|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ ИВАНОВСКОЙ ОБЛАСТИ**  Областное государственное бюджетное профессиональное образовательное учреждение  «Ивановский промышленно-экономический колледж» | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | | | |  | |
|  | | | | |  | |
|  | | | | |  | |
|  | | | | |  | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
| **КУРСОВОЙ ПРОЕКТ** | | | | | | |
|  | | | | | | |
| **Проектирование и разработка автоматизированной информационной системы «Ресторан»** | | | | | | |
|  | | | | | | |
| **ИвПЭК. 09.02.03. 21** | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | **Специальность:** | | | **09.02.03 Программирование в компьютерных системах,**  **базовая подготовка** | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
| Руководитель курсового проекта | | | |  | | Н.С. Агупова |
| Выполнил обучающийся группы 302 | | | |  | | Д.А. Самарин |
|  | | | |  | |  |
|  | | | | | | |
|  | | | | | | |
| Курсовой проект выполнен и защищен с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | | | |
|  | | | | | | |
| «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. | | | | | | |
|  |  | | | | | |
|  | | |  | | |  |
|  | | |  | | |  |
|  | Иваново 2019 | | | | |  |

Содержание

[Введение 3](#_Toc8607588)

[Основные понятия баз данных 4](#_Toc8607589)

[1 Концептуальное проектирование 5](#_Toc8607590)

[1.1 Описание сущностей и предметной области 5](#_Toc8607591)

[1.2 Техническое задание 6](#_Toc8607592)

[2 Техно-рабочий проект 9](#_Toc8607593)

[2.1 Моделирование с использованием UML-диаграмм 9](#_Toc8607594)

[2.2 Проектирование базы данных 10](#_Toc8607595)

[2.3 Разработка макета приложения пользовательского интерфейса 12](#_Toc8607596)

[2.4 Обоснование выбора средств разработки 16](#_Toc8607597)

[3 Рабочая документация 18](#_Toc8607598)

[3.1 Описание разработки приложения 18](#_Toc8607599)

[3.2 Руководство пользователя 21](#_Toc8607600)

[Заключение 38](#_Toc8607601)

[Список использованных источников 39](#_Toc8607602)

[Приложение А 41](#_Toc8607603)

# Введение

В настоящее время в связи с развитием компьютерной техники появилась возможность автоматизировать различные производственные процессы. Эффективно решить эту задачу можно благодаря использованию информационных систем, на основе баз данных. Она позволяет хранить, обрабатывать и выдавать большие объемы данных в наиболее удобной форме. Их так же называют системами управления данными.

Подобные системы используются в различных сферах, в том числе и в ресторанном бизнесе. Увеличение скорости работы, хранение информации о истории всех действий с системой и многое другое, что позволяет увеличить прибыль и снизить нагрузку на персонал.

Информационная система должна содержать данные о меню, продажах, складских запасах. Цель курсового проекта разработать автоматизированную информационную систему для ресторана. К задачам курсового проекта относятся:

* Изучение предметной области;
* Разработка концептуальной, логической и физической модели базы данных;
* Создание запросов;
* Разработка пользовательского интерфейса.

# Основные понятия баз данных

База данных (БД) – это организованная структура, предназначенная для хранения, изменения и обработки взаимосвязанной информации, преимущественно больших объемов.

Система управления базами данных (СУБД) – совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

Целостность базы данных – свойство базы данных, означающее, что БД содержит полную и непротиворечивую информацию, необходимую для корректного функционирования приложений. Для обеспечения целостности накладывают ограничения целостности.

Сущность – это реальный или представляемый тип объекта, информация о котором должна сохраняться и быть доступна. В диаграммах сущность представляется в виде прямоугольника, содержащего имя сущности. При этом имя сущности – это имя типа, а не некоторого конкретного экземпляра этого типа.

Информационная система – автоматизированная система, результатом функционирования которой является представление выходной информации для последующего использования.

Первичный ключ – это ключ, однозначно идентифицирующий одну запись.

Транзитивная зависимость – это зависимость между неключевыми атрибутами.

# 1 Концептуальное проектирование

1.1 Описание сущностей и предметной области

Представление информации с помощью данных требует унифицированного подхода как независимого объекта моделирования. Поэтому выбор модели является одной из важнейших проблем.

В иерархической модели представления данных связи между данными можно охарактеризовать с помощью упорядоченного графа. Данные, расположенные на разных уровнях такой модели, называют сегментами. Сегмент может иметь одного предка и любое количество потомков.

Её простота является одновременно и плюсом, и минусом так как с её помощью описать сложные связи либо сложно и затратно, либо вообще невозможно.

Сетевая модель, в отличие от иерархической может обладать любым количеством предков, что позволяет создавать связи почти любой сложности. В связи с этим универсальность данной модели является одной из самых высоких, но из-за этого увеличивается и сложность понимания её структуры, обработки данных.

Реляционная модель в свою очередь представляет собой набор данных, представленных в виде двухмерных таблиц. Достоинствами реляционной модели представления данных являются ее понятность, простота и удобство практической реализации реляционных баз данных на ЭВМ. К недостаткам можно отнести невозможность представить предметную область в виде таблиц, более низкую скорость доступа к данным.

В результате анализа была выбрана реляционная модель и на её основе создана концептуальная модель, представленная на рисунке 1.

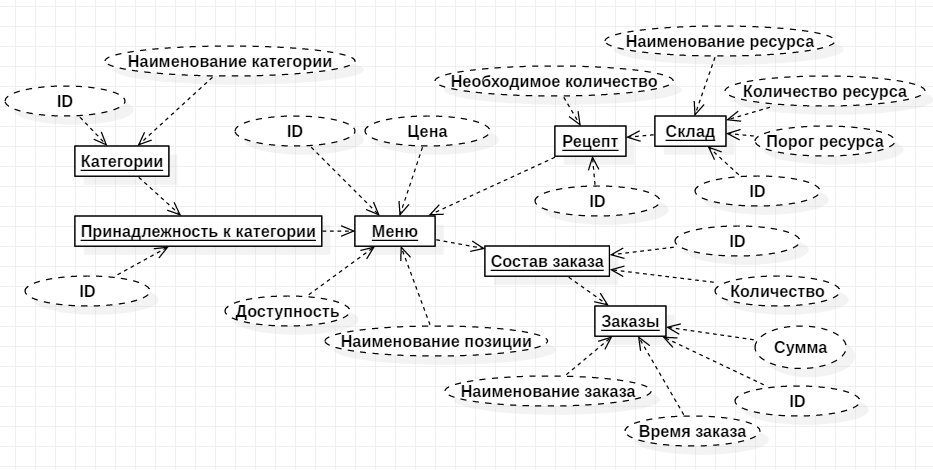


Рисунок 1 – Концептуальная модель

Соответственно, сущности, выделенные в проекте:

* меню;
* категории;
* принадлежность к категориям;
* рецепт;
* склад;
* состав заказа;
* заказ.

1.2 Техническое задание

1.2.1 Введение

Техническое задание – основополагающий документ при разработке программного обеспечения. В нем указаны все требования, используемые при разработке.

1.2.2 Основания для разработки

Основанием для разработки является задание на курсовой проект.

1.2.3 Назначение разработки

Разрабатываемая информационная система предназначена для учета продаж, а также складских запасов ресторана.

1.2.4 Требования к программе

В ресторане необходимо вести учет заказов и складских запасов. Для этого требуется хранить следующую информацию:

* меню (цена, наименование позиции, доступность);
* заказ (наименование заказа, время заказа, сумма заказа);
* содержимое заказа (количество);
* категории (наименование категории);
* рецепт (вид ресурса, количество ресурса);
* склад (наименование ресурса, количество ресурса, порог ресурса).

В базовых таблицах требуется выполнять базовые действия: добавление, редактирование, удаление, поиск, фильтрацию, изменение порядка информации.

Требования к дизайну:

Дизайн приложения должен быть выполнен в стандартных цветах системы. Размещение функциональных элементов должно быть логичным и удобным.

1.2.5 Требования к программной документации

Программная документация соответствует объему курсового проекта.

1.2.6 Технико-экономические показатели

Автоматизированная информационная система ресторан обеспечивает:

* Повышение скорости работы;
* Уменьшение нагрузки на персонал.

1.2.7 Стадии и этапы разработки

При разработке проекты были выделены следующие стадии:

* Концептуальное проектирование;
* Техническое задание;
* Техно-рабочий проект;
* Разработка программного обеспечения;
* Разработка документации.

1.2.8 Порядок приема и контроля

Прием и контроль разработки приложения осуществляется преподавателем. Исполнителем предоставляется вся требуемая документация и разработанное приложение на CD-диске.

# 2 Техно-рабочий проект

2.1 Моделирование с использованием UML-диаграмм

Диаграмма прецедентов (диаграмма вариантов использования) в UML — диаграмма, отражающая отношения между [актёрами](https://ru.wikipedia.org/wiki/%D0%90%D0%BA%D1%82%D1%91%D1%80_(UML)) и [прецедентами](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D1%86%D0%B5%D0%B4%D0%B5%D0%BD%D1%82_(UML)) и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Прецедент — возможность моделируемой системы (часть её функциональности), благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат. Прецедент соответствует отдельному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой. Варианты использования обычно применяются для спецификации внешних требований к системе.

Основное назначение диаграммы — описание функциональности и поведения, позволяющее заказчику, конечному пользователю и разработчику совместно обсуждать проектируемую или существующую систему.

При моделировании системы с помощью диаграммы прецедентов системный аналитик стремится:

* чётко отделить систему от её окружения;
* определить действующих лиц (актёров), их взаимодействие с системой и ожидаемую функциональность системы;
* определить в глоссарии предметной области понятия, относящиеся к детальному описанию функциональности системы (то есть прецедентов).

Работа над диаграммой может начаться с текстового описания, полученного при работе с заказчиком. При этом нефункциональные требования (например, конкретный язык или система программирования) при составлении модели прецедентов опускаются.

Диаграмма прецедентов разрабатываемого проекта размещена на рисунке 2.

