff.Filter = "stuffing\_name LIKE '%" + toolStripTextBox1.Text + "%'";

}

void ClearItems()

{

tbName.Text = "";

tbPrice.Text = "";

tbInfo.Text = "";

pictureBox1.Image = pictureBox1.InitialImage;

}

/// <summary>

/// DeleteData()

/// </summary>

void DeleteData()

{

if (bsStuff.Count > 0)

{

int i = bsStuff.Position;

int ID\_SS = Convert.ToInt32(((DataRowView)this.bsStuff.Current).Row["ID\_stuffing"]);

try

{

DialogResult result = MessageBox.Show("Вы действительно хотите удалить запись", "Внимание", MessageBoxButtons.YesNo);

if (result == DialogResult.No)

{

LoadDataFromTable();

return;

}

if (result == DialogResult.Yes)

{

connection.Close();

connection.Open();

SqlCommand commandDelete = new SqlCommand("Delete From tStuffing where ID\_stuffing = @ID", connection);

commandDelete.Parameters.AddWithValue("@ID", ID\_SS);

commandDelete.ExecuteNonQuery();

ClearItems();

}

}

catch (SqlException exception)

{

MessageBox.Show(exception.ToString());

}

finally

{

connection.Close();

LoadDataFromTable();

}

}

}

void SaveData()

{

int ID\_SS = 0;

try

{

connection.Close();

connection.Open();

if (pictureBox1.Image == null)

{

SqlCommand commandInsert = new SqlCommand("INSERT INTO [tStuffing] VALUES" +

" (@Name,@Info, @Price, NULL)", connection);

commandInsert.Parameters.AddWithValue("@Name", tbName.Text);

commandInsert.Parameters.AddWithValue("@Info", tbInfo.Text);

commandInsert.Parameters.AddWithValue("@Price", tbPrice.Text);

commandInsert.ExecuteNonQuery();

}

else

{

SqlCommand commandInsert = new SqlCommand("INSERT INTO [tStuffing] VALUES" +

" (@Name,@Info, @Price, @photo)", connection);

commandInsert.Parameters.AddWithValue("@Name", tbName.Text);

commandInsert.Parameters.AddWithValue("@Info", tbInfo.Text);

commandInsert.Parameters.AddWithValue("@Price", tbPrice.Text);

commandInsert.Parameters.AddWithValue("@photo", ConvertInBytes(pictureBox1.Image));

commandInsert.ExecuteNonQuery();

}

MessageBox.Show("Запись добавлена");

add\_items = false;

}

catch (SqlException exception)

{

MessageBox.Show(exception.ToString());

}

finally

{

connection.Close();

LoadDataFromTable();

}

}

void UpdateData()

{

if (bsStuff.Count > 0)

{

int i = bsStuff.Position;

int ii = bsStuff.Position;

try

{

connection.Close();

connection.Open();

int ID\_SS = Convert.ToInt32(((DataRowView)this.bsStuff.Current).Row["Id\_stuffing"]);

if (pictureBox1.Image == null)

{

SqlCommand commandUpdate = new SqlCommand("UPDATE tStuffing SET" +

" stuffing\_name=@name, " +

" info=@info, price=@price," +

" WHERE Id\_stuffing = @IDSS", connection);

commandUpdate.Parameters.AddWithValue("@Name", tbName.Text);

commandUpdate.Parameters.AddWithValue("@Info", tbInfo.Text);

commandUpdate.Parameters.AddWithValue("@Price", tbPrice.Text);

commandUpdate.Parameters.AddWithValue("@IDSS", ID\_SS);

commandUpdate.ExecuteNonQuery();

}

else

{

SqlCommand commandUpdate = new SqlCommand("UPDATE tStuffing SET" +

" stuffing\_name=@name, " +

" info=@info, price=@price, photo =@photo" +

" WHERE Id\_stuffing = @IDSS", connection);

commandUpdate.Parameters.AddWithValue("@Name", tbName.Text);

commandUpdate.Parameters.AddWithValue("@Info", tbInfo.Text);

commandUpdate.Parameters.AddWithValue("@Price", tbPrice.Text);

commandUpdate.Parameters.AddWithValue("@Photo", ConvertInBytes(pictureBox1.Image));

commandUpdate.Parameters.AddWithValue("@IDSS", ID\_SS);

commandUpdate.ExecuteNonQuery();

}

MessageBox.Show("Запись обновлена");

}

catch (SqlException exception)

{

MessageBox.Show(exception.ToString());

}

finally

{

connection.Close();

LoadDataFromTable();

bsStuff.Position = ii;

}

}

}

private void dgvTypeTO\_RowPostPaint(object sender, DataGridViewRowPostPaintEventArgs e)

{

var grid = sender as DataGridView;

var rowIdx = (e.RowIndex + 1).ToString();

var centerFormat = new StringFormat()

{

// right alignment might actually make more sense for numbers

Alignment = StringAlignment.Center,

LineAlignment = StringAlignment.Center

};

var headerBounds = new Rectangle(e.RowBounds.Left, e.RowBounds.Top, grid.RowHeadersWidth, e.RowBounds.Height);

e.Graphics.DrawString(rowIdx, this.Font, SystemBrushes.ControlText, headerBounds, centerFormat);

}

void LoadCombo()

{

if ((bsStuff.Count > 0))

{

pictureBox1.Image = LoadImage();

} } }}

**Форма Оформление тортов**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Configuration;

using System.Data.SqlClient;

using System.Drawing.Imaging;

using System.IO;

using System.Windows.Forms.VisualStyles;

namespace Konditer

{

public partial class DecorForm : Form

{

string connectionString = ConfigurationManager.ConnectionStrings["SqlCon"].ConnectionString;

SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["SqlCon"].ConnectionString);

DataTable dtDecor;

SqlDataAdapter dataDecorAdapter;

BindingSource bsDecor;

bool add\_items;

public DecorForm()

{

InitializeComponent();

LoadDataFromTable();

add\_items = false;

}

void LoadDataFromTable()

{

try

{

dtDecor = new DataTable();

dataDecorAdapter = new SqlDataAdapter();

dataDecorAdapter.SelectCommand = new SqlCommand("SELECT \* " +

"FROM tDecor", connection);

dataDecorAdapter.Fill(dtDecor);

bsDecor = new BindingSource();

bsDecor.DataSource = dtDecor;

bindingNavigator1.BindingSource = bsDecor;

dgvTypeTO.DataSource = bsDecor;

dgvTypeTO.Columns[0].Visible = false;

dgvTypeTO.Columns[1].HeaderText = "Название";

dgvTypeTO.Columns[2].HeaderText = "Описание";

dgvTypeTO.Columns[3].HeaderText = "Цена за кг.";

dgvTypeTO.Columns[4].HeaderText = "Фото";

dgvTypeTO.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

((DataGridViewImageColumn)dgvTypeTO.Columns[4]).ImageLayout = DataGridViewImageCellLayout.Zoom;

tbName.DataBindings.Clear();

tbName.DataBindings.Add(new Binding("Text", bsDecor, "decor\_name"));

tbPrice.DataBindings.Clear();

tbPrice.DataBindings.Add(new Binding("Text", bsDecor, "price"));

tbInfo.DataBindings.Clear();

tbInfo.DataBindings.Add(new Binding("Text", bsDecor, "info"));

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

private void tbPrice\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) &&

(e.KeyChar != '.'))

