Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ЛЕСОСИБИРСКИЙ ПЕДАГОГИЧЕСКИЙ ИНСТИТУТ –
филиал Сибирского федерального университета

Высшей математики, информатики, экономики и естествознания
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
Л.Н. Храмова
подпись инициалы, фамилия
«11» июня 2025 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.02 Информационные системы и технологии
код-наименование направления

ПРОЕКТИРОВАНИЕ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ УПРАВЛЕНИЯ ПРОИЗВОДСТВЕННЫМ ПРОЦЕССОМ В СФЕРЕ БЫСТРОГО ПИТАНИЯ

Руководитель 11.06.2025 доцент, канд. пед. наук Е. В. Киргизова
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник 11.06.2025 П. А. Куприянов
подпись, дата инициалы, фамилия

Нормоконтролер 11.06.2025 Е. В. Киргизова
подпись, дата инициалы, фамилия

Лесосибирск 2025

РЕФЕРАТ

Выпускная квалификационная работа по теме «Проектирование автоматизированной системы управления производственным процессом в сфере быстрого питания» содержит 70 страница текстового документа, 35 иллюстраций, 3 таблицы, 40 использованных источников.

C#, Visual Studio Community, Windows Forms, SQLite.

Цель исследования – теоретически обосновать и разработать автоматизированную систему управления производственным процессом в сфере быстрого питания.

Объект исследования – методы и технологические решения проектировании автоматизированной системы.

Предмет исследования – процесс разработки автоматизированной системы управления производственным процессом в сфере быстрого питания.

Для достижения поставленной цели необходимо решить следующие задачи:

– на основе анализа учебной и научно-технической литературы по теме исследования определить понятийно-категориальный аппарат, инструментальные средства разработки автоматизированной системы управления производственным процессом в сфере быстрого питания, языки программирования и их области;

– рассмотреть методы проектирования автоматизированной системы управления производственным процессом в сфере быстрого питания;

– провести сравнительный анализ автоматизированных систем управления производственным процессом в сфере быстрого питания;

– разработать автоматизированную систему управления производственным процессом в сфере быстрого питания.

В результате выполнения выпускной квалификационной работы разработана автоматизированная система управления производственным процессом в сфере быстрого питания.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современное общество невозможно представить без информационных технологий. Они проникают во все сферы жизни – от быта до сложных производственных процессов. Особенно значима роль информационных систем в области общественного питания, где важна скорость обслуживания, точный учёт ресурсов и стабильность процессов. Своевременное получение достоверной информации о состоянии заказов, запасов и оборудования позволяет принимать эффективные управленческие решения и повышать качество обслуживания клиентов.

Организация производственного процесса в заведениях быстрого питания включает множество этапов: от приёма продукции и контроля качества до приготовления блюд и оформления заказов. Автоматизация данных процессов позволяет снизить влияние человеческого фактора, минимизировать ошибки и повысить общую эффективность работы предприятия.

Актуальность исследования определяется необходимостью разработки программного обеспечения, сочетающего автономность, простоту эксплуатации и отсутствие требований к постоянному сетевому подключению. Современные коммерческие аналоги, обладая избыточной функциональностью и ориентацией на облачные технологии, не удовлетворяют потребностям сегмента малых предприятий, функционирующих в офлайн-режиме.

Настоящая работа направлена на создание компактного, но функционального приложения, способного автоматизировать ключевые процессы на предприятии быстрого питания, повысить прозрачность учёта и упростить взаимодействие персонала с системой.

Цель исследования – теоретически обосновать и разработать автоматизированную систему управления производственным процессом в сфере быстрого питания.

Объект исследования – методы и технологические решения проектировании автоматизированной системы.

Предмет исследования – процесс разработки автоматизированной системы управления производственным процессом в сфере быстрого питания.

Для достижения поставленной цели необходимо решить следующие задачи:

– на основе анализа учебной и научно-технической литературы по теме исследования определить понятийно-категориальный аппарат, инструментальные средства разработки автоматизированной системы управления производственным процессом в сфере быстрого питания, языки программирования и области их;

– рассмотреть методы проектирования автоматизированной системы управления производственным процессом в сфере быстрого питания;

– провести сравнительный анализ автоматизированных систем управления производственным процессом в сфере быстрого питания;

– разработать автоматизированную систему управления производственным процессом в сфере быстрого питания.

Методы исследования:

1. теоретические: анализ научной и учебной литературы по информационным системам, сравнительный анализ существующих автоматизированных систем, моделирование производственного процесса с использованием методологии IDEF0, проектирование архитектуры программной системы и базы данных.

2. эмпирические: наблюдение за работой предприятия во время практики, разработка и тестирование прототипа системы, функциональное и модульное тестирование, отладка и проверка корректности работы приложения.

В процессе разработки автоматизированной системы управления производственным процессом в сфере быстрого питания используются современные средства программирования и проектирования: язык C#, платформа .NET, технология Windows Forms, а также встраиваемая база данных SQLite. Такой выбор технологий обусловлен требованиями к автономности, скорости работы и простоте установки конечного продукта.

Результаты исследования представлены на следующих научных мероприятиях:

1. IV Всероссийская научно-практическая конференция «Современное педагогическое образование: теоретический и прикладной аспекты» в секции «Информатика, информационные технологии и экономика», ЛПИ – филиал СФУ (сертификат).

2. VIII Всероссийская научно-практическая конференция «Актуальные проблемы преподавания дисциплин естественнонаучного цикла», ЛПИ – филиал СФУ (сертификат).

По результатам исследования приняты к публикации статей в сборниках конференций:

1. Куприянов, П. А. Разработка автоматизированной системы управления производственным процессом кофейни / П. А. Куприянов / Актуальные проблемы преподавания дисциплин естественнонаучного цикла, Тезисы докладов VIII Всероссийской научно-практической конференции преподавателей, учителей, студентов и молодых ученых, Лесосибирск.

2. Куприянов, П. А. Современные средства для моделирования производственных процессов / П. А. Куприянов / // Актуальные проблемы преподавания дисциплин естественнонаучного цикла, Тезисы докладов VIII Всероссийской научно-практической конференции преподавателей, учителей, студентов и молодых ученых, ЛПИ – филиал СФУ – Лесосибирск.

Структура работы – работа состоит из введения, трех глав, заключения, приложения и списка использованных источников, включающего 40 наименований. Результаты работы представлены в 35 иллюстрациях и 3 таблицах. Общий объем работы – 70 страниц.

1 Теоретические основы проектирования и разработки автоматизированной системы управления производственным процессом в сфере быстрого питания

1.1 Производственный процесс на предприятии

Производственный процесс в заведении быстрого питания представляет собой комплекс взаимосвязанных операций, направленных на своевременное и качественное обслуживание клиентов. В отличие от предприятий других отраслей, здесь производственный цикл тесно интегрирован с процессами обслуживания, логистики и управления персоналом. Эффективность данных процессов напрямую влияет на удовлетворённость клиентов и финансовые показатели предприятия [16].

Производственный процесс включает следующие основные этапы:

– приём и учёт поставляемых продуктов и ингредиентов;

– контроль качества поступающего сырья;

– хранение продукции с соблюдением условий хранения;

– приготовление блюд в соответствии с технологическими картами и рецептурой;

– оформление и выдача заказов;

– учёт остатков и списание продукции.

В условиях высокой динамики и потока клиентов, характерных для формата быстрого питания, важнейшим требованием становится минимизация времени на выполнение каждой операции при сохранении высокого качества продукции. Для этого необходимо обеспечить чёткую координацию действий сотрудников, автоматизированный учёт сырья и готовой продукции, а также быстрое реагирование на изменения спроса [11].

Информационные технологии играют ключевую роль в управлении такими производственными процессами. В частности, автоматизированные системы позволяют:

– отслеживать наличие ингредиентов на складе в режиме реального времени;

– оперативно формировать и обрабатывать заказы;

– контролировать расход сырья в зависимости от конкретных заказов;

– вести статистику продаж и анализировать популярность блюд;

– минимизировать влияние человеческого фактора на расчёт и контроль производственных показателей.

Внедрение автоматизированной системы управления производственным процессом позволяет предприятию не только повысить операционную эффективность, но и заложить основу для масштабирования бизнеса и повышения уровня управляемости. Это особенно актуально для сетевых форматов заведений быстрого питания, где единые стандарты и чёткие регламенты играют определяющую роль [29].

Таким образом, производственный процесс в заведении быстрого питания – это чётко организованная система, где скорость и качество обслуживания зависят от слаженной работы всех этапов: от поставки сырья до выдачи заказов. Автоматизация и IT-решения помогают оптимизировать процессы, снижать ошибки и повышать эффективность, что особенно важно для сетевых заведений. Грамотное управление этими процессами напрямую влияет на успех бизнеса.

1.2 Имитационное моделирование производственного процесса предприятия

Самым верхним уровнем структурно-функциональной декомпозиции производственного процесса является контекстная диаграмма. На рисунке 1 представлена диаграмма производственного процесса [12].

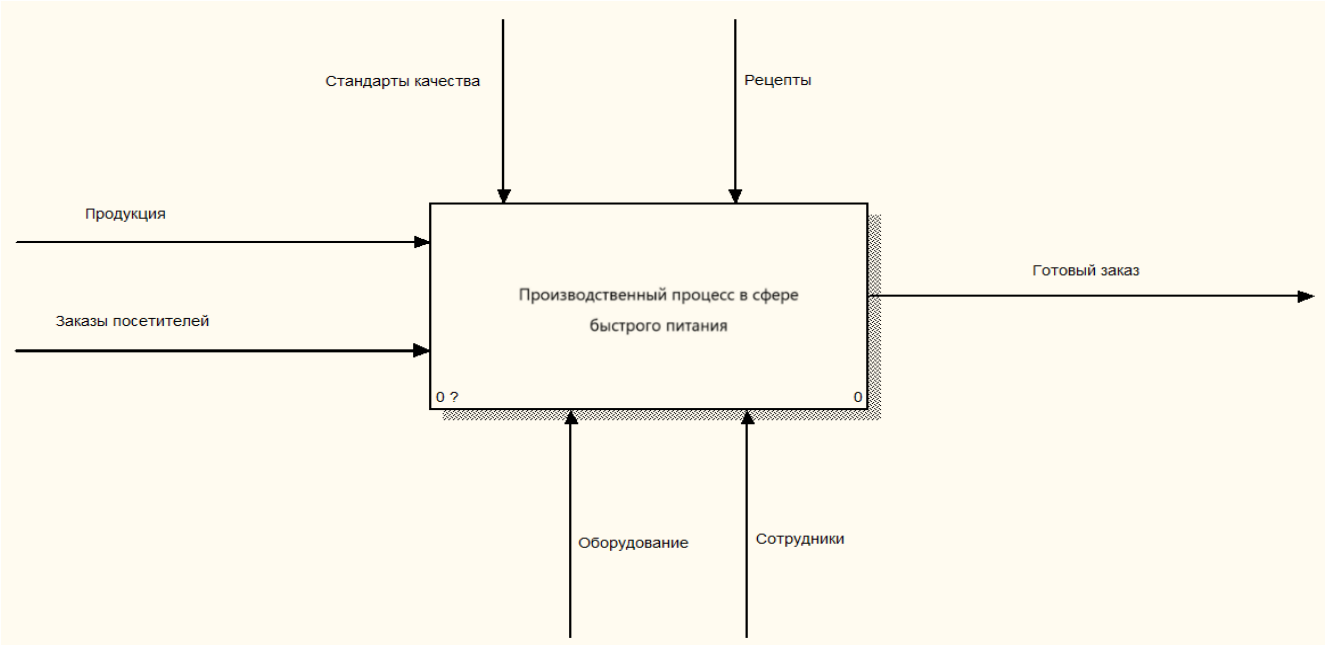

Рисунок 1 – Контекстная диаграмма производственного процесса

Контекстная диаграмма отображает общий производственный процесс, который далее включает несколько уровней декомпозиции. Чем ниже уровень, тем детальней разбивается тот или иной процесс. При всем многообразии процессов, в нотации IDEF0 каждый из них имеет несколько потоков. Входные потоки указывают на данные или материалы необходимые для производства. Управляющие потоки, символизируют контроль выполнения процесса, согласно определенным правилам, стандартам и инструкциям. Механизмы указывают на ресурсы или средства, задействованные в процессе – оборудование, персонал, расходные материалы. Выходные потоки представляют собой результаты, генерируемые в ходе выполнения технологического процесса [3].

Декомпозиция контекстной диаграммы имеет несколько уровней связей отображаемых процессов. Первый уровень декомпозиции, представленный на рисунке 2 демонстрирует четыре последовательных процесса: прием продукции, изготовление блюд, формирование меню и витрины, обработка заказа посетителя [23].

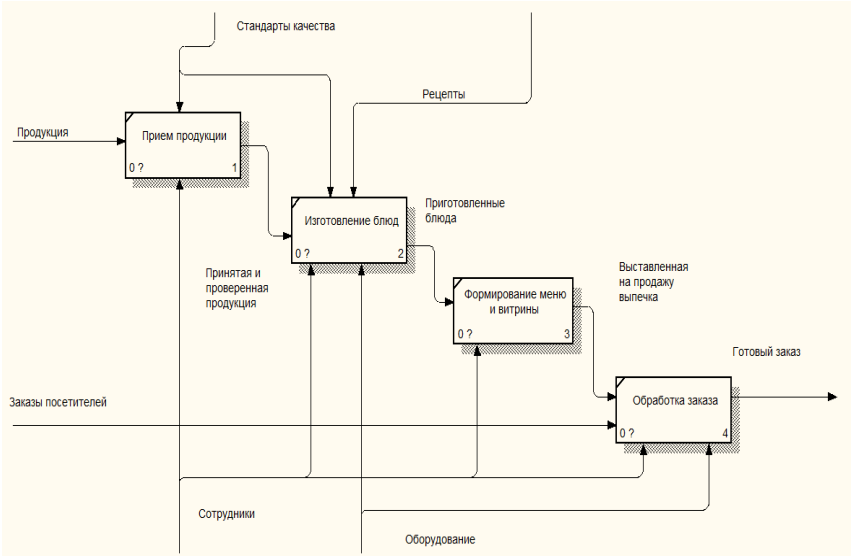

Рисунок 2 – Первый уровень декомпозиции контекстной диаграммы

Прием продукции необходим для формирования склада и входного контроля качества. Изготовление блюд является неотъемлемой частью производственного процесса и привлечения клиентов, так как меню включает в себя множество позиций. Формирование меню и витрины является заключительным этапом приготовления, так как данный процесс предоставляет клиентам выбор из имеющихся угощений [16].

