АННОТАЦИЯ

Отчет о курсовой работе: 82с., 38рис., 13табл., 1 приложение, 8 источников.

Объект исследования – процесс предоставление информации.

Цель работы – разработка базы данных и приложения поиска, просмотра и редактирования кулинарных рецептов.

Метод исследования – изучение принципов разработки и проектирования приложений средствами .NET Framework, принципов работы с базой данных MS SQL.

В работе были использованы технологии: C#, .NET, VisualStudio 2015, WindowsForm, MSSQL, Visual Studio Install Project.

В результате решения задачи было разработано приложение, для получения справки блюдах, а также база данных для приложения.

Дальнейшее развитие программы связано с расширением ее возможностей, увеличением числа опций.

БАЗА ДАННЫХ, ПРИЛОЖЕНИЕ, WINDOWS FORM, VISUAL STUDIO, .NET, FRAMEWORK, СУБД, ADO.NET, SQL, MS SQL, MS SQL VISUAL STUDIO TOOLS

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Мы живём в век информационных технологий, в век в котором техника может выполнять задачи, которые обременяют нас в повседневности. В настоящий момент существует масса мультимедийных систем, которые контролируют и упрощают наш повседневный быт. И при этом каждый день, все мы сталкиваемся с такими тривиальными задачами, как приготовление еды. Нужно знать, ЧТО ты хочешь приготовить, КАК и какие ПРОДУКТЫ необходимы. И при решении этой задачи мы сталкиваемся с проблемой упорядочения этих трех параметров: «ЧТО, КАК, Продукты?». А также актуальной задачей является экономия времени и средств.

Поэтому целесообразным инструментом решения поставленных задач, на мой взгляд, является приложение, подключенное к базе данных, представляющее собой «Книгу рецептов». Функциями этого приложения является: поиск рецептов блюд по широкому выбору критериев, просмотр детальной информации о приготовлении, составе продукта и так далее.

Целью курсовой работы является разработка приложения «Кулинарный справочник». Исходя из указанной цели, можно выделить частные задачи, поставленные в курсовой работе:

1. Проанализировать основные, наиболее интересуемые, критерии, которые интересуют человека, который ищет рецепты.
2. Изучить технологии: .NET, Windows Forms, C#, Visual Studio 2015, MS SQL.
3. Разработать структуру программы и базы данных, разработать функционал программы, соответствующий техническому заданию.
4. Разработать интерфейс и базу данных для приложения.

1 Анализ предметной области

1.1 Состояние вопроса

Для примера рассмотрим один из первых найденных в интернете кулинарных сайтов с рецептами, а также рассмотрим один книжный кулинарный справочник. В ходе рассмотрения будут анализированы недостатки и преимущества каждого из источников, на основании чего можно создавать технические требования приложения.

Для начала обратим внимание на книжный кулинарный справочник «Миллион рецептов праздничных блюд» [1], после чего исследуем сайт «Приятного аппетита» (http://priyatnogo-appetita.com)[2].

Таблица 1 – Анализ книжного кулинарного справочника «Миллион рецептов блюд»

Достоинства	Недостатки
1. «Всегда под рукой». Для использования этого ресурса нет необходимости в интернете либо каком-нибудь девайсе. 2. По мимо разбиения по категориям (салаты, закуски и т.д.) в книге присутствуют блюда к определенным праздникам, так же правила этикета, разнообразные способы украшения и подачи блюд.	1.Не смотря на название книги, рецептов в книге отнюдь не миллион, а всего около семисот. 2. Рецепты в книге нельзя изменить, не испортив саму книгу. Так же плохи дела и с добавлением новых рецептов. 3. При том, что в книге рецептов не так уж и много, так и рецепты описаны очень минималистично (по 2 рецепта на 1 сторону страницы) и без всяких иллюстраций. 4. Поиск самих рецептов ограничен лишь только по названиям в содержании.

Таблица 2 – Анализ Интернет-ресурса «Приятного аппетита»

Достоинства	Недостатки
1. Наличие иллюстраций и подробного описания. 2. Возможность оставлять комментарии к блюдам и выставлять оценки. 3. Возможность добавить рецепт (отправить заявку, которую должны будут подтвердить администраторы).	1. Поиск возможен только по слову в названии. 2. Ресурс теряет свою работоспособность в случае отсутствия интернета. .

В ходе анализа были найдены достоинства и недостатки типичных кулинарных справочников. На основе данного анализа можно выделить основной функционал и требования, которые будет необходимо выполнить. Основным критерием является информативность рецептов, далее следует обратить внимание на удобный, быстрый поиск блюд. Для этого следует ввести необходимый функционал:

1. Поиск блюд, по категориям, названию, описанию и ингредиентам.
2. Возможность добавлять в рецепт фотографию.
3. Возможность в любой момент добавить новый рецепт или изменить существующий.

1.2 Актуальность и цель работы

Вопрос о поиске кулинарных блюд будет всегда актуален, так как связан с одной из основных и бытовых проблем, с которых человек связывается в повседневной жизни. Кулинария не перестает развиваться, так что и информационным ресурсам в этой сфере не стоит стоять на месте. При

изучении состояния вопроса было уяснено, что текущие справочники и ресурсы, связанные с этим, имеют свои недостатки, поэтому целью моей курсовой будет создание приложения, которое позволит:

1. Искать блюдо для приготовления быстро, так как в приложении будет предусмотрена функция быстрого поиска, а также пользователю не обязательно будет регистрироваться или заходить на свой аккаунт.
2. Создавать закладки понравившихся и заинтересовавших блюд.
3. Искать блюда для приготовления подробно, с возможностью указывать необходимые бля блюда ингредиенты, категории и слова.
4. Делиться со всем миром своими кулинарными рецептами.

Целью курсовой работы является разработка приложения «Кулинарный справочник». Исходя из указанной цели, можно выделить частные задачи, поставленные в курсовой работе:

1. Выделить основные, наиболее интересуемые, моменты, которые интересуют человека, который ищет рецепты.
2. Изучитьтехнологии: .NET,Windows Forms, C#, Visual Studio 2015, MS SQL.
3. Разработать структуру программы и базы данных, разработать функционал программы, соответствующий техническому заданию.
4. Разработать интерфейс и базу данных для приложения.

2 Техническое задание

2.1 Общие требования к продукту

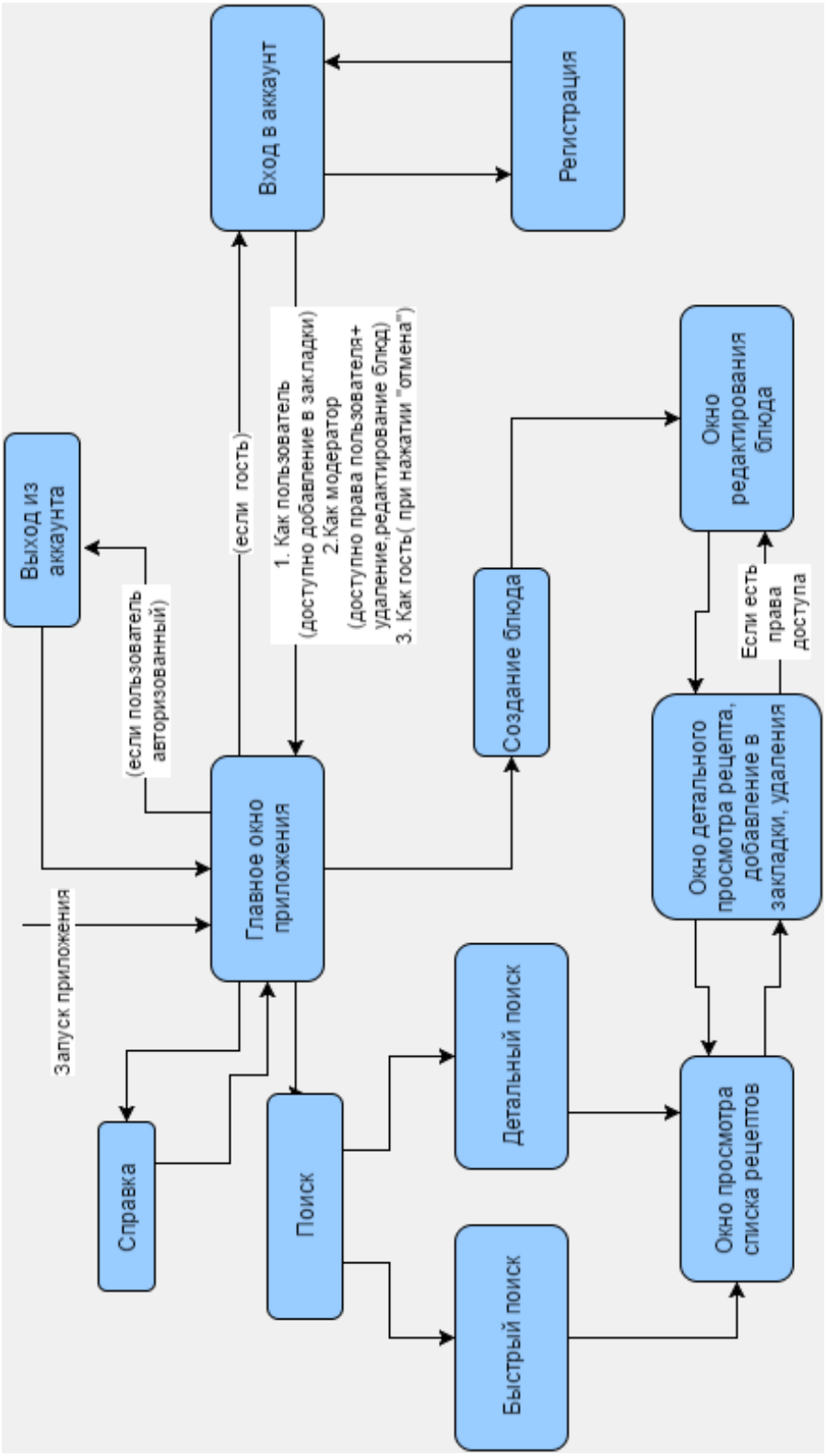

Рисунок 1- Функциональная схема программы

При запуске, в приложении, пользователь авторизирован, как гость. При этом он должен иметь возможность искать рецепты на таких же правах, как и зарегистрированный пользователь. Поиск должен быть двух типов: быстрый и детальный. Каждый вид поиска нужен в той или иной ситуации. Так же обязательно должна быть возможность просматривать подробную информации о способе приготовления блюда. При желании, пользователь должен иметь возможность зарегистрироваться, или войти в систему с помощью существующего аккаунта. Зарегистрированному пользователю добавляется возможность добавлять/удалять закладки всех блюд, а если пользователь отмечен, как модератор, то должна добавляться возможность редактирования, создания и удаления рецептов.

По мимо основного функционала, для всех категорий пользователей должна присутствовать справка, а также возможность смены аккаунта и настройки приложения.

2.2 Позиционирование продукта

2.2.1 Требование к пользовательским интерфейсам

Интерфейс программы должен быть не особо сложным, приятным на вид и учитывать существующие стандарты дизайна. Для облегчения взаимодействия программы с пользователем следует визуально разбить интерфейс на части, каждая из которых будет иметь свой функционал и предназначение.

Требования к интерфейсу приложения, представляемым пользователям:

Основной интерфейс представлен на рисунке 2

1. Главное окно
2. Блок, в котором будет отображаться список рецептов, подробный

рецепт, редактирование рецепта, справочная информация

3. Управляющая панель
4. Отображение состояния авторизации (имя пользователя, кнопка вода/выхода)
5. Панель с основными командами приложения (выход, настройки, вызов справки в блоке 2, создание рецепта, открытие закладок в блоке 2)
6. Кнопка запуска выбранного поиска
7. Панель настройки выбранного поиска
8. Блок поиска
9. Переключение на быстрый поиск
10. Переключение на детальный поиск

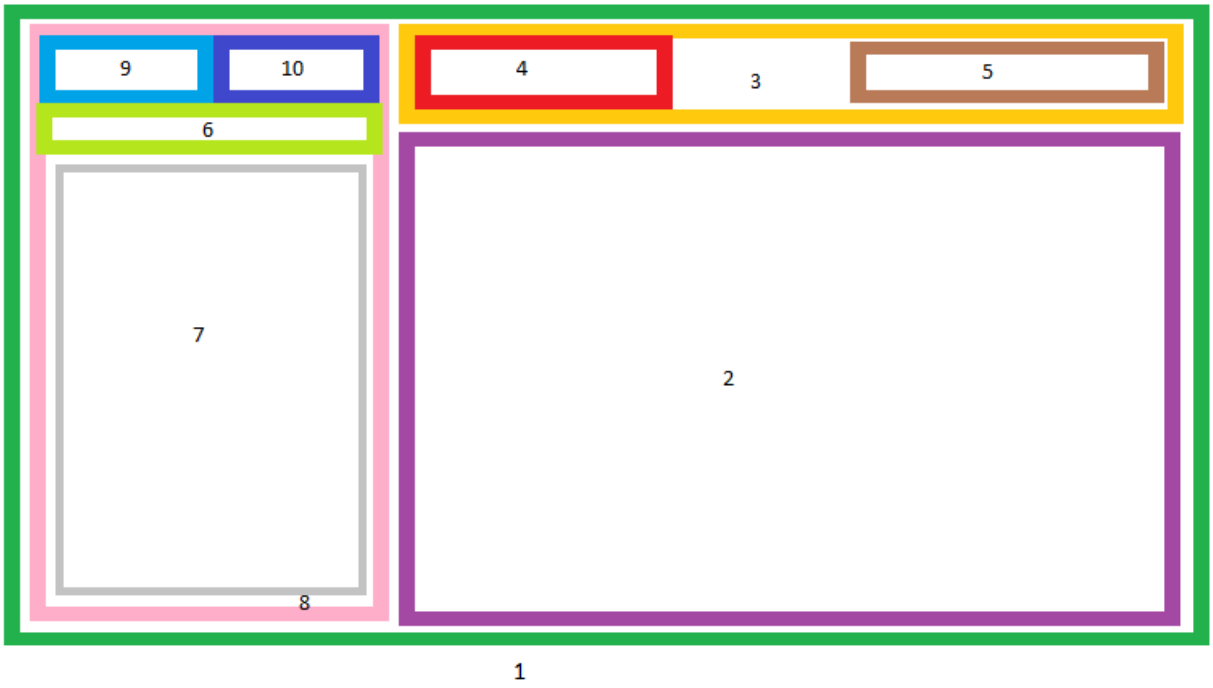

Рисунок 2

Данное окно является родительским, для всех окон. При запуске в главном блоке (блок 2) отображается справка по приложению, в блоке управления (блок 3) отображается имя «Гость» и все доступные ему операции по работе с приложением. В блоке поиска по умолчанию выбран «Быстрый» поиск. В

блоке быстрого поиска содержится столько дочерних панелей, сколько всего будет придумано вариантов разбиения на категории. Блок-схема блока поиска и его панелей описана на рисунке 3

1. Панель быстрого поиска
2. Панель категории
3. Название категории
4. Список элементов категории

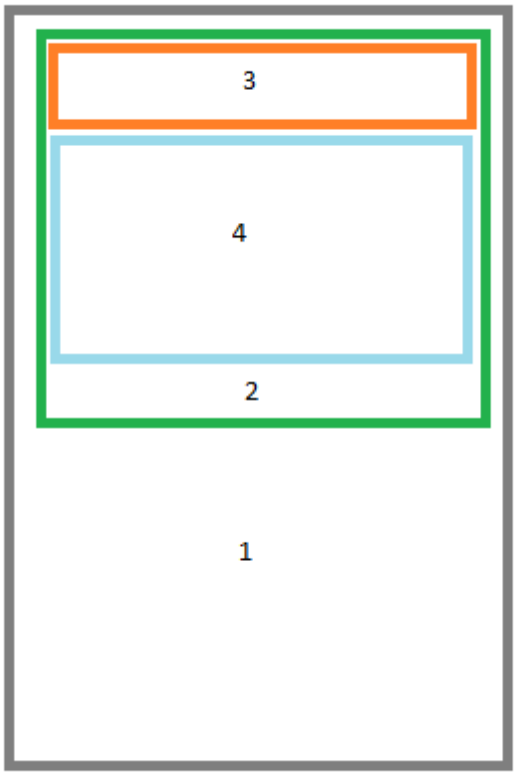

Рисунок 3

При переключении на детальный поиск в блоке поиска блок быстрого поиска заменяется. Блок-схема панели дательного поиска выгляди следующим образом (рисунок 4):

1. Панель детального поиска
2. Список ингредиентов для поиска
3. Список элементов категорий для поиска
4. Панель ингредиента в поиск

5. Панель добавления элементов категорий
6. Строка для ввода поискового слова

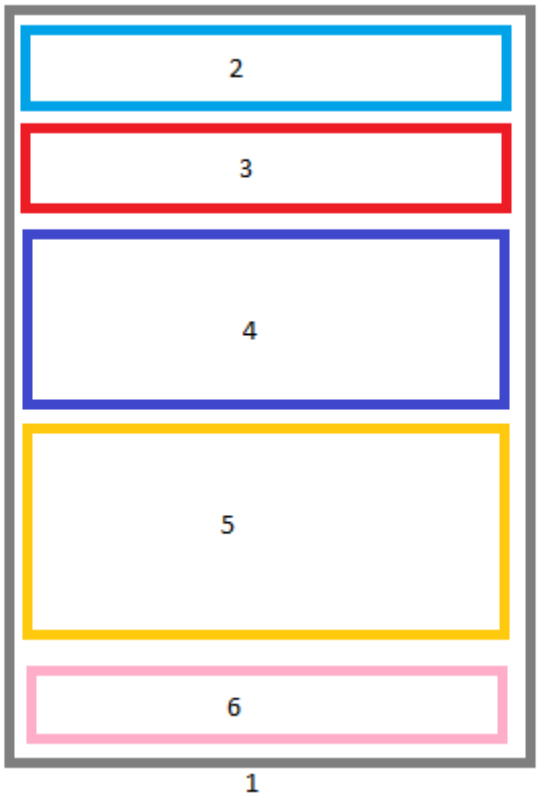

Рисунок 4

После выбора опций поиска и непосредственно запуска поиска в главном блоке открывается пустая панель просмотра списка рецептов и заполняется соответствующими рецептами. Каждый рецепт в этой панели можно изобразить следующей блок-схемой (рисунок 5):

1. Панель минимального отображения рецепта
2. Название блюда
3. Ингредиенты
4. Часть описания приготовления

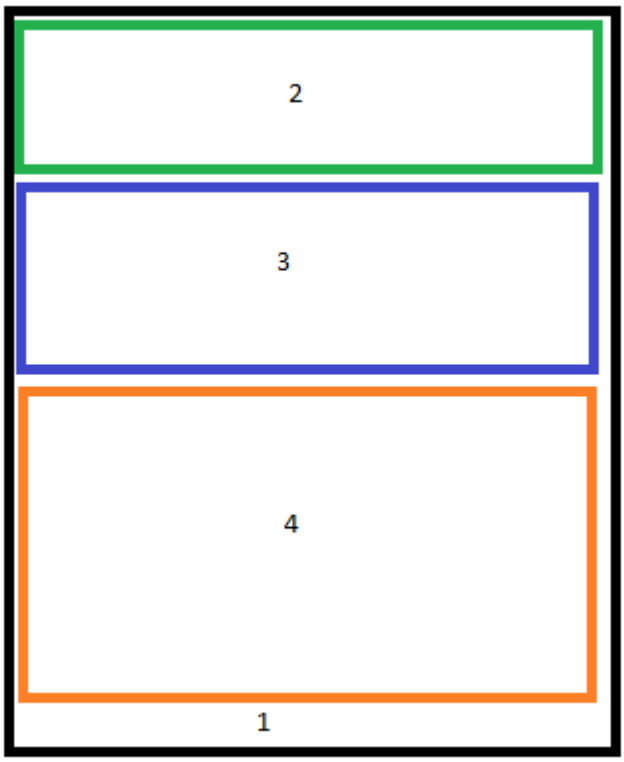

Рисунок 5

При желании, пользователь может нажать на любой из рецептов для детального просмотра.