{

e.Handled = true;

}

}

Image LoadImage()

{

if (bsDecor.Count > 0)

{

int i = bsDecor.Position;

if ((((DataRowView)this.bsDecor.Current).Row["photo"] != System.DBNull.Value))

{

byte[] byteArrayIn = (byte[])(((DataRowView)this.bsDecor.Current).Row["photo"]);

using (var ms = new MemoryStream(byteArrayIn))

{

return Image.FromStream(ms);

}

}

return null;

}

else return null;

}

private void deleteStripButton2\_Click(object sender, EventArgs e)

{

if (add\_items)

{

LoadDataFromTable();

add\_items = false;

return;

}

DeleteData();

}

/// <summary>

/// преобразует картинку в биты

/// </summary>

/// <param name="img"></param>

/// <returns></returns>

byte[] ConvertInBytes(Image img)

{

byte[] bytes;

using (MemoryStream ms = new MemoryStream())

{

img.Save(ms, ImageFormat.Png);

bytes = ms.ToArray();

}

return bytes;

}

private void saveToolStripButton\_Click(object sender, EventArgs e)

{

if ((tbName.Text == "") || (tbPrice.Text == "") || (tbInfo.Text == ""))

return;

if (add\_items)

{

SaveData();

}

else

{

UpdateData();

}

}

private void dgvTypeTO\_SelectionChanged(object sender, EventArgs e)

{

LoadCombo();

}

private void pictureBox1\_DoubleClick(object sender, EventArgs e)

{

if (openFileDialog1.ShowDialog() == DialogResult.Cancel)

return;

// получаем выбранный файл

try

{

string filename = openFileDialog1.FileName;

pictureBox1.Image = Image.FromFile(filename);// читаем файл в строку

}

catch

{

MessageBox.Show("Ошибка загрузки файла", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void btnAdd\_Click(object sender, EventArgs e)

{

ClearItems();

add\_items = true;

}

private void toolStripTextBox1\_TextChanged(object sender, EventArgs e)

{

// bsGoods.Filter = "ID\_good LIKE '%" + toolStripTextBox1.Text + "%'";

bsDecor.Filter = "decor\_name LIKE '%" + toolStripTextBox1.Text + "%'";

}

void ClearItems()

{

tbName.Text = "";

tbPrice.Text = "";

tbInfo.Text = "";

pictureBox1.Image = pictureBox1.InitialImage;

}

/// <summary>

/// DeleteData()

/// </summary>

void DeleteData()

{

if (bsDecor.Count > 0)

{

int i = bsDecor.Position;

int ID\_SS = Convert.ToInt32(((DataRowView)this.bsDecor.Current).Row["ID\_decor"]);

try

{

DialogResult result = MessageBox.Show("Вы действительно хотите удалить запись", "Внимание", MessageBoxButtons.YesNo);

if (result == DialogResult.No)

{

LoadDataFromTable();

return;

}

if (result == DialogResult.Yes)

{

connection.Close();

connection.Open();

SqlCommand commandDelete = new SqlCommand("Delete From tStuffing where ID\_stuffing = @ID", connection);

commandDelete.Parameters.AddWithValue("@ID", ID\_SS);

commandDelete.ExecuteNonQuery();

ClearItems();

}

}

catch (SqlException exception)

{

MessageBox.Show(exception.ToString());

}

finally

{

connection.Close();

LoadDataFromTable();

}

}

}

void SaveData()

{

int ID\_SS = 0;

try

{

connection.Close();

connection.Open();

if (pictureBox1.Image == null)

{

SqlCommand commandInsert = new SqlCommand("INSERT INTO [tDecor] VALUES" +

" (@Name,@Info, @Price, NULL)", connection);

commandInsert.Parameters.AddWithValue("@Name", tbName.Text);

commandInsert.Parameters.AddWithValue("@Info", tbInfo.Text);

commandInsert.Parameters.AddWithValue("@Price", tbPrice.Text);

commandInsert.ExecuteNonQuery();

}

else

{

SqlCommand commandInsert = new SqlCommand("INSERT INTO [tDecor] VALUES" +

" (@Name,@Info, @Price, @photo)", connection);

commandInsert.Parameters.AddWithValue("@Name", tbName.Text);

commandInsert.Parameters.AddWithValue("@Info", tbInfo.Text);

commandInsert.Parameters.AddWithValue("@Price", tbPrice.Text);

commandInsert.Parameters.AddWithValue("@photo", ConvertInBytes(pictureBox1.Image));

commandInsert.ExecuteNonQuery();

}

MessageBox.Show("Запись добавлена");

add\_items = false;

}

catch (SqlException exception)

{

MessageBox.Show(exception.ToString());

}

finally

{

connection.Close();

LoadDataFromTable();

}

}

void UpdateData()

{

if (bsDecor.Count > 0)

{

int i = bsDecor.Position;

int ii = bsDecor.Position;

try

{

connection.Close();

connection.Open();

int ID\_SS = Convert.ToInt32(((DataRowView)this.bsDecor.Current).Row["Id\_decor"]);

if (pictureBox1.Image == null)

{

SqlCommand commandUpdate = new SqlCommand("UPDATE tDecor SET" +

" decor\_name=@name, " +

" info=@info, price=@price," +

" WHERE Id\_decor = @IDSS", connection);

commandUpdate.Parameters.AddWithValue("@Name", tbName.Text);

commandUpdate.Parameters.AddWithValue("@Info", tbInfo.Text);

commandUpdate.Parameters.AddWithValue("@Price", tbPrice.Text);

commandUpdate.Parameters.AddWithValue("@IDSS", ID\_SS);

commandUpdate.ExecuteNonQuery();

}

else

{

SqlCommand commandUpdate = new SqlCommand("UPDATE tDecor SET" +

" decor\_name=@name, " +

" info=@info, price=@price, photo =@photo" +

" WHERE Id\_decor = @IDSS", connection);

commandUpdate.Parameters.AddWithValue("@Name", tbName.Text);

commandUpdate.Parameters.AddWithValue("@Info", tbInfo.Text);

commandUpdate.Parameters.AddWithValue("@Price", tbPrice.Text);

commandUpdate.Parameters.AddWithValue("@Photo", ConvertInBytes(pictureBox1.Image));

commandUpdate.Parameters.AddWithValue("@IDSS", ID\_SS);

commandUpdate.ExecuteNonQuery();