Второй уровень декомпозиции включает работу с дочерними процессами каждого из четырех главных этапов. При декомпозиции приема продукции, который представлен на рисунке 3, отражены его составляющие дочерние процессы.

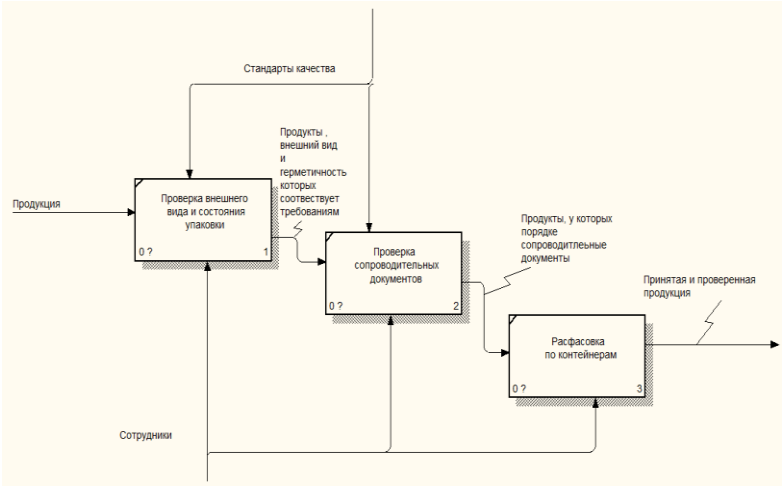

Рисунок 3 – Декомпозиция процесса приема продукции

Система обеспечения качества на производстве начинается с многоуровневой проверки входящего сырья, включающей визуальный осмотр, документальный аудит и верификацию поставщиков. Удовлетворяющее требованиям сырьё подвергается предварительной фасовке для создания эффективной системы складирования и оперативного доступа в процессе производства [31].

После того как продукция подготовлена можно приступать к изготовлению, согласно наименованию позиции. Декомпозиция процесса изготовления представлена на рисунке 4.

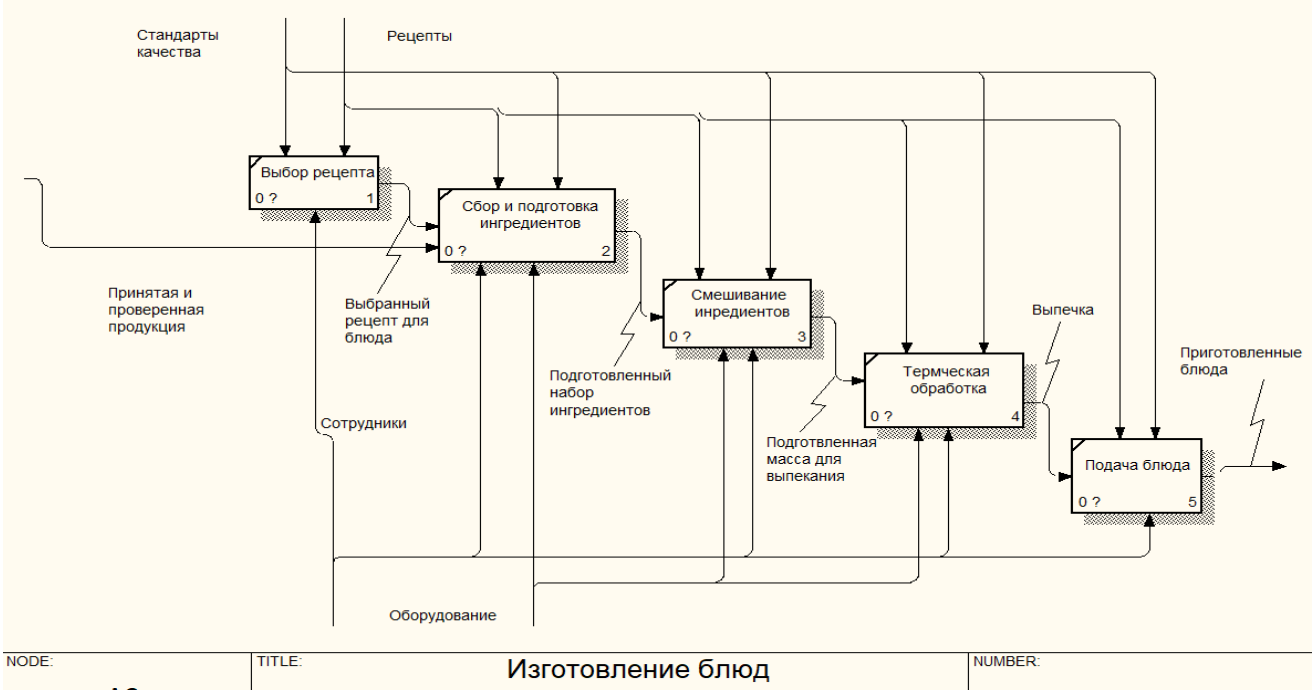

Рисунок 4 – Декомпозиция процесса изготовления блюд

Сотрудник по наименованию блюда собирает набор необходимых продуктов, после чего происходит его приготовление. Как только блюдо готово, его необходимо подать. Это включает сервировку и украшение при необходимости [10].

После того как продукция готова, она отправляется на витрину. Процесс формирования меню и витрины представлен на рисунке 5.

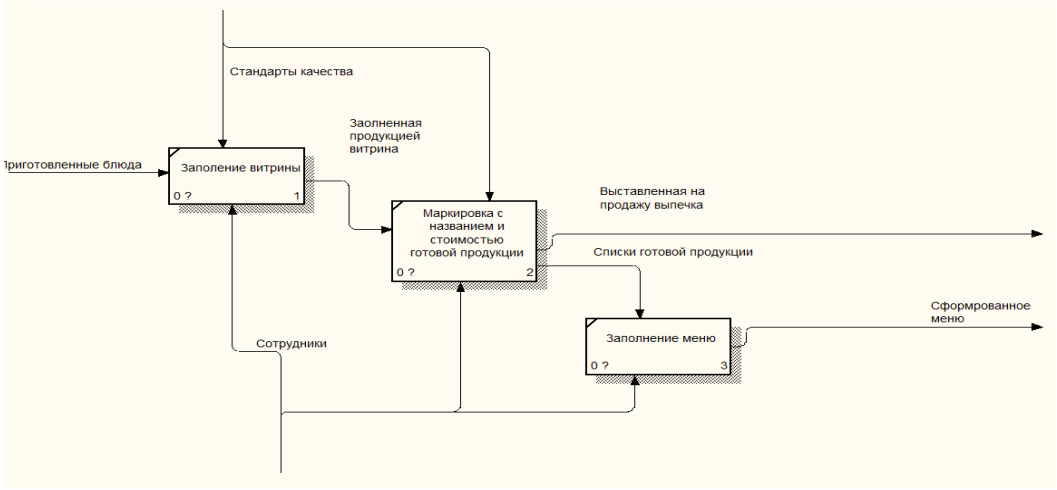

Рисунок 5 – Декомпозиция процесса формирования витрины и меню

Готовая продукция на витрине маркируется ценниками с названиями и стоимостью. Списки готовой продукции добавляются в пункты меню. Кроме этого, в меню добавляется список напитков, которые готовятся в течение дня при обработке заказов посетителей [25].

Функционирование в режиме приема заказов допускается исключительно при полном выполнении всех требований подготовительного этапа, что обеспечивает соответствие стандартам качества обслуживания. Процесс обработки заказа представлен на рисунке 6.

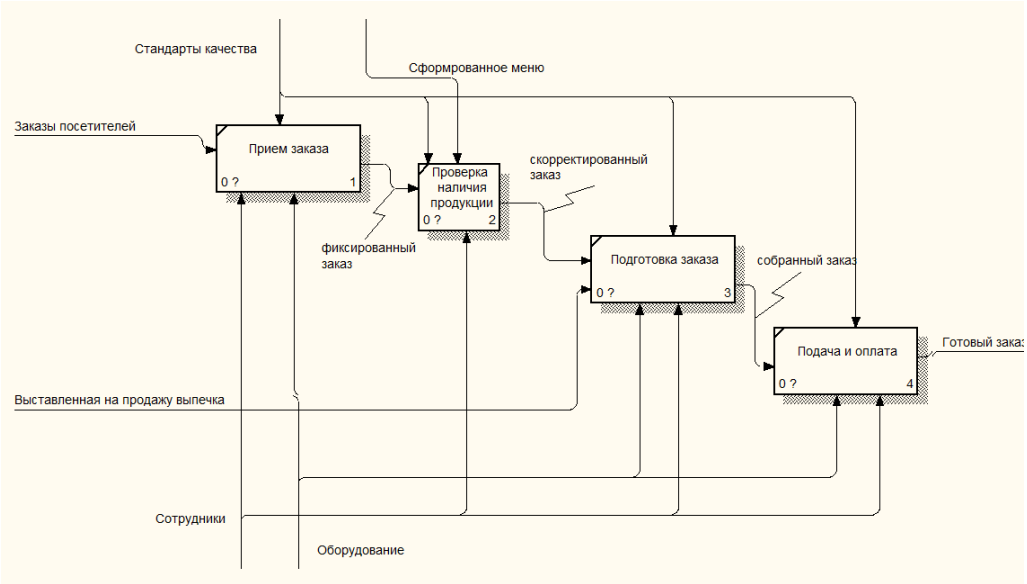

Рисунок 6 – Декомпозиция процесса обработки заказа посетителя

Обслуживание реализуется через последовательность:

1) сбор требований + верификация наличия;

2) системная регистрация;

3) производственный цикл;

4) сервировка + оплата.

Каждая стадия документируется для обеспечения контроля качества [14].

Производственный процесс в заведении быстрого питания представлен в виде структурно-функциональной модели на основе нотации IDEF0. Таким образом, декомпозиция позволила выделить ключевые этапы: прием продукции, изготовление блюд, формирование витрины и обработку заказов. Такой подход обеспечивает наглядное представление процессов и может служить основой для их оптимизации и автоматизации.

1.3 Сравнительный анализ современных автоматизированных систем управления производственным процессом в сфере быстрого питания

Современный рынок программных решений предлагает широкий спектр автоматизированных систем управления производственными и бизнес-процессами в сфере быстрого питания. Такие системы, как правило, включают модули учёта, управления заказами, анализа продаж и складского учёта. Однако при выборе конкретного решения необходимо учитывать специфику предприятия, его масштаб, формат работы и технические возможности [23].

Для анализа выбраны две наиболее распространённые системы: «RemOnline» и «1С:Предприятие 8. Общепит». Рассмотрим их особенности, преимущества и недостатки [27].

1. RemOnline

Программа подходит для широкой сферы оказываемых услуг. Предоставляется большое количество функционала: база клиентов, управление заказами, история взаимодействия с клиентом, системы лояльности, мониторинг эффективности персонала, тайм-менеджмент, интеграция с почтой, e-mail-рассылки, отчёты. Программа позволяет контролировать детали и комплектующие на складе, все этапы взаимодействия с клиентом (история

обращений и работ по каждому устройству), также контроль сервиса через мобильное приложение и многое другое. Цена за сервис зависит от количества сотрудников и филиалов [33].

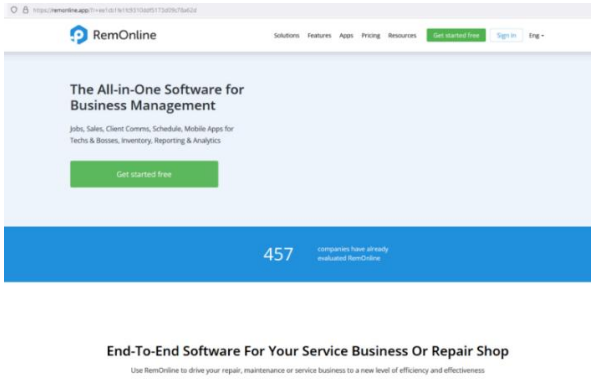

Рисунок 7 – Стартовая страница RemOnline

Преимущества и недостатки RemOnline представлены в таблице 1.

Таблица 1 – Преимущества и недостатки RemOnline

Преимущества	Недостатки
хорошая скорость работы интерфейса, быстро и легко работать	высокая стоимость тарифов
много шаблонов для готового бизнеса, простая и быстрая настройка	если закончилась подписка на тариф, то нет возможности посмотреть данные
оперативная техническая поддержка, отвечают живые люди	цены только в евро, при оплате в рублях происходит конвертация по текущему курсу
шаблоны для документов, которые позволяют очень экономить время на формировании документов	
проект активно развивается, и постоянно обновляется	не всем подходит облачное решение, версии для персональных ПК на рабочий стол у RemOnline нет
полноценный складской учёт, позволяет контролировать все передвижения комплектующих, их наличие и т.д.	

2. 1С:Предприятие 8. Общепит

Отраслевое решение предназначено для автоматизации оперативного, бухгалтерского и налогового учета в независимых и сетевых предприятиях общественного питания различных форматов и концепций, таких как: кафе, бары, рестораны, столовые и буфеты (в том числе – при предприятиях различного профиля), кейтеринговые компании и операторы питания, цеха по

производству кондитерских изделий, полуфабрикатов и кулинарии, а также другие предприятия питания [37].