Блок-схемы детального просмотра блюда и редактирования блюда очень схожи. Отличаются они только дополнительными элементами в окне редактирования (4,6).

1. Панель, содержащая все элементы
2. Название блюда
3. Функциональная панель (содержит команды «сохранить», «назад», «редактировать» и т.д.)
4. Панель добавления подкатегории блюду
5. Список всех подкатегорий блюда
6. Добавление ингредиентов блюда
7. Список ингредиентов блюда
8. Редактирование или просмотр описания

9. Просмотр или изменение изображения результата

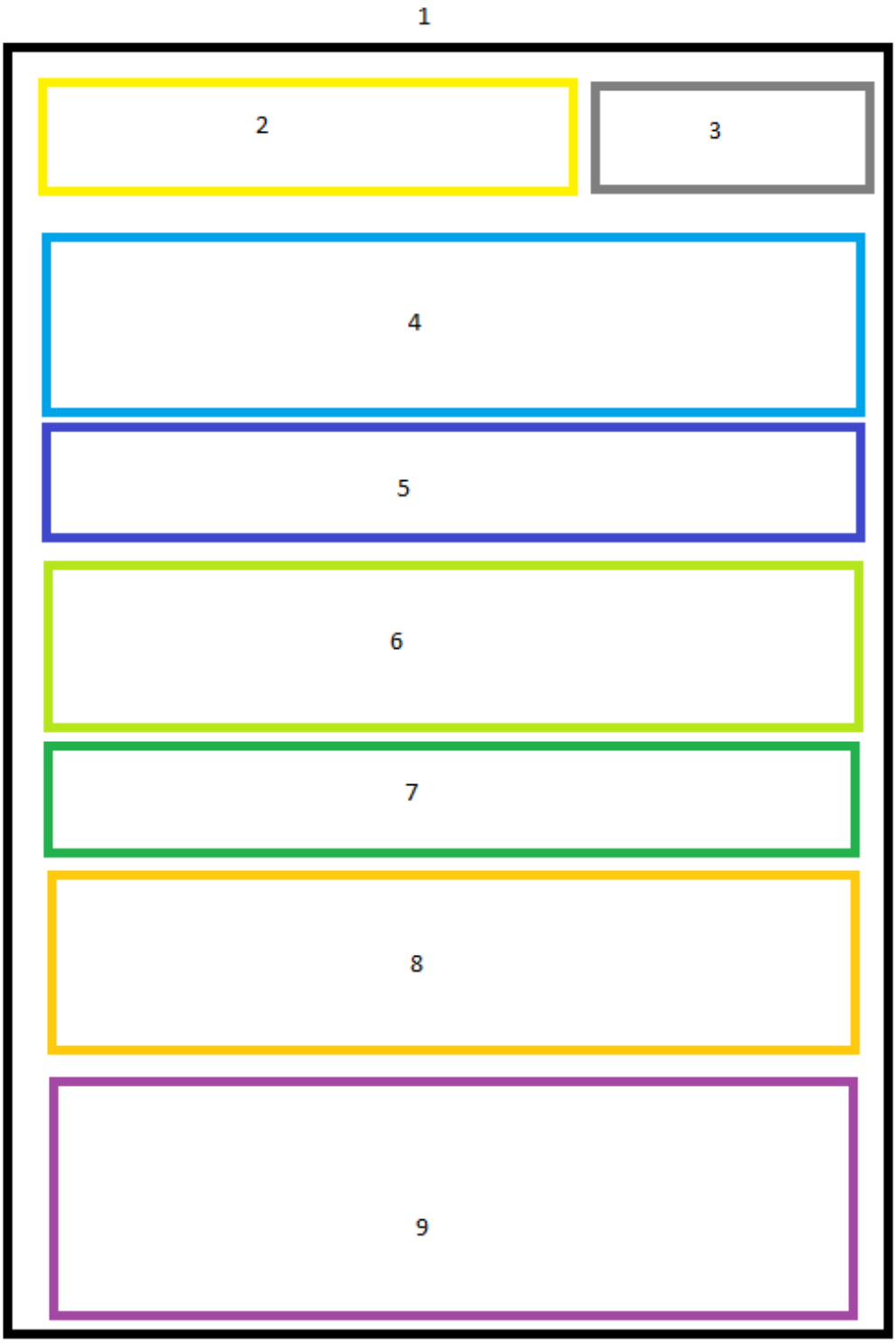

Рисунок 6

2.2.2 Требования к программным интерфейсам

Код программы должен быть понятным, содержать комментарии, не содержать повторяющегося кода. Отделить работу с базой данных от логики приложения в отдельный класс.

2.2.3 Требования к базе данных

База данных должна содержать данные о пользователях для успешного входа, полную информацию о блюдах, категориях, ингредиентах и закладках пользователя. Также потребуется обеспечить связи между таблицами. База данных должна хранить большие объемы данных и обеспечить удовлетворительную скорость работы. Для создания базы данных в данном курсовом проекте была выбрана СУБД MSSQL.

2.2.4 Требования к пользователям продукта

– Подключение к сети Интернет
– Базовый навык работы с компьютером

2.3 Функции продукта

Основной функционал продукта:

1. Возможность быстрого поиска
2. Возможность гибкого поиска
3. Просмотр детальной и минимальной информации о рецепте
4. Настройка приложения
5. Получение справки
6. Добавление и редактирование блюд
7. Создание аккаунта и вход на существующие
8. Управление закладками (просмотр, удаление и добавление)

2.4 Сценарии использования продукта

Пользователь запускает приложение. Изначально пользователь авторизирован как гость. Открывается окно со справкой.

Основные сценарии

Сценарий 1: Пользователь нажимает на кнопку входа и появляется модальное окно входа в аккаунт.

Сценарий 2: Пользователь нажимает на кнопку быстрого поиска, в блоке поиска открывается окно с разными видами категорий для выбора.

Сценарий 3: Пользователь нажимает на кнопку детального поиска, в блоке поиска открывается окно детального поиска.

Сценарий 4: Пользователь нажимает на кнопку закладки (только для авторизированных пользователей), в главном блоке открывается окно для просмотра списка блюд, заполненное блюдами с закладок.

Сценарий 5: Пользователь нажимает на кнопку добавить блюдо (только для авторизированных пользователей, отмеченных как «доверенные»), в главном блоке открывается окно создания рецепта блюда.

Сценарий 6: Пользователь нажимает на кнопку справки, в главном блоке открывается окно с информацией об приложении и его использовании.

Сценарий 7: Пользователь нажимает на кнопку настроек и открывается модальный диалог настроек.

Сценарий 8: Пользователь нажимает на кнопку закрытия окна и осуществляется выход из приложения.

Сценарий 9: Пользователь нажимает на кнопку выхода с аккаунта (кнопка видима или включена только когда пользователь авторизирован) и в приложении становиться гостем.

Окно входа

Сценарий 1: пользователь вводит свои данные для входа.

Сценарий 1.1: пользователь успешно авторизуется и возвращается на главное окно, где в панели управления аккаунтом имя изменилось на имя пользователя, и все панели обновились согласно с правами доступа пользователя.

Сценарий 1.2: пользователь вводит данные неверно и нажимает кнопку входа. Появляется всплывающее окно с предупреждением о неверно набранном логине или пароле.

Сценарий 1.3: пользователь вводит данные верно, но база данных не отвечает, о чем пользователя уведомляет всплывающее через некоторое время окно.

Сценарий 2: пользователь нажимает кнопку регистрации и открывается окно регистрации.

Сценарий 3: кнопка отмены возвращает пользователя в главное окно, не изменивши при этом ничего.

Окно регистрации

Сценарий 1: пользователь вводит данные для регистрации.

Сценарий 1.1: данные не корректны - всплывает окно с предупреждением.

Сценарий 1.2: данные корректны, но база не отвечает - всплывает окно с предупреждением.

Сценарий 1.3: успешная регистрация, пользователь возвращается к окну входа.

Сценарий 2: отмена действия и возврат к окну входа.

Окно редактирования или создания блюда

Сценарий 1: пользователь вводит корректные данные и жмет кнопку сохранить, после чего перенаправляется в окно просмотра созданного блюда.

Сценарий 2: пользователь вводит некорректные данные.

Сценарий 2.1: неподходящее (пустое или существующее) имя. Пользователя предупреждают об этом.

Сценарий 2.2: несуществующий ингредиент. Если пользователь обладает правами, то пользователь может выбрать «занести ингредиент базу».

Окно отображения списка рецептов

Сценарий 1: пользователь нажимает на название блюда и переходит на странице детального просмотра.

Окно просмотра блюда

Сценарий 1: пользователь нажимает кнопку «Назад» и возвращается к просмотру списка рецептов блюд.

Сценарий 2: данное блюдо у пользователя в закладках. Пользователь нажимает на кнопку «Удалить из закладок», после чего закладка на блюдо удаляется и появляется кнопка «Добавить в закладки».

Сценарий 3: данное блюдо у пользователя не в закладках. Пользователь нажимает на кнопку «Добавить в закладки», после чего закладка на блюдо добавляется и появляется кнопка «Удалить из закладок».

Сценарий 4: пользователь нажимает кнопку «Редактировать» и открывается вкладка редактирования данного блюда.

Сценарий 5: пользователь нажимает кнопку «Удалить», после чего блюдо удаляется из базы, а пользователя возвращает к просмотру списка рецептов блюд.

2.5 Функциональные требования

Таблица 3 – Сценарий регистрации

Сценарий использования	Регистрация
Приоритет	Важно/Желательно/Необязательно
Триггер	Нажатие на кнопку «Регистрация»

Окончание таблицы 3

Предусловие	Открытие формы регистрации Корректное заполнение обязательных полей формы
Основной сценарий	Пользователь уведомлен об успешной регистрации
Постусловие	Добавление нового пользователя в базу данных Перенаправление пользователя в окно входа
Сценарий исключительных ситуаций	Пользователь не добавлен в базу, появляется окно с предупреждением.

Таблица 4 – Сценарий авторизации

Сценарий использования	Вход в систему
Приоритет	Важно/Желательно/Необязательно
Триггер	Нажатие на кнопку «Вход»
Предусловие	Открытие формы входа Корректное заполнение обязательных полей формы
Основной сценарий	Проверка пользователя в базе
Постусловие	Открывается главное окно приложения, пользователь авторизирован
Сценарий исключительных ситуаций	Пользователя нет в базе, появляется окно с предупреждением.

Таблица 5 – Сценарий создания блюда (рецепта блюда)

Сценарий использования	Создание блюда
Приоритет	Важно/Желательно/Необязательно
Триггер	Нажатие на кнопку «Сохранить» на управляющей панели вкладки редактирования.
Предусловие	Открытое главное окно с помощью кнопки «добавить рецепт»
Основной сценарий	Добавление блюдо в базу
Постусловие	Перенаправление пользователя во вкладку просмотра рецепта.
Сценарий исключительных ситуаций	База не отвечает. Появляется окно с предупреждением.

Таблица 6 – Сценарий редактирования блюда

Сценарий использования	Редактирование блюда
Приоритет	Важно/Желательно/Необязательно
Триггер	Нажатие на кнопку «Сохранить» на управляющей панели вкладки редактирования.
Предусловие	Открытие окна с помощью нажатия на клавишу редактирования блюда во вкладке просмотра блюда
Основной сценарий	Изменение блюда в базе
Постусловие	Перенаправление пользователя во вкладку просмотра рецепта.
Сценарий исключительных ситуаций	Запись была удалена из базы или база не отвечает. Появляется окно с предупреждением.

Таблица 7 – Сценарий создания закладки

Сценарий использования	Создание закладки
Приоритет	Важно/Желательно/Необязательно
Триггер	Нажатие на кнопку добавления закладки на управляющей панели вкладки просмотра рецепта.
Предусловие	Открытое окно просмотра рецепта, авторизация пользователя
Основной сценарий	Добавление закладки в базу
Постусловие	Изменение кнопки на кнопку удаления.
Сценарий исключительных ситуаций	Запись была удалена из базы или база не отвечает. Появляется окно с предупреждением.

3 Реализация программного продукта

3.1 Обоснование средств разработки

3.1.1 Использование.NET Framework и Windows Form

Для создания GUI приложения и разработки в целом было выбран Microsoft .NETFramework и его компонент WindowsForms. .NETFramework — программная платформа, выпущенная компанией Microsoft в 2002 году. Основой платформы является общеязыковая среда исполнения CommonLanguageRuntime (CLR), которая подходит для разных языков программирования. Функциональные возможности CLR доступны в любых языках программирования, использующих эту среду.

Программа для .NET Framework, написанная на любом поддерживаемом языке программирования, сначала переводится компилятором в единый для .NET промежуточный байт-код CommonIntermediateLanguage (CIL) (ранее назывался MicrosoftIntermediateLanguage, MSIL). В терминах .NET получается сборка, англ. assembly. Затем код либо исполняется виртуальной машиной CommonLanguageRuntime (CLR), либо транслируется утилитой NGen.exe в исполняемый код для конкретного целевого процессора

WindowsForms - новый стиль построения приложения на базе классов .NETFrameworkclasslibrary. Они имеют собственную модель программирования, которая более совершеннее, чем модели, основанные на Win32 API или MFC, и они выполняются в управляемой среде .NET Common Language Runtime (CLR). Это целая новая парадигма программирования, которая реализована на библиотеке классов .NETFramework class library и содержит более единую модель программирования, улучшенную защиту и более богатые возможности для написания полнофункциональных веб-приложений. И это только начало.

Главная выгода от написания Windows-приложений с использованием — WindowsForms - это то, что WindowsForms гомогенизируют (создают более однородную (гомогенную) структуру) программную модель и устраняют многие ошибки и противоречия от использования WindowsAPI. Другая выгода от WindowsForms - вас используете тот же самый API, независимо от языка программирования, который вы выбрали. В прошлом, выбор языка программирования управлял выбором API. Если вы программировали в VisualBasic, вы использовали один API (реализованный на языке VisualBasic), в то время как программисты C использовали Win32 API, а программисты C++, вообще говоря, использовали MFC. MFC-программисту было трудно переключиться на VisualBasic и наоборот. Но теперь такого больше нет. Все приложения, которые используют WindowsForms, используют один API из .NETFrameworkclasslibrary. Знание одного API достаточно позволит программисту писать приложения фактически на любом языке, который он выберет.

WindowsForms никак не меньше, чем современная модель программирования для GUI приложений. В отличие от модели программирования Win32, в которой многое идет еще от Windows 1.0, новая модель была разработана с учетом всех современных требований.

3.1.2 Язык программирования C#

C# — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как язык разработки приложений для платформы Microsoft .NET Framework и впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую

типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Переняв многое от своих предшественников — языков C++, Pascal, Модула, Smalltalk и, в особенности, Java — C#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем.

3.1.3 VisualStudio 2015

MicrosoftVisualStudio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии WindowsForms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, WindowsMobile, Windows CE, .NET Framework, Xbox, WindowsPhone .NET CompactFramework и Silverlight.

VisualStudio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. VisualStudio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий

исходного кода (как, например, Subversion и VisualSourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспеченияТак, например, одним из используемых инструментов VisualStudioв данной курсовой работе является инструмент «MicrosoftVisualStudio 2015 InstallerProjects». Данный инструмент позволит создавать установочный паект для данного приложения за пару минут.

3.2 Моделирование бизнес-процессов

Моделирование бизнес-процесса является крайне важным этапом разработки, так как позволяет оценить работу приложения и в дальнейшем оптимизировать его работу.

На основании технического задания была создана блок-схема бизнес-процессов приложения (рисунок 7)

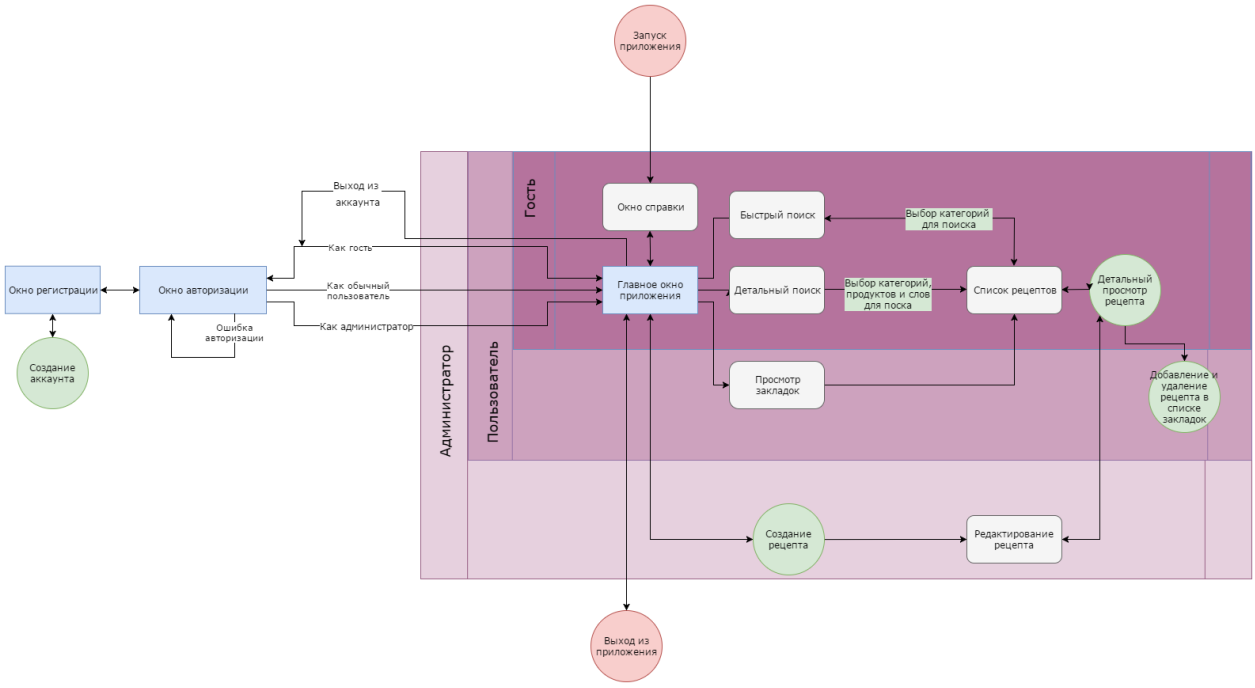

Рисунок 7

Для составления схемы был выбран сайт Draw.io [6], который позволяет создавать и редактировать блок-схемы онлайн. Данный сайт помог обозначить элементы схемы:

- Зеленые блоки означают, что данные блоки обозначают функции, требуемые в ТЗ.
- Синие блоки подсказывают, в какое окно открыто в определенный момент времени.
- Серые блоки обозначают работу с элементами интерфейса (кнопками, панелями и т.д.).
- Фиолетовые блоки разного оттенка обозначают с какими правами доступа выполняется операция. Данные блоки вложены, так как права пользователей имеют подобную иерархическую структуру.
- Стрелки между блоками показывают, доступ одной операции из другой. Они могут быть как односторонними, так и двусторонними, так как при некоторых операциях должна быть возможность вернуться назад.
- Красные круги означают начало и конец приложения.

Из главного окна в окно авторизации можно попасть только как гость, это объясняется тем, что в конкретный момент (по ТЗ) может быть видна либо кнопка авторизации, либо выхода.

3.3 Проектирование БД

3.3.1 Концептуальное проектирование БД

Назначение концептуальной модели состоит в том, чтобы представить формализованную информацию о предметной области таким образом, чтобы она было достаточно емкой для оценки глубины и корректности проработки проекта базы данных.