}

MessageBox.Show("Запись обновлена");

}

catch (SqlException exception)

{

MessageBox.Show(exception.ToString());

}

finally

{

connection.Close();

LoadDataFromTable();

bsDecor.Position = ii;

}

}

}

private void dgvTypeTO\_RowPostPaint(object sender, DataGridViewRowPostPaintEventArgs e)

{

var grid = sender as DataGridView;

var rowIdx = (e.RowIndex + 1).ToString();

var centerFormat = new StringFormat()

{

// right alignment might actually make more sense for numbers

Alignment = StringAlignment.Center,

LineAlignment = StringAlignment.Center

};

var headerBounds = new Rectangle(e.RowBounds.Left, e.RowBounds.Top, grid.RowHeadersWidth, e.RowBounds.Height);

e.Graphics.DrawString(rowIdx, this.Font, SystemBrushes.ControlText, headerBounds, centerFormat);

}

void LoadCombo()

{

if ((bsDecor.Count > 0))

{

pictureBox1.Image = LoadImage();

} } }}

**Форма Заказы**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Configuration;

using System.Data.SqlClient;

using System.Drawing.Imaging;

using System.IO;

using System.Windows.Forms.VisualStyles;

using Excel = Microsoft.Office.Interop.Excel;

namespace Konditer

{

public partial class OrdersForm : Form

{

string connectionString = ConfigurationManager.ConnectionStrings["SqlCon"].ConnectionString;

SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["SqlCon"].ConnectionString);

DataTable dtOrders;

SqlDataAdapter dataAdapterOrder;

BindingSource bsOrders;

private void dgvTypeTO\_DoubleClick(object sender, EventArgs e)

{

ShowOrderToChange();

}

private void tsbChange\_Click(object sender, EventArgs e)

{

ShowOrderToChange();

}

public OrdersForm()

{

InitializeComponent();

LoadDataFromTable();

dtpBegin.Value = DateTime.Today;

dtpEnd.Value = DateTime.Today;

}

private void chbStatus\_CheckedChanged(object sender, EventArgs e)

{

FilterData();

}

private void toolStripTextBox1\_TextChanged(object sender, EventArgs e)

{

FilterData();

}

private void chbDate\_CheckedChanged(object sender, EventArgs e)

{

FilterData();

}

private void tsbExcel\_Click(object sender, EventArgs e)

{

PrintExcel();

}

void FilterData()

{

bsOrders.RemoveFilter();

var queries = new List<string>();

if (chbStatus.Checked)

{

queries.Add(string.Format("[status]={0}", true));

}

if (chbDate.Checked)

{

queries.Add(string.Format("[date\_start] >=#{0}# AND [date\_start]<=#{1}#",

dtpBegin.Value.ToString(System.Globalization.CultureInfo.InvariantCulture.DateTimeFormat),

dtpEnd.Value.ToString(System.Globalization.CultureInfo.InvariantCulture.DateTimeFormat)));

}

if (toolStripTextBox1.Text != "")

{

queries.Add(string.Format("[ID\_order] = {0}", toolStripTextBox1.Text));

}

if (queries.Count >= 1)

{

var queryFilter = String.Join(" AND ", queries);

bsOrders.Filter = queryFilter;

}

}

void ShowOrderToChange()

{

if ((bsOrders.Count > 0) && (dgvTypeTO.SelectedRows.Count > 0))

{

int ID\_SS = Convert.ToInt32(((DataRowView)this.bsOrders.Current).Row["ID\_order"]);

NewOrderForm flat\_form = new NewOrderForm(ID\_SS);

flat\_form.ShowDialog();

LoadDataFromTable();

bsOrders.Position = bsOrders.Find("ID\_order", (ID\_SS));

}

}

private void tsbAdd\_Click(object sender, EventArgs e)

{

NewOrderForm x = new NewOrderForm();

x.ShowDialog();

LoadDataFromTable();

}

private void tsbDelete\_Click(object sender, EventArgs e)

{

if (bsOrders.Count > 0)

{

int i = bsOrders.Position;

int ID\_SS = Convert.ToInt32(((DataRowView)this.bsOrders.Current).Row["ID\_Order"]);

try

{

DialogResult result = MessageBox.Show("Вы действительно хотите удалить запись", "Внимание", MessageBoxButtons.YesNo);

if (result == DialogResult.No)

{

LoadDataFromTable();

return;

}

if (result == DialogResult.Yes)

{

connection.Close();

connection.Open();

SqlCommand commandDelete = new SqlCommand("Delete From tOrder where ID\_order = @ID\_order", connection);

commandDelete.Parameters.AddWithValue("@ID\_order", ID\_SS);

commandDelete.ExecuteNonQuery();

}

}

catch (SqlException exception)

{

if (exception.HResult == -2146232060)

MessageBox.Show("Ошибка удаления, есть связанные записи");

else MessageBox.Show(exception.Message);

}

finally

{

connection.Close();

LoadDataFromTable();

}

}

}

void LoadDataFromTable()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

try

{

dtOrders = new DataTable();

dataAdapterOrder = new SqlDataAdapter();

dataAdapterOrder.SelectCommand = new SqlCommand(" SELECT " +

" tOrder.ID\_order," +

" tOrder.price," +

" tOrder.date\_start," +

" tOrder.date\_end," +

" tOrder.comment," +

" tOrder.customer\_name," +

" tOrder.customer\_phone," +

" tOrder.customer\_email," +

" tOrder.ID\_cake," +

" tOrder.ID\_stuffing," +

" tOrder.status, " + " tOrder.weight " +

"FROM tOrder ", connection);

dataAdapterOrder.Fill(dtOrders);

bsOrders = new BindingSource();

bsOrders.DataSource = dtOrders;

bindingNavigator1.BindingSource = bsOrders;

dgvTypeTO.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

dgvTypeTO.DataSource = bsOrders;

dgvTypeTO.Columns[0].HeaderText = "Номер заказа";

dgvTypeTO.Columns[1].HeaderText = "Стоимость";

dgvTypeTO.Columns[3].HeaderText = "Дата подачи";

dgvTypeTO.Columns[2].HeaderText = "Дата выдачи";

dgvTypeTO.Columns[4].HeaderText = "Информация";

dgvTypeTO.Columns[5].HeaderText = "Клиент";

dgvTypeTO.Columns[6].HeaderText = "Телефон клиента";

dgvTypeTO.Columns[7].HeaderText = "e-mail клиента";

dgvTypeTO.Columns[8].Visible = false;

dgvTypeTO.Columns[9].Visible = false;

dgvTypeTO.Columns[10].HeaderText = "выполнен";

dgvTypeTO.Columns[11].HeaderText = "вес";

if (bsOrders.Count <= 0) tsbDelete.Enabled = false;

else tsbDelete.Enabled = true;