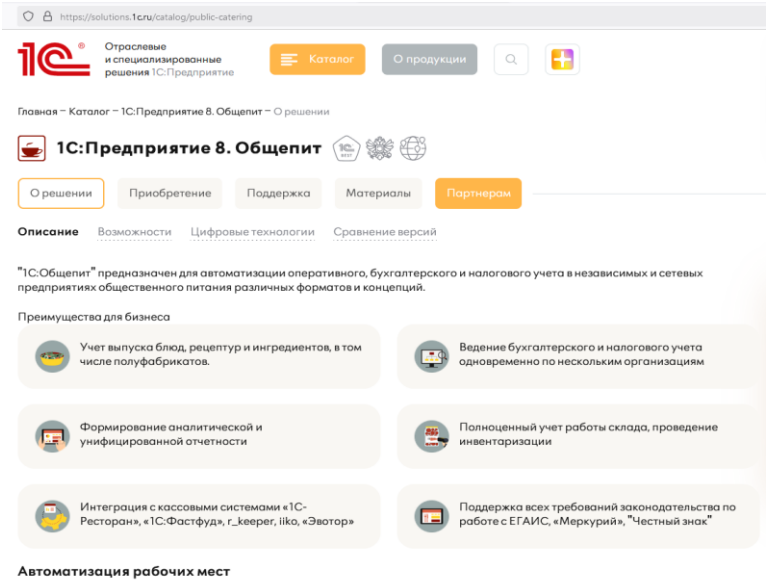

Рисунок 8 – Страница покупки решения от 1С

Решение использует возможности типовой конфигурации «1С:Предприятие 8. Общепит» [20].

Преимущества и недостатки «1С:Предприятие 8. Общепит» представлены в таблице 2.

Таблица 2 – Преимущества и недостатки «1С:Предприятие 8. Общепит»

Преимущества	**Недостатки**
позволяет учитывать движение товара в каждом отделе предприятия	ошибки приложения, возникающие в процессе работы либо после обновления системы
ее можно использовать на предприятии различной формы	отсутствие части функционала, который необходимо дорабатывать, под специализированные задачи
можно вести калькуляцию, хранить разработанные рецепты	для поддержки системы необходим обученный специалист, которого содержат в штате, либо нанимают
программа помогает подобрать взаимозаменяемые продукты, рассчитать их использование в определенном рецепте	
учитывает поступление и перемещение товаров, приготовление и наличие полуфабрикатов, списанный товар	функции приложения распространяется по подписке, которая экономит время на поиски информации, и деньги, уменьшая вероятность ошибок в работе сотрудников
составлять аналитическую отчетность (отчет о получаемой прибыли и реализации блюд, остатков блюд, расхода товаров, калькуляции за определенный период).	

Программа поддерживает две схемы оборота документации, она не ограничивает ведение количества приготовленных блюд и применяемых рецептов. При этом все инструменты управления доступны для пользователей, на рабочем столе расположены отчеты, справочники, нужная документация и журналы [9].

Проведенный анализ показал, что рассмотренные программные продукты, несмотря на широкий функционал и универсальность, способны удовлетворить базовые потребности предприятий быстрого питания. Однако ни одно из представленных решений не соответствует ключевым требованиям проекта, а именно:

– автономность работы (функционирование при нестабильном или отсутствующем интернет-соединении);

– минималистичная архитектура (отсутствие избыточного функционала, характерного для корпоративных систем);

– простота развертывания (возможность запуска со съемного носителя без сложных настроек).

Вышеизложенное позволяет сделать вывод о том, что большинство коммерческих систем ориентированы на облачные технологии, требуют постоянного подключения к сети и обладают избыточной сложностью конфигурации, что делает их непригодными для локального использования в условиях ограниченной инфраструктуры.

1.4 Описание предметной области. Требования к разрабатываемой автоматизированной системе по управлению производственным процессом

Заведение в сфере быстрого питания представляют собой предприятия общественного питания, основной задачей которых является оперативное обслуживание клиентов с минимальными затратами времени на приготовление и выдачу заказов. Такие заведения, как правило, предлагают ограниченный ассортимент блюд и напитков, стандартизированную рецептуру, а также повторяемые процессы приготовления [5].

Организационная структура заведения включает следующие функциональные единицы:

– кассовая зона – приём и оформление заказов, расчёт с клиентами;

– кухонный блок – непосредственное приготовление продукции;

– склад – хранение сырья, полуфабрикатов и вспомогательных материалов;

– управленческий уровень – контроль остатков, планирование закупок, анализ продаж и формирование отчётности.

Важнейшей особенностью предприятий быстрого питания является высокая интенсивность операций, большой объём транзакций в течение коротких временных интервалов и необходимость строгого соблюдения технологических карт и стандартов приготовления. В таких условиях критически важным становится использование программных решений для управления и оптимизации производственного процесса [7].

Рассмотрим требования к разрабатываемой автоматизированной системе управления производственным процессом в сфере быстрого питания.

С учётом анализа предметной области и специфики работы заведений в сфере быстрого питания, к автоматизированной системе управления предъявляются следующие функциональные и технические требования:

1) Общие требования:

– локальная работа без постоянного подключения к интернету;

– простота установки и эксплуатации (возможность запуска на любом ПК под Windows);

– компактность и переносимость (возможность запуска с внешнего накопителя).

2) Требования к функциональности:

– регистрация заказов и формирование состава заказа из готовых блюд;

– ведение справочников блюд, ингредиентов, рецептур и категорий продукции;

– учёт складских остатков, списание ингредиентов при оформлении заказов;

– возможность добавления, редактирования и удаления записей;

– формирование отчётов по заказам, расходу ингредиентов, остаткам продукции.

3) Требования к пользовательскому интерфейсу:

– наличие графического интерфейса с интуитивной навигацией;

– формы отображения списков сущностей (блюда, заказы, ингредиенты и т.д.);

– всплывающие окна для ввода данных и подтверждения действий;

– сообщения об ошибках и валидация пользовательского ввода.

4) Требования к архитектуре и надёжности:

– использование встроенной СУБД (SQLite) для хранения данных;

– поддержка связей между таблицами через внешние ключи;

– изоляция логики работы с базой данных в отдельные модули;

– устойчивость к ошибкам и сбоям, сохранение целостности данных;

– минимальная зависимость от внешних компонентов и настроек системы.

5) Перспективы развития:

– возможность миграции интерфейса с Windows Forms на WPF;

– поддержка мобильных платформ (в будущем – разработка на базе MAUI или Xamarin);

– реализация паттерна MVVM для повышения масштабируемости проекта.

Таким образом, создаваемая система должна обеспечивать стабильную и автономную работу, охватывая все ключевые процессы производственной деятельности небольшого предприятия в сфере быстрого питания. Автоматизированная система должна быть ориентирована на конечного пользователя – персонал заведения, не обладающий специальной технической

подготовкой, что требует особого внимания к удобству интерфейса и надёжности исполнения всех функций [2].

1.5 Выбор инструментальных средств разработки автоматизированной системы управления производственным процессом в сфере быстрого питания

От правильности выбора технологий зависит не только скорость и удобство разработки, но и производительность, стабильность и расширяемость конечной системы [32].

1. Язык программирования: C#

Язык C# выбран по следующим причинам:

– полная интеграция с операционной системой Windows;

– наличие визуальных средств разработки интерфейса – Windows Forms и WPF;

– поддержка объектно-ориентированного подхода, что упрощает масштабирование и сопровождение проекта;

– широкое сообщество и хорошая документация;

– совместимость с встраиваемыми СУБД, такими как SQLite, через стандартные библиотеки (System.Data.SQLite, Microsoft.Data.Sqlite).

В отличие от Java, где визуальная часть реализуется через Swing/JavaFX, C# обеспечивает более нативный внешний вид и простую интеграцию с системными компонентами Windows, что критично для настольного ПО.

2. Среда разработки: Visual Studio Community Edition

Среда разработки Visual Studio выбрана благодаря:

– поддержке языка C# и WinForms «из коробки»;

– интегрированному визуальному редактору форм;

– инструментам для отладки, анализа кода и работы с базами данных;

– возможности быстрого создания исполняемых файлов (*.exe) и установки без сторонних зависимостей;

– бесплатной лицензии для учебных и небольших коммерческих проектов.

Альтернативы, такие как JetBrains Rider или MonoDevelop, уступают в удобстве визуального редактора и настройке шаблонов Windows-приложений [35].

3. Графическая библиотека: Windows Forms

Выбор пал на WinForms по следующим причинам:

– простота реализации оконного интерфейса;

– возможность быстрого создания форм с кнопками, таблицами, текстовыми полями и обработчиками событий;

– поддержка работы без интернета, без сторонних зависимостей;

– низкий порог вхождения и высокая скорость разработки.

Хотя WPF является более современным решением с поддержкой MVVM и гибкой системой стилизации, WinForms лучше подходит для начальной реализации – быстрее создаётся прототип и проще тестировать функциональность [40].

4. Система управления базами данных: SQLite

Для хранения информации о заказах, рецептах, ингредиентах и складах используется СУБД SQLite, которая представляет собой компактную, высокопроизводительную СУБД, работающую в виде одного файла [16].

Причины выбора:

– автономность – база данных не требует отдельного сервера или установки драйверов;

– компактность – единый файл может быть легко скопирован или перемещён;

– производительность – высокая скорость чтения/записи при небольших и средних объёмах данных;

– совместимость с C# – через библиотеку System.Data.SQLite;

– поддержка SQL и внешних ключей, что критично для обеспечения связности данных (например, между таблицами «Заказы», «Блюда», «Ингредиенты»).

Недостатки SQLite, такие как отсутствие многопоточной записи и ограниченные типы данных, не являются критичными в рамках настольного приложения для одного пользователя [34].

Результаты сравнения выбранных инструментов с альтернативами, представлены в таблице 3.

Таблица 3 – Сравнение с альтернативами

СУБД	Преимущества	Недостатки
SQLite	Быстрая, автономная, простая в установке	Нет поддержки одновременной записи, базовые типы
MSSQL Server	Мощная, масштабируемая, развитая система	Требует установки сервера, больше объём и ресурсы
MongoDB	Гибкая, без схем	Требует отдельного сервера, сложность структуры
XML/JSON	Простота встраивания	Нет связности, высокая вероятность ошибок, медленно

5. Архитектурные подходы и шаблоны

Для реализации приложения используется трёхуровневая архитектура:

– уровень данных – классы моделей и контексты взаимодействия с БД;

– уровень логики – обработка действий пользователя, валидация данных;

– уровень представления (UI) – формы, таблицы, поля ввода, кнопки.

Данный подход облегчает отладку, модульное тестирование и последующее масштабирование (например, переход с WinForms на WPF).

Также в будущем возможно внедрение паттерна MVVM – это особенно актуально при переходе на WPF. Это повысит уровень разделения бизнес-логики и интерфейса и упростит поддержку проекта [18].

Таким образом, выбор инструментов разработки обоснован потребностями в автономности, простоте развертывания, надёжности и скорости реализации. Комбинация C# + Visual Studio + WinForms + SQLite обеспечит:

– быструю разработку без зависимости от сторонних сервисов;

– удобную реализацию пользовательского интерфейса;

– высокую надёжность хранения данных и целостность записей;

– простоту поддержки и возможное масштабирование в будущем.

2 Разработка системы управления производственным процессом в сфере быстрого питания

2.1 Разработка модулей автоматизированной системы управления производственным процессом в сфере быстрого питания

Как только пользователь запускает приложение, оно отвечает появлению пользовательского интерфейса, через который происходит управление всего приложения.

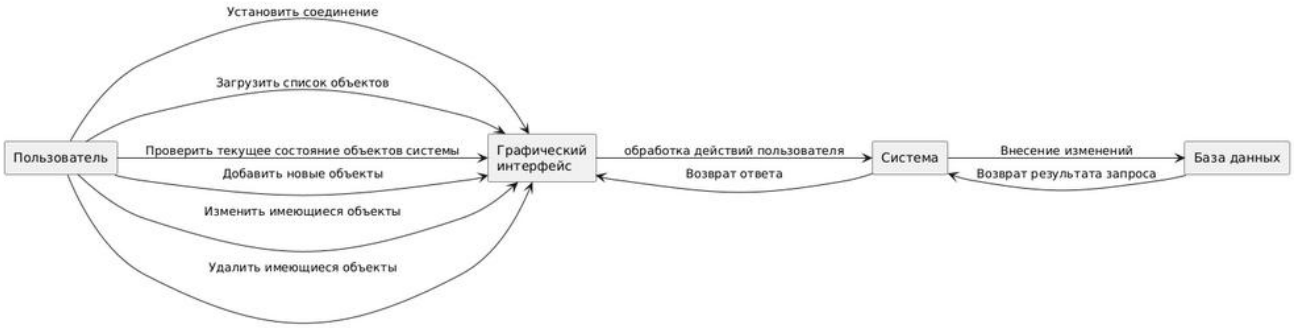

Рисунок 8 – Общий принцип работы приложения

Как видно из схемы, приложение состоит из двух модулей – это графический интерфейс и модуль для работы с базой данных. В графическом интерфейсе расположены визуальные компоненты и методы обработки действий пользователя, а в модуле для работы с базой данных расположены элементы, которые конвертируют действия над объектами системы в код запросов, которые отправляются в базу данных.

Если база данных содержит таблицы, которые характеризуют объекты внешнего мира, которые проецирует информационная система, то само приложение должно иметь аналоги подобных таблиц у себя в виде классов. Подобные классы именуются моделями дынных и содержат в себе одноименные поля или свойства соответствующих таблиц [8].

Структура моделей данных может быть выражена в диаграмме классов, изображенных на рисунке 9.