Были выделены следующие сущности, необходимые для решения задачи:

1. Пользователь
2. Блюдо
3. Ингредиент
4. Продукт
5. Категория
6. Элемент категории
7. Закладка

Основными сущностями БД, как можно заметить, являются блюдо (рецепт) и пользователь. При детальном рассмотрении ТЗ и требуемого функционала следует, что следует ввести еще роли. Т.е. каждый пользователь должен обладать ограниченными правами на доступ. На рисунке 8 представлена концептуальная схема базы данных, которая описывает сущности-объекты, необходимые для решения поставленной задачи в данной курсовой работе.

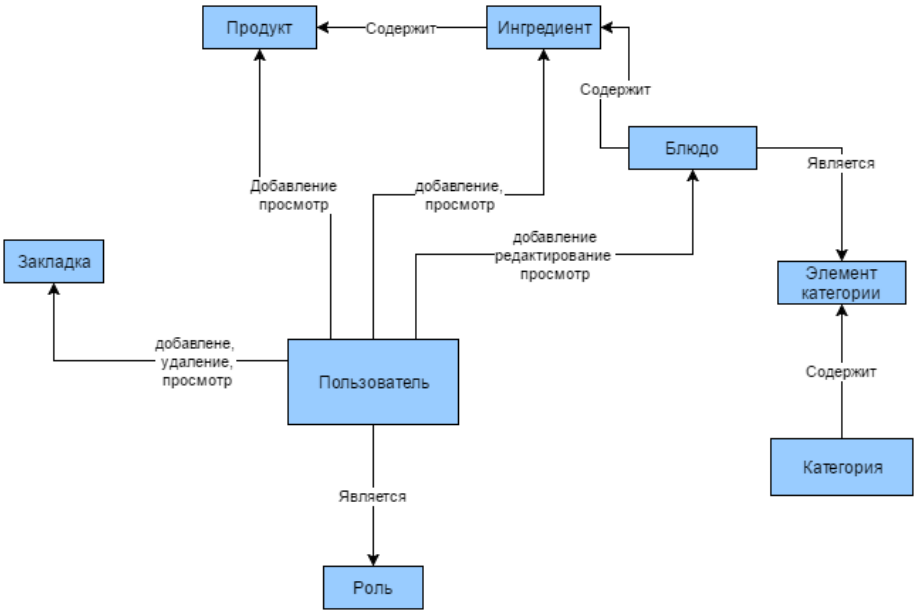

Рисунок 8

3.3.2 Логическое проектирование

При концептуальном проектировании были выделены основные сущности базы данных. Теперь, при логическом проектировании, можно выделить их атрибуты и связь между ними.

Пользователь имеет поля для: логина, пароля, имени и уровня доверия к пользователю. К пользователю привязаны блюда, которые он создал и закладки, которые, также, привязаны к блюдам. По сути, закладки являются просто связующей таблицей в отношении многие-ко-многим.

Блюдо имеет поля: имя, описание и изображение. Блюдо привязано только к пользователю, а к блюду привязаны ингредиенты и подклассы. Подклассы связаны отношением многие-ко-многим с блюдом с помощью дополнительной таблицы, так же они связанны с определенным классом.

Классы (категории) нужны для создания групп блюд. Например, можно разделить блюда по национальности или по времени употребления. Подклассы же являются критериями для разделения внутри классов. Примерами подклассов могут быть (для разделения по времени употребления): обед, завтрак, ужин.

В таблице ингредиента есть лишь ссылка на продукт и строка для представления количества, продукта. Строка выбрана из-за того, что не всегда получается представить измерение продуктов в виде числа. Представление в виде строки позволяет записывать ингредиенты понятными для большинства людей фразами: «2 штуки», «щепотка», «по вкусу».

На изображении ниже представлена логическая схема базы данных для курсового проекта. Для построения данной схемы было выбрано веб-приложение DbDesigner [7].

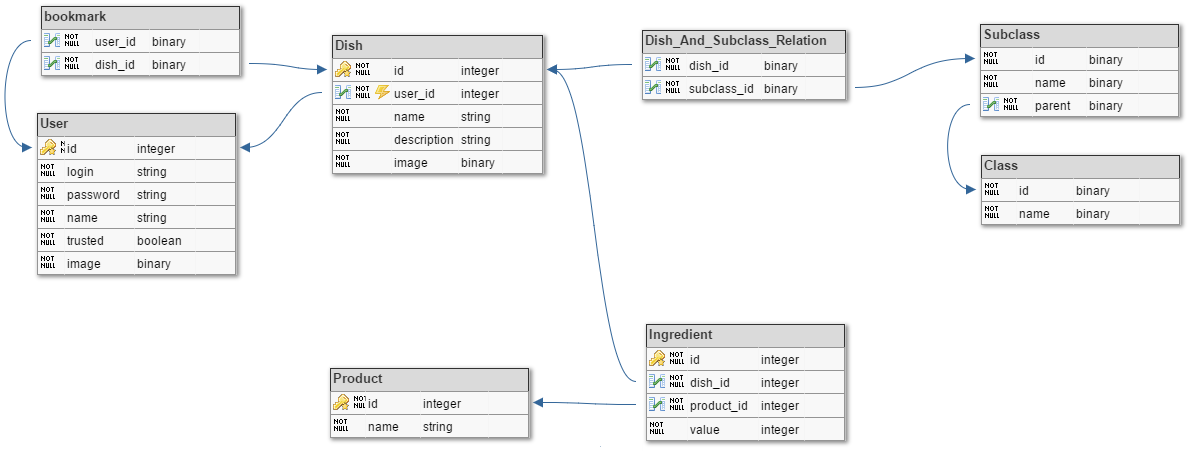

Рисунок 9

3.3.3 Физическая модель БД

Физическое проектирование базы данных подразумевает процесс переноса реализации базы данных на определенную СУБД. Поскольку, была выбранна MS SQL LocalDB, то файл базы данных лежит в папке с проектом с названием CulinaryGuideDB и расширением mdf, а не расположен на сервере.

Был создан файл баз данных, с помощью Visual Studio, с которым и будет работать приложение. Особенность работы Visual Studio с файлами данного типа в том, что при запуске база копируется в каталог Debug и там используется. Т.е. изменения при разработке приложения не влияют на данные в исходной базе. Это удобно, т.к. не приходится либо копировать базу, либо восстанавливать ее вручную.

Идентификаторы в таблицах используют тип int и присваиваются экземплярам данных начиная с единицы, и увеличиваясь каждый раз на единицу, что делает их уникальными. Все таблицы имеют первичный ключ, за исключением таблиц, которые организуют связь многие-ко-многим. Связи один-ко-многим были реализованы с помощью вторичного ключа, который имеет такой же тип данных, что и вторичный ключ. И в результате были получены таблицы, как на рисунке 10.

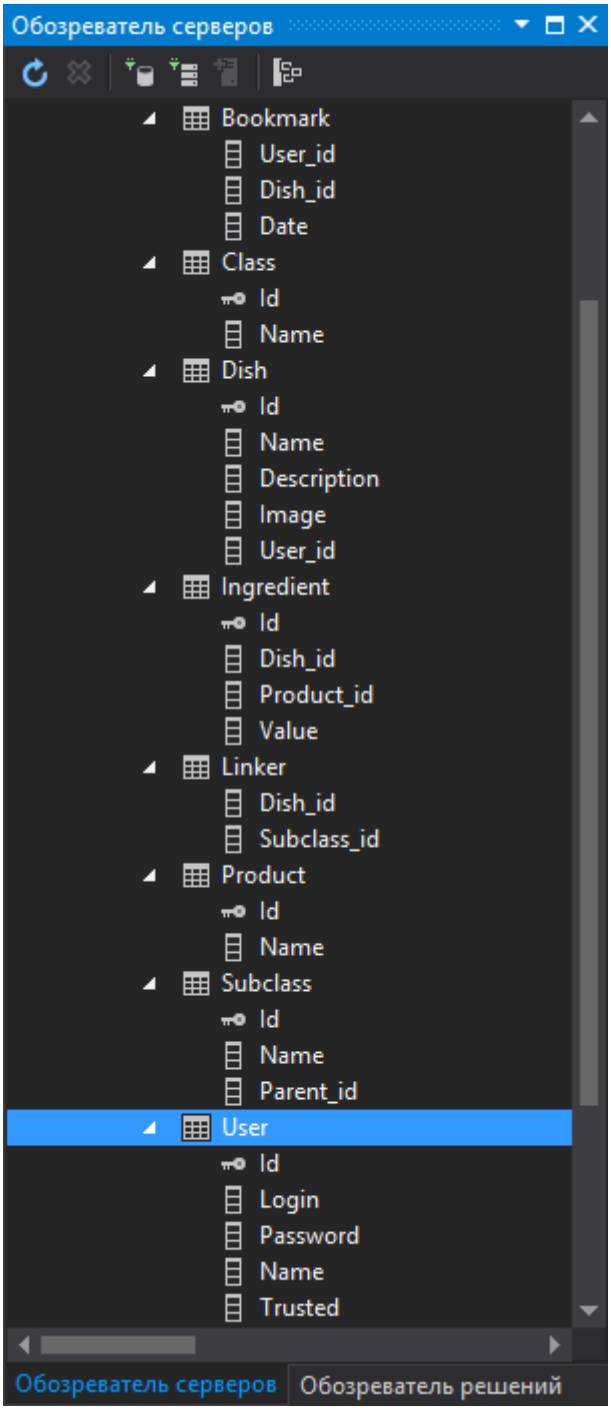

Рисунок 10

Учитывая особенности работы проекта с локальной БД, которые были указаны в данном пункте, база была заполнена первичными данными, чтобы можно было в полной мере тестировать функции приложения.

4 Разработка электронного кулинарного справочника

4.1 Разработка пользовательского интерфейса

Интерфейс приложения – важная особенность приложения, так как это связь между пользователем и программой. Основные характерные черты пользовательского интерфейса данного приложения:

- В WindowsForms интерфейс приложения строится на стандартных элементах Windows, что упрощает понимание пользователем того, как с ними взаимодействовать.
- В дизайне приложения был использован популярный на данный момент стиль Flat.
- Основным цветом приложения выбран – зеленый, а в особенности: seagreen, panelgreen иlightgreen.
- Способ организации графического интерфейса – MDI (*multipledocumentinterface).*

В соответствии вышесказанным, при запуске приложения пользователем открывается главное окно в нем открыта вкладка со справкой, а слева в блоке поиска открыт быстрый поиск (рисунок 11). Как можно увидеть, почти все кнопки оформлены в виде черно белых изображений, каждое из которых характеризует операцию, с которой эта кнопка или

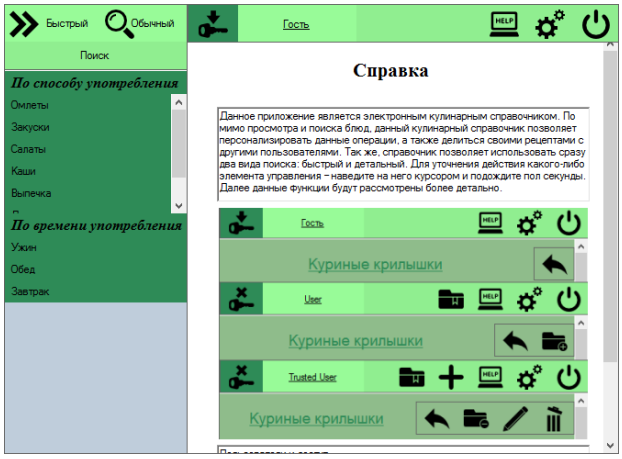

Рисунок 11

переключатель связанны.

На рисунке 12 изображены иконки всех кнопок и описание их функций.

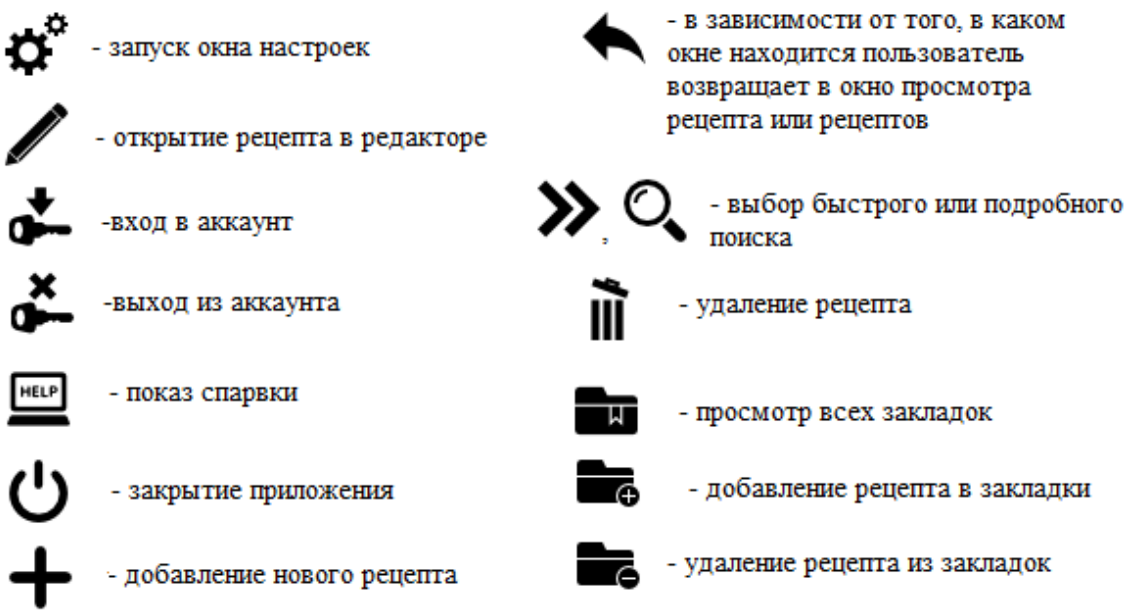

Рисунок 12

В соответствии с ТЗ главное окно состоит из трех логических частей: панель управления приложением и аккаунтом, панель поиска и главная панель. Главная панель и панель поиска реализованы с помощью элемента tabcontrol. Данная особенность позволяет проектировать пользовательский интерфейс не программно (не создавая дочернее окно или меня свойство Visiblекаждый раз, при клике), а в визуальном редакторе VisualStudio (рисунок 13), также избавляет от проверки повторного нажатия клавиши выбора меню, что уберегает от перезагрузки формы, которое пользователь увидит, как «мигание». Блок поиска состоит из двух кнопок переключения, которые контролируют вкладки поискового tabcontrol и кнопки запуска поиска, а главный tabcontrol состоит из четырех страниц, все из которых будут рассмотрены далее.

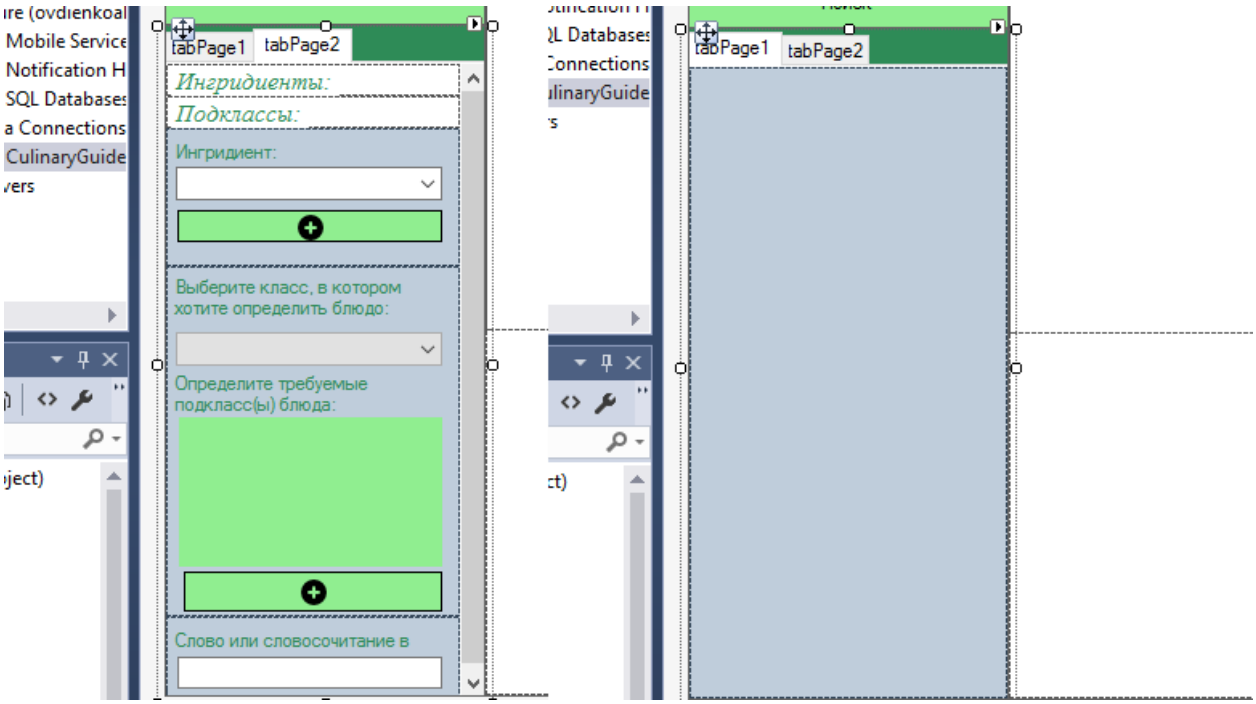

Рисунок 13

Начальная страница поискового tabcontrol изначально пуста и заполняется лишь при старте приложения. Как можно заметить на рисунке 13 заполняется эта страница пользовательскими элементами. Данный пользовательский элемент был разработан специально для отображения списка элементов категории, поэтому состоит из верхушки (название категории) и панели с переключателями checkbox (элементы категории), стилизованными под приложение.

Вторая страница поискового tabcontrolимеет два списка listview, один для списка ингредиентов, второй для списка элементов категории. Каждый элемент списка – пользовательский элемент, который уничтожается при нажатии на него. Данная функция важна, так как это простой способ удалять элементы из списка, а не пересоздавать список каждый раз. После списков расположена панель с combobox для выбора продукта. Список продуктов будет загружаться из базы, и при этом пользователь может ввести свой продукт. При нажатии на клавишу выбранный продукт добавляется в список ингредиентов. Если продукта нет в базе, то, обладая должными правами, пользователь может добавить продукт в базу (рисунок14). Далее идет панель

для добавления элементов категории с помощью combobox и уже знакомым пользовательским элементом для просмотра списка элементов категории. Работает это так, что, когда пользователь выбирает категорию из combobox в пользовательский элемент загружается список из элементов этой категории, далее пользователь может выделить некоторые из них и нажать на кнопку добавления, после чего выделенные элементы категории появятся в списке элементов категорий. И в самом низу панели находится textbox, введенное туда слово обязательно должно содержаться в описании или названии блюда.