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

}

private void dgvTypeTO\_RowPostPaint(object sender, DataGridViewRowPostPaintEventArgs e)

{

var grid = sender as DataGridView;

var rowIdx = (e.RowIndex + 1).ToString();

var centerFormat = new StringFormat()

{

// right alignment might actually make more sense for numbers

Alignment = StringAlignment.Center,

LineAlignment = StringAlignment.Center

};

var headerBounds = new Rectangle(e.RowBounds.Left, e.RowBounds.Top, grid.RowHeadersWidth, e.RowBounds.Height);

e.Graphics.DrawString(rowIdx, this.Font, SystemBrushes.ControlText, headerBounds, centerFormat);

}

private void PrintExcel()

{

string fileName = System.Windows.Forms.Application.StartupPath + "\\" + "Orders" + ".xltx";

Excel.Application xlApp = new Excel.Application();

Excel.Worksheet xlSheet = new Excel.Worksheet();

try

{

//добавляем книгу

xlApp.Workbooks.Open(fileName, Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing);

//делаем временно неактивным документ

xlApp.Interactive = false;

xlApp.EnableEvents = false;

Excel.Range xlSheetRange;

//выбираем лист на котором будем работать (Лист 1)

xlSheet = (Excel.Worksheet)xlApp.Sheets[1];

//Название листа

xlSheet.Name = "Список";

int row = 2;

int i = 0;

if (dgvTypeTO.RowCount > 0)

{

for (i = 0; i < dgvTypeTO.RowCount; i++)

{

xlSheet.Cells[row, 1] = dgvTypeTO.Rows[i].Cells[0].Value.ToString();

string x = "";

xlSheet.Cells[row, 2] = dgvTypeTO.Rows[i].Cells[1].Value.ToString();

if (dgvTypeTO.Rows[i].Cells[2].Value != DBNull.Value)

x = Convert.ToDateTime(dgvTypeTO.Rows[i].Cells[2].Value).ToShortDateString();

xlSheet.Cells[row, 3] = x;

x = "";

if (dgvTypeTO.Rows[i].Cells[3].Value != DBNull.Value)

x = Convert.ToDateTime(dgvTypeTO.Rows[i].Cells[3].Value).ToShortDateString();

xlSheet.Cells[row, 4] = x;

xlSheet.Cells[row, 5] = dgvTypeTO.Rows[i].Cells[4].Value.ToString();

xlSheet.Cells[row, 6] = dgvTypeTO.Rows[i].Cells[5].Value.ToString();

xlSheet.Cells[row, 7] = dgvTypeTO.Rows[i].Cells[6].Value.ToString();

xlSheet.Cells[row, 8] = dgvTypeTO.Rows[i].Cells[7].Value.ToString();

xlSheet.Cells[row, 9] = dgvTypeTO.Rows[i].Cells[11].Value.ToString();

x = "не выполнен";

if (Convert.ToBoolean(dgvTypeTO.Rows[i].Cells[10].Value) == true)

x = "выполнен";

xlSheet.Cells[row, 10] = x;

row++;

Excel.Range r = xlSheet.get\_Range("A" + row.ToString(), "J" + row.ToString());

r.Insert(Excel.XlInsertShiftDirection.xlShiftDown);

}

}

row--;

xlSheetRange = xlSheet.get\_Range("A2:J" + row.ToString(), Type.Missing);

xlSheetRange.Borders.LineStyle = true;

row++;

//выбираем всю область данных\*/

xlSheetRange = xlSheet.UsedRange;

//выравниваем строки и колонки по их содержимому

xlSheetRange.Columns.AutoFit();

xlSheetRange.Rows.AutoFit();

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

finally

{

//Показываем ексель

xlApp.Visible = true;

xlApp.Interactive = true;

xlApp.ScreenUpdating = true;

xlApp.UserControl = true;

} } }}

**Форма Заказ**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Configuration;

using System.Data.SqlClient;

using System.IO;

using System.Drawing.Imaging;

using Konditer.models;

using Excel = Microsoft.Office.Interop.Excel;

namespace Konditer

{

public partial class NewOrderForm : Form

{

string connectionString = ConfigurationManager.ConnectionStrings["SqlCon"].ConnectionString;

SqlDataAdapter dataCategoryAdapter;

BindingSource bsCategory;

double DecorPrice = 0;

List<Tort> tortMainList, tortSecondList;

List<Stuffing> stuffMainList, stuffSecondList;

List<Decor> decorMainList, decorSecondList;

List<int> cakesCategoryList;

private ListViewItem lastItemChecked, lastItemChecked1, lastItemChecked2;

int ID\_order = -1;

int ID\_cake = -1;

int ID\_stuffing = -1;

bool add\_items;

Order order;

public NewOrderForm()

{

InitializeComponent();

LoadTort();

LoadStuffing();

LoadDecor();

loadCheckboxlist();

label1.Text = "";

add\_items = false;

ID\_order = -1;

dtpStopDate.Enabled = false;

}

public NewOrderForm(int order)

{

InitializeComponent();

LoadTort();

LoadStuffing();

LoadDecor();

loadCheckboxlist();

label1.Text = "";

add\_items = true;

ID\_order = order;

}

private void listView1\_ItemCheck(object sender, ItemCheckEventArgs e)

{

// if we have the lastItem set as checked, and it is different

// item than the one that fired the event, uncheck it

if (lastItemChecked != null && lastItemChecked.Checked

&& lastItemChecked != listViewTort.Items[e.Index])

{

// uncheck the last item and store the new one

lastItemChecked.Checked = false;

}

// store current item

ID\_cake = Convert.ToInt32(listViewTort.Items[e.Index].Tag);

// MessageBox.Show(ID\_cake.ToString());

lastItemChecked = listViewTort.Items[e.Index];

}

void loadCheckboxlist()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

dataCategoryAdapter = new SqlDataAdapter();

dataCategoryAdapter.SelectCommand = new SqlCommand("SELECT \* FROM tCakeCategory Order by category\_name", connection);

DataTable dt = new DataTable();

dataCategoryAdapter.Fill(dt);

bsCategory = new BindingSource();

bsCategory.DataSource = dt;

clbType.DataSource = bsCategory;

clbType.ValueMember = "ID\_cake\_category";

clbType.DisplayMember = "category\_name";

}

}

void FilterData()

{

if ((chbTort.Checked == false))