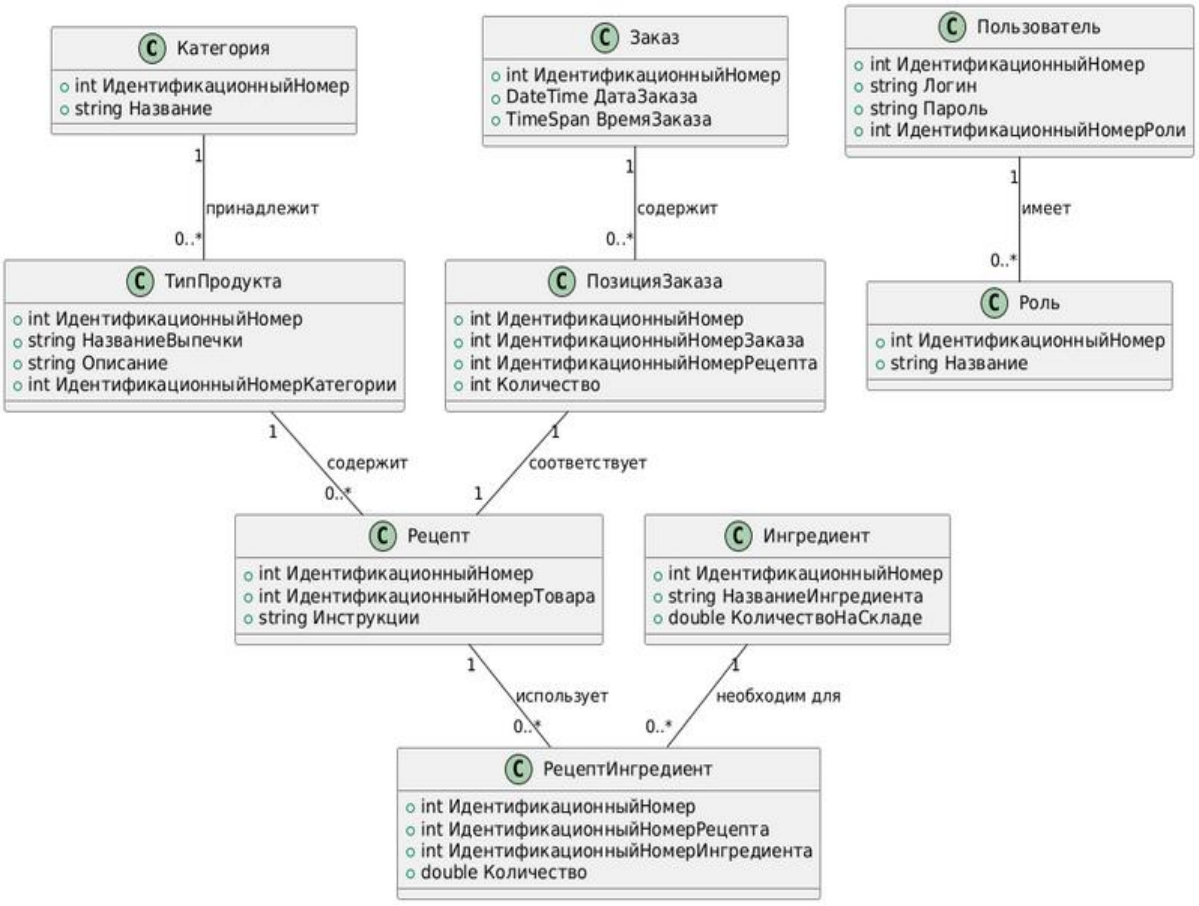

Рисунок 9 – Диаграмма классов моделей данных

Данные классы не содержат методов, так как нужны только для отражения самих объектов реального мира и сохранения информации в СУБД. Кроме набора моделей в архитектуре системы используется еще один слой, который обеспечивает взаимодействие приложения и базы данных. Каждый из классов подобного типа именуется классом контекста соответствующей модели данных и содержит лишь методы, которые возвращают список объектов модели, производят поиск объекта, обновляют и удаляют объект [28].

На рисунке 10 изображена диаграмма классов контекстов и моделей данных.

В класс контекста данных передается строка соединения, где далее в каждом их вызываемых методов открывается соединение с базой данных, производятся манипуляции с таблицами, после чего соединение закрывается. Таким образом, работа с базой данных изолирована от остальной логики приложения [30].

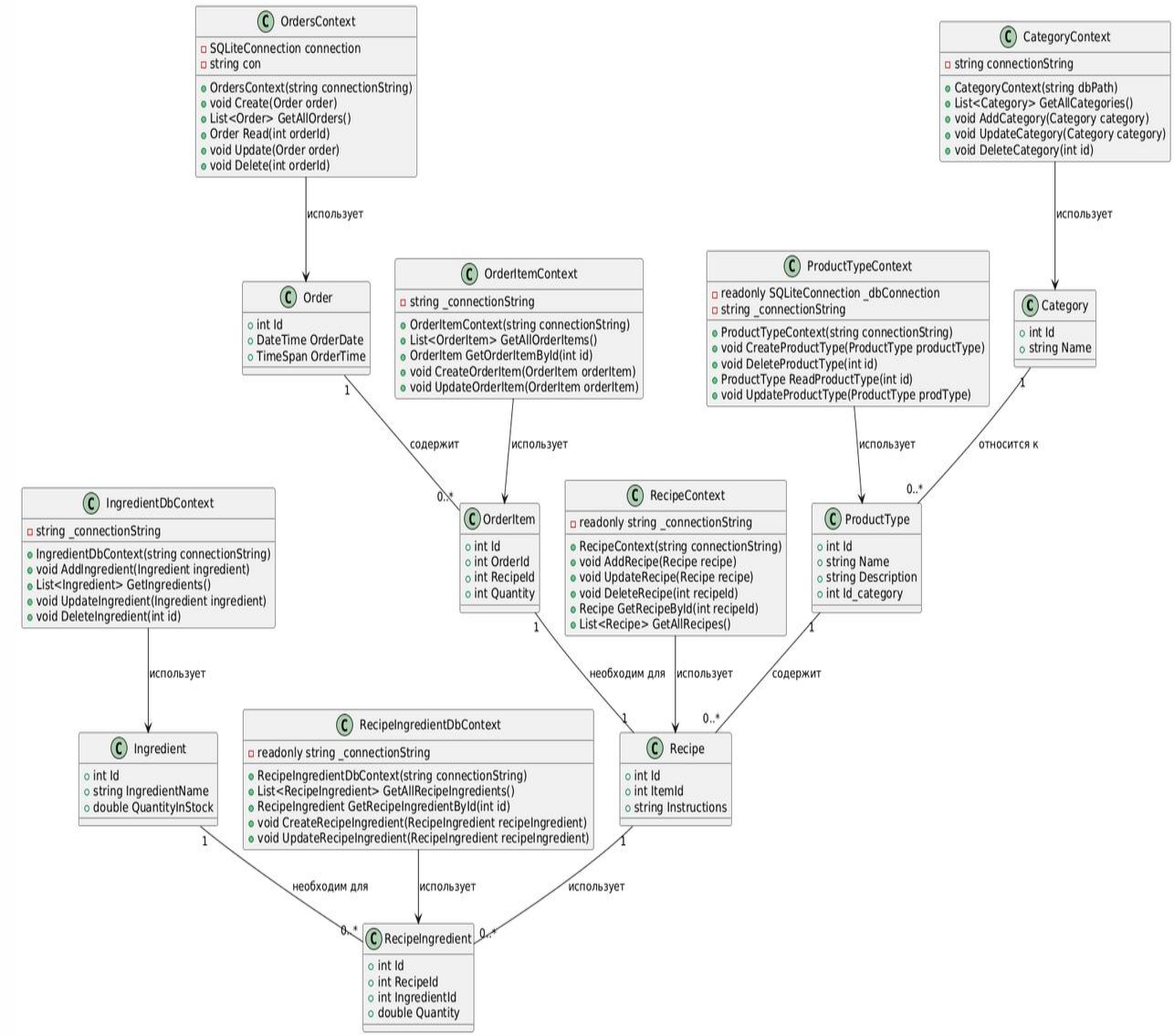

Рисунок 10 – Диаграмма классов контекстов и моделей данных

Класс контекста содержит набор методов, который эквивалентен стандартному набору операций с данными (CRUD).

Аббревиатура CRUD расшифровывается следующим образом:

– create (Создание) – операция создания новой записи в базе данных или добавления нового объекта;

– read (Чтение) – операция получения данных или чтения информации из базы данных или другого хранилища;

– update (Обновление) – операция изменения или обновления существующей записи или объекта в базе данных;

– delete (Удаление) – операция удаления записи или объекта из базы данных или другого хранилища данных;

Эти операции представляют основные функции для работы с данными. Одним из свойств контекста данных является проверка базы данных на существование и действие при отсутствии в ней необходимых таблиц.

Перед тем как производить различные манипуляции с объектами, необходимо убедится в работоспособности и целостности базы данных, а также наличия таблицы, которая будет хранить записи для определенного объекта. После создания объекта контекста вызывается метод, который ищет таблицу в базе данных и создает ее при необходимости.

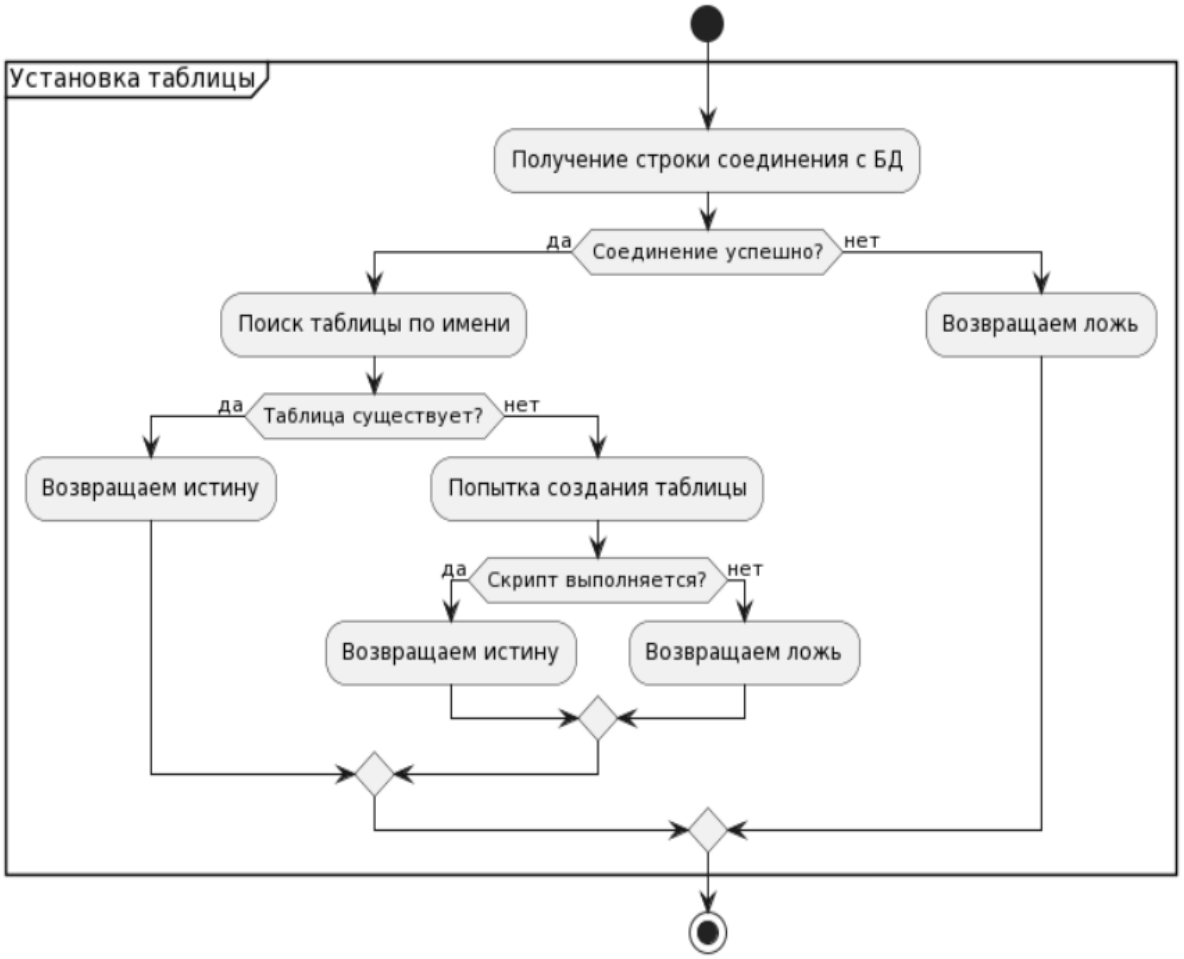

Рисунок 11 – Алгоритм проверки существования базы данных и таблицы

Далее в классе контекста реализуется метод, который производит поиск в базе данных и возвращает объект искомого типа по его идентификационному номеру, аналогично происходит удаление.

Поиск объекта по номеру в базу данных представлен на рисунке 12.

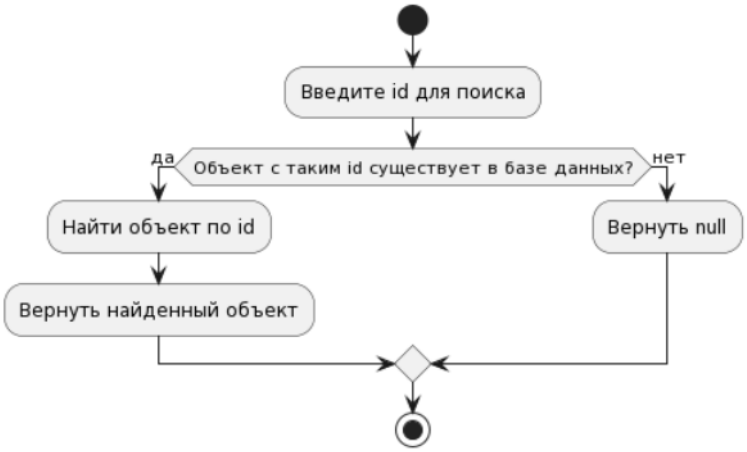

Рисунок 12 – Поиск объекта по номеру в базе данных

Чтобы редактировать объект в базе данных, необходимо в метод контекста добавить обновленный объект (модель), по которому будет производиться поиск и редактирование объекта в базе данных, алгоритм которого изображен на рисунке 13.