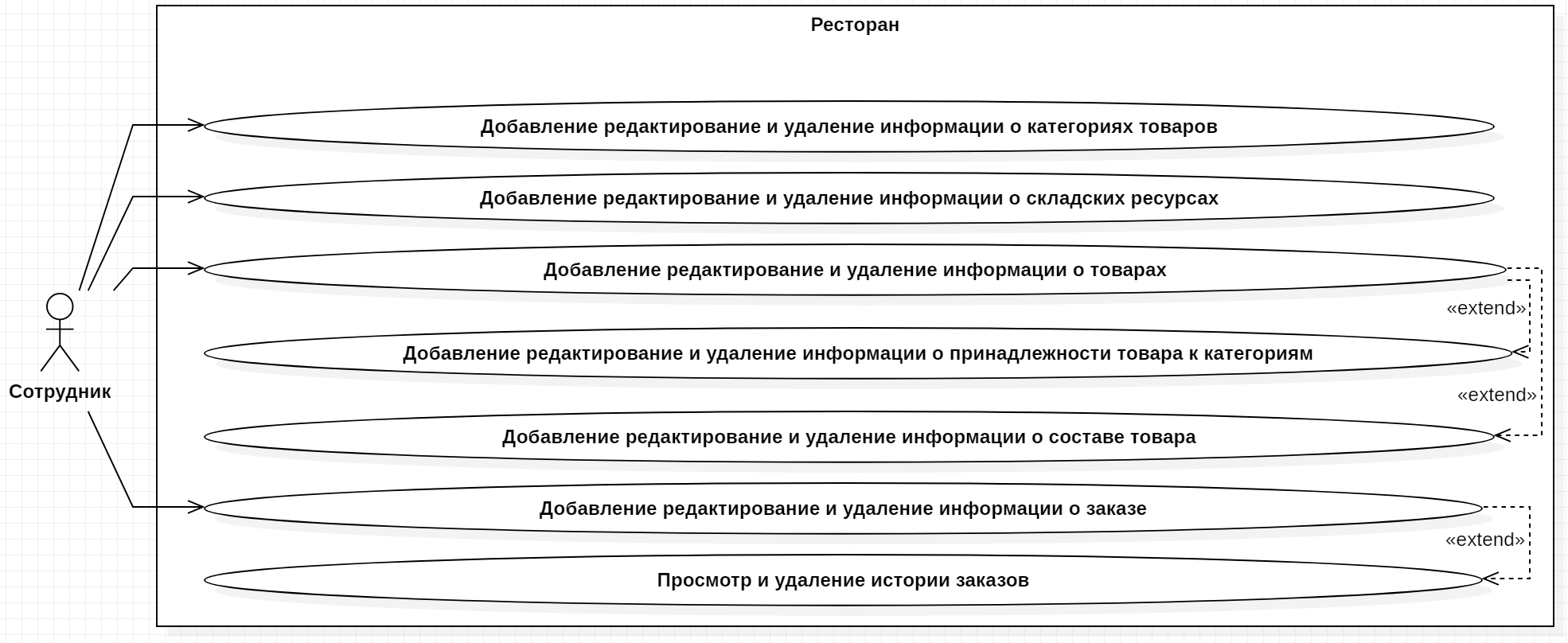


Рисунок 2 – Диаграмма прецедентов

2.2 Проектирование базы данных

Нормальная форма — свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, потенциально приводящей к логически ошибочным результатам выборки или изменения данных. Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение.

Процесс преобразования отношений базы данных к виду, отвечающему нормальным формам, называется нормализацией. Нормализация предназначена для приведения структуры БД к виду, обеспечивающему минимальную логическую избыточность, и не имеет целью уменьшение или увеличение производительности работы или же уменьшение, или увеличение физического объёма базы данных. Конечной целью нормализации является уменьшение потенциальной противоречивости, хранимой в базе данных информации. Как отмечает К. Дейт, общее назначение процесса нормализации заключается в следующем:

* исключение некоторых типов избыточности;
* устранение некоторых аномалий обновления;
* разработка проекта базы данных, который является достаточно «качественным» представлением реального мира, интуитивно понятен и может служить хорошей основой для последующего расширения;
* упрощение процедуры применения необходимых ограничений целостности.

Устранение избыточности производится, как правило, за счёт [декомпозиции](https://ru.wikipedia.org/wiki/%D0%9F%D1%8F%D1%82%D0%B0%D1%8F_%D0%BD%D0%BE%D1%80%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%84%D0%BE%D1%80%D0%BC%D0%B0#%D0%94%D0%B5%D0%BA%D0%BE%D0%BC%D0%BF%D0%BE%D0%B7%D0%B8%D1%86%D0%B8%D1%8F_%D0%B1%D0%B5%D0%B7_%D0%BF%D0%BE%D1%82%D0%B5%D1%80%D1%8C) отношений таким образом, чтобы в каждом отношении хранились только первичные факты (то есть факты, не выводимые из других хранимых фактов).

Первая нормальная форма (1NF): Переменная отношения находится в первой нормальной форме (1НФ) тогда и только тогда, когда в любом допустимом значении отношения каждый его кортеж содержит только одно значение для каждого из атрибутов. В реляционной модели отношение всегда находится в первой нормальной форме по определению понятия отношение. Что же касается различных таблиц, то они могут не быть правильными представлениями отношений и, соответственно, могут не находиться в 1НФ.

Вторая нормальная форма (2NF): Переменная отношения находится во второй нормальной форме тогда и только тогда, когда она находится в первой нормальной форме и каждый неключевой атрибут неприводимо (функционально полно) зависит от её потенциального ключа.

Третья нормальная форма (3NF): Переменная отношения находится в третьей нормальной форме тогда и только тогда, когда она находится во второй нормальной форме, и отсутствуют транзитивные функциональные зависимости неключевых атрибутов от ключевых.

После нормализации структуры базы данных были получены логическая и физическая модели базы данных, представленные на рисунках 3 и 4 соответственно.

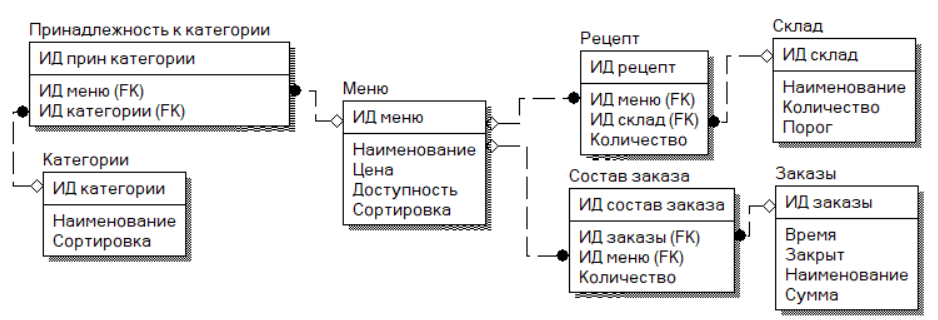


Рисунок 3 – Логическая модель базы данных

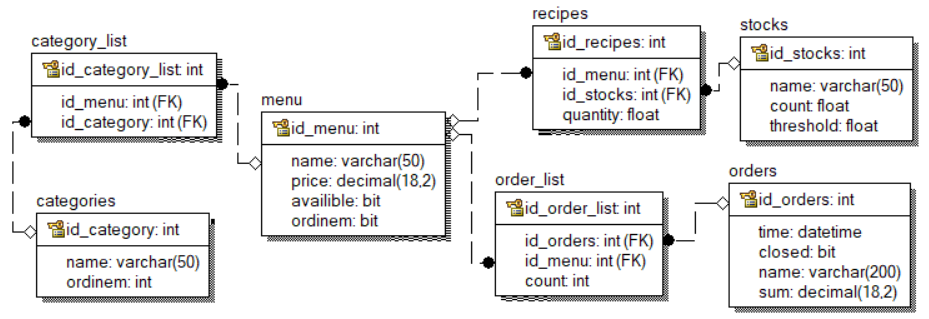


Рисунок 4 – Физическая модель базы данных

2.3 Разработка макета приложения пользовательского интерфейса

Для разработки приложения были разработаны макеты каждой формы, представленные на рисунках 5 – 14.