{

FullTortImageList(tortMainList);

return;

}

List<Tort> z = new List<Tort>();

tortSecondList = new List<Tort>();

tortSecondList = tortMainList;

List<int> y = new List<int>();

foreach (DataRowView L in clbType.CheckedItems)

{

y.Add(Convert.ToInt32(L[0]));

}

foreach (Tort s in tortSecondList)

{

if (s.cake\_category.Intersect(y).Any())

{

z.Add(s);

}

}

tortSecondList = z;

FullTortImageList(tortSecondList);

}

private void listView2\_ItemCheck(object sender, ItemCheckEventArgs e)

{

// if we have the lastItem set as checked, and it is different

// item than the one that fired the event, uncheck it

if (lastItemChecked1 != null && lastItemChecked1.Checked

&& lastItemChecked1 != listViewStuffing.Items[e.Index])

{

// uncheck the last item and store the new one

lastItemChecked1.Checked = false;

}

// store current item

ID\_stuffing = Convert.ToInt32(listViewStuffing.Items[e.Index].Tag);

//MessageBox.Show(ID\_stuffing.ToString());

lastItemChecked1 = listViewStuffing.Items[e.Index];

}

private void listView2\_ItemChecked(object sender, ItemCheckedEventArgs e)

{

if (e.Item.Checked)

{

tbCostStuffing.Text = e.Item.SubItems[2].Text.ToString();

}

else tbCostStuffing.Text = "0";

PriceForWeight();

}

private void chbTort\_CheckedChanged(object sender, EventArgs e)

{

FilterData();

}

private void NewOrderForm\_Load(object sender, EventArgs e)

{

LoadDataFromTable();

}

private void tsbSave\_Click(object sender, EventArgs e)

{

if ((tbCustomerName.Text == "")

|| (tbCustomerPhone.Text == "")

|| (tbCustomerEmail.Text == "")

|| (tbCostDecor.Text == "0")

|| (tbCostStuffing.Text == "")

)

{

MessageBox.Show("Ключевые поля пустые");

return;

}

if (add\_items == true)

{

UpdateData();

}

else

{

SaveData();

}

}

private void listView3\_ItemChecked(object sender, ItemCheckedEventArgs e)

{

PriceForWeight();

}

private void numericUpDown2\_ValueChanged(object sender, EventArgs e)

{

PriceForWeight();

}

private void chbStatus\_CheckedChanged(object sender, EventArgs e)

{

if (chbStatus.Checked)

{

dtpStopDate.Enabled = true;

tsbAktDone.Enabled = true;

}

else

{

dtpStopDate.Enabled = false;

tsbAktDone.Enabled = false;

}

}

private void tsbAktPriema\_Click(object sender, EventArgs e)

{

PrintAktPriema();

}

private void PrintAktPriema()

{

string fileName = System.Windows.Forms.Application.StartupPath + "\\" + "AktPriema" + ".xltx";

Excel.Application xlApp = new Excel.Application();

Excel.Worksheet xlSheet = new Excel.Worksheet();

try

{

//добавляем книгу

xlApp.Workbooks.Open(fileName, Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing);

//делаем временно неактивным документ

xlApp.Interactive = false;

xlApp.EnableEvents = false;

Excel.Range xlSheetRange;

//выбираем лист на котором будем работать (Лист 1)

xlSheet = (Excel.Worksheet)xlApp.Sheets[1];

//Название листа

xlSheet.Name = "Акт Приема";

string tort\_name = "";

foreach (ListViewItem x in listViewTort.CheckedItems)

{

tort\_name = x.Text;

}

string stuffing\_name = "";

foreach (ListViewItem x in listViewStuffing.CheckedItems)

{

stuffing\_name = x.Text;

}

string decor\_name = "";

foreach (ListViewItem x in listViewDecor.CheckedItems)

{

decor\_name = decor\_name +", " + x.Text;

}

decor\_name = decor\_name.Substring(2);

xlSheet.Cells[2, 2] = order.ID\_order.ToString();

xlSheet.Cells[4, 2] = order.customer\_name;

xlSheet.Cells[5, 2] = order.customer\_phone;

xlSheet.Cells[6, 2] = order.customer\_email;

xlSheet.Cells[8, 2] = order.date\_start.ToShortDateString();

xlSheet.Cells[9, 2] = order.weight.ToString();

xlSheet.Cells[10, 2] = tort\_name;

xlSheet.Cells[11, 2] = stuffing\_name;

xlSheet.Cells[12, 2] = decor\_name;

xlSheet.Cells[13, 2] = order.comment;

xlSheet.Cells[14, 2] = order.price.ToString();

xlSheet.Cells[17, 2] = "/" + getFam(order.customer\_name) + "/";

//выбираем всю область данных\*/

xlSheetRange = xlSheet.UsedRange;

//выравниваем строки и колонки по их содержимому

//xlSheetRange.Columns.AutoFit();

//xlSheetRange.Rows.AutoFit();

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

finally

{

//Показываем ексель

xlApp.Visible = true;

xlApp.Interactive = true;

xlApp.ScreenUpdating = true;

xlApp.UserControl = true;

}

}

string getFam(string s)

{

int k = s.Length;

string Fam = "", imya = "", otch = "";

Fam = s.Substring(0, s.IndexOf(' '));

int l = s.IndexOf(' ');

if (l > 0 && s.Length > 0)

{

s = s.Remove(0, s.IndexOf(' ') + 1);

imya = s.Substring(0, 1) + ".";

}

else s = "";

l = s.IndexOf(' ');

if (l > 0 && s.Length > 0)

{

s = s.Remove(0, s.IndexOf(' ') + 1);

otch = s.Substring(0, 1) + ". ";

}

return imya + otch + " " + Fam;

}

private void PrintAktDone()

{

string fileName = System.Windows.Forms.Application.StartupPath + "\\" + "AktDone" + ".xltx";

Excel.Application xlApp = new Excel.Application();

Excel.Worksheet xlSheet = new Excel.Worksheet();

try

{

//добавляем книгу

xlApp.Workbooks.Open(fileName, Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing, Type.Missing, Type.Missing,

Type.Missing, Type.Missing);

//делаем временно неактивным документ

xlApp.Interactive = false;

xlApp.EnableEvents = false;

Excel.Range xlSheetRange;

//выбираем лист на котором будем работать (Лист 1)

xlSheet = (Excel.Worksheet)xlApp.Sheets[1];

//Название листа

xlSheet.Name = "Акт выполненных работ";

string tort\_name = "";

foreach (ListViewItem x in listViewTort.CheckedItems)

{

tort\_name = x.Text;

}

string stuffing\_name = "";

foreach (ListViewItem x in listViewStuffing.CheckedItems)

{

stuffing\_name = x.Text;