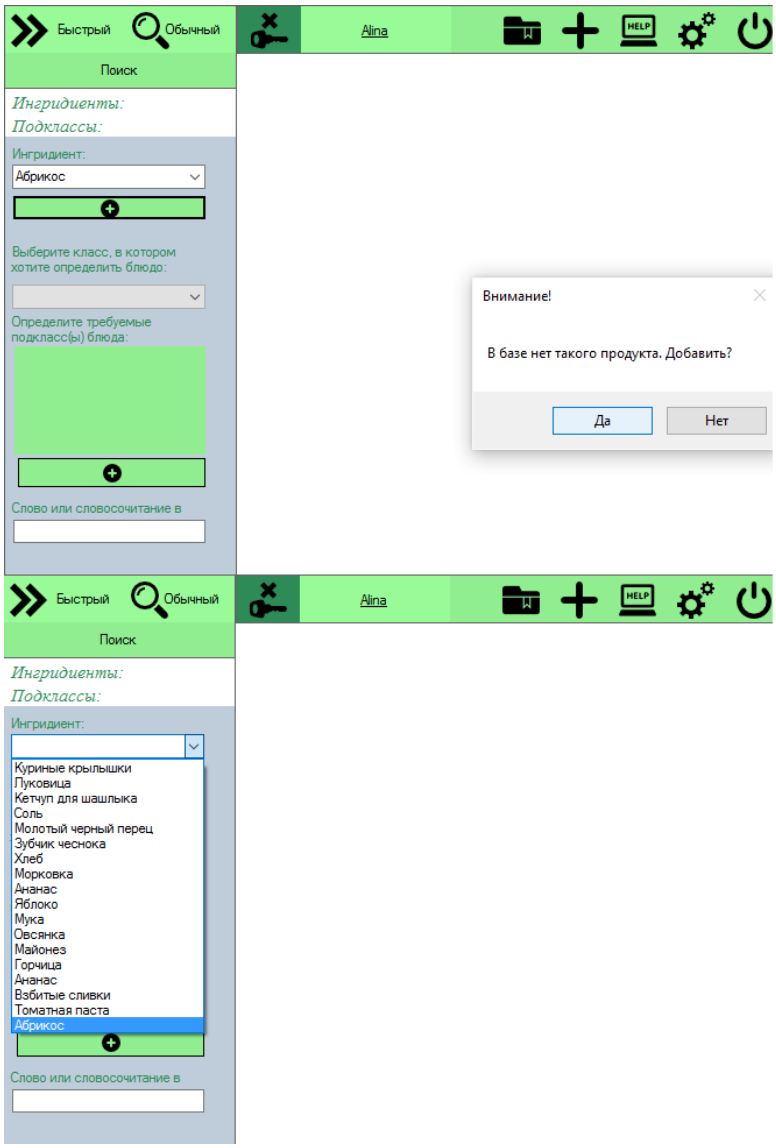

Рисунок 14

Первая страница главного tabcontrol – это панель для просмотра списка рецептов (рисунок 15). Рецепты на этой панели представлены в виде минимизированного окна. Оно содержит название блюда, ингредиенты (показывает первые 6) и описание, которое вмещается в свободное пространство, отведенное под элемент.

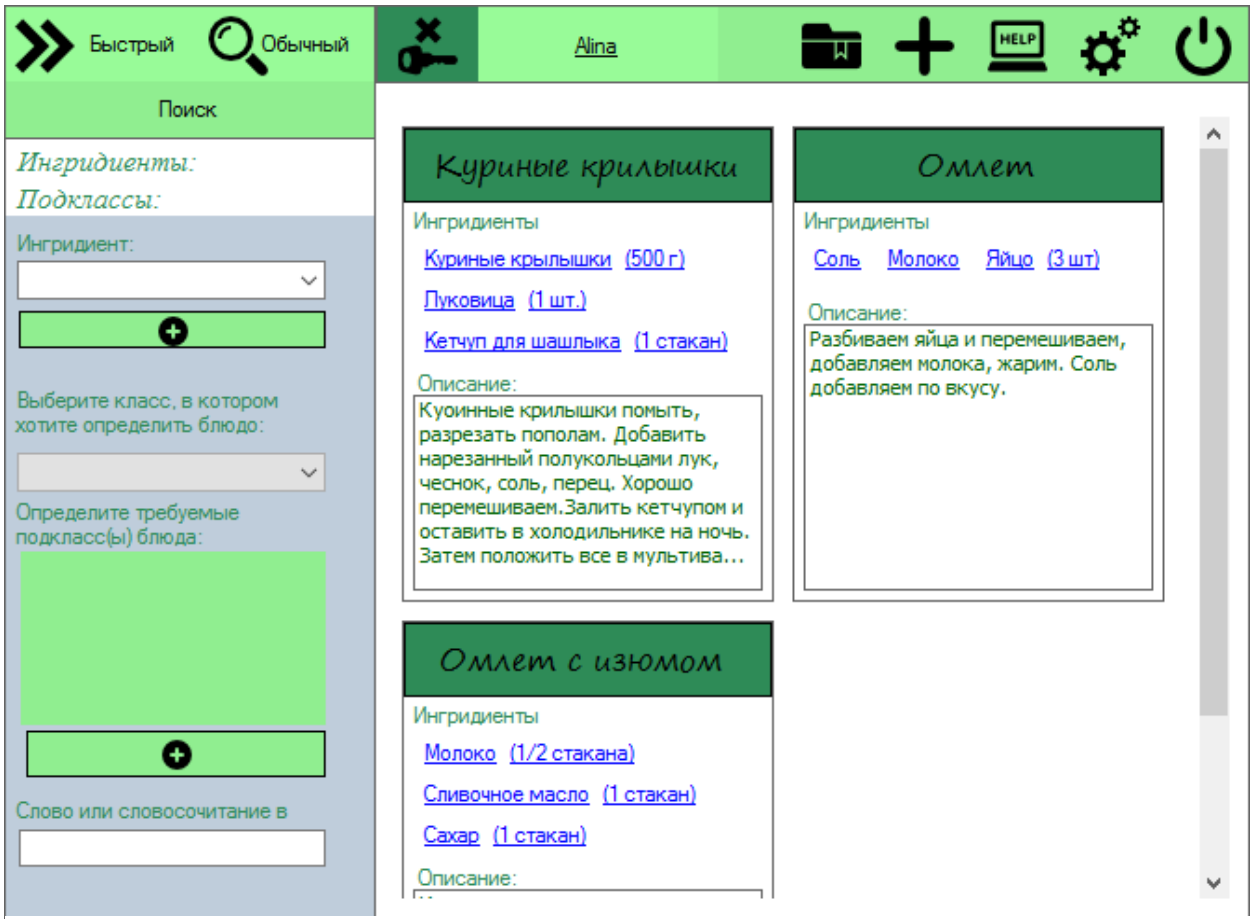

Рисунок 15

При нажатии на название текста для пользователя в главном tabcontrol открывается страница просмотра блюда. На данной странице отображаются все атрибуты блюда: название, ингредиенты, описание, картинку и все категории, к которым блюдо принадлежит. По мимо атрибутов блюда, на данной странице содержится меню управления с кнопками: «назад», «добавить/удалить закладку», «редактирование» и «удаление». Данные элементы доступны только тогда, когда пользователь имеет права на их

использование, как можно увидеть на рисунке 16. Например, пользователь не увидит кнопки «удалить», если он не доверенный или же в один момент времени можно увидеть только одну из двух кнопок редактирования закладок добавление/удаление. Чтобы увидеть следующую страницу главного tabcontrol нужно нажать на кнопку редактирования в вышеописанной странице либо нажать на кнопку «добавить» на главной управляющей панели.

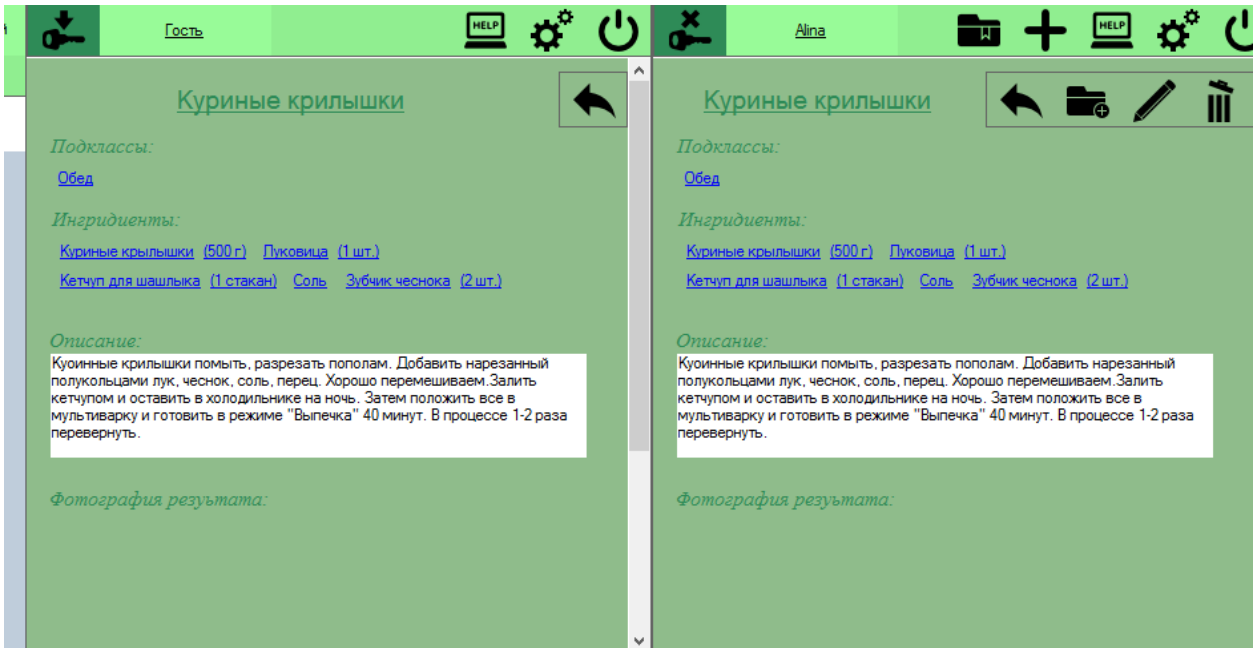

Рисунок 16

Страница редактирования (рисунок 17) очень схожа со страницей просмотра, только на данной странице текстовые поля редактируемые, а также добавлены две панели для добавления ингредиентов и элементов категории. Действие этих панелей аналогично действию панелей в детальном поиске. Меню управления на данной странице состоит лишь из двух элементов: «назад» и «сохранить». Оставшаяся страница – справка, ее можно увидеть на рисунке 11. На ней содержится инструкция по использованию, а также информация о приложении.

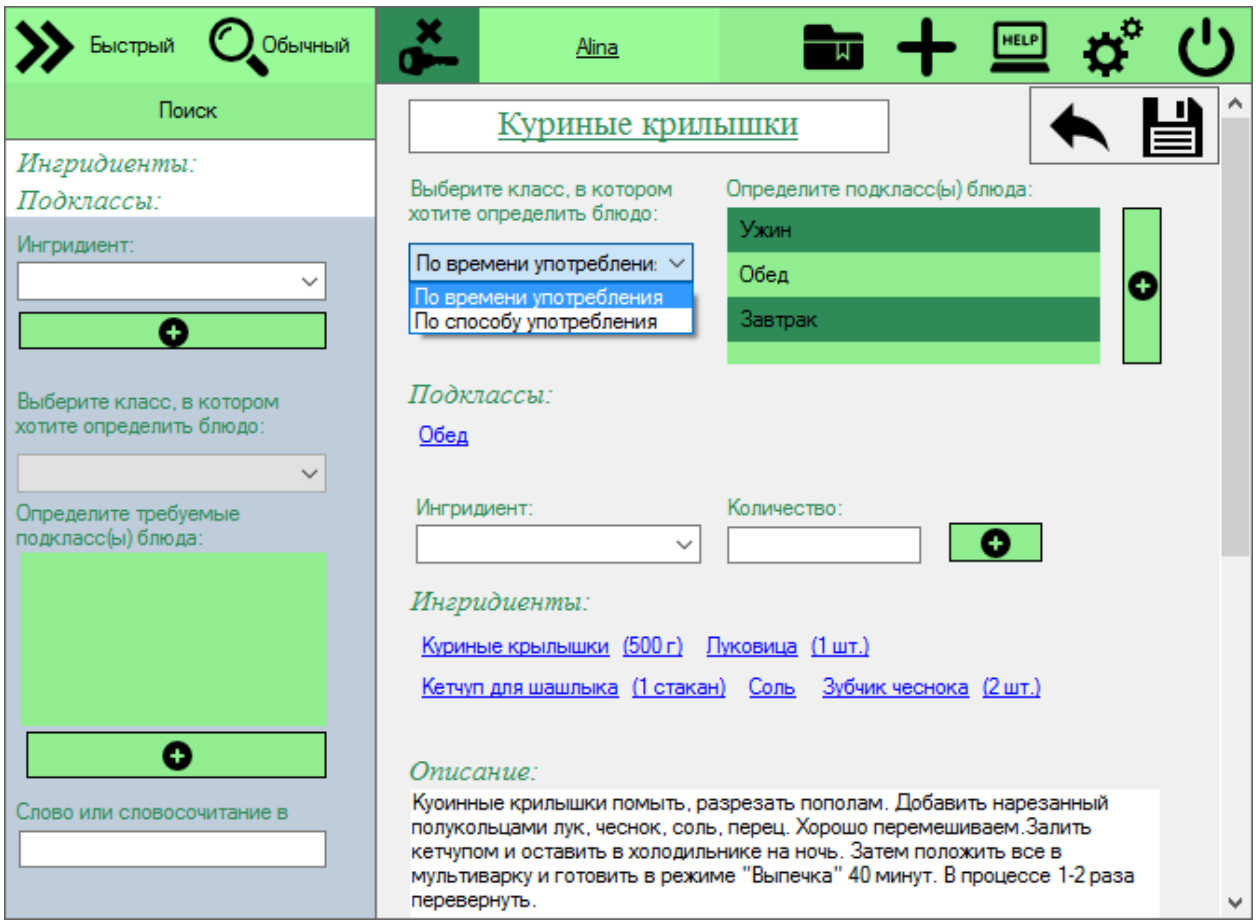

Рисунок 17

Главную панель управления приложением можно разбить на две чести: левую и правую. Справа находятся управляющие кнопки, функционал которых описан ранее (рисунок 12). Некоторые из них скрываются, если у пользователя недостаточно прав. Слева находится две кнопки и текстовое поле. Если пользователь не авторизирован, то в текстовом поле написано «Гость» и видна кнопка «зайти в аккаунт», иначе в текстовом поле находится имя пользователя, а рядом кнопка «выйти из аккаунта».

При нажатии на кнопку «войти в аккаунт» для пользователя открывается модальное диалоговое окно, изображенное на рисунке 18 в котором он сможет ввести свои логин и пароль, вернуться в главное меню или открыть окно регистрации (рисунок 19). При открытии окна регистрации окно входа в систему не закрывается, а скрывается, тогда как закрытии окна регистрации окно входа снова становится видимым.

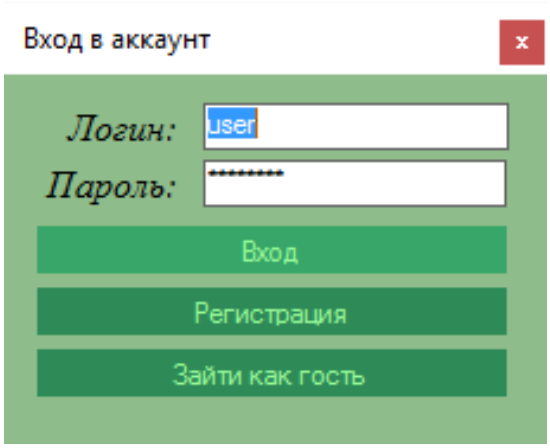

Рисунок 19

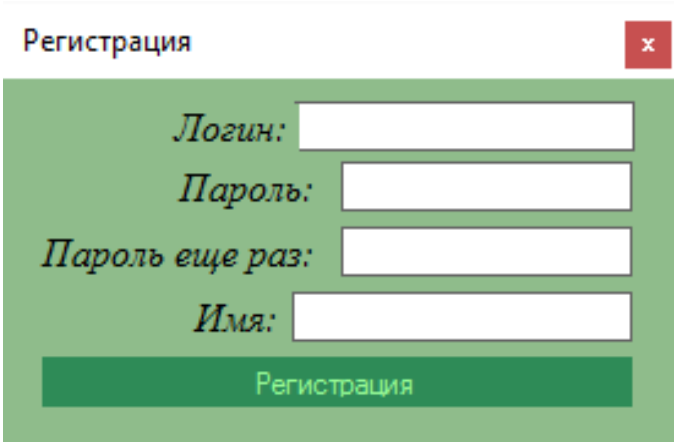

Рисунок 18

4.2 Разработка логики приложения

После создания каркаса приложения его стоит наполнить функционалом. Это легко осуществить с помощью привязки определенных методов к элементам интерфейса. Но перед началом создания логики, в приложение стоит ввести основные объекты, которые в дальнейшем будут часто использованы. Это позволит улучшить переносимость приложения,а также ряд других факторов, таких, например, как читабельность кода.

Из всех объектов следует выделить:

- Пользователь
- Рецепт приготовления
- Ингредиент
- Продукт
- Категория блюда
- Элемент категории
- Класс с данными для поиска

Все данные объекты больше схожи на структуры и имеют лишь поля атрибутов и конструктор, но при этом некоторые из них могу содержат другие.

В соответствии с интерфейсом приложения, функциональные требования из ТЗ были разбиты на следующие методы:

Таблица 8 – Методы класса MainForm

Название метода	Описание
RefreshLocalData	Обновляет из базы локальные переменные: список всех продуктов в базе, информацию о пользователе, классах, блюдах. Используется почти во всех методах, где осуществляется работа с данными базы.
FillPanel(List<DishClass>)	Заполняет страницу просмотра списка блюд, принимая в качестве параметра этот самый список. Вызывается из метода начала поиска.
OpenDish	Открывает страницу просмотра рецепта и заполняет данными из локальной переменной dish. В данную переменную заносится информация о рецепте при нажатии на название рецепта в просмотре списка рецептов.
EditDish	Открываетя при редактировании или создании блюда. При редактировании поля панели заполняются данными из локальной переменной dish, а при создании остаются пустыми.
StartSearchBtn_Click	В зависимости от выбранной страницы поиска запускает быстрый или детальный поиск. Заполняет класс SearchData данными о поиске, отсеивает неподходящие блюда и выводит подходящие с помощью метода FillPanel

Окончание таблицы 8

SaveDishBtn_Click	Создает или изменяет рецепт блюда на основе элементов из страницы редактирования рецепта.
editAddIngBtn_Click, addIngToDSBtn_Click	Методы для добавления ингредиентов, которые позволяют добавлять в базу новые продукты.
loginBtn_Click	Открывает модальное окно логирования.
logoutBtn_Click	Выход из текущего аккаунта пользователя.

Таблица 9 – Методы форм и пользовательских элементов

Название формы или пользовательского элемента	Название метода	Описание
loginForm	loginBtn_Click	Проверяет наличие в базе пользователя по логину и паролю.
registrationForm	registrationBtn_Click	СоставляетобъектUser на основе текстовых полей формы и регистрирует пользователя в базе.
MinDishForm	MinDishForm(MainForm, DishClass)	Заполняет элементы формы на основе полученного в форму класса рецепта.
MinDishForm	nameLinkBtn_Click	Открывает в главном окне блюдо, которое описанное в данной форме.

Окончание таблицы 9

DishClassLBUC	SetList(DishCategoryClass)	Заполняет пользовательский элемент элементами категории, переданной в параметрах.
DishClassLBUC	List<SubcategoryClass>GetSelectedList	Возвращает список элементов категории, выделенных в пользовательском элементе.

4.3 Проектирование взаимодействия с БД

Учитывая потребности приложения было бы оптимально созданы класс для работы с базой данных. Данный класс позволит обращаться к базе не из методов форм приложения, а через вызов его методов. Это убережет от повторения кода, а также позволит отделить логику работы с данными от логики приложения.

Для работы с базой данных в .NETFramework есть замечательная библиотека ADO.NET. Библиотеки ADO.NET можно применять тремя концептуально различными способами: в подключенном режиме, в автономном режиме и с помощью технологии EntityFramework.

В рамках курсовой работы был использован подключенный уровень работы с БД. При использовании подключенного уровня (connectedlayer), кодовая база явно подключается к соответствующему хранилищу данных и отключается от него. При таком способе использования ADO.NET обычно

происходит взаимодействие с хранилищем данных с помощью объектов подключения, объектов команд и объектов чтения данных.

С точки зрения программиста, тело ADO.NET составляет базовая сборка с именем System.Data.dll. В этом двоичном файле находится значительное количество пространств имен, многие из которых представляют типы конкретного поставщика данных ADO.NET (рисунок 20).

Рисунок 20

Для работы MS SQL в ADO.NET содержит System.Data.SqlClient, это пространство имен содержит классы, используемые для подключения к базе данных Microsoft SQL Server, в том числе SqlDbCommand, SqlDbConnection и SqlDbDataAdapter. Эти классы оптимизированы для использования интерфейса TDS к SQL Server.