Рисунок 5 – Главная форма приложения



Рисунок 6 – Форма Категории



Рисунок 7 – Форма Изменение категории

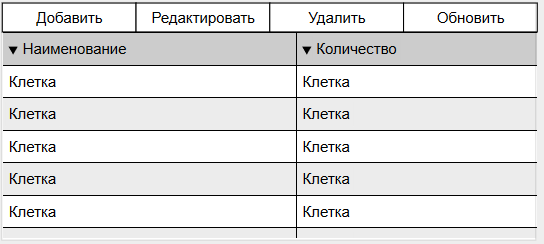


Рисунок 8 – Форма Склад

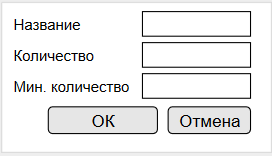


Рисунок 9 – Форма Изменение склада



Рисунок 10 – Форма Товары

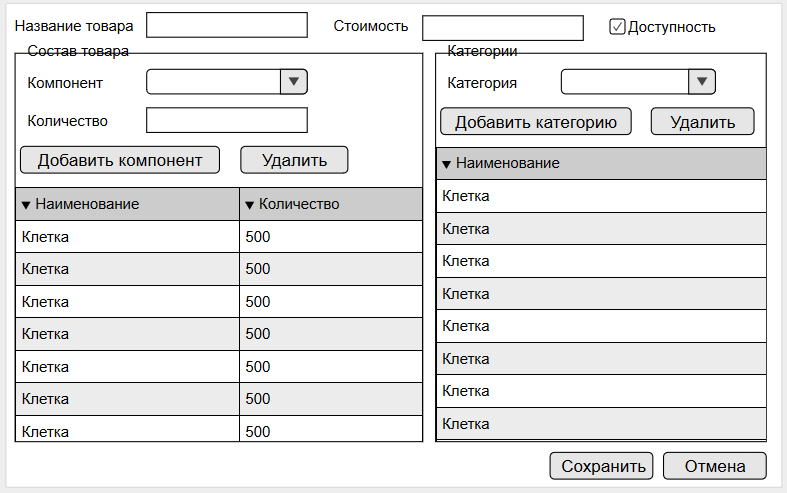


Рисунок 11 – Форма Изменение товара

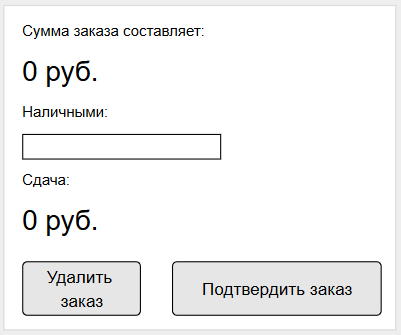


Рисунок 12 – Форма Закрытие заказа

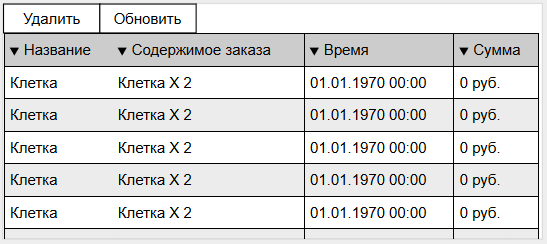


Рисунок 13 – Форма История заказов

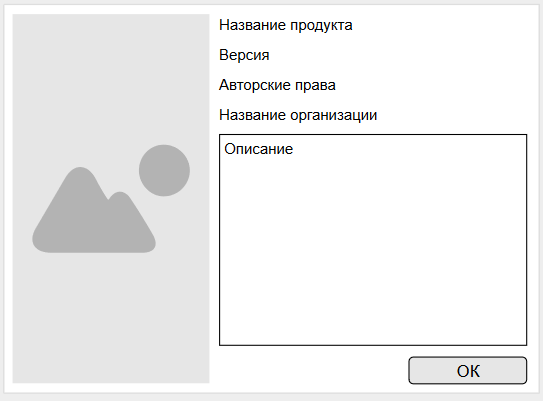


Рисунок 14 – Форма О программе

2.4 Обоснование выбора средств разработки

C# был выбран в качестве языка программирования из-за того, что он объектно-ориентированный язык программирования, в нем присутствует большое количество синтаксического сахара, работа на базе .NET Framework, который позволяет приложениям сначала транслироваться в IL, а уже при выполнении преобразовываться дальше в машинный код. Данная особенность позволяет приложениям использовать специфичные функции компьютера, на котором оно запускается.

В качестве среды разработки используется Visual Studio так как она обладает наилучшей поддержкой C#, имеет бесплатную версию для сообщества, наличие большого количества плагинов и шаблонов.

Большая часть программистов программирующих на C# используют Visual Studio и благодаря этому становится возможным найти решению почти любой проблемы при их использовании.

# 3 Рабочая документация

3.1 Описание разработки приложения

Так как приложение имеет клиент-серверную архитектуру база данных была размещена на сервере. Диаграмма БД изображена на рисунке 15.

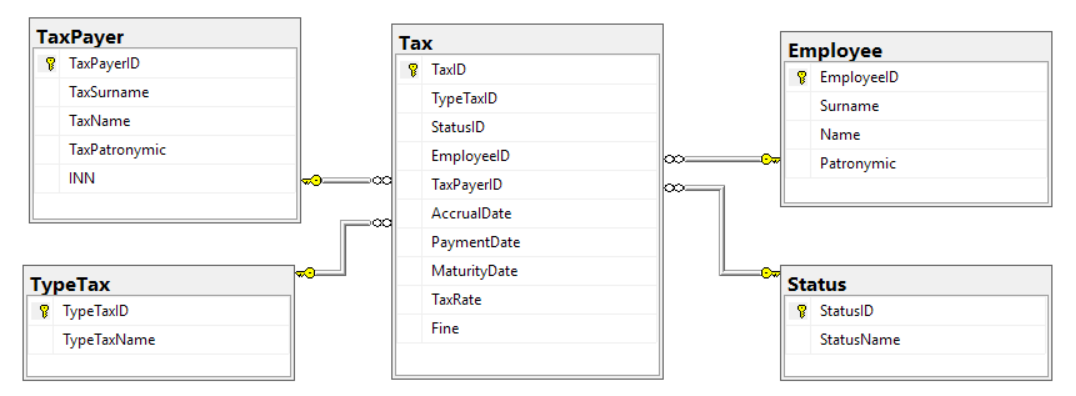


Рисунок 15– Диаграмма базы данных

После создания базы данных, необходимо спроектировать клиентскую часть приложения.

Для создания клиентской части необходимо запустить Microsoft Visual Studio и создать проект (рисунок 16).

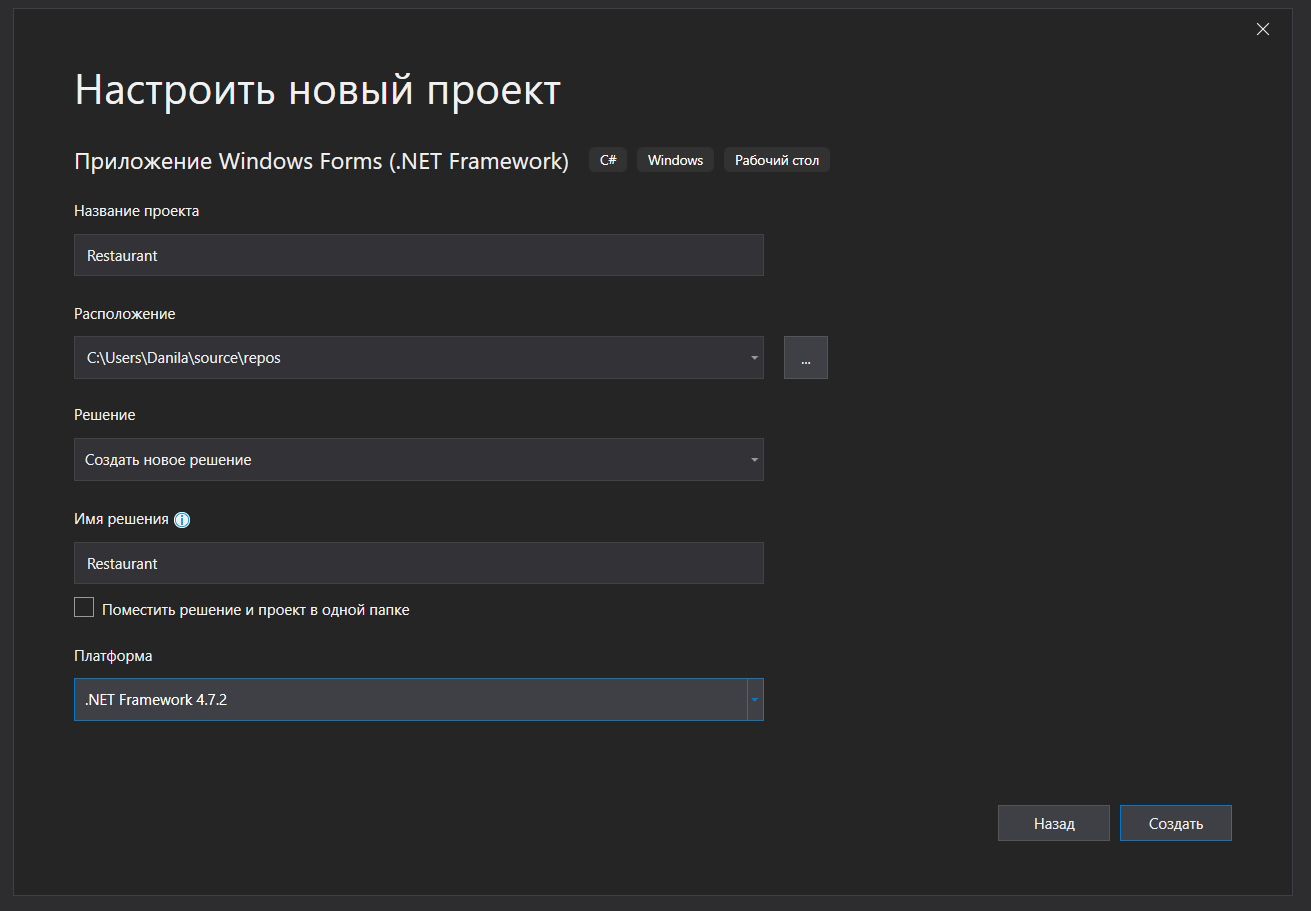


Рисунок 16 – Создание проекта в Microsoft Visual Studio

Затем создать подключение к БД с использованием конфигурационного файла:

Конфигурационный файл:

<?xml version="1.0" encoding="utf-8"?>

<configuration>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2"/>

</startup>

<connectionStrings>

<add name="RestourantDB" connectionString="Data Source=localhost;Initial Catalog=Restaurant\_DB;User ID=admin;Password=1234" providerName="System.Data.SqlClient"/>

</connectionStrings>

</configuration>

Подключение к БД:

string conectionString = ConfigurationManager.ConnectionStrings["RestourantDB"].ConnectionString;

sqlConnection = new SqlConnection(conectionString+ ";MultipleActiveResultSets=True");

sqlConnection.Open();

Код основных действий:

Добавление информации:

SqlCommand addStock = new SqlCommand("INSERT INTO [stocks] (name, count, threshold) VALUES (@name, @count, @threshold)", sqlConnection);

addStock.Parameters.AddWithValue("name", nameTextBox.Text);

addStock.Parameters.AddWithValue("count", countTextBox.Text);

addStock.Parameters.AddWithValue("threshold", minCountTextBox.Text);

try

{

await addStock.ExecuteNonQueryAsync();

Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

Редактирование:

SqlCommand updateStock = new SqlCommand("UPDATE [stocks] SET name=@name, count=@count, threshold=@threshold WHERE id\_stocks=@id", sqlConnection);

updateStock.Parameters.AddWithValue("name", nameTextBox.Text);

updateStock.Parameters.AddWithValue("count", countTextBox.Text);

updateStock.Parameters.AddWithValue("threshold", minCountTextBox.Text);

pdateStock.Parameters.AddWithValue("id", id);

try

{

await updateStock.ExecuteNonQueryAsync();

Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

Удаление:

if (listCategories.SelectedRows.Count > 0)

{

if (MessageBox.Show("Вы действительно хотите удалить эту категорию?", "Удаление категории", MessageBoxButtons.OKCancel, MessageBoxIcon.Exclamation) == DialogResult.OK) {

SqlCommand deleteCategory = new SqlCommand("DELETE FROM [categories] WHERE id\_category=@id; DELETE FROM [category\_list] WHERE id\_category=@id", sqlConnection);

deleteCategory.Parameters.AddWithValue("id", Convert.ToInt32(listCategories.SelectedRows[0].Cells[2].Value));

try

{

await deleteCategory.ExecuteNonQueryAsync();

await LoadCategoriesAsync();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

else

{

MessageBox.Show("Не выделена ни одна строка", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

Выборка:

menuDataGridView.Rows.Clear();

SqlDataReader sqlReader = null;

string query = "";

if (menuFilterTextBox.TextLength != 0 && menuFilterComboBox.SelectedIndex == 0) {

query = "SELECT \* FROM [menu] WHERE UPPER(name) like UPPER(@name)+'%' AND available='1' ORDER BY ordinem";

}

if (menuFilterTextBox.TextLength == 0 && menuFilterComboBox.SelectedIndex != 0)

{

query = "SELECT menu.\* FROM [menu], [category\_list] WHERE category\_list.id\_category=@categoryID AND category\_list.id\_menu=menu.id\_menu AND menu.available='1' ORDER BY ordinem";

}

if (menuFilterTextBox.TextLength != 0 && menuFilterComboBox.SelectedIndex != 0)

{

query = "SELECT menu.\* FROM [menu], [category\_list] WHERE category\_list.id\_category=@categoryID AND category\_list.id\_menu=menu.id\_menu AND UPPER(name) like UPPER(@name)+'%' AND menu.available='1' ORDER BY ordinem";

}

if (menuFilterTextBox.TextLength == 0 && menuFilterComboBox.SelectedIndex == 0)

{

query = "SELECT \* FROM [menu] WHERE available='1' ORDER BY ordinem";

}

SqlCommand loadCategoryList = new SqlCommand(query, sqlConnection);

if (menuFilterTextBox.TextLength != 0)

loadCategoryList.Parameters.AddWithValue("name", menuFilterTextBox.Text);

if (menuFilterComboBox.SelectedIndex != 0)

{

ComboboxValue combo = (ComboboxValue)menuFilterComboBox.SelectedItem;

loadCategoryList.Parameters.AddWithValue("categoryID", combo.Id);

}

try

{

sqlReader = loadCategoryList.ExecuteReader();

menuDataGridView.Rows.Clear();

while (sqlReader.Read())

{

int rowNumber = menuDataGridView.Rows.Add();

menuDataGridView.Rows[rowNumber].Cells[0].Value = Convert.ToInt32(sqlReader["id\_menu"]);

menuDataGridView.Rows[rowNumber].Cells[1].Value = Convert.ToString(sqlReader["name"]);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

finally

{

if (sqlReader != null && !sqlReader.IsClosed)

sqlReader.Close();

}

3.2 Руководство пользователя

После запуска приложения нужно заполнить список категорий. Для этого через меню приложения нужно выбрать пункт Категории (рисунок 17).

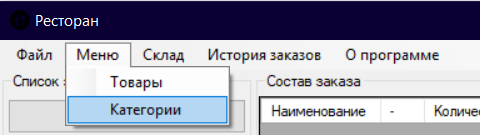


Рисунок 17 – Меню: Категории

Список категорий представлен на рисунке 18.

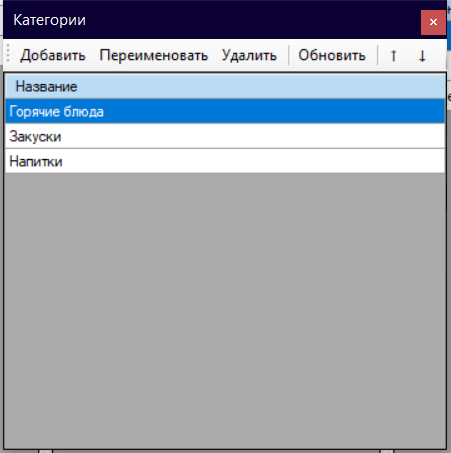


Рисунок 18 – Категории

Для добавления категории необходимо нажать кнопку Добавить на панели инструментов (рисунок 19), ввести наименование категории и нажать кнопку Сохранить (рисунок 20).

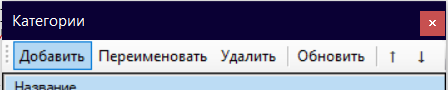


Рисунок 19 – Категории: Инструменты: Добавить

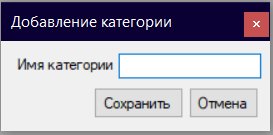


Рисунок 20 – Добавление категории

Для редактирования наименования категории необходимо выбрать одну из категорий в списке (рисунок 18), нажать кнопку Переименовать на панели инструментов (рисунок 21), изменить имя категории и нажать кнопку Сохранить (рисунок 22).

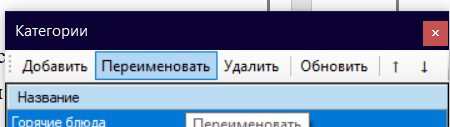


Рисунок 21 – Категории: Инструменты: Переименовать

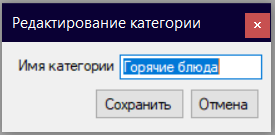


Рисунок 22 – Редактирование категории

Для удаления категории необходимо выбрать одну из категорий в списке (рисунок 18), нажать кнопку Удалить на панели инструментов (рисунок 23), подтвердить удаление кнопкой ОК (рисунок 24).

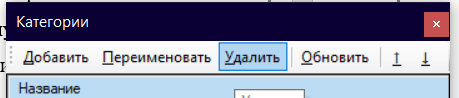


Рисунок 23 – Категории: Инструменты: Удалить

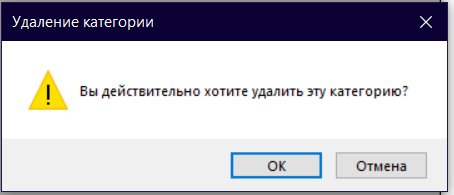


Рисунок 24 – Подтверждение удаления категории

Для принудительного обновления списка категорий необходимо нажать кнопку Обновить на панели инструментов (рисунок 25).

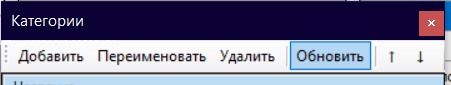


Рисунок 25 – Категории: Инструменты: Обновить

Для изменения порядка отображения категорий выбрать одну из категорий в списке (рисунок 18), нажать кнопку ↑ или ↓ на панели инструментов (рисунок 26).



Рисунок 26 – Категории: Инструменты: ↑ и ↓

После создания заполнения списка категорий нужно заполнить список складских запасов. Для этого через меню приложения нужно выбрать пункт Склад (рисунок 27).

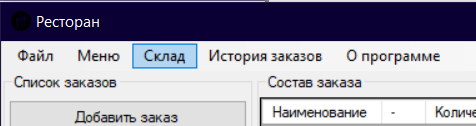


Рисунок 27 – Меню: Склад

Список ресурсов на складе представлен на рисунке 28. Ресурсы количество которых опустилось ниже минимального количества выделены красным цветом и подняты вверх списка.

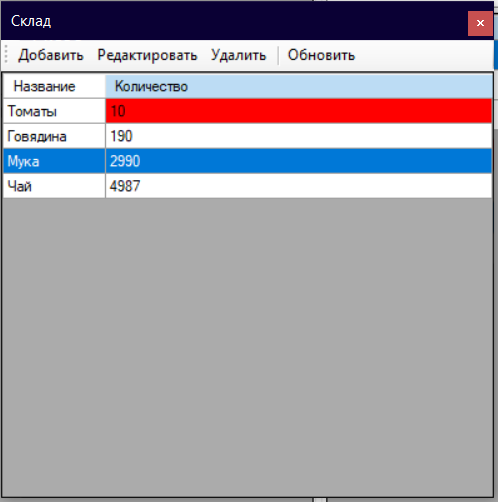


Рисунок 28 – Склад

Для добавления ресурса на складе необходимо нажать кнопку Добавить на панели инструментов (рисунок 29), ввести наименование ресурса, его количество, минимальное количество и нажать кнопку Сохранить (рисунок 30).

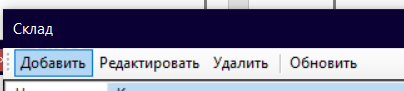


Рисунок 29 – Склад: Инструменты: Добавить

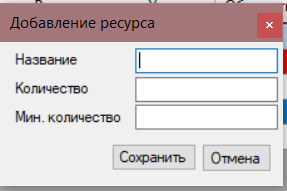


Рисунок 30 – Добавление ресурса

Для редактирования ресурса необходимо выбрать один из ресурсов в списке (рисунок 28), нажать кнопку Редактировать на панели инструментов (рисунок 31), изменить необходимые свойства и нажать кнопку Сохранить (рисунок 32).

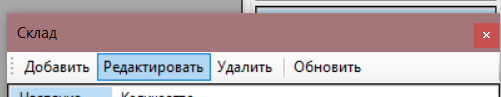


Рисунок 31 – Склад: Инструменты: Редактировать

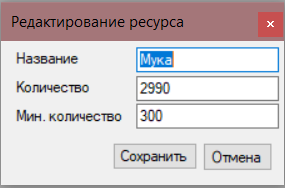


Рисунок 32 – Редактирование ресурса

Для удаления ресурса необходимо выбрать одну из них в списке (рисунок 28), нажать кнопку Удалить на панели инструментов (рисунок 33), подтвердить удаление кнопкой ОК (рисунок 34).