}

string decor\_name = "";

foreach (ListViewItem x in listViewDecor.CheckedItems)

{

decor\_name = decor\_name + ", " + x.Text;

}

decor\_name = decor\_name.Substring(2);

xlSheet.Cells[2, 2] = order.ID\_order.ToString();

xlSheet.Cells[4, 2] = order.customer\_name;

xlSheet.Cells[5, 2] = order.customer\_phone;

xlSheet.Cells[6, 2] = order.customer\_email;

xlSheet.Cells[8, 2] = order.date\_start.ToShortDateString();

xlSheet.Cells[9, 2] = order.date\_end.ToShortDateString();

xlSheet.Cells[10, 2] = order.weight.ToString();

xlSheet.Cells[11, 2] = tort\_name;

xlSheet.Cells[12, 2] = stuffing\_name;

xlSheet.Cells[13, 2] = decor\_name;

xlSheet.Cells[14, 2] = order.comment;

xlSheet.Cells[15, 2] = order.price.ToString();

xlSheet.Cells[18, 2] = "/"+getFam(order.customer\_name)+"/";

//выбираем всю область данных\*/

xlSheetRange = xlSheet.UsedRange;

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

finally

{

//Показываем ексель

xlApp.Visible = true;

xlApp.Interactive = true;

xlApp.ScreenUpdating = true;

xlApp.UserControl = true;

}

}

private void tsbAktDone\_Click(object sender, EventArgs e)

{

PrintAktDone();

}

byte[] ConvertInBytes(Image img)

{

byte[] bytes;

using (MemoryStream ms = new MemoryStream())

{

img.Save(ms, ImageFormat.Png);

bytes = ms.ToArray();

}

return bytes;

}

void LoadDataFromTable()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

try

{

SqlCommand SelectCommand = new SqlCommand(" SELECT " +

" tOrder.ID\_order," +

" tOrder.price," +

" tOrder.date\_start," +

" tOrder.date\_end," +

" tOrder.comment," +

" tOrder.customer\_name," +

" tOrder.customer\_phone," +

" tOrder.customer\_email," +

" tOrder.ID\_cake," +

" tOrder.ID\_stuffing," +

" tOrder.status, " +

" tOrder.weight " +

"FROM tOrder WHERE ID\_order = " + ID\_order, connection);

SqlDataReader reader = SelectCommand.ExecuteReader();

if (reader.HasRows)

{

reader.Read();

order = new Order();

order.ID\_order = Convert.ToInt32(reader.GetValue(0));

order.price = Convert.ToDouble(reader.GetValue(1));

order.date\_start = Convert.ToDateTime(reader.GetValue(2));

if (reader.GetValue(3) != DBNull.Value)

{

order.date\_end = Convert.ToDateTime(reader.GetValue(3));

dtpStopDate.Value = order.date\_end;

}

order.comment = Convert.ToString(reader.GetValue(4));

order.customer\_name = Convert.ToString(reader.GetValue(5));

order.customer\_phone = Convert.ToString(reader.GetValue(6));

order.customer\_email = Convert.ToString(reader.GetValue(7));

order.ID\_cake = Convert.ToInt32(reader.GetValue(8));

order.ID\_stuffing = Convert.ToInt32(reader.GetValue(9));

order.status = Convert.ToBoolean(reader.GetValue(10));

order.weight = Convert.ToDouble(reader.GetValue(11));

label1.Text = order.ID\_order.ToString();

dtpStartDate.Value = order.date\_start;

tbComment.Text = order.comment;

tbITOGO.Text = order.price.ToString();

if (order.status)

{

dtpStopDate.Enabled = true;

tsbAktDone.Enabled = true;

}

else

{

dtpStopDate.Enabled = false;

tsbAktDone.Enabled = false;

}

chbStatus.Checked = order.status;

tbCustomerName.Text = order.customer\_name;

tbCustomerPhone.Text = order.customer\_phone;

tbCustomerEmail.Text = order.customer\_email;

numericUpDown2.Value = Convert.ToDecimal(order.weight);

foreach (ListViewItem x in listViewTort.Items)

{

if (x.Tag.ToString() == order.ID\_cake.ToString())

x.Checked = true;

}

foreach (ListViewItem x in listViewStuffing.Items)

{

if (x.Tag.ToString() == order.ID\_stuffing.ToString())

x.Checked = true;

}

LoadDecorOfOrder(order);

ID\_order = order.ID\_order;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

}

void LoadDecorOfOrder(Order new\_order)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

try

{

for (int i = 0; i < clbType.Items.Count; i++)

{

clbType.SetItemChecked(i, false);

}

SqlCommand SelectCommand = new SqlCommand("select ID\_decor FROM tDecorCake WHERE ID\_order =" + new\_order.ID\_order.ToString(), connection);

SqlDataReader dataReader = SelectCommand.ExecuteReader();

if (dataReader.HasRows) // если есть данные

{

List<int> x = new List<int>();

while (dataReader.Read()) // построчно считываем данные

{

x.Add(Convert.ToInt32(dataReader.GetValue(0)));

}

new\_order.iddecor = x;

foreach (ListViewItem item in listViewDecor.Items)

{

if (x.Contains(Convert.ToInt32(item.Tag)))

item.Checked = true;

}

}

}

catch

(Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

}

void SaveData()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

int ID\_SS = 0;

try

{

SqlCommand commandInsert = new SqlCommand("INSERT INTO [tOrder] VALUES(" +

"@price," +

"@date\_start," +

"@date\_end," +

"@comment," +

"@customer\_name," +

"@customer\_phone," +

"@customer\_email," +

"@ID\_cake," +

"@ID\_stuffing," +

"@status," +

"@weight" +

") ; SELECT SCOPE\_IDENTITY()", connection);

commandInsert.Parameters.AddWithValue("@price", Convert.ToDouble(tbITOGO.Text));

commandInsert.Parameters.AddWithValue("@date\_start", dtpStartDate.Value);

if (chbStatus.Checked)

commandInsert.Parameters.AddWithValue("@date\_end", dtpStopDate.Value);

else

commandInsert.Parameters.AddWithValue("@date\_end", DBNull.Value);

commandInsert.Parameters.AddWithValue("@comment",tbComment.Text);

commandInsert.Parameters.AddWithValue("@customer\_name", tbCustomerName.Text);

commandInsert.Parameters.AddWithValue("@customer\_phone", tbCustomerPhone.Text);

commandInsert.Parameters.AddWithValue("@customer\_email", tbCustomerEmail.Text);

commandInsert.Parameters.AddWithValue("@ID\_cake", ID\_cake);

commandInsert.Parameters.AddWithValue("@ID\_stuffing", ID\_stuffing);

commandInsert.Parameters.AddWithValue("@status", Convert.ToBoolean(chbStatus.Checked));

commandInsert.Parameters.AddWithValue("@weight", Convert.ToDouble(numericUpDown2.Value));