Таблица 10 - Основные объекты поставщиков данных ADO.NET

Тип объекта	Базовый класс	Интерфейсы	Назначение
Connection	DbConnection	IDbConnection	Позволяет подключаться к хранилищу данных и отключаться от него. Кроме того, объекты подключения обеспечивают доступ к соответствующим объектам транзакций

Окончание таблицы 10

Command	DbCommand	IDbCommand	Представляет SQL-запрос или хранимую процедуру. Кроме того, объекты команд предоставляют доступ к объекту чтения данных конкретного поставщика данных
DataReader	DbDataReader	IDataReader, IDataRecord	Предоставляет доступ к данным только для чтения в прямом направлении с помощью курсора на стороне сервера
DataAdapter	DbDataAdapter	IDataAdapter, IDbDataAdapter	Пересылает наборы данных из хранилища данных к вызывающему процессу и обратно. Адаптеры данных содержат подключение и набор из четырех внутренних объектов команд для выборки, вставки, изменения и удаления информации в хранилище данных
Parameter	DbParameter	IDataParameter, IDbDataParameter	Представляет именованный параметр в параметризованном запросе
Transaction	DbTransaction	IDbTransaction	Инкапсулирует транзакцию в базе данных

Конкретные имена этих основных классов различаются у различных поставщиков (например, SqlConnection, OracleConnection, OdbcConnection и MySqlConnection), но все эти объекты порождены от одного и того же базового класса (в случае объектов подключения это DbConnection), который реализует идентичные интерфейсы (вроде IDbConnection).

Разработанный класс для работы с БД содержит множество методов, которые по функциональному назначению можно разбить на несколько типов:

Таблица 11 – Методы класса DBWorker

Тип функции	Экземпляры функций	Функционал данного типа функций
Для работы с соединением	Connect,Close,IsOpen, RefreshConnect.	Все методы, необходимые для управления соединением.
Для возвращения объектов	GetSubCategoryById(int), GetUserByLoginPassword(string,string), GetSimpleClassById(int), GetProductIdByName(string) ит.д.	Методы, возвращающие один объект по какому-либо параметру, характеризующему таблицу в базе.
Для возвращения списков объектов	GetAllDishClasses(), GetAllSubCategoryByCategoryId(int), GetSubCategoryById(int), GetAllDishIdFromUserBookmarks(int) и т.д.	Методы, возвращающие списки объектов.
Для удаления объектов	DeleteAllDishIngredientsByDishId(int), DeleteAllDishSubclassesByDihId(int), DeleteTableById(int), DeleteBookmark(int, int).	Удаление объектов. Класс DeleteTableById – универсальный для всех таблиц, имеющих атрибут id.

Окончание таблицы 11

Для создания объектов	AddSubclass(int, int), AddIngredient(IngredientClass), AddUser(UserClass), AddBookmark(int, int) ит.д.	Добавление экземпляров таблиц в базу данных. В большинстве случаем эти методы принимают класс того объекта, который представлять данную таблицу.
Для изменения объектов	UpdateDish(Dish)	Обновление существующего экземпляра таблицы.
Для проверки наличия существования объекта	UserExist(string)	Возвращает true, если пользователь с переданным в метод логином есть в базе

5 Описание программного продукта

5.1 Описание объектов и их взаимодействия

На рисунке 21 изображены основные классы и их взаимодействие. Связь есть, как и между элементарными классами, представляющими объекты БД, так и между классами формы приложения, которые буферизируют классы из базы данных, позволяя переносить эти объекты между методами. Это один из способов связи в проекте.

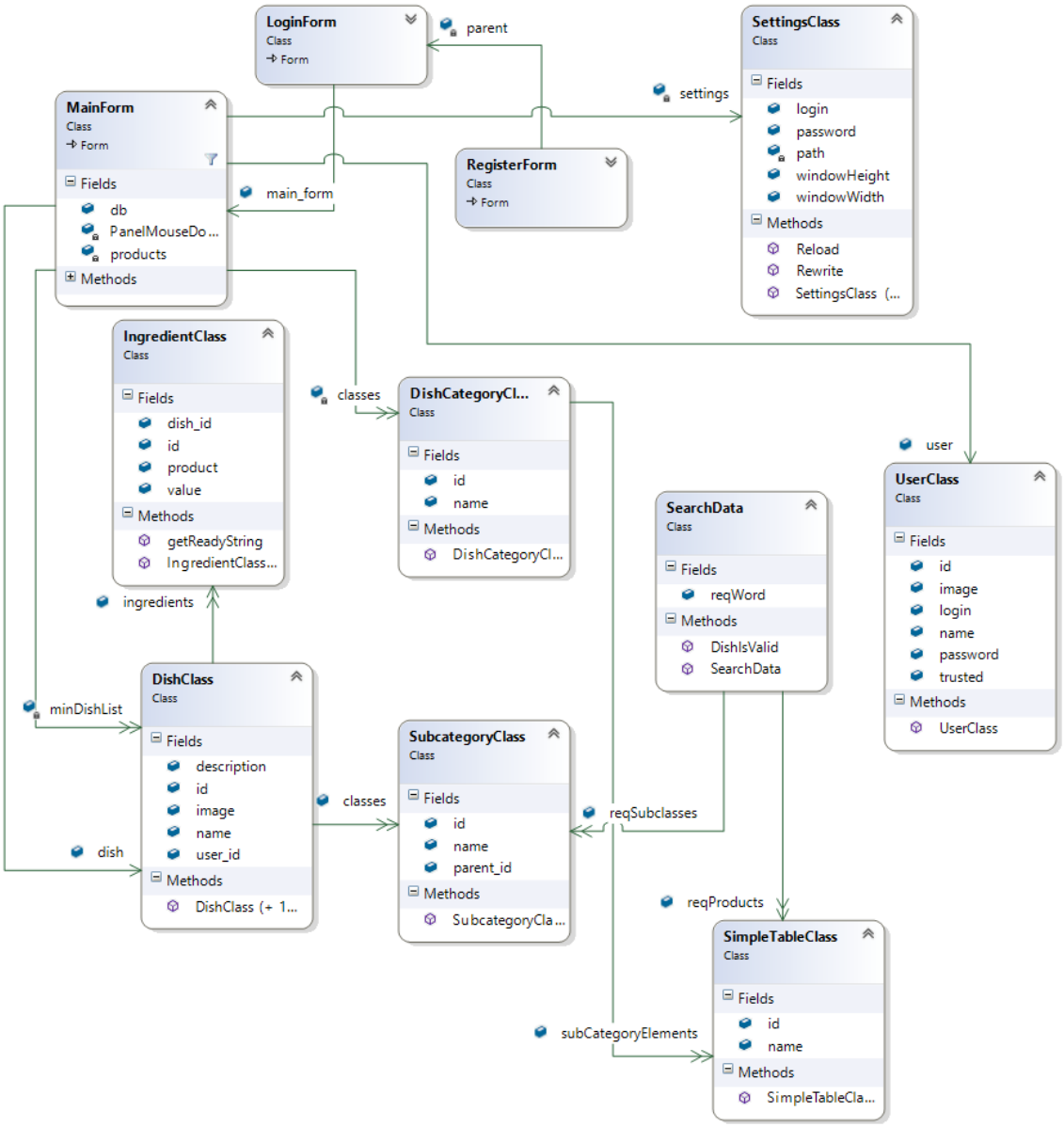

Рисунок 21

Так же в приложении есть класс для работы с базой данных, связи этого класса с формами приложения изображены на рисунке 22.

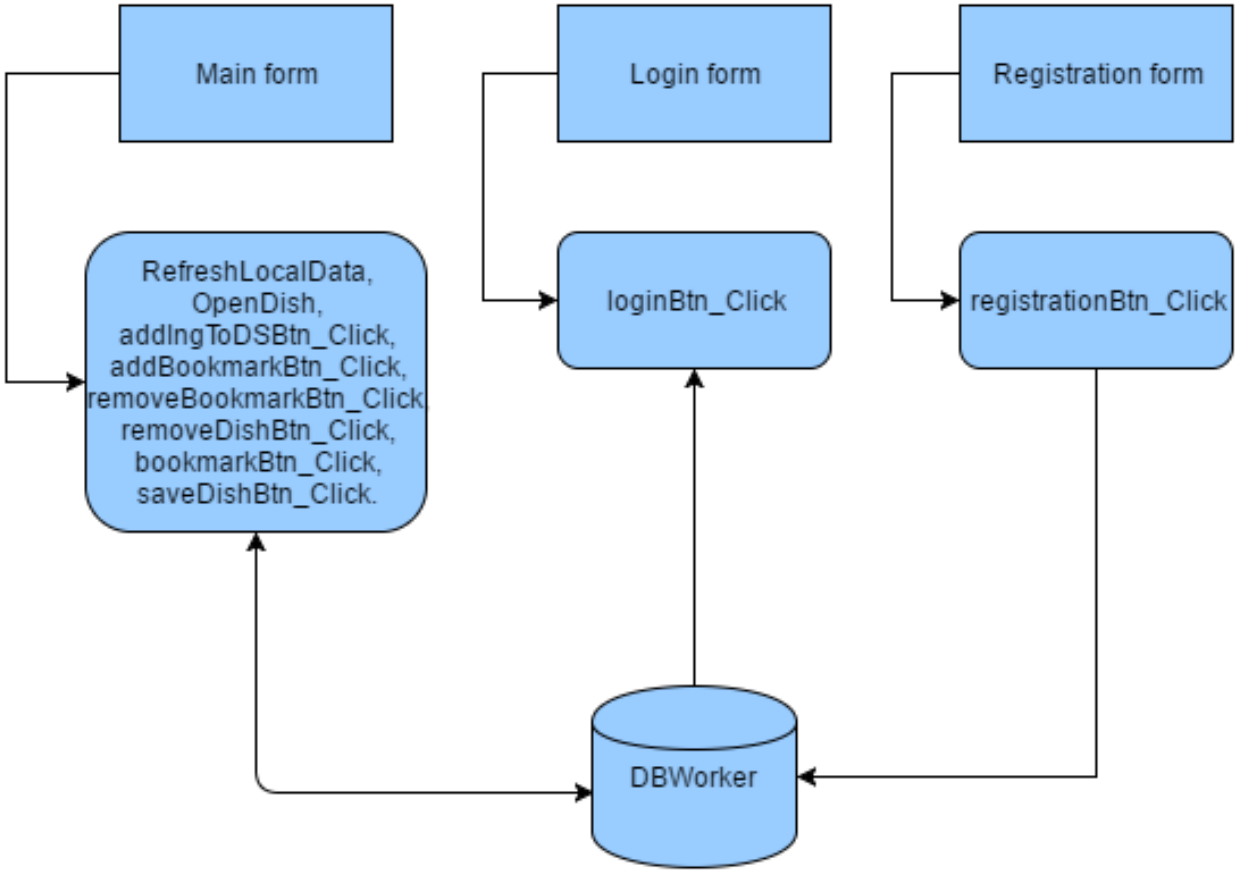

Рисунок 22

5.2 Описание Sql-запросов

Все sql-запросы приложения содержаться в методах класса DBWorker и строятся данные запросы с помощью стандартного метода string.Format(string, agrs*). Это позволяет переносить код из одного метода в другой, а также улучшить читабельность кода. Название функций было написано с учетом того, какой запрос находится в данной функции. Некоторые из них можно увидеть в таблице 12.

Таблица 12 – sql-запросы, существующие в методах класса DBWorker

Метод	Выражение для создания Sql-запроса
UserExist(stringlogin)	string.Format("SELECT * FROM [User] WHERE Login ='{0}'", login)
GetDishById(intid)	string.Format("SELECT * FROM [Dish] WHERE Id ='{0}'", id)
GetDishCatigoriesByDishId(intid)	string.Format("SELECT * FROM [Linker] where Dish_id = '{0}'",id)
GetAllMinDishs()	string.Format("SELECT * FROM [Dish]");
UpdateDish(DishClassdish)	string.Format("UPDATE [Dish] set Name=@Name,Description=@Description, Image=@Image where id='{0}'", dish.id.ToString())
AddDish(DishClassdish)	"INSERT INTO [Dish](Description,Name,User_id,Image) Values(@Description,@Name,@User_id,@Image)"
DeleteAllDishIngredientsByDishId (intid)	string.Format("Delete from [Ingredient] where Dish_id = '{0}'", id)
DeleteBookmark(intuser_id, intdish_id)	string.Format("Delete from [Bookmark] where User_id = '{0}' and Dish_id = '{1}'", user_id,dish_id)

Созданные команды выполняются с помощью класса SqlCommand. Команда вместе с открытым подключением посылаются в конструктор этого класса, а затем выполняется с помощью одного из следующих методов: ExecuteReader или ExecuteNonQuery. Первый возвращает объект, с помощью которого производится считывание данных из БД, а второй возвращает количество элементов БД, затронутых выполнимой командой. При выполнении операций INSERT или UPDATE следует так же задать параметры, передаваемые в базу. Пример работы с ADO.NET изображен на рисунке 23.

```
1 reference
public bool AddLink(int dish_id, int subclass_id)
{

    bool allFine = false;
    string Command = "INSERT INTO [Linker](Dish_id,Subclass_id) Values(@Dish_id,@Subclass_id)";
    try
    {
        using (SqlCommand cmd = new SqlCommand(Command, connection))
        {
            // Добавляем параметры
            cmd.Parameters.AddWithValue("@Dish_id", dish_id);
            cmd.Parameters.AddWithValue("@Subclass_id", subclass_id);

            if (cmd.ExecuteNonQuery() > 0)
                allFine = true;
        }

    }
    catch (SqlException ex)
    {
        MessageBox.Show(ex.Data.ToString(), "Ошибка подключения к базе!");
    }

    return allFine;
}
```

Рисунок 23

6 Тестирование и внедрение

6.1 Тестирование качества ПО

Анализ надежности отображен в таблице 13. Назначение, область применения и требования к надежности программного продукта:

- Назначение - предоставление справки о рецептах
- Область применения: поиск информации
- Требования к надежности: Скорость и стабильность работы

Таблица 13 - Анализ надежности программного продукта

№ Теста	Данные для теста (вводимая информация)	Время функционирования программы	Результат (успешно\отказ)	Результат
1.	Запуск приложения	1-2 секунды	успешно	Открытие главной формы приложения
2.	Попытка добавить несуществующий продукт в поиск без прав доверенного пользователя	Менее 1 секунды	успешно	Предупреждение о том, что элемента нет в базе
3.	Попытка добавить несуществующий продукт в поиск с правами доверенного пользователя	Менее 1 секунды	успешно	Предложение добавить элемент в базу
4.	Нажатие на кнопку выхода из аккаунта	Меньше секунды	успешно	Очищение локальной переменной пользователя, обновление элементов интерфейса

Окончание таблицы 13

5.	Нажатие на кнопку входа	Меньше секунды	успешно	Вызов окна авторизации
6.	Ввод корректных данных для входа	Меньше секунды	успешно	Авторизация в приложении
7.	Ввод некорректных данных для входа	Моментально	успешно	Предупреждение о неверном вводе данных
8.	Ввод корректных данных для регистрации	Меньше секунды	успешно	Регистрация в базе, предупреждение перенаправление пользователя к окну входа.
9.	Попытка создать блюдо, с уже существующим названием	Меньше секунды	успешно	Предупреждение об не оригинальности названия

Как можно заметить из таблицы 13 - приложение работает быстро и надежно. В надежность осуществлена с помощью динамичных панелей, которые исчезают, когда у пользователя нет прав на их использования и т.д. Данный шаг уменьшает количество возможных ошибок в приложении, а также упрощает интерфейс пользователя.

6.2 Руководство пользователя

6.2.1 Использование продукта

Данное приложение является электронным кулинарным справочником. По мимо просмотра и поиска блюд, данный кулинарный справочник позволяет персонализировать данные операции, а также делиться своими рецептами с другими пользователями. Так же, справочник позволяет использовать сразу два вида поиска: быстрый и детальный. Для уточнения действия какого-либо элемента управления – наведите на него курсором и подождите пол секунды. Вид приложения описан на рисунке 24. Далее данные функции будут рассмотрены более детально.

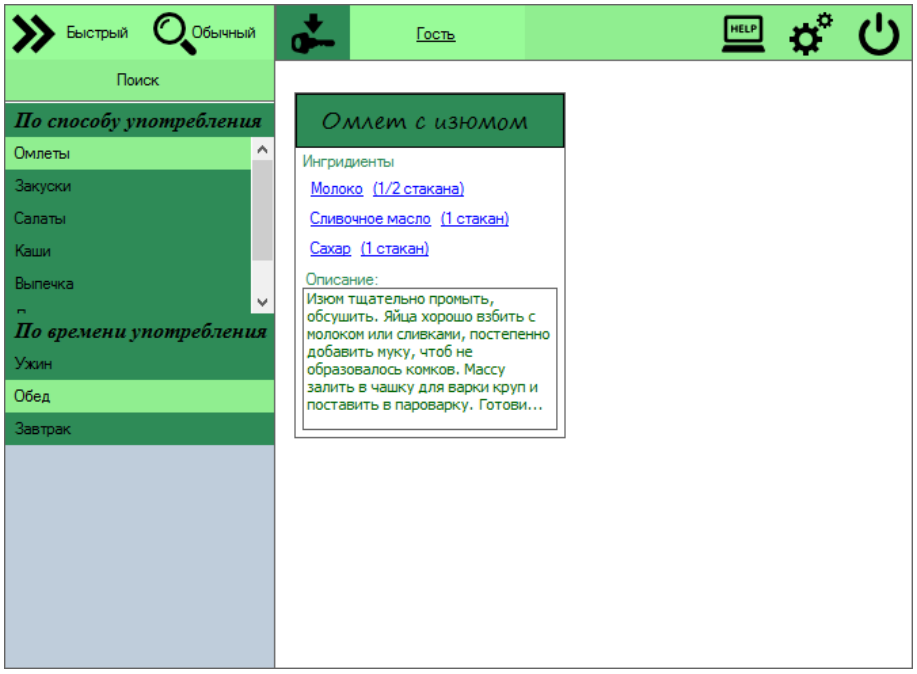

Рисунок 24 – Вид интерфейса приложения

Пользователи и доступ

В данном приложении есть три типа пользователей:

1. Гость. Гость может искать и изучать рецепты.
2. Зарегистрированный пользователь. Он может сохранять блюда в

закладки.

3. Доверенный пользователь. Данный пользователь может создавать и редактировать свои рецепты.

Каждый последующий тип имеет возможности всех предыдущих.

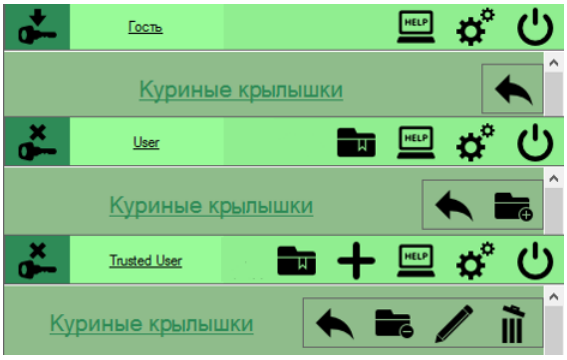

Рисунок 25 – вид панели управления блюдом
от лица разных пользователей

Вход на существующий аккаунт

Для входа в аккаунт надо нажать на кнопку входа и верно ввести в поля логин и пароль свои данные. При неверном вводе вас уведомят об ошибке (рисунок 25). Если по у вас нет аккаунта или по какой-то причине не можете зайти на свой, то можете зарегистрировать новый аккаунт.