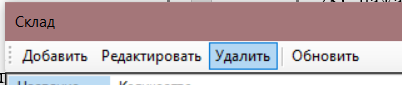


Рисунок 33 – Склад: Инструменты: Удалить

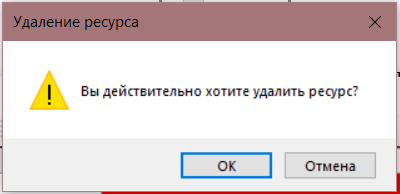


Рисунок 34 – Подтверждение удаления ресурса

Для принудительного обновления списка ресурсов необходимо нажать кнопку Обновить на панели инструментов (рисунок 35).

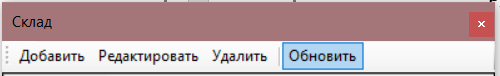


Рисунок 35 – Склад: Инструменты: Обновить

После создания заполнения списка складских запасов нужно заполнить список товаров. Для этого через меню приложения нужно выбрать пункт Товары (рисунок 36).

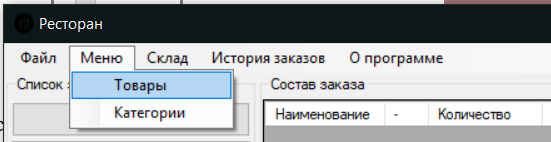


Рисунок 36 – Меню: Товары

Список товаров представлен на рисунке 37. Недоступные товары помечаются серым цветом и помещаются в конец списка. Крайне не рекомендуется удалять товары, вместо этого делайте их недоступными.

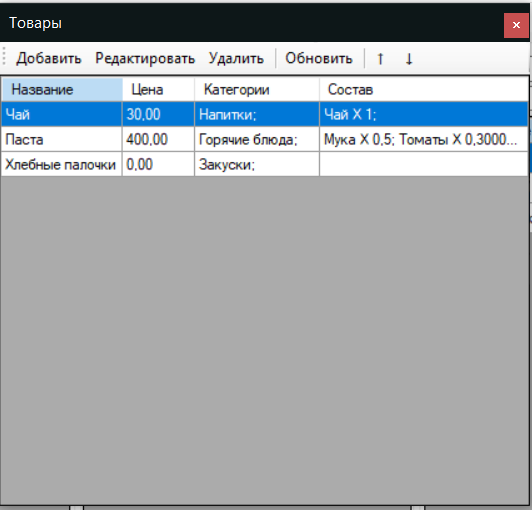


Рисунок 36 – Товары

Для добавления товара необходимо нажать кнопку Добавить на панели инструментов (рисунок 37), ввести наименование товара, его стоимость, доступность, состав, принадлежность к категориям и нажать кнопку Сохранить (рисунок 38). Для добавления ресурса в состав необходимо выбрать его из выпадающего списка в левой части формы, ниже ввести его количество и нажать кнопку Добавить компонент. Удаление ресурса из состава производится с выделения желаемого компонента из левой таблицы и нажатием кнопки Удалить. Для добавления принадлежности к категории выбрать её из выпадающего списка в правой части формы и нажать кнопку Добавить категорию. Удаление принадлежности к категории производится с выделения желаемой категории из правой таблицы и нажатием кнопки Удалить.

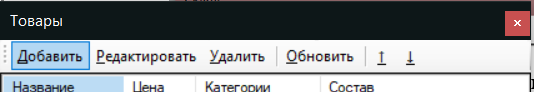


Рисунок 37 – Товары: Инструменты: Добавить

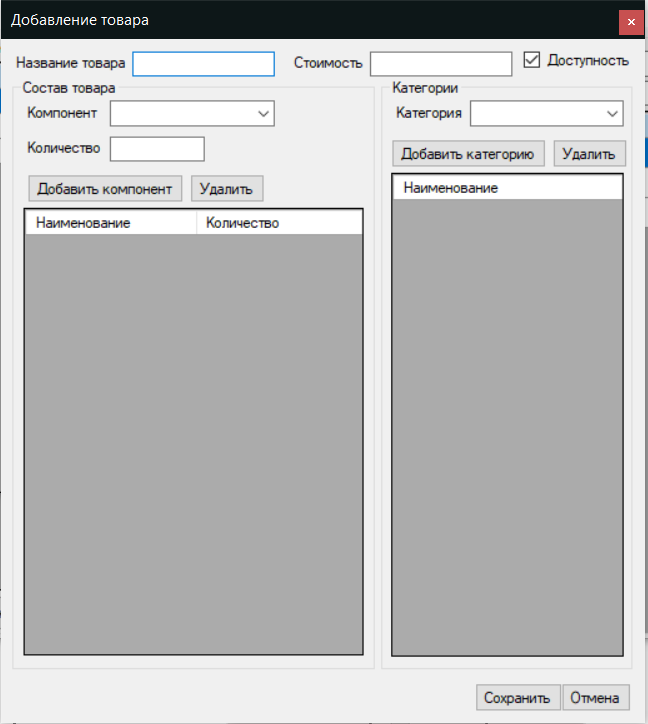


Рисунок 38 – Добавление товара

Для редактирования товара необходимо выбрать один из товаров в списке (рисунок 36), нажать кнопку Редактировать на панели инструментов (рисунок 39), изменить необходимые свойства по аналогии с процессом добавления нового товара и нажать кнопку Сохранить (рисунок 40).

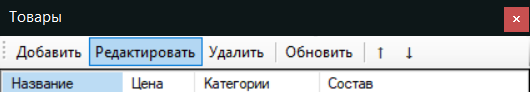


Рисунок 39 – Товары: Инструменты: Редактировать

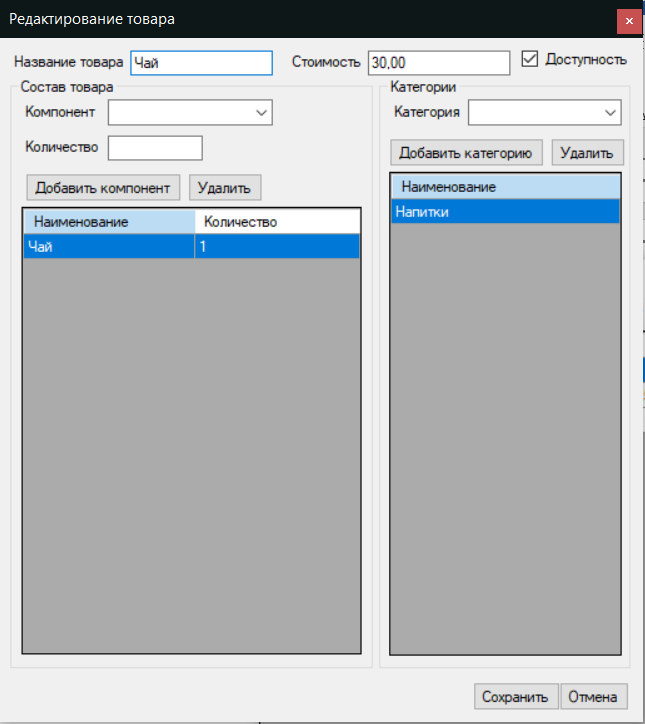


Рисунок 40 – Редактирование ресурса

Для удаления товара необходимо выбрать один из них в списке (рисунок 36), нажать кнопку Удалить на панели инструментов (рисунок 41), подтвердить удаление кнопкой ОК (рисунок 42). Рекомендуется вместо удаления отключать доступность товара.

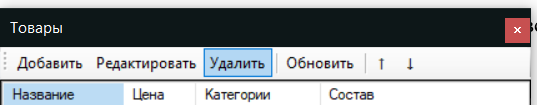


Рисунок 41 – Товары: Инструменты: Удалить

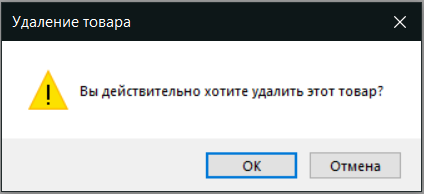


Рисунок 42 – Подтверждение удаления товара

Для принудительного обновления списка товаров необходимо нажать кнопку Обновить на панели инструментов (рисунок 43).

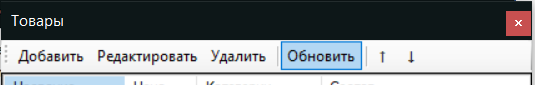


Рисунок 43 – Товары: Инструменты: Обновить

Для изменения порядка отображения товаров выбрать один из товаров в списке (рисунок 36), нажать кнопку ↑ или ↓ на панели инструментов (рисунок 44).

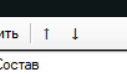


Рисунок 44 – Товары: Инструменты: ↑ и ↓

Форма заказов представлена на рисунке 45. Левая часть формы содержит список текущих заказов, кнопку для добавления нового заказа и кнопку для закрытия заказа. Средняя часть формы содержит информацию о составе выбранного заказа, с возможностью его редактирования. Правая часть содержит список меню с возможностью фильтрации по категории и нерегистрозависимого поиска.

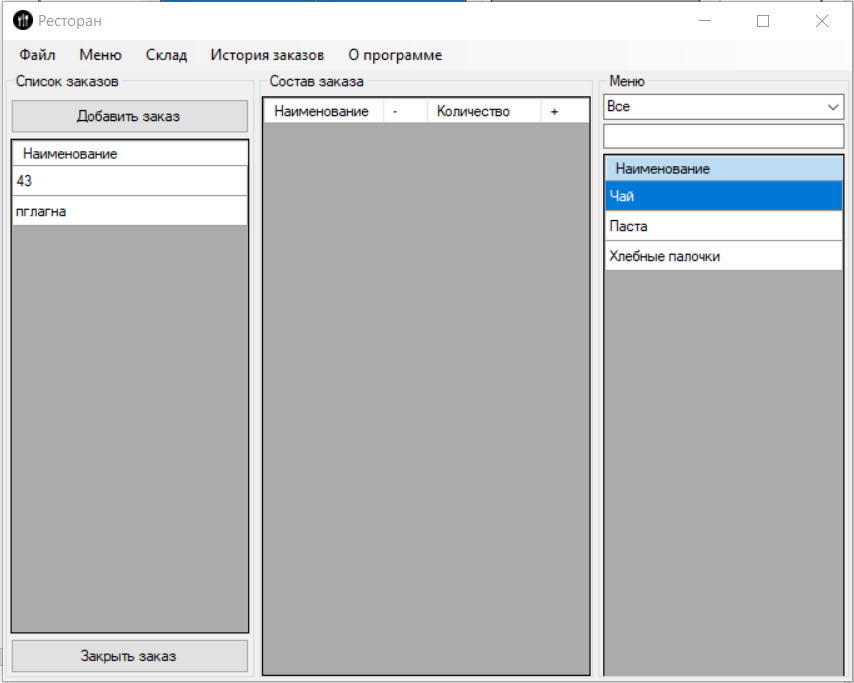


Рисунок 45 – Заказы

Для создания нового заказа нажмите кнопку Добавить заказ, после чего заказ создаться и Вам будет предложено изменить название заказа (рисунок 46). Для изменения названия в дальнейшем необходимо перевести поле в режим редактирования (двойной клик).