ID\_order = Convert.ToInt32(commandInsert.ExecuteScalar());

foreach (ListViewItem x in listViewDecor.CheckedItems)

{

commandInsert = new SqlCommand("INSERT INTO [tDecorCake] VALUES(" +

"@ID\_order," +

"@ID\_decor)" , connection);

commandInsert.Parameters.AddWithValue("@ID\_order", ID\_order);

commandInsert.Parameters.AddWithValue("@ID\_decor", Convert.ToInt32(x.Tag));

commandInsert.ExecuteNonQuery();

}

MessageBox.Show("Запись добавлена");

add\_items = true;

}

catch (SqlException exception)

{

MessageBox.Show(exception.ToString());

}

finally

{

LoadDataFromTable();

}

}

}

void UpdateData()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

try

{

SqlCommand commandUpdate = new SqlCommand("UPDATE tOrder SET" +

" price=@price," +

"date\_start=@date\_start," +

"date\_end=@date\_end," +

"comment=@comment," +

"customer\_name=@customer\_name," +

"customer\_phone=@customer\_phone," +

"customer\_email=@customer\_email," +

"ID\_cake=@ID\_cake," +

"ID\_stuffing=@ID\_stuffing," +

"status=@status, weight=@weight" +

" WHERE ID\_order= @IDSS", connection);

commandUpdate.Parameters.AddWithValue("@price", Convert.ToDouble(tbITOGO.Text));

commandUpdate.Parameters.AddWithValue("@date\_start", dtpStartDate.Value);

if (chbStatus.Checked)

commandUpdate.Parameters.AddWithValue("@date\_end", dtpStopDate.Value);

else

commandUpdate.Parameters.AddWithValue("@date\_end", DBNull.Value);

commandUpdate.Parameters.AddWithValue("@comment", tbComment.Text);

commandUpdate.Parameters.AddWithValue("@customer\_name", tbCustomerName.Text);

commandUpdate.Parameters.AddWithValue("@customer\_phone", tbCustomerPhone.Text);

commandUpdate.Parameters.AddWithValue("@customer\_email", tbCustomerEmail.Text);

commandUpdate.Parameters.AddWithValue("@ID\_cake", ID\_cake);

commandUpdate.Parameters.AddWithValue("@ID\_stuffing", ID\_stuffing);

commandUpdate.Parameters.AddWithValue("@status", Convert.ToBoolean(chbStatus.Checked));

commandUpdate.Parameters.AddWithValue("@weight", Convert.ToDouble(numericUpDown2.Value));

commandUpdate.Parameters.AddWithValue("@IDSS", order.ID\_order);

commandUpdate.ExecuteNonQuery();

List<int> s = new List<int>();

foreach (ListViewItem x in listViewDecor.CheckedItems)

{

if (!(order.iddecor.Contains(Convert.ToInt32(x.Tag))))

{

SqlCommand commandInsert = new SqlCommand("INSERT INTO [tDecorCake] VALUES(" +

"@ID\_order," +

"@ID\_decor)", connection);

commandInsert.Parameters.AddWithValue("@ID\_order", ID\_order);

commandInsert.Parameters.AddWithValue("@ID\_decor", Convert.ToInt32(x.Tag));

commandInsert.ExecuteNonQuery();

}

s.Add(Convert.ToInt32(x.Tag));

}

foreach (int x in order.iddecor)

{

if (!s.Contains(x))

{

SqlCommand delete = new SqlCommand("DELETE FROM [tDecorCake] WHERE "+

"(ID\_order= @ID\_order) and (ID\_decor = @ID\_decor)", connection);

delete.Parameters.AddWithValue("@ID\_order", ID\_order);

delete.Parameters.AddWithValue("@ID\_decor", Convert.ToInt32(x));

delete.ExecuteNonQuery();

}

}

MessageBox.Show("Запись обновлена");

}

catch (SqlException exception)

{

MessageBox.Show(exception.ToString());

}

finally

{

LoadDataFromTable();

}

}

}

void LoadTort()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlDataAdapter daTort = new SqlDataAdapter("SELECT \* FROm tCake", connection);

DataTable dt2 = new DataTable();

daTort.Fill(dt2);

tortMainList = new List<Tort>();

for (int i = 0; i < dt2.Rows.Count; i++)

{

DataRow dr = dt2.Rows[i];

Tort X = new Tort

{

ID\_cake = Convert.ToInt32(dr[0]),

cake\_name = dr[1].ToString(),

photo = (byte[])(dr[2])

};

LoadCakeCategory(X);

tortMainList.Add(X);

}

FullTortImageList(tortMainList);

}

}

void LoadCakeCategory(Tort cake)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

try

{

for (int i = 0; i < clbType.Items.Count; i++)

{

clbType.SetItemChecked(i, false);

}

SqlCommand SelectCommand =

new SqlCommand("select ID\_cake\_category FROM dbo.tCakeAndCategory WHERE ID\_cake =" + cake.ID\_cake.ToString(),

connection);

SqlDataReader dataReader = SelectCommand.ExecuteReader();

if (dataReader.HasRows) // если есть данные

{

List<int> x = new List<int>();

while (dataReader.Read()) // построчно считываем данные

{

x.Add(Convert.ToInt32(dataReader.GetValue(0)));

}

cake.cake\_category = x;

}

}

catch

(Exception ex)

{

MessageBox.Show(ex.ToString());

}

}

}

void FullTortImageList(List<Tort> X)

{

imageList1.Images.Clear();

listViewTort.Items.Clear();

listViewTort.LargeImageList = imageList1;

int ImageIndex = 0;

foreach (Tort s in X)

{

ListViewItem listitem = new ListViewItem();

listitem.Text = s.cake\_name;

listitem.Tag = s.ID\_cake.ToString();

listitem.ForeColor = Color.Black;

listitem.Font = new Font("Arial", 10, FontStyle.Bold);

listitem.UseItemStyleForSubItems = false;

byte[] byteArrayIn = s.photo;

Image im;

using (var ms = new MemoryStream(byteArrayIn))

{

im = Image.FromStream(ms);

}

imageList1.Images.Add(im);

listitem.ImageIndex = ImageIndex;

ImageIndex++;

listViewTort.Items.Add(listitem);

}

}

void LoadStuffing()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlDataAdapter daStuff = new SqlDataAdapter("SELECT \* FROm tStuffing", connection);

DataTable dt2 = new DataTable();

daStuff.Fill(dt2);

stuffMainList = new List<Stuffing>();

for (int i = 0; i < dt2.Rows.Count; i++)

{

DataRow dr = dt2.Rows[i];