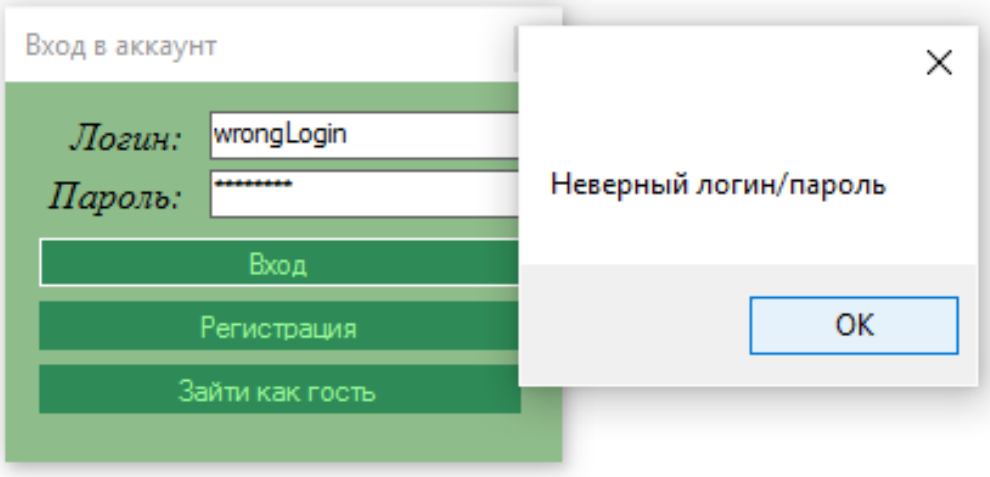

Рисунок 26 – Предупреждение пользователя о
неверно введенных данных

Регистрация

Для регистрации следует перейти из главного меню в меню входа в аккаунт, а в нем нажать на кнопку «Регистрация», после чего вам будет предложено зарегистрироваться. Для успешной регистрации стоит:

- Ввести уникальный логин.
- Заполнить все поля.
- Убедится, что пароль длиннее 8 символов.
- Убедится, что пароли в элементах «Пароль» и «Пароль еще раз» совпадают.

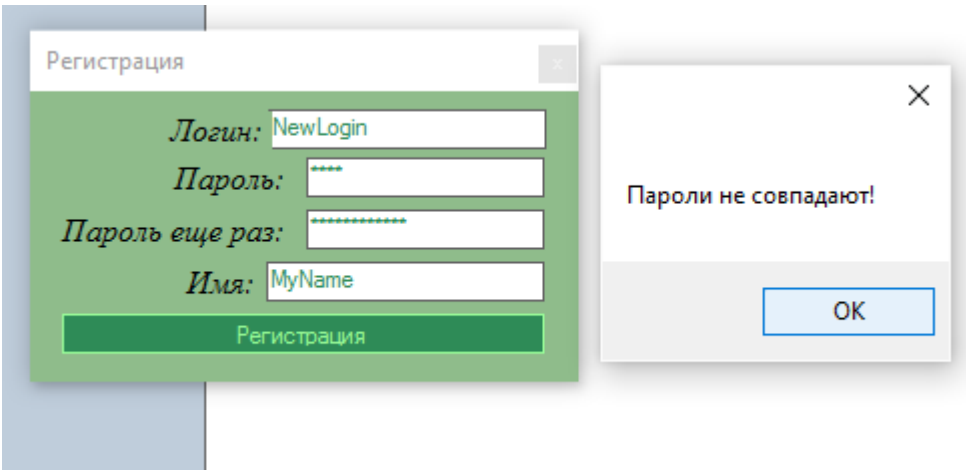

Рисунок 27 - Предупреждение пользователя
о неверно введенных данных

После нажатия клавиши при успешной регистрации вы будете перенаправлены назад в окно входа в аккаунт, либо будете предупреждены о проблемах в приложении.

Быстрый поиск блюд:

1. Откройте вкладку быстрого поиска
2. Выберите интересующие категории блюд
3. Нажмите поиск

Пример использования быстрого поиска показан на рисунке 28.

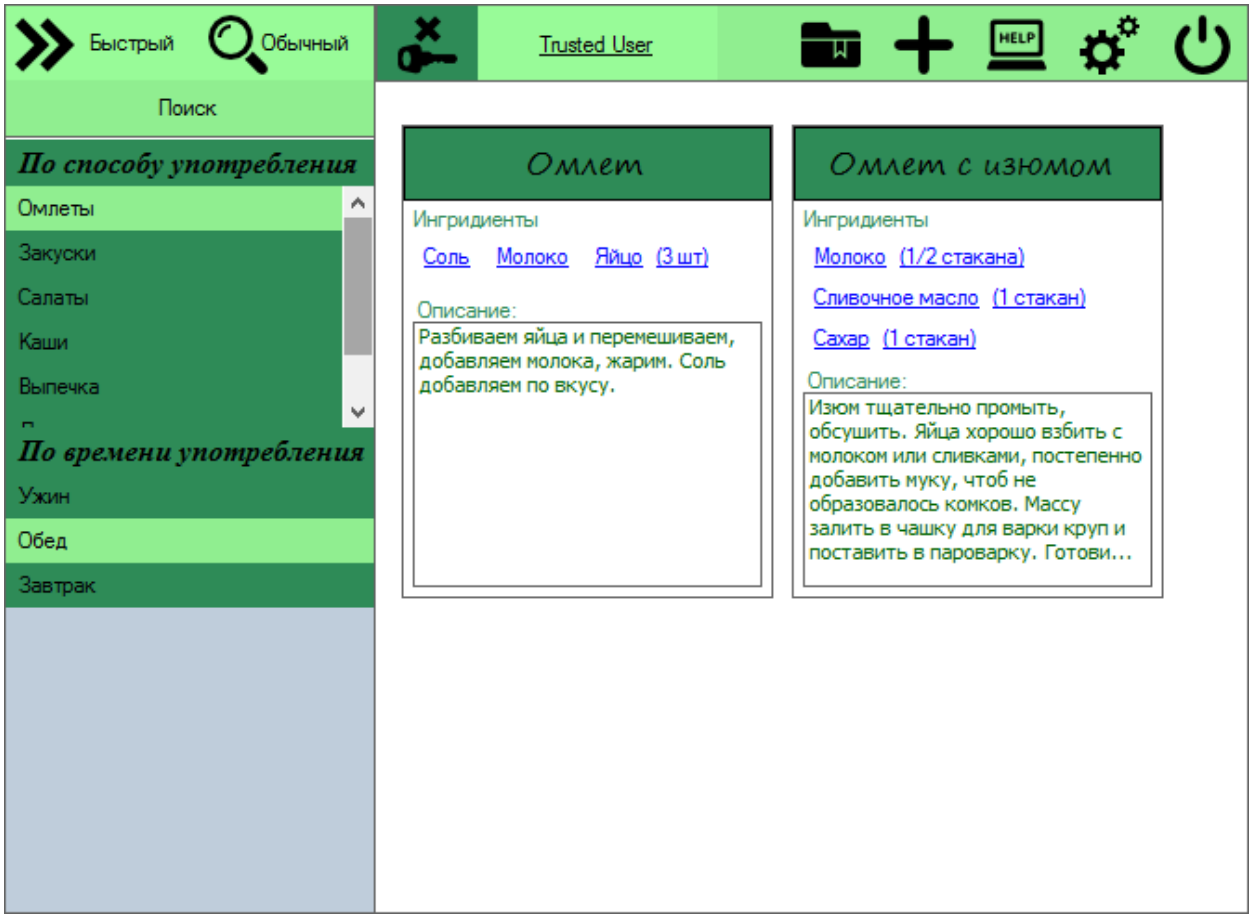

Рисунок 28

Детальный поиск блюд:

1. Откройте вкладку детального поиска
2. Выберите нужные вам критерии поиска
3. Нажмите на кнопку поиска

Пример использования детального поиска показан на рисунке 29.

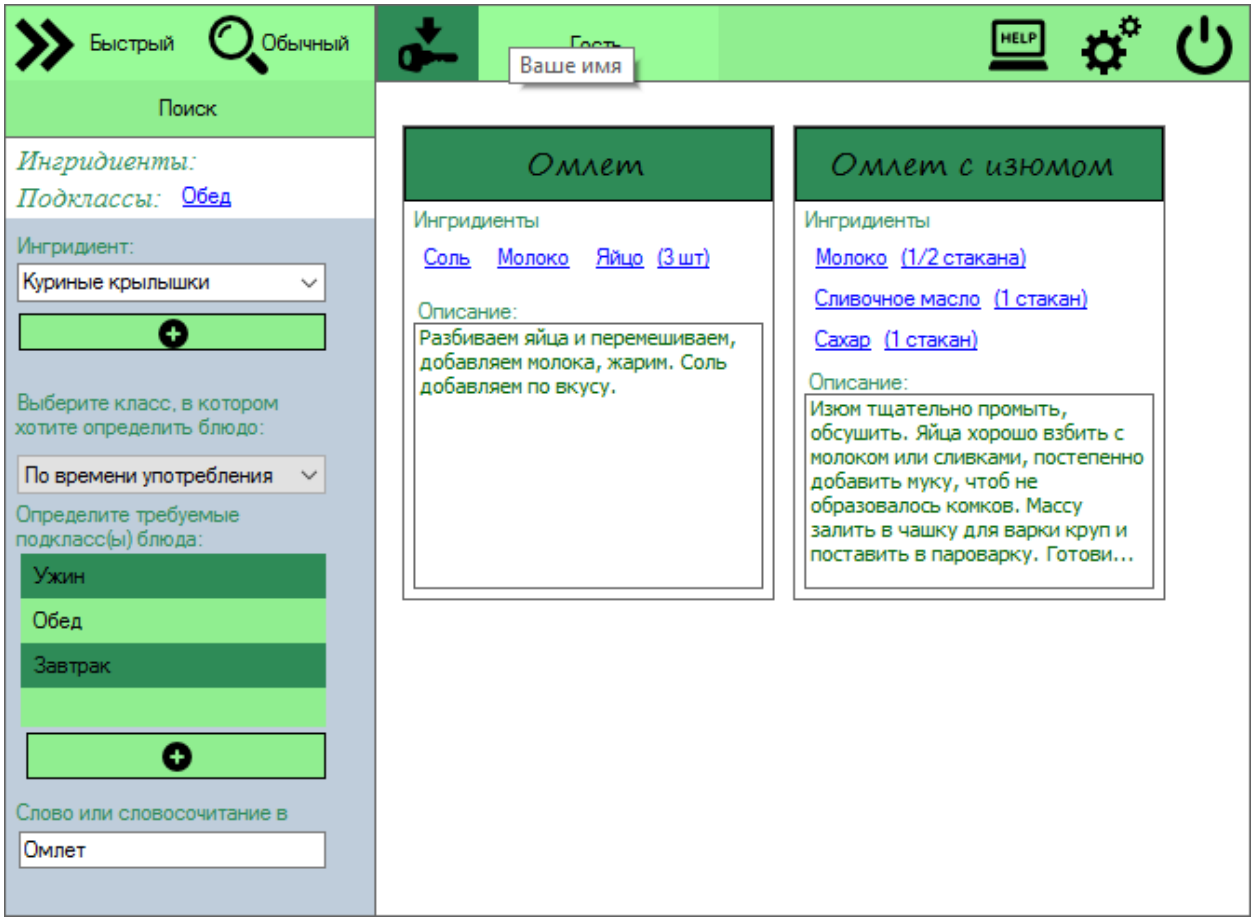

Рисунок 29

Если вам нужно найти рецепты с упоминаниями некоторого слова или словосочетания, то введите его в текстовом элементе внизу формы. При желании отсеять блюда по категориям – воспользуйтесь панелью добавления категорий: выберите интересующую вас классификацию блюд, щелчком мыши выделите нужные категории после чего нажмите на кнопку добавления категорий. Когда нужно учитывать и наличие определенных продуктов, тогда следует добавить продукты для поиска с помощью элемента добавления продукта. Введите название продукта. Если он есть в базе, то нужный вариант можно будет выбрать и с помощью авто заполнения. Если продукта в базе нет, то при наличии специальных прав, вы можете добавить продукт в базу (рисунок 30), но в этом нет смысла, так как если продукта в базе нет, то и рецептов с ним тоже.

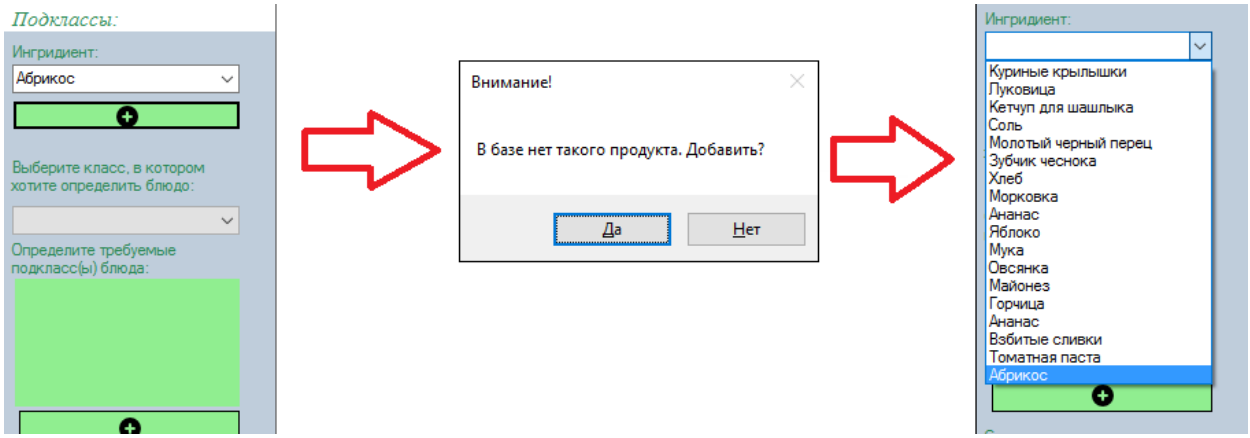

Рисунок 30 – Схема добавления продукта в базу

Просмотр списка рецептов

На данную вкладку главного окна можно попасть из закладок или при поиске. Рецепт в минимальном представлении содержит название, ингредиенты и часть описания, которая поместилась в элементе. При нажатии на название блюда, данное открывается в окне просмотра блюд.

Работа с закладками

Работа с закладками очень проста и понятна. Для этого существует всего три кнопки: добавления, удаления и просмотра. При нажатии на первую блюдо добавляется в список закладок, при нажатии на удаление – удаляется. В один момент времени может быть видна лишь одна из этих кнопок (рисунок 31) Для того, чтоб просмотреть все свои закладки нужно нажать на кнопку закладок, расположенную на главной панели управления (рисунок 32).

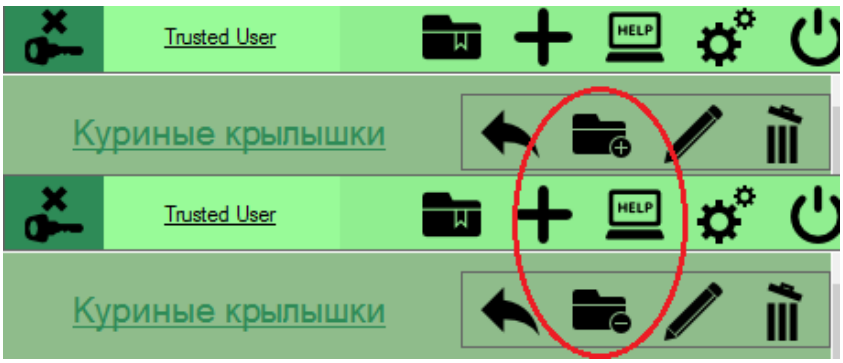

Рисунок 31

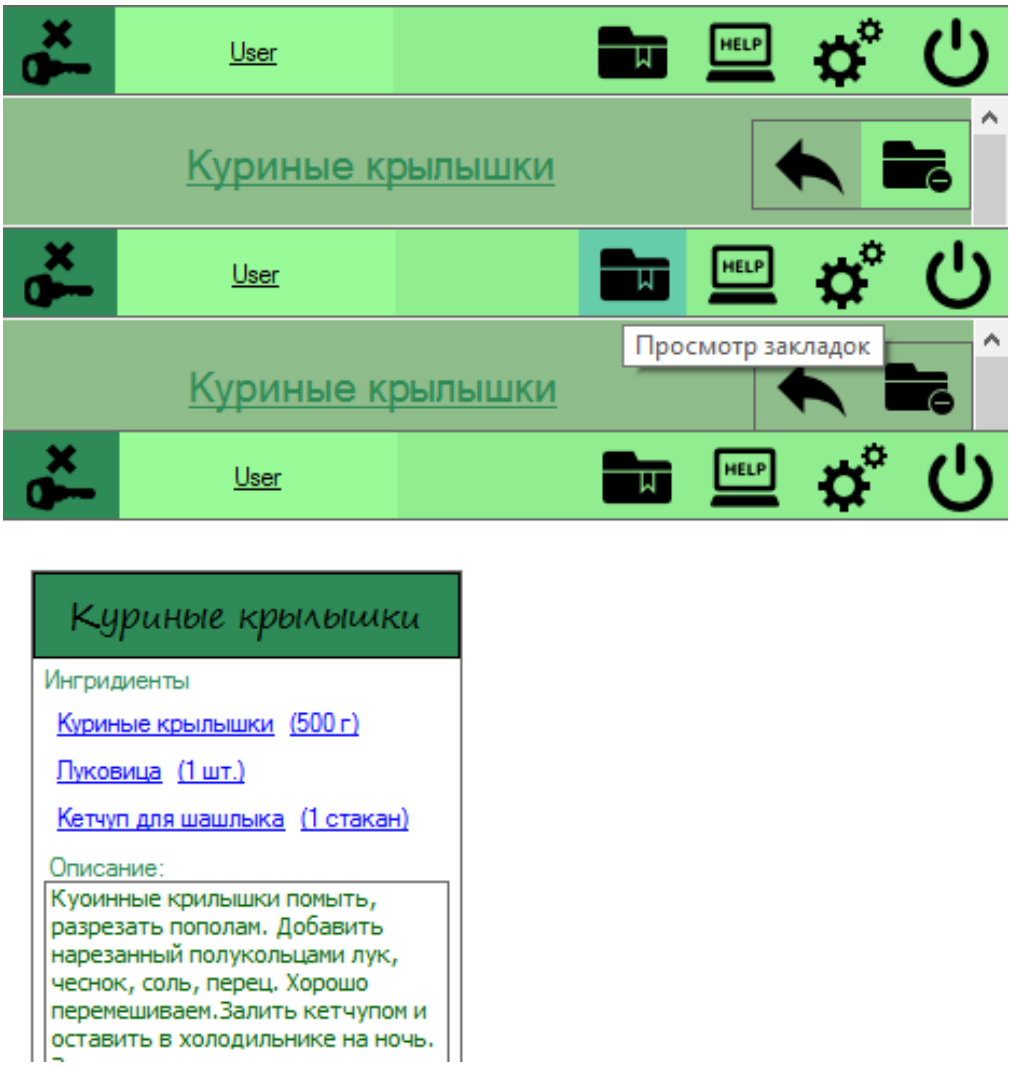

Рисунок 32 – Процесс вызова отображения закладок

Просмотр рецепта

В данном окне видна вся информация о рецепте, а на панели управления расположены следующие кнопки:

- «Назад». Возвращает в предыдущее окно.
- «Добавить в закладки» / «Удалить закладку». В конкретный момент времени отображается или одни из кнопок, или ни одной.
- «Редактировать». Открывает окно редактирования данного рецепта.
- «Удалить». Удаляет рецепт.