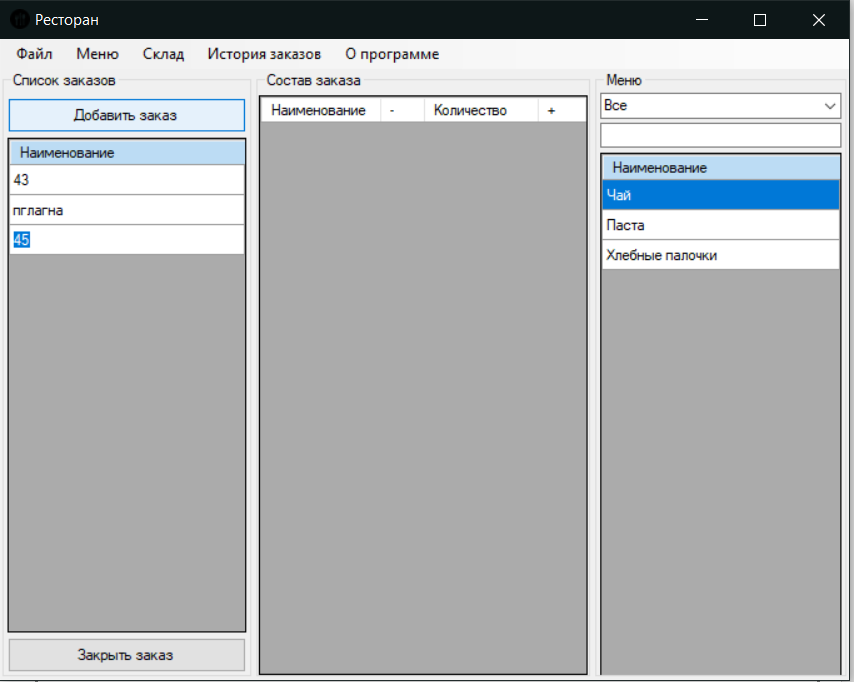


Рисунок 46 – Заказы с активным редактированием названия заказа

Для добавления содержимого в заказ необходимо, чтобы он был выбран в левой таблице (при выборе заказа автоматически загружается его актуальный состав), кликом по товару из правой таблицы он добавляется к выбранному заказу, и Вы можете сразу ввести необходимое количество товара т.к. количество товара автоматически переключается в режим редактирования (рисунок 47), можете перейти в режим редактирования так же с помощью двойного клика для уже существующих записей. Количество так же можно менять с помощью кнопок слева (-) и справа (+) от количества. При повторной попытке добавить тот же товар в заказ, количество товара так же увеличится на 1 и поле количества перейдёт в режим редактирования. В случае если количество товара станет равным или меньшим чем 0, то товар автоматически удалится из заказа.

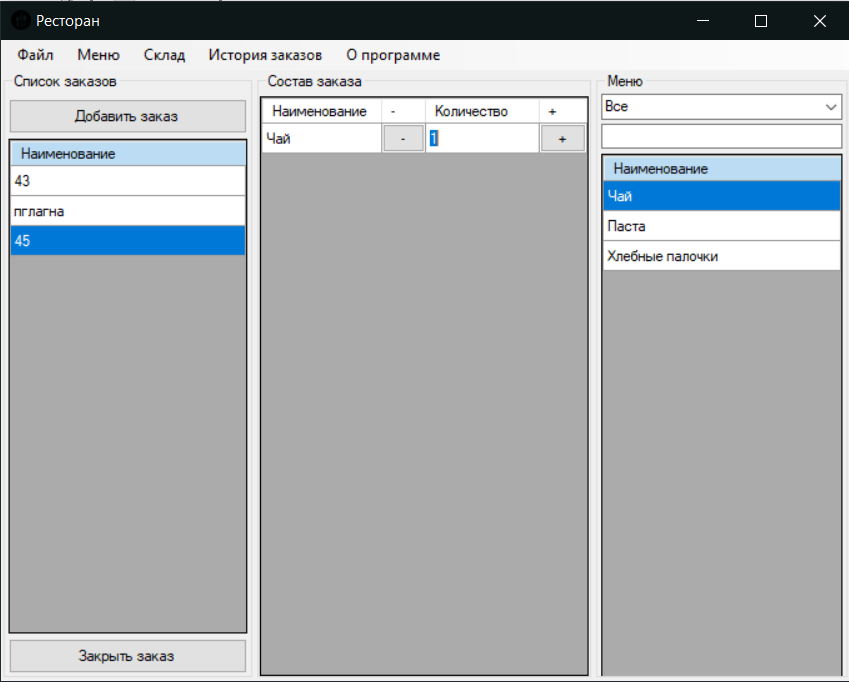


Рисунок 47 – Заказы с активным редактированием названия заказа

Для облегчения и ускорения поиска необходимого товара можно выбрать необходимую категорию товаров, вследствие будут отображаться только товары из данной категории (рисунок 48). Так же поле ниже позволяет вести поиск по товарам, при этом не учитывается регистр теста и достаточно ввести только начало названия (рисунок 49). Поиск и фильтрация могут работать одновременно (рисунок 50, 51)

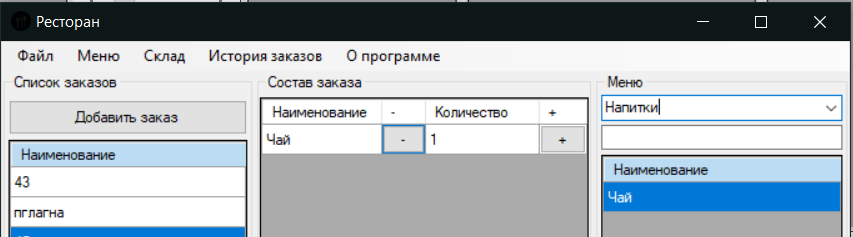


Рисунок 48 – Заказы с фильтрацией по категориям

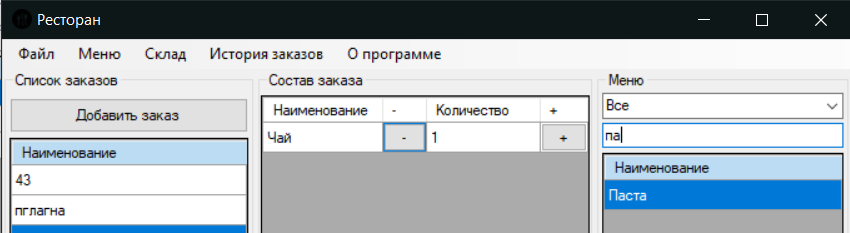


Рисунок 49 – Заказы с поиском

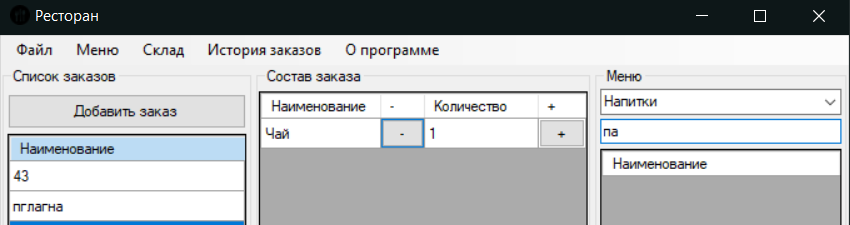


Рисунок 50 – Заказы с поиском и фильтрацией без результатов

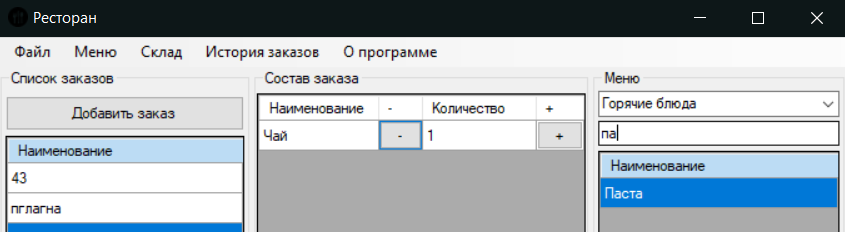


Рисунок 51 – Заказы с поиском и фильтрацией с результатом

Для закрытия или удаления заказа необходимо выбрать заказ в левой таблице (рисунок 45), нажать кнопку Закрыть заказ. В появившейся форме (рисунок 52) выводится сумма заказа. При необходимости можно ввести количество данных наличных и количество денег к сдаче рассчитается автоматически. При нажатии кнопки подтвердить заказ, он перемещается в историю заказов, и количество ресурсов необходимых для его производства вычитается со складских запасов. При нажатии кнопки удалить заказ, заказ будет удалён.

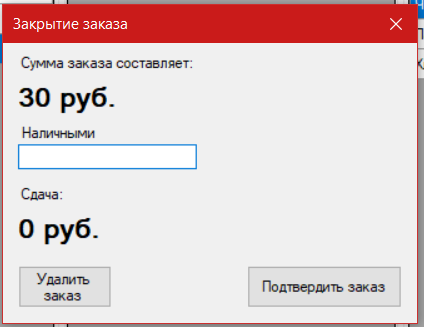


Рисунок 52 – Закрытие заказа

Для просмотра истории заказов через меню приложения необходимо выбрать пункт История заказов (рисунок 53).