Stuffing X = new Stuffing

{

ID\_stuffing = Convert.ToInt32(dr[0]),

stuffing\_name = dr[1].ToString(),

price = Convert.ToDouble(dr[3]),

photo = (byte[])(dr[4])

};

stuffMainList.Add(X);

}

FullStuffingImageList(stuffMainList);

}

}

void FullStuffingImageList(List<Stuffing> X)

{

imageList2.Images.Clear();

listViewStuffing.Items.Clear();

listViewStuffing.LargeImageList = imageList2;

int ImageIndex = 0;

foreach (Stuffing s in X)

{

ListViewItem listitem = new ListViewItem();

listitem.Text = s.stuffing\_name;

listitem.Tag = s.ID\_stuffing.ToString();

listitem.ForeColor = Color.Black;

listitem.Font = new Font("Arial", 10, FontStyle.Bold);

listitem.UseItemStyleForSubItems = false;

byte[] byteArrayIn = s.photo;

Image im;

using (var ms = new MemoryStream(byteArrayIn))

{

im = Image.FromStream(ms);

}

imageList2.Images.Add(im);

listitem.ImageIndex = ImageIndex;

ListViewItem.ListViewSubItem subsItem = listitem.SubItems.Add("цена =" + s.price.ToString() + "р. за кг.");

subsItem.ForeColor = Color.DarkBlue;

subsItem.Font = new Font("Arial", 8, FontStyle.Bold);

listitem.SubItems.Add(s.price.ToString());

subsItem.ForeColor = Color.DarkBlue;

subsItem.Font = new Font("Arial", 8, FontStyle.Bold);

ImageIndex++;

listViewStuffing.Items.Add(listitem);

}

}

void LoadDecor()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlDataAdapter daDecor = new SqlDataAdapter("SELECT \* FROm tDecor", connection);

DataTable dt2 = new DataTable();

daDecor.Fill(dt2);

decorMainList = new List<Decor>();

for (int i = 0; i < dt2.Rows.Count; i++)

{

DataRow dr = dt2.Rows[i];

Decor X = new Decor

{

ID\_decor = Convert.ToInt32(dr[0]),

decor\_name = dr[1].ToString(),

price = Convert.ToDouble(dr[3]),

photo = (byte[])(dr[4])

};

decorMainList.Add(X);

}

FullDecorImageList(decorMainList);

}

}

void FullDecorImageList(List<Decor> X)

{

imageList3.Images.Clear();

listViewDecor.Items.Clear();

listViewDecor.LargeImageList = imageList3;

int ImageIndex = 0;

foreach (Decor s in X)

{

ListViewItem listitem = new ListViewItem();

listitem.Text = s.decor\_name;

listitem.Tag = s.ID\_decor.ToString();

listitem.ForeColor = Color.Black;

listitem.Font = new Font("Arial", 10, FontStyle.Bold);

listitem.UseItemStyleForSubItems = false;

byte[] byteArrayIn = s.photo;

Image im;

using (var ms = new MemoryStream(byteArrayIn))

{

im = Image.FromStream(ms);

}

imageList3.Images.Add(im);

listitem.ImageIndex = ImageIndex;

ListViewItem.ListViewSubItem subsItem = listitem.SubItems.Add("цена =" + s.price.ToString() + "р. за кг.");

subsItem.ForeColor = Color.DarkBlue;

subsItem.Font = new Font("Arial", 8, FontStyle.Bold);

listitem.SubItems.Add(s.price.ToString());

subsItem.ForeColor = Color.DarkBlue;

subsItem.Font = new Font("Arial", 8, FontStyle.Bold);

ImageIndex++;

listViewDecor.Items.Add(listitem);

}

}

void PriceForWeight()

{

double s = 0;

foreach (ListViewItem item in listViewDecor.Items)

{

if (item.Checked)

{

s += Convert.ToDouble(item.SubItems[2].Text);

}

}

DecorPrice = s;

tbCostDecor.Text = DecorPrice.ToString();

double x = Convert.ToDouble(tbCostStuffing.Text);

double y = Convert.ToDouble(numericUpDown2.Value);

tbPrice.Text = Convert.ToString(x\*y);

tbITOGO.Text = (DecorPrice + x \* y).ToString();

} }}

**Форма Загрузка**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Gurantii

{

public partial class StartForm : Form

{

public StartForm()

{

InitializeComponent();

progressBar1.Value = 0;

timer1.Start(); }

private void timer1\_Tick(object sender, EventArgs e)

{

if (progressBar1.Value == 100)

{

MainForm fMain = new MainForm();

fMain.Show();

this.Hide();

timer1.Stop();

}

else progressBar1.Value += 10;

} }}

**Класс Models**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Konditer.models

{

public class Tort

{

public int ID\_cake { get; set; }

public string cake\_name { get; set; }

public byte[] photo { get; set; }

public List<int> cake\_category { get; set; }

}

public class Decor

{

public int ID\_decor { get; set; }

public string decor\_name { get; set; }

public double price { get; set; }

public byte[] photo { get; set; }

}

public class Stuffing

{

public int ID\_stuffing { get; set; }

public string stuffing\_name { get; set; }

public double price { get; set; }

public byte[] photo { get; set; }

}

public class CakeCategory

{

public int ID\_cake\_category { get; set; }

}

public class Order

{

public int ID\_order { get; set; }

public double price { get; set; }

public DateTime date\_start { get; set; }

public DateTime date\_end { get; set; }

public string comment { get; set; }

public string customer\_name { get; set; }

public string customer\_phone { get; set; }

public string customer\_email { get; set; }

public int ID\_cake { get; set; }

public int ID\_stuffing { get; set; }

public bool status { get; set; }

public List<int> iddecor { get; set; }

public double weight { get; set; }

}}

**Приложение Б**

Автоматизированная информационная система «Кондитерское ателье»

Спецификация

460.ЗМК.00005-01 01

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Обозначение** | | |  | | --- | | **Наименование** | | |  | | --- | | **Примечание** | |
|  | |  | | --- | | Документация | |  |
| 460.ЗМК.00005-01 12 01 | Автоматизированная информационная система |  |
|  | Кондитерское ателье |  |
|  | Текст программы |  |
| 460.ЗМК.00005-01 13 01 | Автоматизированная информационная система |  |
|  | Кондитерское ателье |  |
|  | Описание программы |  |
| 460.ЗМК.00005-01 34 01 | Автоматизированная информационная система |  |
|  | Кондитерское ателье |  |
|  | Руководство пользователя |  |
| 460.ЗМК.00005-01 81 01 | Автоматизированная информационная система |  |
|  | Кондитерское ателье |  |
|  | Пояснительная записка |  |

**Приложение B**

**Диск с программным средством**