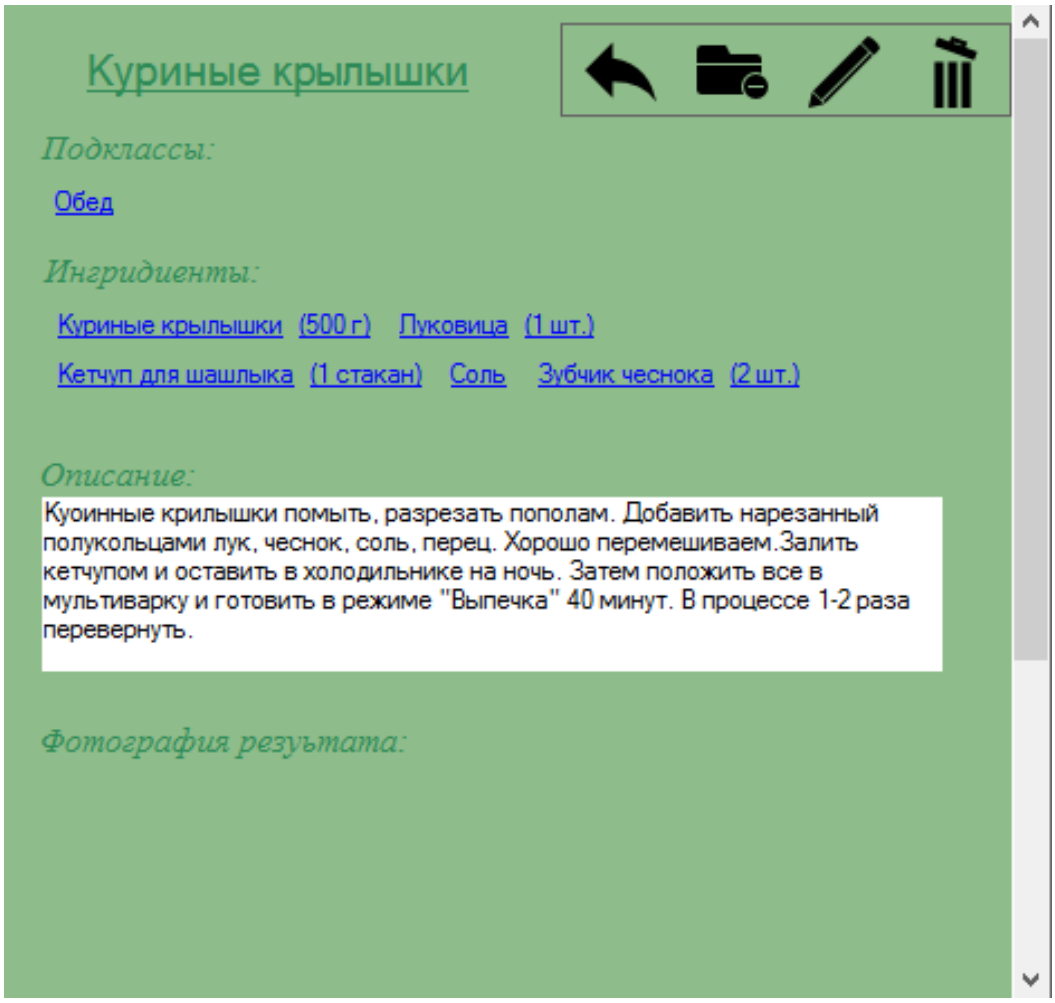

Рисунок 33-Вкладка просмотра блюда

Редактирование и создание рецепта

Очень схожее с окном просмотра блюда, только содержит дополнительные кнопки и редактируемые элементы (риск нок 33). Процесс добавления категорий аналогичен тому, что в детальном поиске, а новый процесс добавления ингредиента отличается от добавления продукта тем, что ингредиент состоит из продукта и значения (словесное описание количества, например, «2 яйца», «3 ложки сахара», «щепотка приправы» и т.д.). Так же есть кнопка выбора фотографии блюда, которая вызывает файловый диалог для выбора фото формата .png, но вы можете выбрать фото с другим расширением, выбрав тип документа на «Allfiles» (рисунок 35).

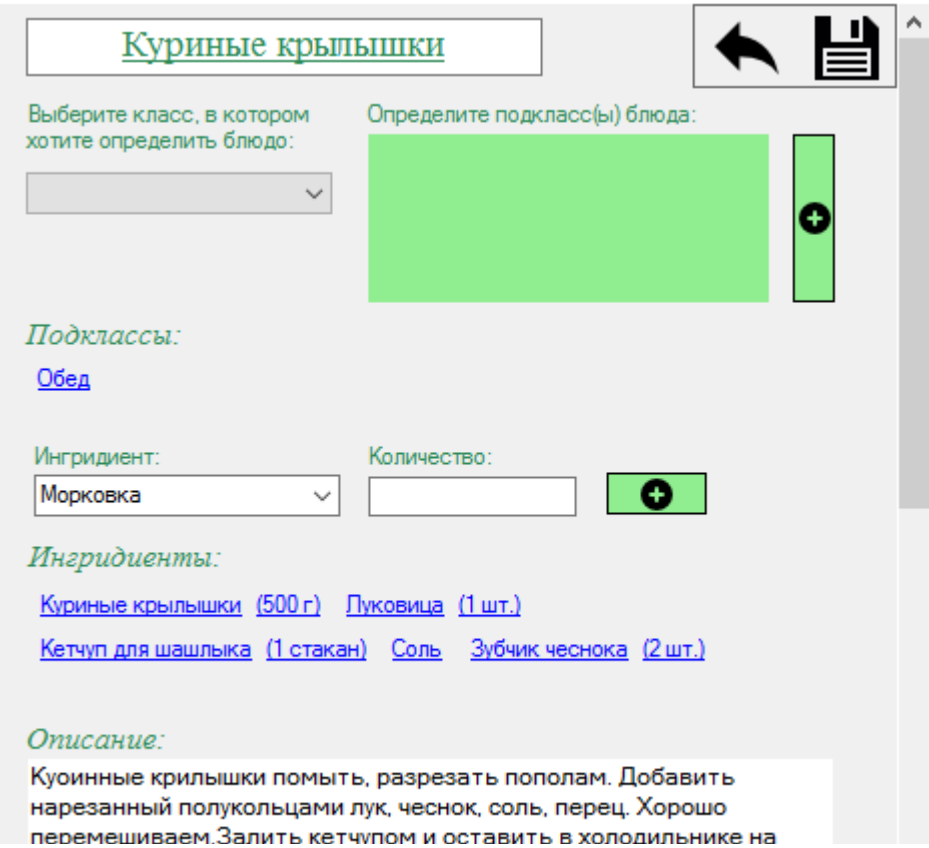

Рисунок 35 – Вид окна редактирования блюда

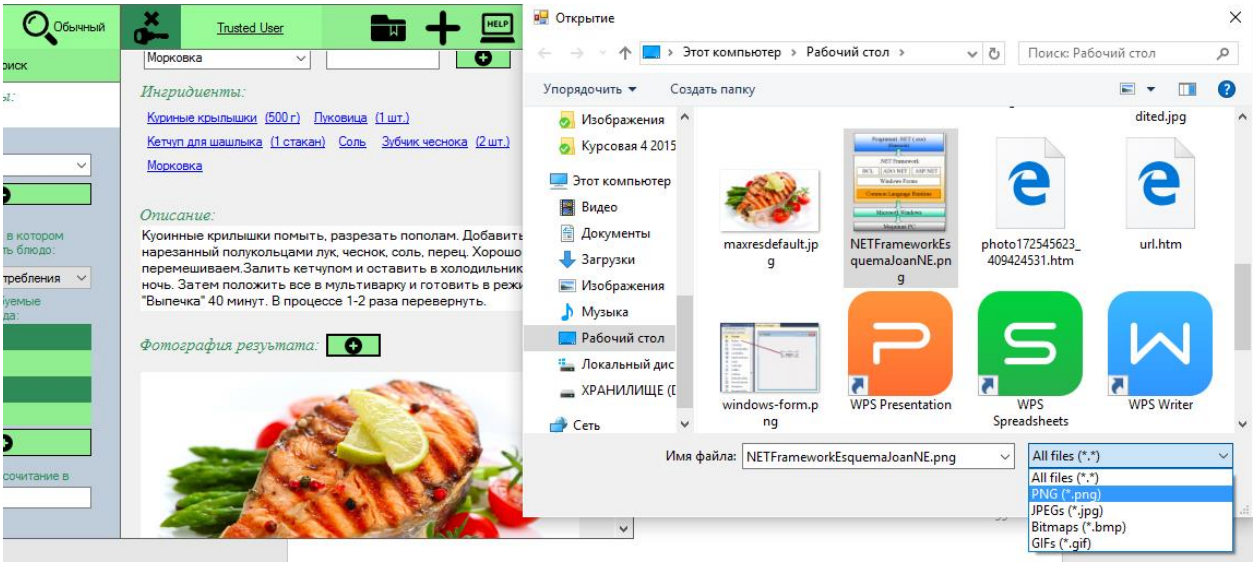

Рисунок 34 – Процесс изменения изображения блюда

В этом окне видны только две кнопки: возврата и сохранения. Кнопка возврата возвращает к просмотру редактируемого блюда или на страницу помощи, если блюдо не редактировалось, а создавалось. Кнопка «Сохранить» действует в зависимости от того, как вы попали в окно редактирования. Если

вы попали в это окно при создавали блюда, то нажатие на вышеупомянутую кнопку создаст новое блюдо, иначе изменит тот рецепт, с которого вы перешли в редактирование.

Настройки

Если вас не устраивает расширение главного окна приложения, то вы можете нажать на кнопку настроек, появится модальное окно настроек. После выбора разрешения нажмите «Ок», после чего размер приложения изменится.

6.2.2 Инструкция по установке

1. Запустите файл CulinaryGuideInstaller.msi от имени администратора, нажмите «Далее» в появившемся окне диалога (рисунок 36).

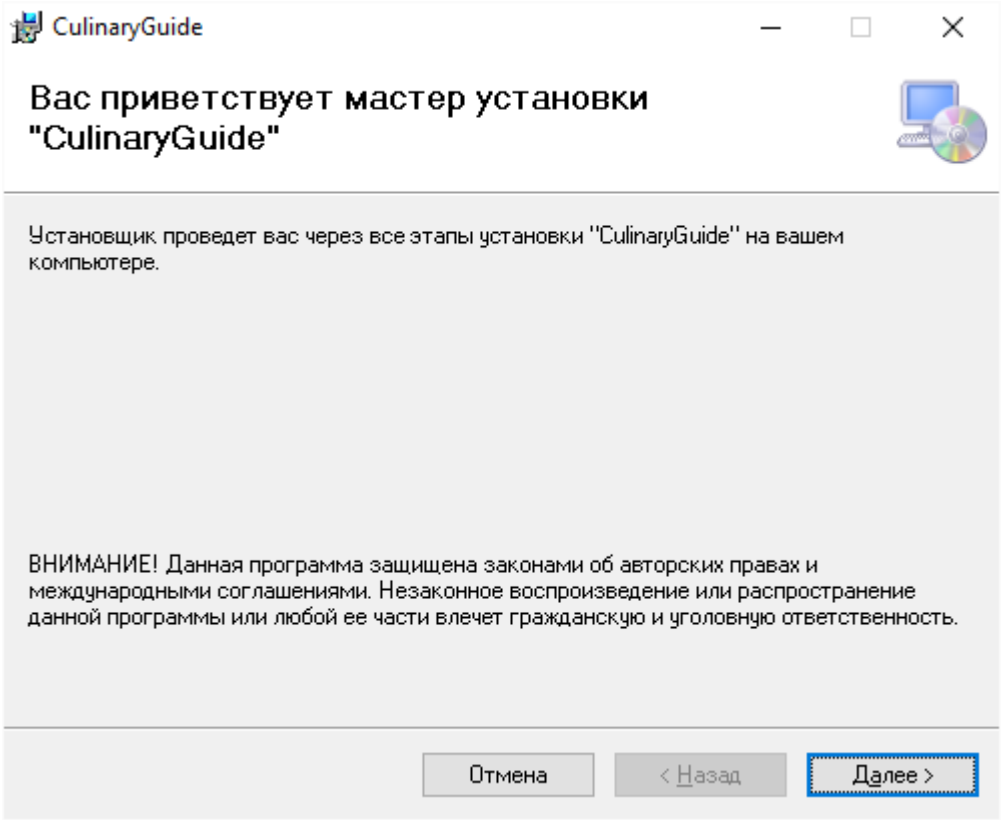

Рисунок 36

1. Выберите папку для установки и пользователя, для которого установите программу (рисунок 37) и нажмите «далее».

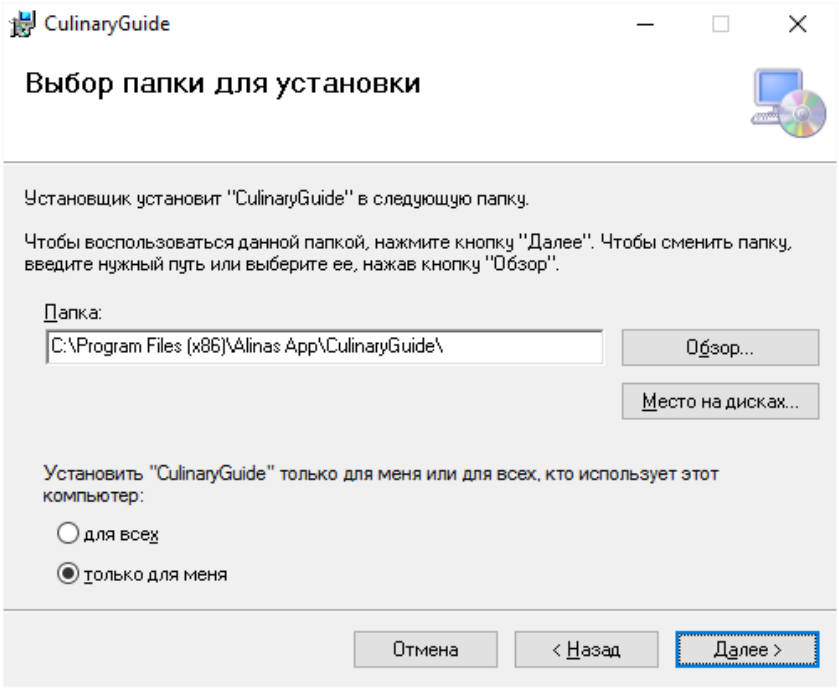

Рисунок 37

2. Нажмите «далее» в подтверждении установки.
3. Дождитесь установки программы и подтвердите завершение установки (рисунок 38).

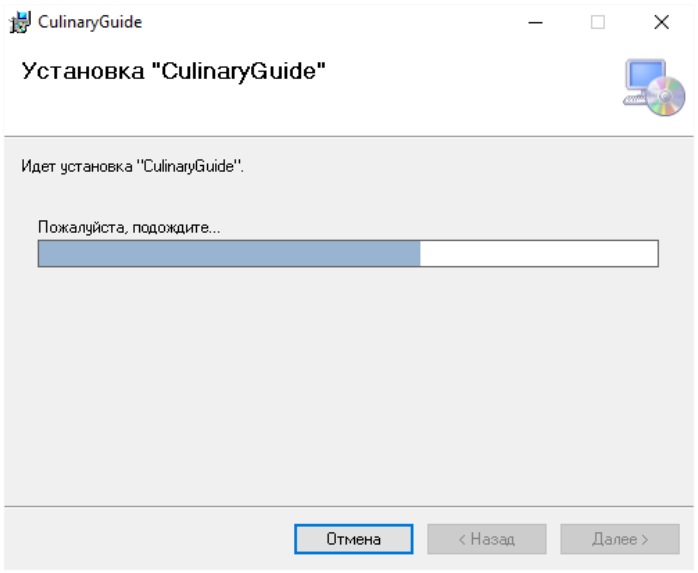

Рисунок 38

6.2.3 Системные и программные средства

Рекомендуемые минимальные системные требования:

- Процессор – Pentium 4, Celeron и выше
- Оперативнаяпамять – 1 GB и выше
- Видеокарта с памятью 256 M и выше
- Свободно место на диске минимум 500 kb

Необходимые программные требования:

- Операционная система WindowsXP/ Windows7/8/10
- .NET framework
- SqlLocalBD (Для текущей, тестовой версии)

ЗАКЛЮЧЕНИЕ

В результате работы над курсовой работой выполнены все поставленные задачи. Спроектированные и разработанные база данных и приложение «Кулинарный справочник». Был изучен ряд технологий для создания проекта: .NET, WindowsForms, C#, VisualStudio 2015, MS SQL. Также было написано обоснование выбора данных технологий в курсовом проекте. Функционал приложения был утвержден согласно разработанной структуре программы и базы данных. На основе разработанной функциональной схемы проекта составлен программный продукт, обладающий интерфейсной частью и базой данных для хранения данных о пользователях, блюдах, категорий блюд, закладках и продуктах.

На данный момент при работе с приложением доступны следующие возможности: регистрация, авторизация пользователей, просмотр блюд, возможность изменения существующих рецептов и создания новых так же есть возможность сохранения блюд в закладки, а также два вида поиска блюд.

Дальнейшее развитие программы связано с расширением ее возможностей, улучшением уровня безопасности, подключением к удаленной БД. Тестирование программы показало ее работоспособность.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Миллион рецептов праздничных блюд. Готовим, украшаем, сервируем. / Скаляр С. С. [Клуб семейного досуга], 2012.г - 386 стр.

2. Информационный Интернет-ресурс: Приятного аппетита URL:http://priyatnogo-appetita.com (дата обращения: 27.04.2016).

3. Wikipedia – информационный ресурс:
URL: https://ru.wikipedia.org/wiki/Windows_Forms
(дата обращения: 25.04.2016).
URL: https://ru.wikipedia.org/wiki/.NET_Framework
(дата обращения: 02.05.2016).
URL: https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio
(дата обращения: 02.05.2016).
URL: https://ru.wikipedia.org/wiki/Установщик _Windows
(дата обращения: 04.05.2016).

4. ProfessionalC# 5.0 and .NET 4.5 / Автор: НейгелК.,ИвьенБ. / Издательство - М.: Вильямс, Год: 2014.

5. DependencyInjectionin .NET Букинистическое издание / Автор: Марк Симан, А. Барышнев, Евгений Зазноба, издательство: Питер.

6. Draw.io // [web-приложение для создания графических схем] URL: https://www.draw.io/ (дата обращения: 05.05.2016).

7. DBDesigner.net // [web-приложение для создания схемы базы данных] URL: http://dbdesigner.net/ (дата обращения: 12.04.2016).

8. Professorweb.ru// [Интернет-ресурс с материалами для работы с ADO.NET].
URL: http://professorweb.ru/my/ADO_NET/base/level1/ado_net_index.php
(дата посещения: 22.04.2016).