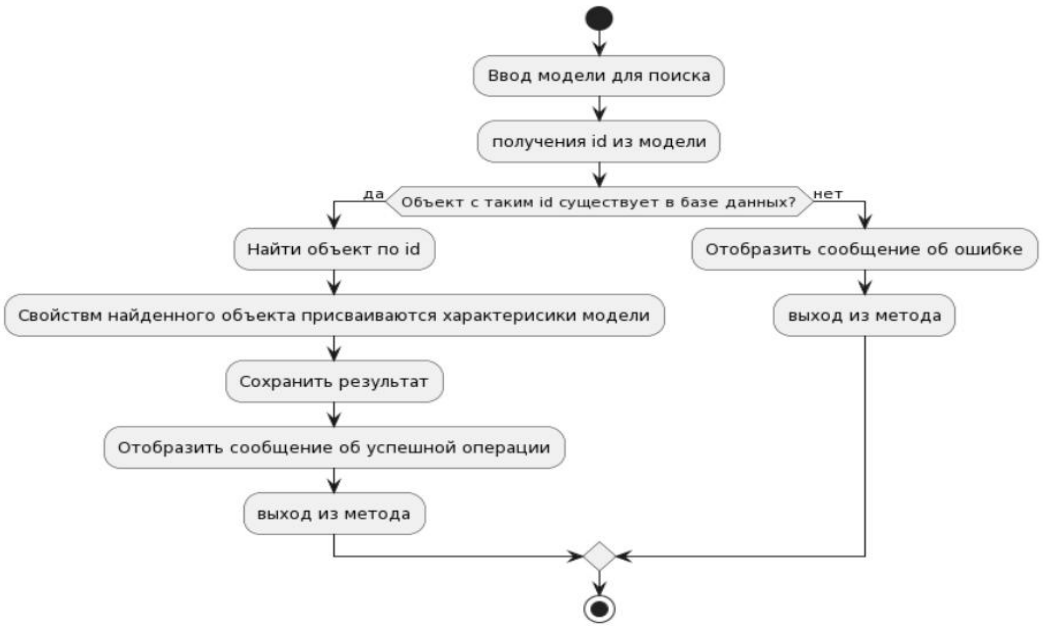

Рисунок 13 – Обновление объекта в базе данных

Алгоритм присваивает объекту в базе данных новые характеристики и сохраняет результат, для добавления новой записи применяется схожий алгоритм. Отличие лишь в том, что нет необходимости в id объекта. В параметрах метода имеется модель новой записи, для которой создается скрипт в базе данных, где добавляются все необходимые поля в таблицу, за

исключением id. Так как в базе данных имеется автоматическое приращение при добавлении записи.

Следующим уровнем архитектуры приложения являются формы – классы пользовательского интерфейса, которые отражают окна с наличием различных элементов взаимодействия: таблицы, поля ввода, кнопки и т.д. Формы содержат в себе встроенные методы обработки пользовательских действий, в которые можно добавлять различные алгоритмы обработки. К примеру, на главной форме, которая изображена на рисунке 14, расположены четыре кнопки [29].

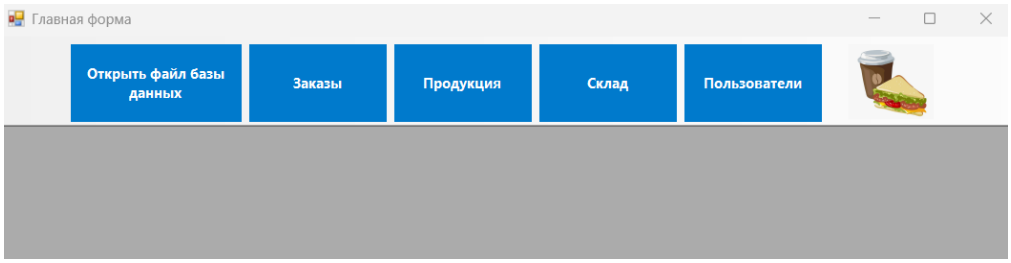

Рисунок 14 – Главная форма приложения

Если открыть код класса формы, то в ней реализовано четыре метода обработчика событий (главная, заказы, продукты, склад) нажатия отображаемых кнопок, каждый из которых будет вызывать другие формы. В диаграмме последовательностей, представленной на рисунке 15, отображен порядок действий [34].

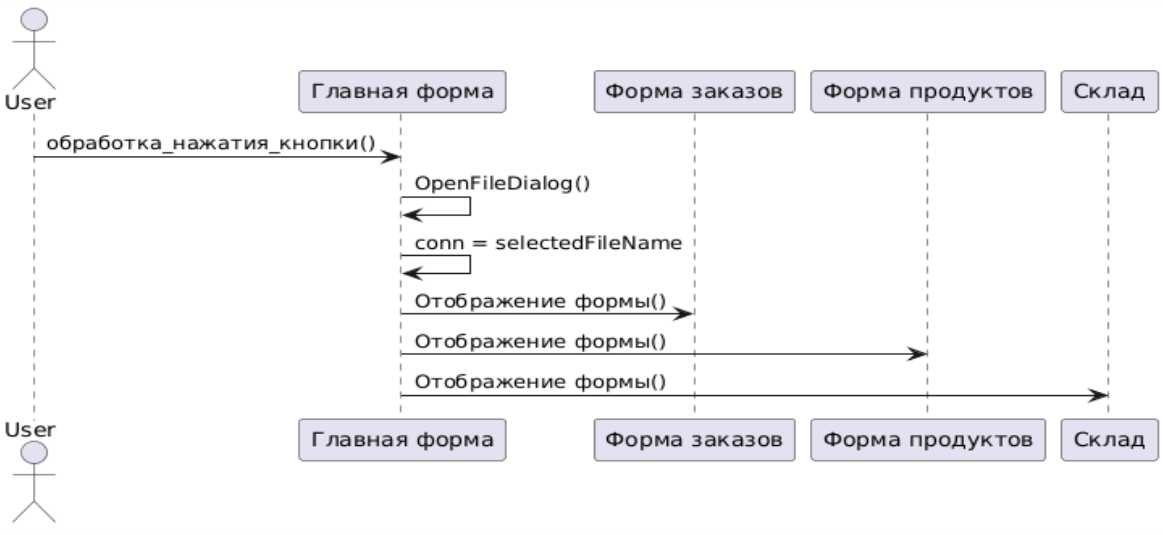

Рисунок 15 – Последовательность вызова форм

Архитектура системы реализована по принципу иерархического наследования форм, где каждый дочерний элемент содержит собственные обработчики событий, ответственные за генерацию следующих вложенных оконных объектов [34].

Взаимодействие форм, отображенное на рисунке 16 в виде диаграммы классов, демонстрирует древовидную структуру.

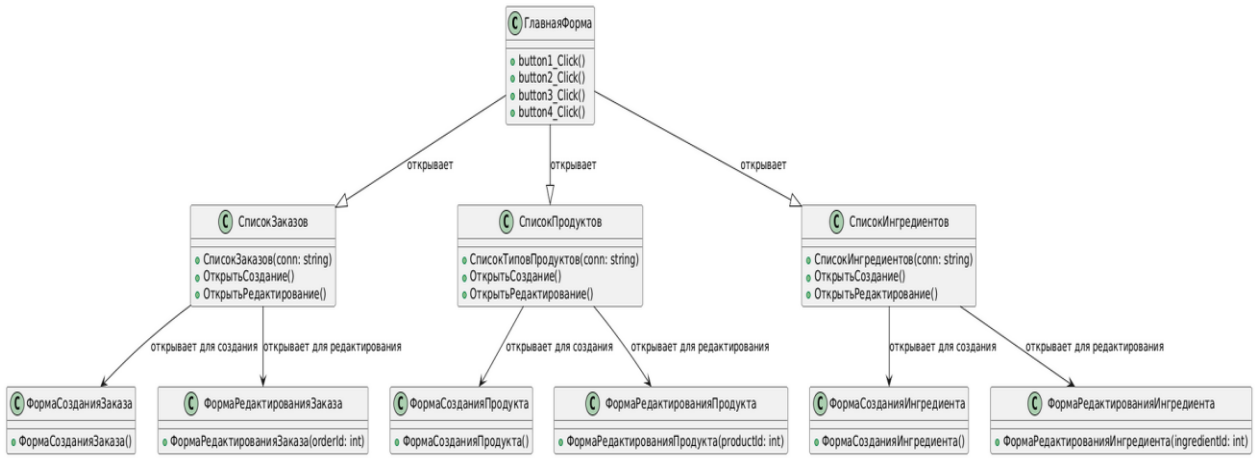

Рисунок 16 – Диаграмма классов форм пользовательского интерфейса

Подход от общего к частному упрощает реализацию приложения, так как декомпозирует большой блок, на отдельные компактные модули, а в связке с отдельным слоем работы с данными, реализация основного механизма – логики обработки действий пользователя является достаточно простым, так как нет смешивания моделей, контекстов данных и других посторонних структур [17].

2.2 Тестирование автоматизированной системы

Тестирование разработанной автоматизированной системы управления производственным процессом в сфере быстрого питания является важнейшим этапом, направленным на подтверждение корректности работы всех компонентов, выявление и устранение возможных ошибок, а также проверку соответствия реализованного функционала заявленным требованиям [17].

Цели тестирования:

– Проверить корректность взаимодействия между пользовательским интерфейсом, логикой приложения и базой данных;

– Оценить стабильность работы приложения при типичных сценариях использования;

– Убедиться в правильности реализации бизнес-логики (списание ингредиентов, расчёт стоимости заказа и т.д.);

– Проверить защиту от некорректного пользовательского ввода и устойчивость к сбоям.

Для работы автоматизированной системы необходимо наличие СУБД, а также наличие в ней необходимых таблиц, их атрибутов и связей. Так как для информационной системы задействована SQLite, то можно с помощью определенных менеджеров открыть файл базы данных и посмотреть на структуру ее таблиц и отношений, которая изображена на рисунке 17.

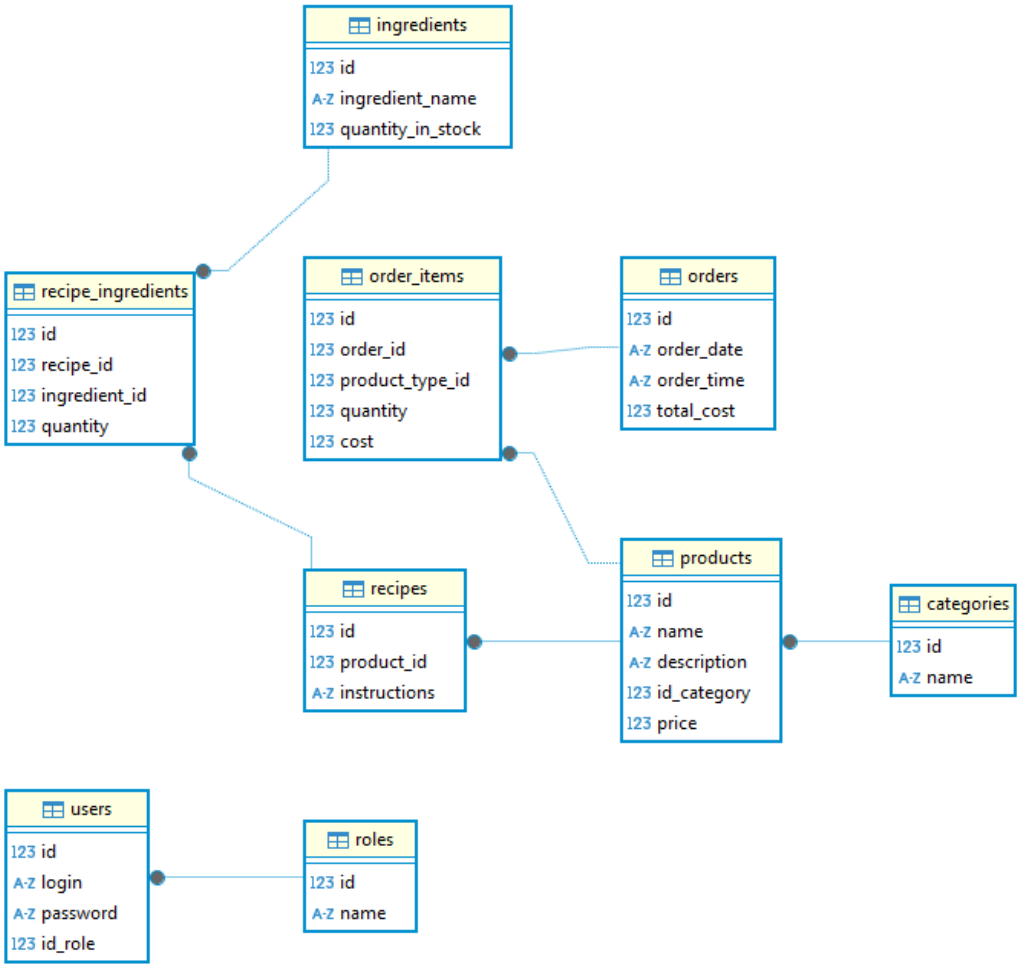

Рисунок 17 – Диаграмма логической схемы базы данных

Полученная структура таблиц схожа с диаграммой классов моделей данных, представленная в параграфе 2.1. Практически все данные, относящиеся непосредственно к работе в сфере быстрого питания связаны между собой внешними ключами [24].

Взаимодействие базы данных и приложения разделено под каждую из таблиц и моделей данных в отдельном классе, что обеспечивает модульность и разделение по задачам каждого из классов. Работоспособность классов контекста можно проверить при отладке приложения, когда отображается входная и входная информация при работе методов. В качестве примера, будет рассмотрим работу класса контекста для работы с готовой продукцией – «ProductContext». При создании объекта в конструктор класса добавляется строка для соединения с базой данных. После чего методы класса открывают соединение с СУБД, производят манипуляции и закрывают соединение. Таким образом открытые соединения не находятся в фоновом режиме во время работы приложения, а открываются, не блокируя друг друга [8].

На форме, изображенной на рисунке 18, при отладке приложения вводятся данные нового продукта:

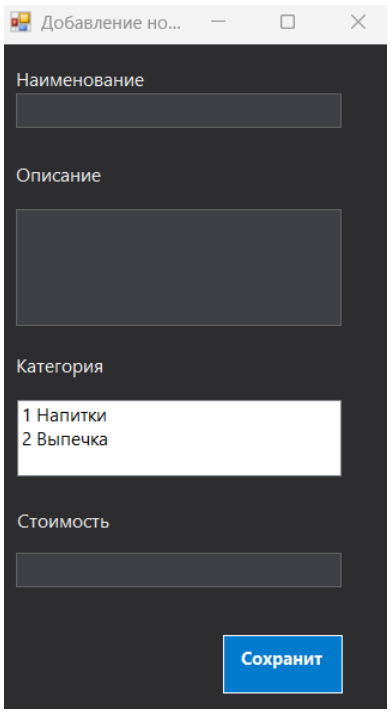

Рисунок 18 – Форма добавления новой записи готовой продукции

В режиме отладки, класс формы принимает информацию в виде значений полей ввода, создает новый объект продукта. Объект отправляется в качестве аргумента метода добавления продукта в класс контекста данных. Событие передачи объекта, отображенное на рисунке 19, может быть перехвачено средой разработки в точке останова [8].