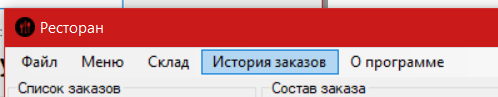


Рисунок 53 – Меню: История заказов

История заказов представлена на рисунке 54.

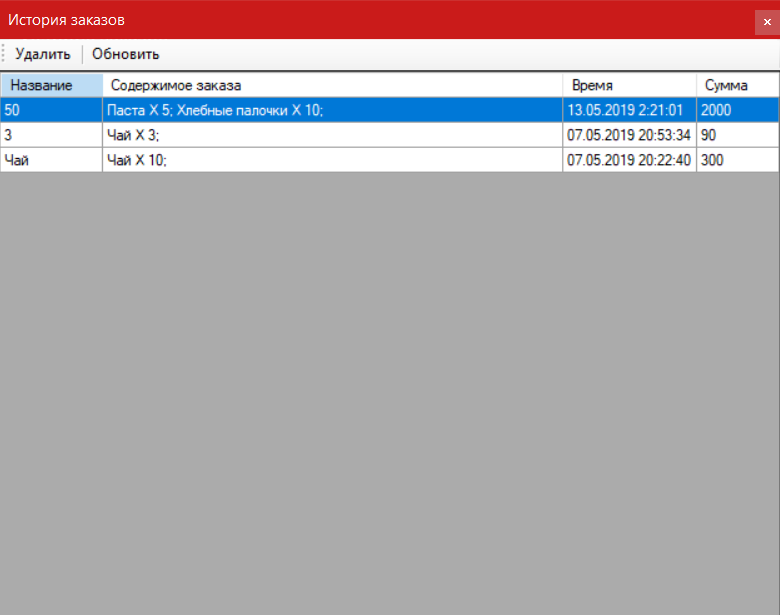


Рисунок 54 – История заказов

Для удаления заказа из истории необходимо выбрать один из них в списке (рисунок 54), нажать кнопку Удалить на панели инструментов (рисунок 55), подтвердить удаление кнопкой ОК (рисунок 56).

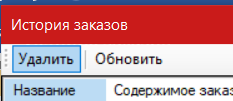


Рисунок 55 – История заказов: Инструменты: Удалить

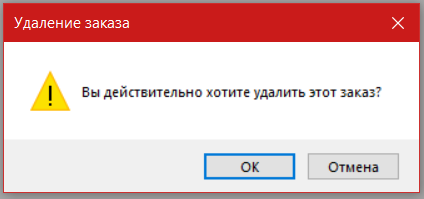


Рисунок 56 – Подтверждение удаления ресурса

Для принудительного обновления списка ресурсов необходимо нажать кнопку Обновить на панели инструментов (рисунок 57).

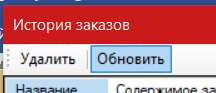


Рисунок 57 – История заказов: Инструменты: Обновить

Для просмотра информации об авторе ПО через меню приложения необходимо выбрать пункт О программе (рисунок 58).

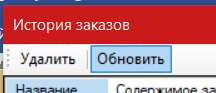


Рисунок 58 – Меню: О программе

Информация об авторе представлена на рисунке 54.

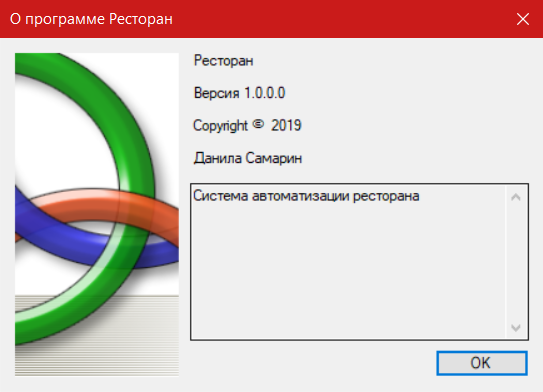


Рисунок 59 – О программе

Для выхода из приложения можно закрыть главную форму или через меню приложения необходимо выбрать пункт Выход (рисунок 60)

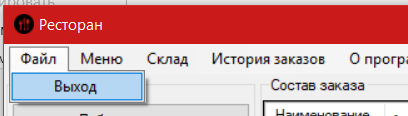


Рисунок 60 – Меню: Выход

# Заключение

В ходе курсового проекта было разработана автоматизированная информационная система для ресторана. Основными функциями которой является учет заказов и складских запасов с целью повышения скорости работы и снижения нагрузки на персонал.

В трех разделах пояснительной записки описано проектирование и разработка клиент-серверного приложения. Был проведен анализ предметной области, разработана концептуальная модель базы данных, а затем и логическая и физическая модели. В курсовом проекте представлена диаграмма базы данных, выполненная на стороне сервера MS SQL Server.

Приложение выполнено с помощью среды разработки Visual Studio и протестировано согласно приведенной программе проведения испытаний.

Цели, поставленные перед разработкой курсового проекта достигнуты.

# Список использованных источников

1. SQL. Язык структурированных запросов [Электронный ресурс] / А.В. Багдуева .— Улан-Удэ : Бурятский государственный университет, 2017 .— 122 с. — ISBN 978-5-9793-1087-9 .— Режим доступа: https://lib.rucont.ru/efd/640318
2. БАЗЫ ДАННЫХ [Электронный ресурс] / Г.И. Занданова, О.А. Лобсанова .— Улан-Удэ : Бурятский государственный университет, 2016 .— 97 с. — ISBN 978-5-9793-0914-9 .— Режим доступа: https://lib.rucont.ru/efd/558886
3. Проектирование баз данных [Электронный ресурс] / В.А. Чулюков, И.П. Половинкин, И.Ф. Астахова .— Воронеж : Издательский дом ВГУ, 2017 .— 74 с. — 74 с. — Режим доступа: https://lib.rucont.ru/efd/683727
4. Основы Microsoft .NET Framework и языка программирования C#: учебное пособие [Электронный ресурс] / И.В. Бачурин, М.В. Суханов .— Архангельск : Северный (Арктический) федеральный университет имени М.В. Ломоносова, 2014 .— 96 с. : ил. — ISBN 978-5-261-00934-4 .— Режим доступа: https://lib.rucont.ru/efd/279618
5. Язык C#: краткое описание и введение в технологии программирования [Электронный ресурс] : учеб. пособие / Урал. федер. ун-т, О.М. Котов .— Екатеринбург : Издательство Уральского университета, 2014 .— 210 с. : ил. — ISBN 978-5-7996-1094-4 .— Режим доступа: https://lib.rucont.ru/efd/293014
6. Разработка приложений баз данных [Электронный ресурс] / М.М. Безрядин, Ю.С. Левицкая, В.Г. Рудалев .— Воронеж : Издательский дом Воронежского государственного университета, 2014 .— 48 с. — 48 с. — Режим доступа: https://lib.rucont.ru/efd/297994
7. Базы данных в высокопроизводительных информационных системах [Электронный ресурс] : учебное пособие. Направление подготовки 09.04.02 – Информационные системы и технологии. Квалификация выпускника – магистр / Е. И. Николаев .— Ставрополь : изд-во СКФУ, 2016 .— 163 с. — Режим доступа: https://lib.rucont.ru/efd/622818
8. Пальмов, С.В. Методы и средства моделирования программного обеспечения [Электронный ресурс] : конспект лекций / С.В. Пальмов .— Самара : Изд-во ПГУТИ, 2016 .— 105 с. — Режим доступа: https://lib.rucont.ru/efd/565120
9. Проектирование объектно-ориентированных программ средствами StarUML (диаграммы прецедентов, классов) [Электронный ресурс] : метод. указания / Оренбургский гос. ун- т, О. Н. Яркова .— Оренбург : ОГУ, 2014 .— 28 с. — Режим доступа: https://lib.rucont.ru/efd/293613
10. Базы данных [Электронный ресурс] : учеб. пособие / С. А. Щелоков .— Оренбург : ОГУ, 2014 .— 298 с. — Режим доступа: https://lib.rucont.ru/efd/278638
11. Информационные системы [Электронный ресурс] : учебник / М.Л. Соболева, А.С. Алфимова, С.А. Жданов .— М. : Издательство Прометей, 2015 .— 302 с. : ил. — ISBN 978-5-9906-2644-7 .— Режим доступа: https://lib.rucont.ru/efd/315777
12. Архитектура систем управления базами данных [Электронный ресурс] : учеб. пособие / А.И. Мирошников .— Липецк : Изд-во Липецкого государственного технического университета, 2018 .— 94 с. : ил. — ISBN 978-5-88247-879-6 .— Режим доступа: https://lib.rucont.ru/efd/682415
13. Современные компьютерные технологии [Электронный ресурс] : учеб. пособие / Р.Г. Хисматов, Р.Г. Сафин, Д.В. Тунцев, Н.Ф. Тимербаев, Казан. нац. исслед. технол. ун-т .— Казань : КНИТУ, 2014 .— 83 с. — ISBN 978-5-7882-1559-4 .— Режим доступа: https://lib.rucont.ru/efd/302846
14. Базы данных. Курс лекций [Электронный ресурс] : [учеб. пособие] / Р.Р. Латыпова .— М. : Проспект, 2016 .— 96 с. — ISBN 978-5-392-19240-3 .— Режим доступа: https://lib.rucont.ru/efd/632783
15. Основы проектирования компонентов автоматизированных систем [Электронный ресурс] : учеб. пособие / Оренбургский гос. ун- т, Т.В. Волкова .— Оренбург : ОГУ, 2016 .— 226 с. — ISBN 978-5-7410-1560-5 .— Режим доступа: https://lib.rucont.ru/efd/618341

# Приложение А

Фрагмент кода

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.Data.SqlClient;

namespace Restaurant

{

public partial class EditingProducts : Form

{

private SqlConnection sqlConnection;

private int id;