ПРИЛОЖЕНИЕ А

(обязательное)

Фрагменты листинга

Листинг А.1 – Класс MainForm

```
using System;
usingSystem.Collections.Generic;
usingSystem.ComponentModel;
usingSystem.Data;
usingSystem.Drawing;
usingSystem.Drawing.Imaging;
using System.IO;
usingSystem.Linq;
usingSystem.Text;
usingSystem.Threading.Tasks;
usingSystem.Windows.Forms;

namespaceCulinaryGuide
{
public partial class MainForm : Form
    {
SettingsClass settings = new SettingsClass();
publicDBWorkerdb = new DBWorker();

        Point PanelMouseDownLocation;
publicUserClass user =null;
publicDishClassdish=null;
        // список с данными о всех блюдах для поиска
        // в переменные не записываются параметры,
        // корорые не участвуют в поиске(картики)
List<DishClass>minDishList;
        List<DishCategoryClass> classes;
        List<SimpleTableClass> products;


publicMainForm()
        {
InitializeComponent();
OpenHelpPage();
if (!db.Connect())
{
MessageBox.Show("Нет соединения с базой данных!");
Close();
            }
            // применяем настройки
ReloadSettings();
            // обновляем локальные переменные
RefreshLocalData();
RefreshMainWindowUIElements();
```

```
OpenFastSearch();
        }

private void ReloadSettings()
{
            // считываем с файла настройки
settings.Reload();

            // применяем настройки
Width = settings.windowWidth;
Height = settings.windowHeight;

        }

public void RefreshMainWindowUIElements()
        {
if (user == null)
            {
userNameLbl.Text = "Гость";
logoutBtn.Visible = false;
loginBtn.Visible = true;
bookmarkBtn.Visible = false;
addDishBtn.Visible = false;
            }
else
            {
userNameLbl.Text = user.name;
logoutBtn.Visible = true;
loginBtn.Visible = false;
bookmarkBtn.Visible = true;
if (user.trusted) addDishBtn.Visible = true;
elseaddDishBtn.Visible = false;
            }

        }

public void OpenDish()
        {
RefreshLocalData();

if (user == null)
            {
addBookmarkBtn.Visible = false;
removeBookmarkBtn.Visible = false;
editDishBtn.Visible = false;
removeDishBtn.Visible = false;
            }
else
            {
if (db.GetAllDishIdFromUserBookmarks(user.id).IndexOf(dish.id)
== -1)
                {
addBookmarkBtn.Visible = true;
```

```
removeBookmarkBtn.Visible = false;
                }
else
                {
addBookmarkBtn.Visible = false;
removeBookmarkBtn.Visible = true;
                }
if (user.trusted)
                {
editDishBtn.Visible = true;
removeDishBtn.Visible = true;
                }
else
                {
editDishBtn.Visible = false;
removeDishBtn.Visible = false;
                }

            }

            // очищаем елементы от данных, что были до этого
showNameLbl.Text = dish.name;
showIngFLP.Controls.Clear();
foreach (IngredientClassic in dish.ingredients)
            {
SmartLabelUCiuc = new SmartLabelUC(ic.product.name, false,
ic.product.id, ic.value);
showIngFLP.Controls.Add(iuc);
            }
showSubclassFLP.Controls.Clear();
foreach (SubcategoryClassic in dish.classes)
            {
SmartLabelUCiuc = new SmartLabelUC(ic.name, false);
showSubclassFLP.Controls.Add(iuc);
            }
if (dish.image != null)
            {
MemoryStreamms = new MemoryStream(dish.image);
showDishImgPB.Image = Image.FromStream(ms);
ms.Close();
            }
else
            {
showDishImgPB.Image = showDishImgPB.InitialImage;
            }
showDiscriptionUC.setText(dish.description);


            //  открываемпанельблюда
mainTC.SelectedTab = dishTP;

        }
```

```
private void panel2_MouseDown(object sender, MouseEventArgs e)
        {
if (e.Button == MouseButtons.Left)
PanelMouseDownLocation = e.Location;

        }
private void panel2_MouseMove(object sender, MouseEventArgs e)
        {
if (e.Button == MouseButtons.Left)
            {
this.Left += e.X + - PanelMouseDownLocation.X;
this.Top += e.Y - PanelMouseDownLocation.Y;
            }

        }
private void exitBtn_Click(object sender, EventArgs e)
        {
Close();
        }

private void label3_Click(object sender, EventArgs e)
        {

        }

private void comboBox1_SelectedIndexChanged(object sender,
EventArgs e)
        {

        }

private void editAddIngBtn_Click(object sender, EventArgs e)
        {
SimpleTableClass prod = new SimpleTableClass();
            prod.name = editProductCB.Text;
            prod.id = -1;

foreach (SimpleTableClass item in products)
            {
if (item.name == prod.name) prod.id = item.id;
            }


if (prod.id != -1)
            {
string value = editValueTB.Text;
editIngFLP.Controls.Add(new SmartLabelUC(prod.name, true,
prod.id, value));
            }
else
            {
if (user != null &&user.trusted)
                {
```

```
DialogResultdialogResult =
MessageBox.Show("Вбазенеттакогопродукта. Добавить?",
"Внимание!", MessageBoxButtons.YesNo);
if (dialogResult == DialogResult.Yes)
                    {
db.AddProduct(prod.name);
RefreshLocalData();
editProductCB.Items.Add(prod.name);
}

                }
else
                {
MessageBox.Show("В базе нет такого продукта.", "Внимание!");
}
            }

        }

private void editAddPhotoBtn_Click_1(object sender, EventArgs e)
        {
OpenFileDialog f = new OpenFileDialog();
f.Filter = "All
files|*.*|PNG|*.png|JPEGs|*.jpg|Bitmaps|*.bmp|GIFs|*.gif";
f.FilterIndex = 1;
if (f.ShowDialog() == DialogResult.OK)
            {
editResultPhotoPB.Image = Image.FromFile(f.FileName);
            }
        }

private void fastSearchBtn_Click(object sender, EventArgs e)
        {
OpenFastSearch();
        }

private void OpenFastSearch()
        {
RefreshLocalData();

fastSearchTP.Controls.Clear();
foreach (DishCategoryClassdishClass in classes)
            {
DishClassLBUCDishClassUC = new DishClassLBUC(dishClass,true);
DishClassUC.Dock = DockStyle.Top;
fastSearchTP.Controls.Add(DishClassUC);
            }

searchTC.SelectedTab = fastSearchTP;
        }

private void defaultSearchBtn_Click(object sender, EventArgs e)
        {
```

```
OpenDefaultSerach();
        }
private void OpenDefaultSerach()
        {
RefreshLocalData();

productDSCB.Items.Clear();
productDSCB.AutoCompleteMode = AutoCompleteMode.Append;
productDSCB.AutoCompleteSource = AutoCompleteSource.ListItems;
foreach (SimpleTableClass item in products)
            {
productDSCB.Items.Add(item.name);
            }

classDSCB.Items.Clear();
foreach (DishCategoryClass item in classes)
            {
classDSCB.Items.Add(item.name);
            }

searchTC.SelectedTab = defaultSearchTP;
        }


        // обновляемюзераиспискипродуктовивеличн
private void RefreshLocalData()
        {
           // загружаемсбазывсеееееее
if (user != null)
            {
user = db.GetUserByLoginPassword(user.login, user.password);
            }
if (dish != null)
            {
dish = db.GetDishById(dish.id);
            }
classes = db.GetAllDishClasses();
products = db.GetAllProducts();
minDishList = db.GetAllMinDishs();
        }

private void FillPanel(List<DishClass> list)
        {
ItemsFLP.Controls.Clear();
foreach (DishClass dc in list)
            {
ItemsFLP.Controls.Add(new MinDishForm(this,dc));
            }
        }


private void addIngToDSBtn_Click(object sender, EventArgs e)
        {
```

```
SimpleTableClass prod = new SimpleTableClass();
            prod.name = productDSCB.Text;
            prod.id = -1;

foreach (SimpleTableClass item in products)
            {
if (item.name == prod.name) prod.id = item.id;
            }
if (prod.id!=-1){
ingredientsDSFLP.Controls.Add(new SmartLabelUC(prod.name,
true,prod.id));}
else{
if (user != null &&user.trusted )
                {
DialogResultdialogResult =
MessageBox.Show("Вбазенеттакогопродукта. Добавить?",
"Внимание!", MessageBoxButtons.YesNo);
if (dialogResult == DialogResult.Yes)
                    {
db.AddProduct(prod.name);
}
 }
else{
MessageBox.Show("В базе нет такого продукта.", "Внимание!");
}}}
private void backToAllDishes_Click(object sender, EventArgs e)
        {
OpenAllDishes();
        }
private void OpenAllDishes()
        {
mainTC.SelectedTab = dishesTP;
        }
private void addBookmarkBtn_Click(object sender, EventArgs e)
        {
if (db.AddBookmark(user.id, dish.id))
            {
addBookmarkBtn.Visible = false;
removeBookmarkBtn.Visible = true;
        }}
private void removeBookmarkBtn_Click(object sender, EventArgs e)
        {
if (db.DeleteBookmark(user.id, dish.id))
            {
addBookmarkBtn.Visible = true;
removeBookmarkBtn.Visible = false;
            }
        }
private void removeDishBtn_Click(object sender, EventArgs e)
        {
db.DeleteAllDishIngredientsByDishId(dish.id);
db.DeleteAllDishSubclassesByDihId(dish.id);
db.DeleteBookmark(user.id, dish.id);
```

```
db.DeleteTableById("Dish", dish.id);
dish = null;
RefreshLocalData();
OpenHelpPage();
        }
private void addDishBtn_Click(object sender, EventArgs e)
        {
dish = null;
EditDish();
        }
private void EditDish()
{RefreshLocalData();
editDishNameTB.Text = "Названиe блюда";
editIngFLP.Controls.Clear();

editSelectClassCB.Items.Clear();
foreach (DishCategoryClass item in classes)
            {
editSelectClassCB.Items.Add(item.name);
            }
editProductCB.Items.Clear();
foreach (SimpleTableClass item in products)
            {
editProductCB.Items.Add(item.name);
            }
editAddSubclassLBUC.SetList(null);
editSubclassFLP.Controls.Clear();
editIngFLP.Controls.Clear();
editResultPhotoPB.Image = editResultPhotoPB.InitialImage;
editValueTB.Text = "";
editDescriptionUC.setText("");
if (dish != null)
{editDishNameTB.Text =dish.name;
foreach (IngredientClassic in dish.ingredients)
                {
SmartLabelUCiuc = new
SmartLabelUC(ic.product.name,true,ic.product.id,ic.value);
editIngFLP.Controls.Add(iuc);
                }
foreach (SubcategoryClassic in dish.classes)
                {
SmartLabelUCiuc = new SmartLabelUC(ic.name,true,ic.id);
editSubclassFLP.Controls.Add(iuc);
                }

if (dish.image != null){
MemoryStreamms = new MemoryStream(dish.image);
editResultPhotoPB.Image = Image.FromStream(ms);
ms.Close();}
editDescriptionUC.setText(dish.description);}
mainTC.SelectedTab = editDishTP;}
```

ЛистингА.2 – ЧастькодаклассаDBWorker

```
public class DBWorker
{
SqlConnectionStringBuilderconnectStringBuilder;
publicSqlConnection connection;
publicDBWorker()
{
connectStringBuilder = new SqlConnectionStringBuilder();
connectStringBuilder.DataSource = @"(localdb)\MSSQLLocalDB";
connectStringBuilder.ConnectTimeout = 30;
connectStringBuilder.IntegratedSecurity = true;
connectStringBuilder.AttachDBFilename =
@"|DataDirectory|CulinaryGuide.mdf";

connection = new SqlConnection();
connection.ConnectionString =
connectStringBuilder.ConnectionString;
}
publicbool Connect()
{
try
    {
        //Открытьподключение
connection.Open();
return true;
    }
catch (SqlException ex)
{
        // Протоколировать исключение
Console.WriteLine(ex.Message);
return false;
    }
}
public void Close()
{
connection.Close();
}
publicboolIsOpen()
{
returnconnection.State == System.Data.ConnectionState.Open;
}

publicboolUserExist(string login)
{
bool result = false;
string command = string.Format("SELECT * FROM [User] WHERE Login
='{0}'", login);

try
    {
SqlCommandcmd = new SqlCommand(command, connection);
SqlDataReaderdr = cmd.ExecuteReader();
```

```
if (dr.Read())
        {
result = true;
        }
dr.Close();
    }
catch (SqlException ex)
{
        // Протоколировать исключение
Console.WriteLine(ex.Message);
}


return result;
}

publicboolRefreshConnect()
{
Close();
return Connect();
}
publicUserClassGetUserByLoginPassword(string login,string
password)
{
UserClass result = new UserClass();

string command = string.Format("SELECT * FROM [User] WHERE Login
='{0}' and Password = '{1}'", login,password);

boolallFine = false;
try
    {
SqlCommandcmd = new SqlCommand(command, connection);
SqlDataReaderdr = cmd.ExecuteReader();
if (dr.Read())
        {
            result.id = (int)dr[dr.GetOrdinal("Id")];
result.login = (string)dr[dr.GetOrdinal("Login")];
result.password = (string)dr[dr.GetOrdinal("Password")];
result.trusted = (((int)dr[dr.GetOrdinal("Trusted")]==1)? true :
false);
            result.name = (string)dr[dr.GetOrdinal("Name")];
if (dr.IsDBNull(dr.GetOrdinal("Image"))) result.image = null;
elseresult.image = (byte[])dr[dr.GetOrdinal("Image")];

allFine = true;
        }
dr.Close();
    }
catch (SqlException ex)
    {
MessageBox.Show(ex.Data.ToString(),"Ошибкаподключениякбазе!");
    }
```

```
if (allFine)
return result;
else
return null;
}
publicDishClassGetDishById(int id)
{
DishClass result = new DishClass();

string command = string.Format("SELECT * FROM [Dish] WHERE Id
='{0}'", id);
boolallFine = false;
try
    {
SqlCommandcmd = new SqlCommand(command, connection);
SqlDataReaderdr = cmd.ExecuteReader();
if (dr.Read())
        {
            result.id = (int)dr[dr.GetOrdinal("Id")];
result.description = (string)dr[dr.GetOrdinal("Description")];
result.user_id = (int)dr[dr.GetOrdinal("User_id")];
            result.name = (string)dr[dr.GetOrdinal("Name")];
if (dr.IsDBNull(dr.GetOrdinal("Image"))) result.image = null;
elseresult.image = (byte[])dr[dr.GetOrdinal("Image")];

allFine = true;
        }
dr.Close();
    }
catch (SqlException )
{
MessageBox.Show("Ошибка подключения к базе!");
}

result.ingredients = GetIngredientsByDishId(result.id);
result.classes = GetDishCatigoriesByDishId(result.id);



if (allFine)
return result;
else
return null;
}
private List<SubcategoryClass>GetDishCatigoriesByDishId(int id)
{
    List<SubcategoryClass> b = new List<SubcategoryClass>();
    List<int>subCategoriesId = new List<int>();
string command = string.Format("SELECT * FROM [Linker] where
Dish_id = '{0}'",id);
boolallFine = false;
```

```
try
    {

SqlCommandcmd = new SqlCommand(command, connection);
SqlDataReaderdr = cmd.ExecuteReader();
while (dr.Read())
        {
subCategoriesId.Add((int)dr[dr.GetOrdinal("Subclass_id")]);
        }
allFine = true;
dr.Close();
    }
catch (SqlException )
{
MessageBox.Show("Ошибка подключения к базе!");
}

foreach(int a in subCategoriesId)
    {
try
        {
string command1 = string.Format("SELECT * FROM [Subclass] where
Id = '{0}'", a);

SqlCommandcmd = new SqlCommand(command1, connection);
SqlDataReaderdr = cmd.ExecuteReader();
while (dr.Read())
            {
SubcategoryClassbuf = new SubcategoryClass();

                buf.id = (int)dr[dr.GetOrdinal("Id")];
                buf.name = (string)dr[dr.GetOrdinal("Name")];
buf.parent_id = (int)dr[dr.GetOrdinal("Parent_id")];

b.Add(buf);
            }
allFine = true;
dr.Close();
        }
catch (SqlException )
        {
MessageBox.Show("Ошибкаподключениякбазе!");
        }
    }
if (allFine)
return b;
else
return null;
}
```

```
// возвращает все блюда только для заполнения минимизированной
формы
// ипоиска (экономияпамяти)
public List<DishClass>GetAllMinDishs()
{
    List<DishClass> result = new List<DishClass>();

string command = string.Format("SELECT * FROM [Dish]");

boolallFine = false;
try
    {

SqlCommandcmd = new SqlCommand(command, connection);
SqlDataReaderdr = cmd.ExecuteReader();
while (dr.Read())
        {
DishClassbuf = new DishClass();

            buf.id = (int)dr[dr.GetOrdinal("Id")];
buf.description = (string)dr[dr.GetOrdinal("Description")];
            buf.name = (string)dr[dr.GetOrdinal("Name")];


result.Add(buf);
        }
allFine = true;
dr.Close();
    }
catch (SqlException )
    {
MessageBox.Show("Ошибкаподключениякбазе!");
    }

foreach (DishClass item in result)
    {
item.ingredients = GetIngredientsByDishId(item.id);
item.classes = GetDishCatigoriesByDishId(item.id);
    }

if (allFine)
return result;
else
return null;
}

public List<IngredientClass>GetIngredientsByDishId(int id)
{
    List<IngredientClass> result = new List<IngredientClass>();

string command = string.Format("SELECT * FROM [Ingredient] where
Dish_id = '{0}'",id);
boolallFine = false;
```

```
try
    {

SqlCommandcmd = new SqlCommand(command, connection);
SqlDataReaderdr = cmd.ExecuteReader();
while (dr.Read())
        {
IngredientClassbuf = new IngredientClass();

            buf.id = (int)dr[dr.GetOrdinal("Id")];
buf.value = (string)dr[dr.GetOrdinal("Value")];

            buf.product.id =
(int)dr[dr.GetOrdinal("Product_id")];


result.Add(buf);
        }
allFine = true;
dr.Close();
    }
catch (SqlException ex)
    {
MessageBox.Show("Ошибкаподключениякбазе!"+ex.Data);
    }

foreach (IngredientClass item in result)
    {
        item.product.name = GetSimpleClassById("Product",
item.product.id);
    }

if (allFine)
return result;
else
return null;
}

// методы, работающие для таблиц Product и Meshure одновременно
public List<SimpleTableClass>GetAllProducts()
{
    List<SimpleTableClass> result = new
List<SimpleTableClass>();

string command = string.Format("SELECT * FROM [Product]");
try
    {
SqlCommandcmd = new SqlCommand(command, connection);
SqlDataReaderdr = cmd.ExecuteReader();
while (dr.Read())
        {
SimpleTableClassbuf = new SimpleTableClass();
```

```
            buf.id = (int)dr[dr.GetOrdinal("Id")];
            buf.name = (string)dr[dr.GetOrdinal("Name")];
result.Add(buf);
        }
dr.Close();
    }catch (SqlException )
    {MessageBox.Show("Ошибкаподключениякбазе!");}
return result;
}
publicintGetProductIdByName(string name)
{
int result = -1;
string command = string.Format("SELECT [Id] FROM [Product] where
name = '{0}'", name);
try
    {
SqlCommandcmd = new SqlCommand(command, connection);
SqlDataReaderdr = cmd.ExecuteReader();
while (dr.Read())
        {
result = (int)dr[dr.GetOrdinal("Id")];
        }
dr.Close();
    }
catch (SqlException )
    {
MessageBox.Show("Ошибкаподключениякбазе!");
    }
return result;
}
public string GetSimpleClassById(string tableName, int id)
{
string result="";
string command = string.Format("SELECT [Name] FROM [{0}] WHERE
Id = '{1}'",tableName,id);
boolallFine = false;
try
    {
          SqlCommandcmd = new SqlCommand(command, connection);
          SqlDataReaderdr = cmd.ExecuteReader();
          while (dr.Read())
          {
               result = (string)dr[dr.GetOrdinal("Name")];
               allFine = true;
          }
          dr.Close();
    }
catch (SqlException )
{MessageBox.Show("Ошибка подключения к базе!");}
if (allFine) return result;
else return null;}
```

ЛистингА.3 – КодклассаSearchData

```
classSearchData
    {
public List<SubcategoryClass>reqSubclasses= new
List<SubcategoryClass>();
public List<SimpleTableClass>reqProducts = new
List<SimpleTableClass>();
public string reqWord = "";
publicSearchData()
        {
        }
publicboolDishIsValid(DishClass item)
        {
if (reqSubclasses!=null)
foreach (SubcategoryClass item1 in reqSubclasses)
                {
boolmachesIsFind = false;
foreach (SubcategoryClass item2 in item.classes)
                    {
if (item1.id == item2.id) machesIsFind = true;
                    }
if (!machesIsFind) return false;
                }
if (reqProducts != null)
foreach (SimpleTableClass item1 in reqProducts)
                {
boolmachesIsFind = false;
foreach (IngredientClass item2 in item.ingredients)
                    {
if (item1.id == item2.product.id) machesIsFind = true;
                    }
if (!machesIsFind) return false;
                }

if (reqWord != "")
            {
                // если поискового слова нет ни в описании, ни в
названии
                // то говорим, что блюдо не прошло проверку
if (item.name.IndexOf(reqWord) == -1
&&item.description.IndexOf(reqWord) == -1)
returnfalse;
            }

            // если до этого не выкинуло из метода, тогда блюдо
            // подходит по критериям поиска
returntrue;
}
    }
```