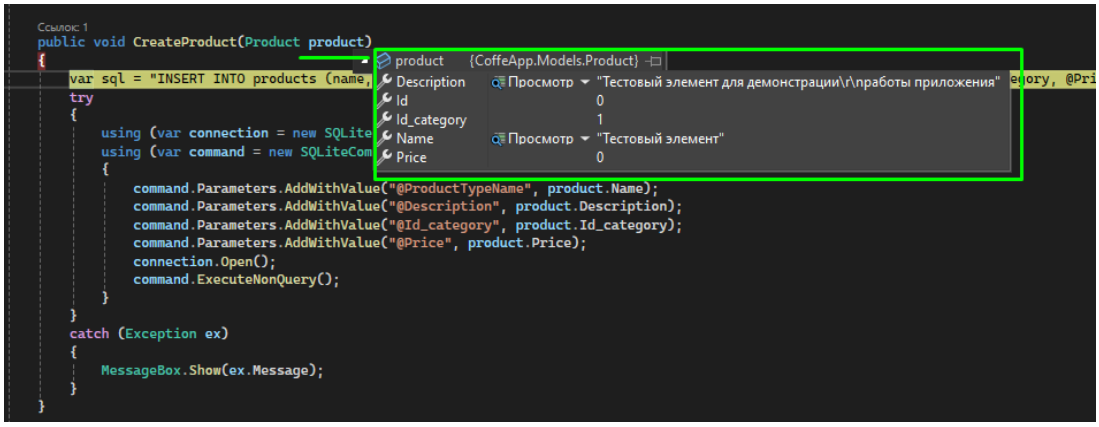

Рисунок 19 – Передача объекта продукта в метод добавления класса контекста

После выполнения работы контекста данных, новая запись добавляется в СУБД. Это можно отследить при работе с графическим клиентом SQlite базы данных, если произвести выборку всех записей, представленной на рисунке 20.

	Фи...	Фильтр	Фильтр	Фильтр	Фил...
1	1	Эспрессо	Классический итальянский кофе, ...	1	100.0
2	2	Американо	Эспрессо с добавлением горячей воды.	1	100.0
3	3	Капучино	Эспрессо с паровым молоком и молочной ...	1	120.0
4	4	Латте	Эспрессо с паровым молоком и небольшим ...	1	130.0
5	5	Мокко	Латте с добавлением шоколадного сиропа.	1	140.0
6	6	Далгоне	Кофе, взбитый до кремовой консистенции с ...	1	120.0
7	7	Флэт Уайт	Латте с меньшим количеством молока и боле...	1	150.0
8	8	Кофе по-восточному	Кофе, приготовленный с кардамоном и ...	1	100.0
9	9	Круассан	Традиционная французская выпечка с нежны...	2	160.0
10	10	Мафин	Пышный сладкий пирожок, часто с фруктами.	2	210.0
11	11	Брауни	Шоколадный десерт с влажной текстурой.	2	110.0
12	12	Тартуф	Миниатюрный десерт с шоколадной начинкой.	2	130.0
13	13	Чизкейк	Десерт на основе сыра с хрустящей основой.	2	260.0
14	14	Пирожное Наполеон	Слоеное пирожное с кремом.	2	310.0
15	15	Пирожное Мадлен	Мягкое пирожное в форме раковины.	2	150.0
16	16	Кекс	Сладкая выпечка с ягодами или орехами.	2	70.0
17	17	Песочное печенье	Овощное печенье с богатым масляным вкусом.	2	40.0
18	20	Тестовый элемент	Тестовый элемент для демонстрации...	1	0.0

Рисунок 20 – Выборка всех записей продукции в графическом клиенте

Как видно из рисунка, взаимодействие приложения и СУБД, происходит корректно. Пользователь может создавать новый объект, данные которого переносятся в базу. Эта же информация отображается в форме списка продукции, которая представлена на рисунке 21.

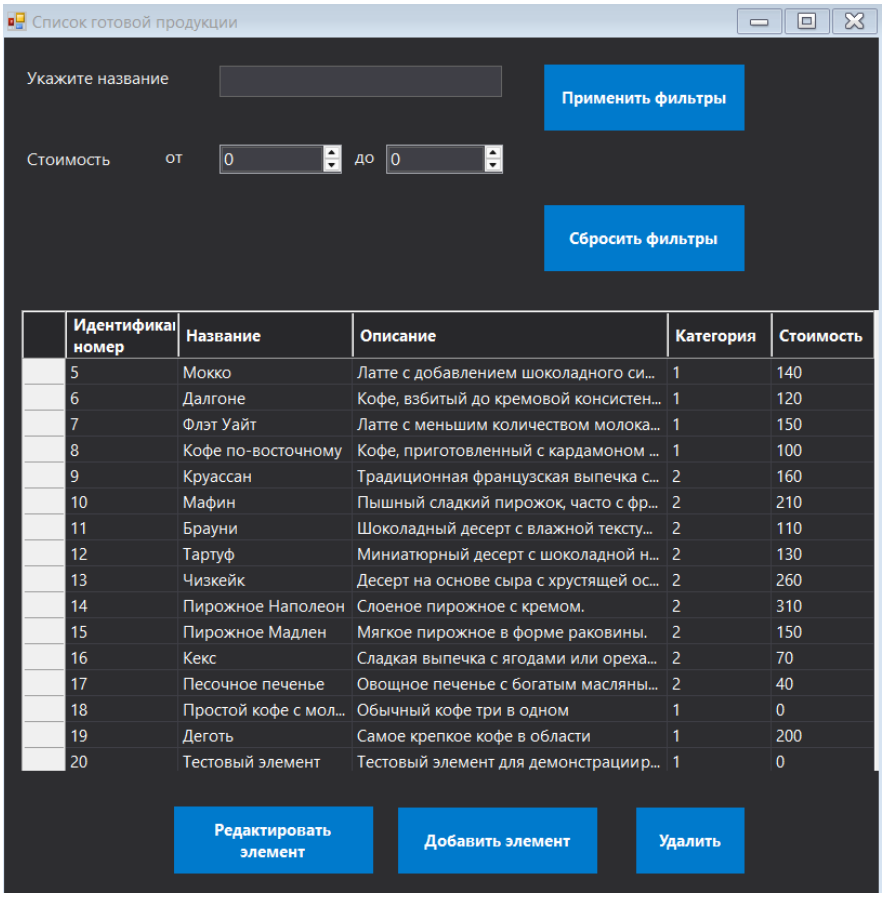

Рисунок 21 – Форма списка готовой продукции

Таким образом, приложение в одной форме создает объект данных и отправляет SQL запрос для изменения базы данных, после чего в родительской форме отправляется запрос на извлечение записей и их отображение [19].

2.3 Руководство пользователя

Приложение представляет собой исполняемый файл windows с расширением exe и работает, используя вспомогательные файлы. Общее количество вспомогательных файлов изображено на рисунке 22. При запуске приложения необходимо установить соединение с базой данных, путем выбора ее файла в диалоговом окне [23].

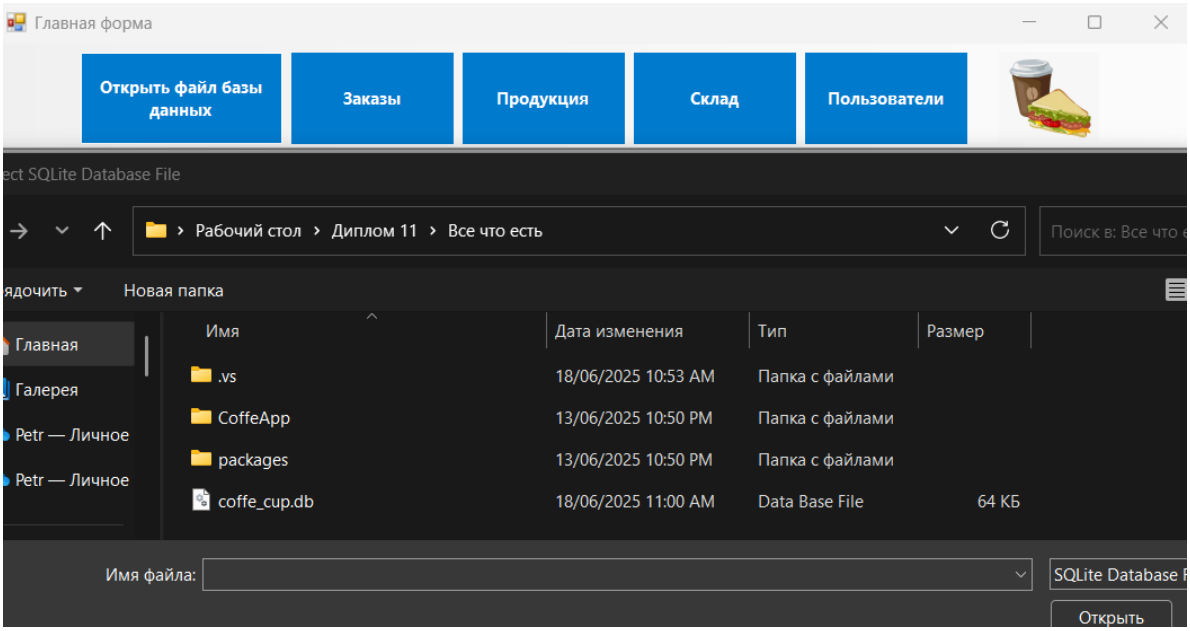

Рисунок 22 – Форма открытия файла в базы данных

Как только соединение с СУБД будет установлено, пользователь может открывать соответствующие окна списков сущностей и работать с записями. Чтобы убрать возможность использования приложения посторонними лицами, в системе реализован механизм авторизации. Чтобы войти в систему необходимо открыть форму входа, которая изображена на рисунке 23.

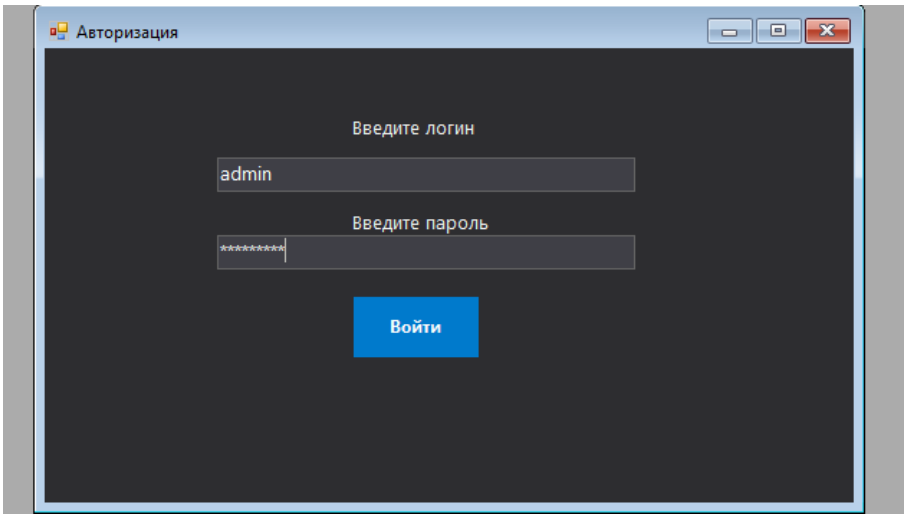

Рисунок 23 – Форма авторизации сотрудников

При вводе логина и пароля система обращается к базе данных и запрашивает список всех сотрудников. Если находится запись, содержащая логин и пароль, то происходи загрузка объекта, где также выделяется номер роли

пользователя. После чего происходит обращение к списку ролей, где определяется значение роли. Исходя из значения роли пользователю предоставляются разные полномочия. К примеру, если войти в режиме пользователя, то главная форма примет вид, изображенный на рисунке 24.

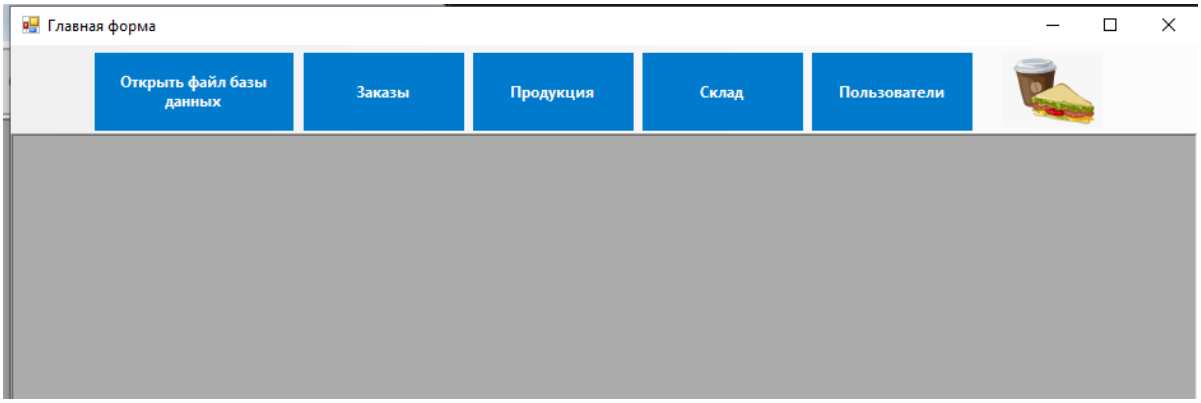

Рисунок 24 – Главная форма работы администратора

Если пользователь войдет в режиме сотрудника, то количество доступных форм для работы сократится. Окно приложения в режиме работы сотрудника изображено на рисунке 25.