//класс для выпадающих списков

class ComboboxValue

{

public int Id { get; private set; }

public string Name { get; private set; }

public ComboboxValue(int id, string name)

{

Id = id;

Name = name;

}

public override string ToString()

{

return Name;

}

}

public EditingProducts(SqlConnection connection, int idTransf = -1)

{

InitializeComponent();

sqlConnection = connection;

id = idTransf;

if (id == -1)

{

this.Text = "Добавление товара";

}

else

{

this.Text = "Редактирование товара";

}

}

private async void EditingProducts\_Load(object sender, EventArgs e)

{

//подгрузка данных в форму в случае переданного ID

if (id > -1)

{

SqlCommand loadProducts = new SqlCommand("SELECT name, price, available FROM [menu] WHERE id\_menu=@id", sqlConnection);

loadProducts.Parameters.AddWithValue("id", id);

try

{

SqlDataReader sqlReader = await loadProducts.ExecuteReaderAsync();

await sqlReader.ReadAsync();

nameTextBox.Text = Convert.ToString(sqlReader["name"]);

priceTextBox.Text = Convert.ToString(sqlReader["price"]);

availableCheckBox.Checked = Convert.ToBoolean(sqlReader["available"]);

if (sqlReader != null && !sqlReader.IsClosed)

sqlReader.Close();

//загрузка состава

SqlCommand loadRecipesGreed = new SqlCommand("SELECT stocks.id\_stocks, stocks.name, recipes.quantity FROM stocks, recipes WHERE recipes.id\_stocks = stocks.id\_stocks AND recipes.id\_menu = @id", sqlConnection);

loadRecipesGreed.Parameters.AddWithValue("id", id);

sqlReader = await loadRecipesGreed.ExecuteReaderAsync();

while (await sqlReader.ReadAsync())

{

int rowNumber = recipesDataGridView.Rows.Add();

recipesDataGridView.Rows[rowNumber].Cells[0].Value = Convert.ToInt32(sqlReader["id\_stocks"]);

recipesDataGridView.Rows[rowNumber].Cells[1].Value = Convert.ToString(sqlReader["name"]);

recipesDataGridView.Rows[rowNumber].Cells[2].Value = Convert.ToDouble(sqlReader["quantity"]);

}

if (sqlReader != null && !sqlReader.IsClosed)

sqlReader.Close();

//загрузка категорий

SqlCommand loadCategoryGreed = new SqlCommand("SELECT categories.name, categories.id\_category FROM [categories], [category\_list] WHERE category\_list.id\_category = categories.id\_category AND category\_list.id\_menu = @id\_menu", sqlConnection);

loadCategoryGreed.Parameters.AddWithValue("id\_menu", id);

sqlReader = await loadCategoryGreed.ExecuteReaderAsync();

while (await sqlReader.ReadAsync())

{

int rowNumber = categoryDataGridView.Rows.Add();

categoryDataGridView.Rows[rowNumber].Cells[1].Value = Convert.ToString(sqlReader[0]);

categoryDataGridView.Rows[rowNumber].Cells[0].Value = Convert.ToString(sqlReader[1]);

}

if (sqlReader != null && !sqlReader.IsClosed)

sqlReader.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

//загрузка содержимого выпадающего списка ресурсов

SqlCommand loadRecipesList = new SqlCommand("SELECT id\_stocks, name FROM [stocks]", sqlConnection);

try

{

SqlDataReader sqlReader = await loadRecipesList.ExecuteReaderAsync();

while (await sqlReader.ReadAsync())

{

recipesComboBox.Items.Add(new ComboboxValue(Convert.ToInt32(sqlReader["id\_stocks"]), Convert.ToString(sqlReader["name"])));

}

if (sqlReader != null && !sqlReader.IsClosed)

sqlReader.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

//загрузка содержимого выпадающего списка категорий

SqlCommand loadCategoryList = new SqlCommand("SELECT id\_category, name FROM [categories]", sqlConnection);

try

{

SqlDataReader sqlReader = await loadCategoryList.ExecuteReaderAsync();

while (await sqlReader.ReadAsync())

{

categoryComboBox.Items.Add(new ComboboxValue(Convert.ToInt32(sqlReader["id\_category"]), Convert.ToString(sqlReader["name"])));

}

if (sqlReader != null && !sqlReader.IsClosed)

sqlReader.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

string error = "";

float count;

ComboboxValue combo = (ComboboxValue)recipesComboBox.SelectedItem;

if (combo == null)

error = "Не выбран компонент\n";

if (!float.TryParse(recipesCountTextBox.Text, out count))

error += "Введиите количество компонента\n";

if (error == "")

{

int rowNumber = recipesDataGridView.Rows.Add();

recipesDataGridView.Rows[rowNumber].Cells[0].Value = Convert.ToInt32(combo.Id);

recipesDataGridView.Rows[rowNumber].Cells[1].Value = Convert.ToString(combo.Name);

recipesDataGridView.Rows[rowNumber].Cells[2].Value = Convert.ToDouble(count);

}

else {

MessageBox.Show(error, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

if (recipesDataGridView.SelectedRows.Count == 1)

{

recipesDataGridView.Rows.Remove(recipesDataGridView.SelectedRows[0]);

}

else {

MessageBox.Show("Выберите строку", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

Close();

}

private async void OkButton\_Click(object sender, EventArgs e)

{

string error = "";

if (nameTextBox.Text == "")

error += "Имя не может быть пустым\n";

if (priceTextBox.Text == "")

error += "Стоимоть не может быть пустой";

if (error != "") {

MessageBox.Show(error, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (id == -1)

{

string query = "INSERT INTO [menu] (name, price, available, ordinem) VALUES (@name, @price, @available, IDENT\_CURRENT('menu'))";

if (recipesDataGridView.RowCount > 0)

query = query + "; INSERT INTO [recipes] (id\_menu, id\_stocks, quantity) VALUES";

for (int i = 0; i < recipesDataGridView.RowCount; i++)

{

query = query + " (IDENT\_CURRENT('menu'), @id\_menu" + i + ", @quantity" + i + ")";

if (i < recipesDataGridView.RowCount - 1)

query = query + ",";

}

if (categoryDataGridView.RowCount > 0)

query = query + "; INSERT INTO [category\_list] (id\_menu, id\_category) VALUES";

for (int i = 0; i < categoryDataGridView.RowCount; i++)

{

query = query + " (IDENT\_CURRENT('menu'), @id\_menuC" + i + ")";

if (i < categoryDataGridView.RowCount - 1)

query = query + ",";

}

SqlCommand addProduct = new SqlCommand(query, sqlConnection);

addProduct.Parameters.AddWithValue("name", nameTextBox.Text);

addProduct.Parameters.AddWithValue("price", priceTextBox.Text);

addProduct.Parameters.AddWithValue("available", availableCheckBox.Checked);

for (int i = 0; i < recipesDataGridView.RowCount; i++)

{

addProduct.Parameters.AddWithValue("id\_menu" + i, recipesDataGridView.Rows[i].Cells[0].Value);

addProduct.Parameters.AddWithValue("quantity" + i, recipesDataGridView.Rows[i].Cells[2].Value);

}

for (int i = 0; i < categoryDataGridView.RowCount; i++)

{

addProduct.Parameters.AddWithValue("id\_menuC" + i, categoryDataGridView.Rows[i].Cells[0].Value);

}

try

{

await addProduct.ExecuteNonQueryAsync();

Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else {

string query = "UPDATE [menu] SET name=@name, price=@price, available=@available WHERE id\_menu=@id\_menu; DELETE FROM [recipes] WHERE id\_menu=@id\_menu; DELETE FROM [category\_list] WHERE id\_menu=@id\_menu";

if (recipesDataGridView.RowCount > 0)

query = query + "; INSERT INTO [recipes] (id\_menu, id\_stocks, quantity) VALUES";

for (int i = 0; i < recipesDataGridView.RowCount; i++)

{

query = query + " (@id\_menu, @id\_menu" + i + ", @quantity" + i + ")";

if (i < recipesDataGridView.RowCount - 1)

query = query + ",";

}

if (categoryDataGridView.RowCount > 0)

query = query + "; INSERT INTO [category\_list] (id\_menu, id\_category) VALUES";

for (int i = 0; i < categoryDataGridView.RowCount; i++)

{

query = query + " (@id\_menu, @id\_menuC" + i + ")";

if (i < categoryDataGridView.RowCount - 1)

query = query + ",";

}

SqlCommand updateProduct = new SqlCommand(query, sqlConnection);

updateProduct.Parameters.AddWithValue("name", nameTextBox.Text);

updateProduct.Parameters.AddWithValue("price", Convert.ToDouble(priceTextBox.Text));

updateProduct.Parameters.AddWithValue("available", availableCheckBox.Checked);

updateProduct.Parameters.AddWithValue("id\_menu", id);

for (int i = 0; i < recipesDataGridView.RowCount; i++)

{

updateProduct.Parameters.AddWithValue("id\_menu" + i, recipesDataGridView.Rows[i].Cells[0].Value);

updateProduct.Parameters.AddWithValue("quantity" + i, recipesDataGridView.Rows[i].Cells[2].Value);

}

for (int i = 0; i < categoryDataGridView.RowCount; i++)

{

updateProduct.Parameters.AddWithValue("id\_menuC" + i, categoryDataGridView.Rows[i].Cells[0].Value);

}

try

{

await updateProduct.ExecuteNonQueryAsync();

Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

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

{

string error = "";

ComboboxValue combo = (ComboboxValue)categoryComboBox.SelectedItem;

if (combo == null)

error = "Не выбрана категория";

if (error == "")

{

int rowNumber = categoryDataGridView.Rows.Add();

categoryDataGridView.Rows[rowNumber].Cells[0].Value = Convert.ToInt32(combo.Id);

categoryDataGridView.Rows[rowNumber].Cells[1].Value = Convert.ToString(combo.Name);

}

else

{

MessageBox.Show(error, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

if (categoryDataGridView.SelectedRows.Count == 1)

{

categoryDataGridView.Rows.Remove(categoryDataGridView.SelectedRows[0]);

}

else

{

MessageBox.Show("Выберите строку", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}