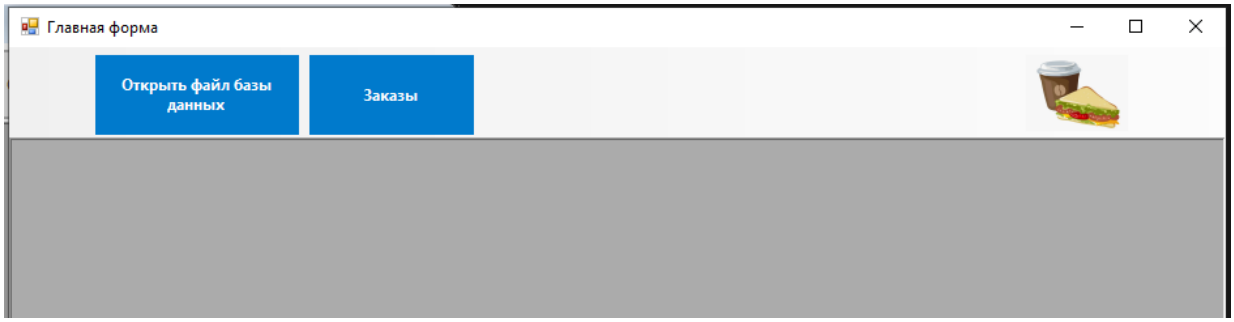

Рисунок 25 – Форма работы для сотрудника

Для создания нового заказа необходимо открыть форму с заказами, где будет отображен список сделанных заказов, представленный на рисунке 26. При открытии данной формы приложение создаст объект контекста данных, который обратится к базе данных, загрузит из нее список имеющихся заказов. Полученные записи будут сконвертированы в списки объектов заказа, из которых будет сформирован источник данных для объекта DataGridView.

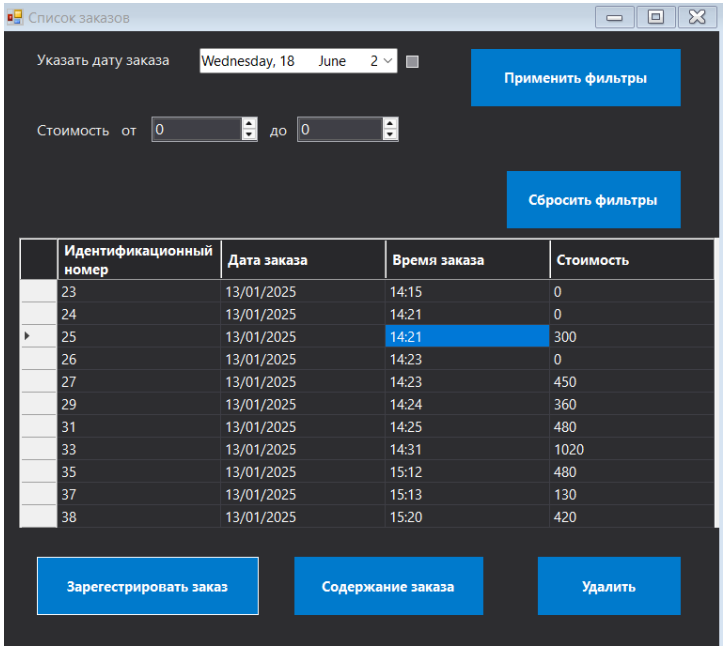

Рисунок 26 – Форма списка заказов

На форме можно просматривать имеющиеся заказы, удалять их, а также открыть форму для добавления нового заказа. Чтобы это сделать, необходимо нажать кнопку «Зарегистрировать заказ». После чего текущая форма станет недоступна, а пользователь сможет создать новую запись [13].

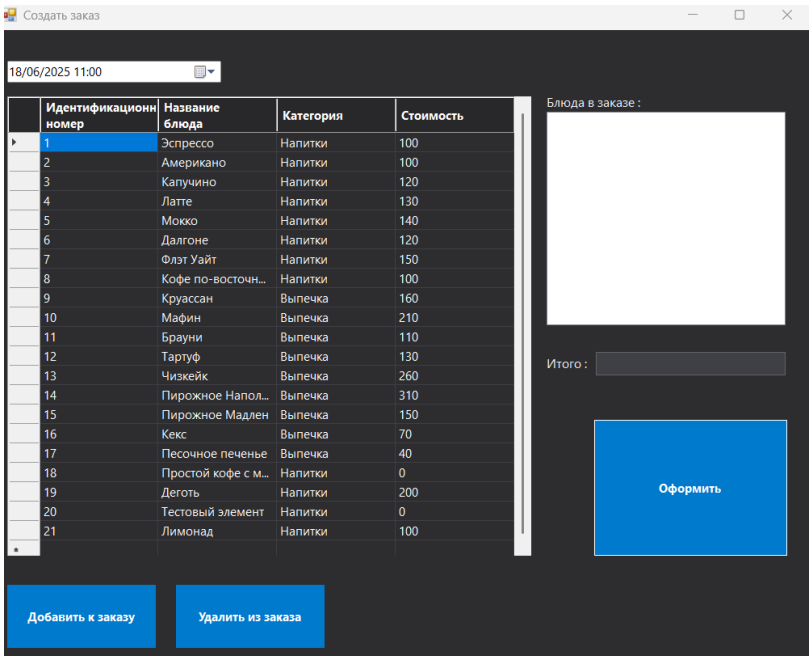

Рисунок 27 – Форма добавления нового заказа

Чтобы оформить заказ, пользователь выбирает необходимое блюдо из списка имеющихся и при выделении строки щелкает по кнопке «Добавить к

заказу», после чего появляется вспомогательное диалоговое окно с указанием количества блюд, представленное на рисунке 28.

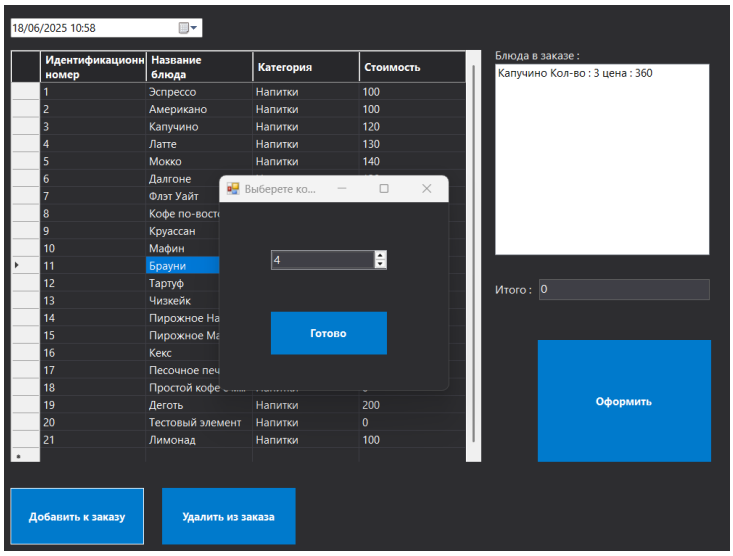

Рисунок 28 – Диалоговое окно с указанием количества блюд

После указания количества блюд, на форме заказа отображается дополнительная информация о количестве блюд, их названии и стоимости [18].

После чего пользователь может оформить заказ, так как общая сумма продуктов составила 890 рублей, то при закрытии формы добавления, заказ будет отображаться в общем списке с датой, временем и стоимостью покупки.

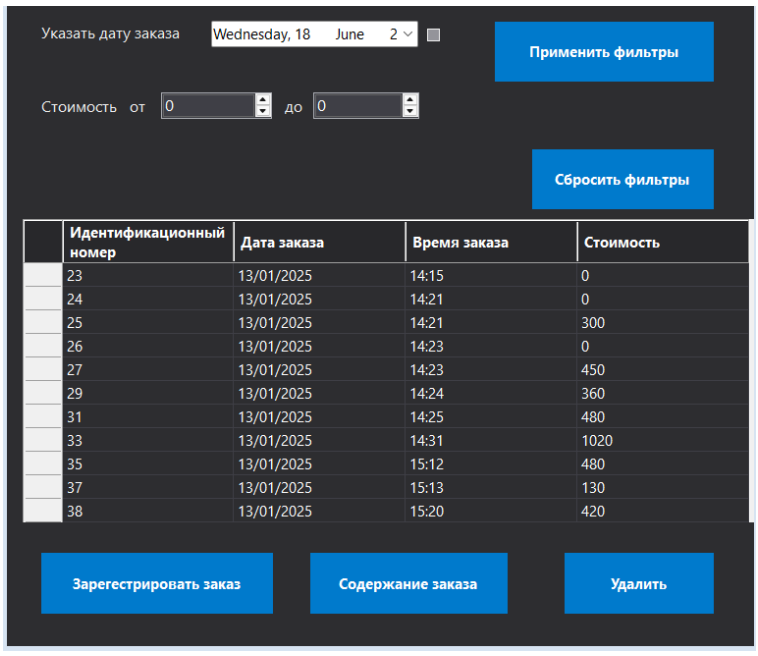

Рисунок 29 – Обновленный список заказов

Кроме простой работы со списками заказов и пользователь может добавлять новые блюда, ингредиенты и редактировать их количество на складе. Кроме этого, в приложении реализована бизнес-логика, при которой каждый заказ влияет на количество продукции на складе. Так как при выполнении заказа, тратится определенное количество продукции, то это также отражается на остатках по складу [18].

Если вызвать форму работы склада, то отобразится список имеющихся продуктов, изображённый на рисунке 30.

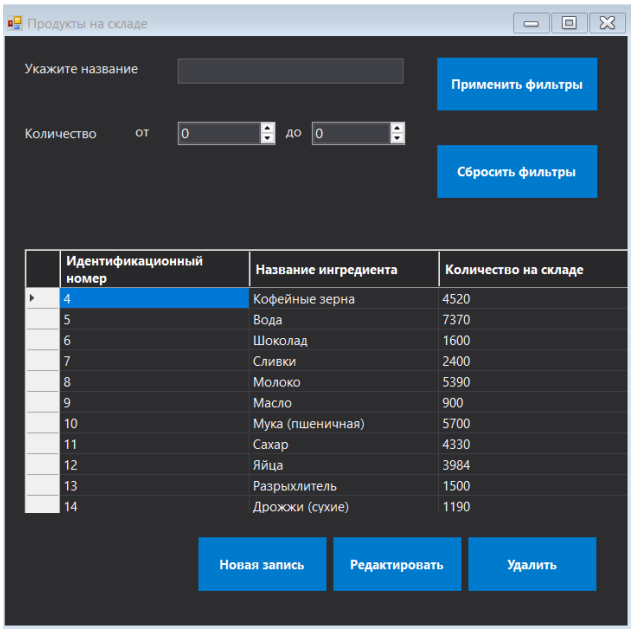

Рисунок 30 – Форма списка ингредиентов на складе

Если оформить заказ обычного кофе, где необходимы кофейные зерна и вода, то их количество изменится [10].

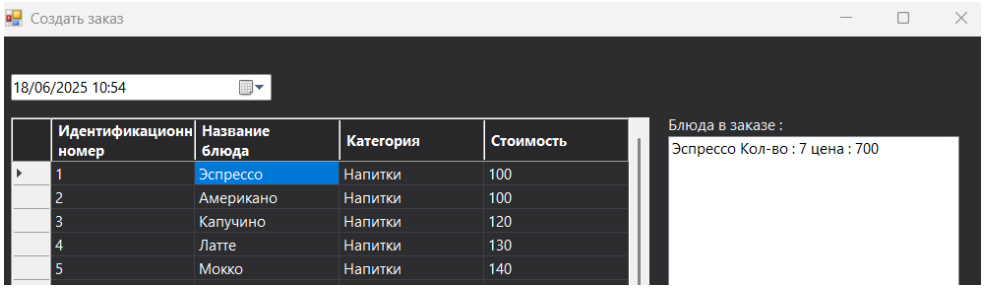

Рисунок 31 – Оформление заказа

Это подтверждается после повторного открытия формы для склада.

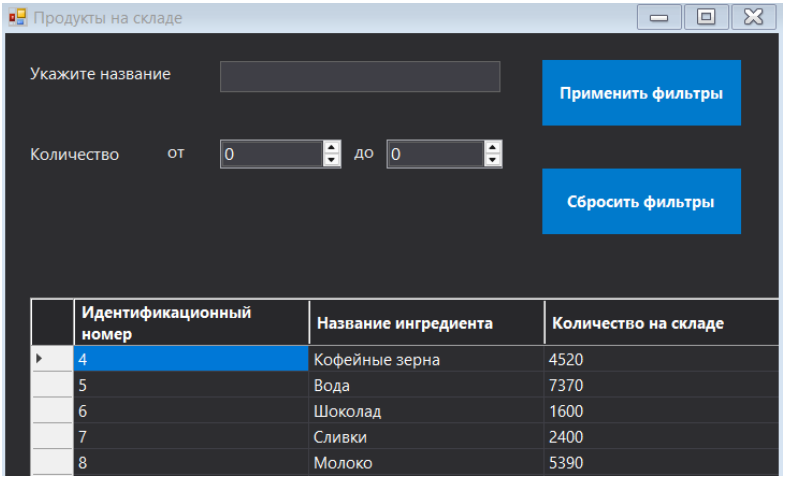

Рисунок 32 – Обновленный список продукции на складе

В отличие от режима работы для сотрудника, использование приложение со стороны администратора предусматривает добавление новых продуктов, пользователей и работу со складом. Чтобы добавить пользователя необходимо вызвать соответствующую форму и установить значение логина и пароля. Форма создания пользователя изображена на рисунке 33.

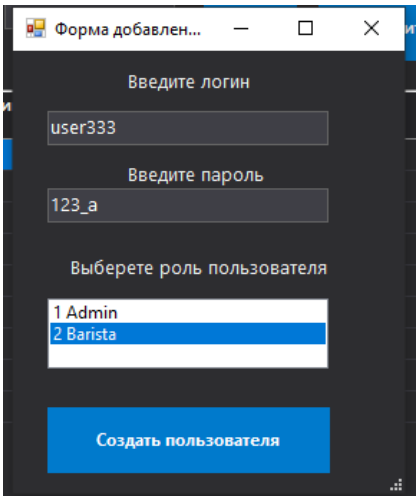

Рисунок 33 – Форма добавления нового пользователя

Из рисунка видно, что администратор вводит не только текстовые данные, но и выбирает роль пользователя. Кроме того, при добавлении записи предусмотрена логика фильтрации логина и пароля, в которой обозначено минимальное количество символов, а также содержание специальных символов, заглавных букв и цифр. Поэтому если новая запись не удовлетворяет требованиям, пользователь получает предупреждение, рисунок 34.

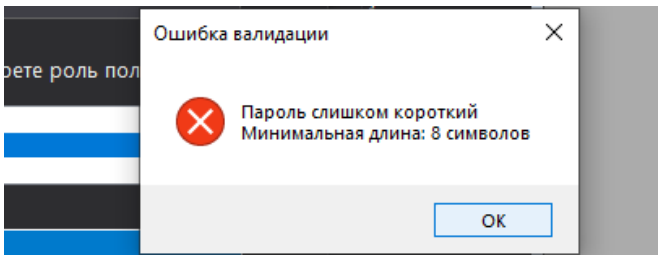

Рисунок 34 – Предупреждение пользователя о некорректном значении пароля

При корректном вводе, форма добавления пользователей закрывается, а в родительской форме отображается обновленный список, который представлен на рисунке 35.

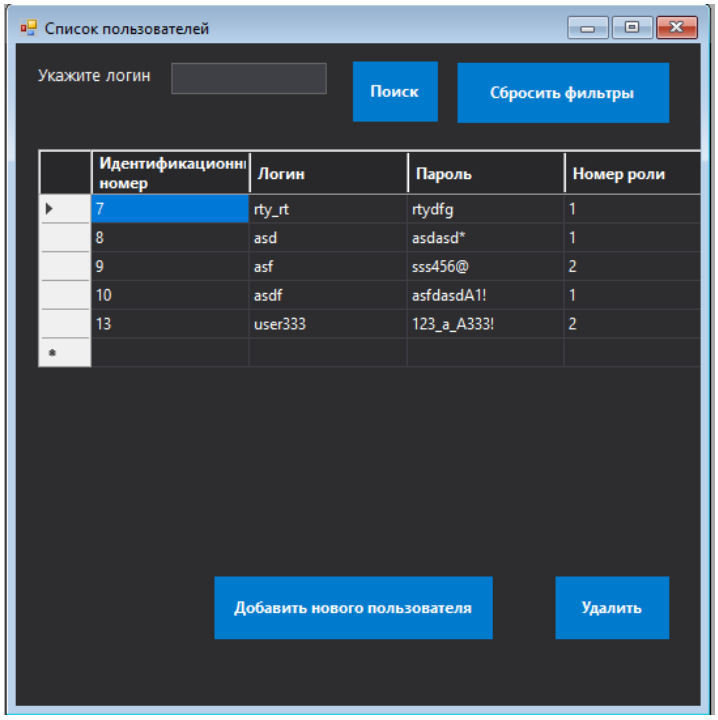

Рисунок 35 – Обновленная форма со списком добавленных сотрудников

Таким образом, каждый из блоков приложения влияет не только на базу данных, но и на само приложение в целом. Поэтому даже небольшой проект необходимо делать из небольших деталей, а не использовать монолитную конструкцию, что усложнит работу по добавлению функционала и отладке. В результате получается компактное автономное приложение, которое работает на самой распространенной системе Windows. Что делает его автономным и простым в использовании [19].

ЗАКЛЮЧЕНИЕ

В ходе работы разработана автоматизированная система для управления производственным процессом в сфере быстрого питания. Для работы задействованы инструменты, которые используют возможности языка программирования C#. Начиная от построения пользовательского интерфейса и заканчивая работой с данными. Кроме того средства разработки позволяют создать файл готового к запуску рабочего приложения там, где установлена операционная система Windows.

При реализации приложения решены следующие задачи:

– созданы модели, отражающие характеристики и поведение объектов из реальной жизни;

– реализованы контексты данных – механизмы, обеспечивающие обмен данными;

– построены представления – механизмы, обеспечивающие взаимодействие с пользователем;

– разработана структура базы данных, в которой содержится вся необходимая информация.

Кроме этого, улучшены навыки работы с различными компонентами и модулями среды разработки, это касается визуального редактора, инструментов рефакторинга и тестирования. Кроме этого, на практике подтверждена стратегия декомпозиции при проектировании приложения, которая позволила создавать и отлаживать элементы без ущерба всему приложению в целом. Несмотря на то, что приложение выполняет задачи, указанные в требованиях, в плане реализации еще есть к чему стремиться. В частности, можно улучшить пользовательский интерфейс переведя приложения из Windows Forms на WPF. Данная библиотека содержит больше графических элементов, с более гибкой системой настройки. Часть программы уже к этому готова, так как классы для работы с данными и модели данных автономны и могут использоваться в другом приложении. Данный подход хорошо применим к паттерну MVVM, который предусматривает

работа с WPF. Либо можно будет поработать в сторону наращивания функционала, либо прибегнуть к реализации приложения на мобильной платформе, что сделает приложение еще проще в работе, так как можно будет работать без ноутбука.

В любом случае рассмотренные технологии и опыт при работе с ними позволяет увеличивать скорость разработки и повышать качество проектируемых систем.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Аносов, А. Критерии выбора СУБД при создании информационных систем. / Аносов А., URL: http://www.interface.ru/home.asp?artId=2147 (Дата обращения 12.11.2024).

2. Аншина, М. Л. Технологии создания распределенных систем / А.А. Цимбал, М.Л. Аншина. – СПб.: Питер, 2003. – 576 с.

3. Балдин, К.В. Информационные системы в экономике [Текст]: Учеб. пособие / К.В. Балдин. – М.: НИЦ ИНФРА-М, 2016. – 218 с. – (Учебное пособие). (Дата обращения 25.02.2025).

4. Варфоломеева, А.О. Информационные системы предприятия [Текст]: Учебное пособие / А.О. Варфоломеева, А.В. Коряковский, В.П. Романов. – М.: НИЦ ИНФРА-М, 2017. – 283 с. – (Учебное пособие). (Дата обращения 18.12.2024).

5. Гарсия-Молина, Системы баз данных. Полный курс.: Пер с англ. Гарсия-Молина Г., Ульман Дж., Уидом Дж. – М.: ООО «И. Д Вильямс», 2017. – 1088с. (Дата обращения 05.01.2025).

6. Голицына О. Л. Программирование на языках высокого уровня: Учебное пособие / О.Л. Голицына, И.И. Попов. – М.: Форум, 2016. – 496 с. [Электронный ресурс]. URL: http://znanium.com/bookread.php?book=139428. (Дата обращения 23.12.2024).

7. Граничин О.Н. Информационные технологии в управлении [Электронный ресурс]: Учебное пособие / В.И. Кияев, О.Н. Граничин – Электрон. текстовые данные. – М.: Национальный Открытый Университет Информационных Технологий (ИНТУИТ), 2016. – 401 с. – URL: https://intuit.ru/studies/courses/1055/271/lecture/6876?page=5 – Загл. с экрана. (Дата обращения 22.12.2024).

8. Джон, С. C# для профессионалов. Тонкости программирования / С. Джон – М.: Вильямс, 2014. – 408 с.

9. Диаграмма состоянии (UML) // Википедия URL: https://ru.wikipedia.org/wiki/Диаграмма_состояний_(UML) (Дата обращения: 06.01.2025).

10. Дунаев, В. В. Базы данных. Язык SQL для студента / В.В. Дунаев – М.: БХВ-Петербург, 2017. – 288 с.

11. Жданов, С.А. Информационные системы [Электронный ресурс]: учебное пособие для студентов учреждений высшего образования / С.А. Жданов, М. Л. Соболев, А. С. Алфимова – Электрон. текстовые данные. – М.: ООО «Прометей», 2017. – 302 с. – URL: https://avidreaders.ru/readbook/informacionnye-sistemy1.html – Загл. с экрана. (Дата обращения 03.01.2025).

12. Ипатова, Э. Р. Методологии и технологии системного проектирования информационных систем [Текст]: Учебник: моногр. / Э.Р. Ипатова. – М.: Флинта, 2016. – 300 с. – (Учебник).

13. Кравченко, Т. К. Системы поддержки принятия решений [Электронный ресурс]: учебник и практикум для вузов / Т. К. Кравченко, Д. В. Исаев. – Электрон. текстовые данные – Москва: Издательство Юрайт, 2020. – 292 с. – URL: https://urait.ru/book/sistemy-podderzhki-prinyatiya-resheniy-450834 – Загл. с экрана (Дата обращения 12.11.2024).

14. Кузин, А. В. Базы данных. / А. В. Кузин, С. В. Левонисова – 5-е изд. – М.: Академия, 2012. – 320 с.

15. Маркин, А. В. Программирование на sql. Учебное пособие для СПО. / А. В. Маркин – М.: Юрайт, 2019. – 434 с.

16. Мезенцев, К. Н. Автоматизированные информационные системы: Учебник для студентов учреждений среднего профессионального образования. / К. Н. Мезенцев – М.: ИЦ Академия, 2017. – 176 с.

17. Мезенцев, К.Н. Автоматизированные информационные системы [Текст]: Учебник для студентов учреждений среднего профессионального образования / К.Н. Мезенцев. – М.: ИЦ Академия, 2017. – 176 с. – (Учебник).

18. Одинцов, Б.Е. Информационные системы управления эффективностью бизнеса [Текст]: Учебник и практикум / Б.Е. Одинцов. – М.: Юрайт, 2017. – 208 с.

19. Организационная диаграмма // businessstudio URL: https://www.businessstudio.ru/wiki/docs/current/doku.php/ru/manual/org_struct/working_org_struct (Дата обращения: 26.11.2024)

20. Паршин, К. А. Методы и средства проектирования информационных систем: учеб. – метод. Пособие / К. А. Паршин. – Екатеринбург: УрГУПС, 2018. – 129 с.

21. Перчини [Электронный ресурс]: – URL: http://perchini.ru/ – Загл. с экрана. (Дата обращения 23.12.2024).

22. Принципы разработки пользовательских интерфейсов [Электронный ресурс] – URL: https://medium.com/начинающемуux-дизайнеру/ – Загл. с экрана. (Дата обращения 26.12.2024).

23. Рыжко, А.Л. Информационные системы управления производственной компанией: Учебник / А.Л. Рыжко, А.И. Рыбников, Н.А. Рыжко. – М.: Юрайт, 2016. – 356 с.

24. СБИС Pesto [Электронный ресурс] – URL: https://sbis.ru/presto – Загл. с экрана. (Дата обращения 19.12.2024).

25. Системы управления базами данных. Типы баз данных // Инфо-урок URL: https://infourok.ru/subd-sistemi-upravleniya-bazami-dannih-2114145.html (Дата обращения: 03.12.2024).

26. Советов, Б.Я. Базы данных: теория и практика: Учебник для бакалавров / Б.Я. Советов, В.В. Цехановский, В.Д. Чертовской. – М.: Юрайт, 2013. – 463 с.

27. Таненбаум, Э. Компьютерные сети 5-е изд. / Э. Таненбаум, Д. Уэзеролл. – СПб.: Питер, 2022 – 960с.

28. Титоренко, Г.А. Информационные системы в экономике [Электронный ресурс]: учебник / ред.: Г.А. Титоренко. – 2-е изд., перераб. и доп. – Электрон. текстовые данные. – М.: ЮНИТИ-ДАНА, 2016. – 464 с.: ил. – ISBN 978-5-238-

01167-7. – URL: https://lib.rucont.ru/efd/351822. – Загл. с экрана. (Дата обращения 01.12.2024).

29. Троелсен, Э. Язык программирования C# 6.0 и платформа .NET 4.6. – М.:Вильямс. / Э. Троелсен, Ф. Джепикс. 2016. – 1440 с.

30. Федорова, Г. Н. Информационные системы: Учебник / Г. Н. Федорова – М.: Academia, 2018. – 384 с.

31. Фуфаев, Э.В. Базы данных: Учебное пособие для студентов учреждений среднего профессионального образования / Э.В. Фуфаев, Д.Э. Фуфаев. – М.: ИЦ Академия, 2012. – 320 с.

32. Хабр. Форум помощи разработки – статья о расчетах в MS Project. URL: https://habr.com/ru/post/218885/. Текст электронный. (дата обращения 21.10.2024)

33. Халимон, В. И. Базы данных: учебное пособие / В. И. Халимон, Г. А. Мамаева, А. Ю. Рогов, В. Н. Чепикова – СПб.: СПбГТИ(ТУ), 2017. – 118 с.

34. Халин, В. Г. Системы поддержки принятия решений [Текст]: учебник и практикум для бакалавриата и магистратуры / В. Г. Халин [и др.]; под редакцией В. Г. Халина, Г. В. Черновой. – Москва: Издательство Юрайт, 2019. – 494 с.

35. Хортсман, К.С Java. Библиотека профессионала, том 1. Основы 11-е изд.: Пер. с англ. – СПб.: ООО «Диалектика», 2019. – 864с.

36. Чистов, Д. В. Информационные системы в экономике: Учебное пособие / Д. В. Чистов – М.: Инфра-М, 2019. – 248 с.

37. CRM для ресторанов [Электронный ресурсс] – URL: https://crmindex.ru/for/horeca – Загл. с экрана (Дата обращения 17.02.2025).

38. METANIT.COM Сайт о программировании C# URL: https://metanit.com/sharp/ – (Дата обращения 11.02.2025).

39. R-Keeper [Электронный ресурс] – URL: https://rkeeper.ru/products/r_keeper/ – Загл. с экрана (Дата обращения 26.02.2025).

40. SQLAlcemy. Docs: официальный сайт. – URL: https://www.sqlalchemy.org (Дата обращения 24.02.2025).

Приложение А.

Листинг кода функциональной части системы

Классы моделей данных

```
public class Category
{
   [Display(Name = "Идентификационный номер")]
   public int Id { get; set; }
   [Display(Name = "Наименование категории")]
   public string Name { get; set; }
}

public class Ingredient
{
   [Display(Name = "Идентификационный номер")]
   public int Id { get; set; }
   [Display(Name = "Название ингредиента")]

   public string IngredientName { get; set; }
   [Display(Name = "Количество на складе")]

   public double QuantityInStock { get; set; }

}

public class Order
 {
    [Display(Name = "Идентификационный номер")]
    public int Id { get; set; }

    [Display(Name = "Дата заказа")]
    public DateTime OrderDate { get; set; }
    [Display(Name = "Время заказа")]
    public TimeSpan OrderTime { get; set; }
    [Display(Name = "Стоимость")]
    public double TotalCost { get; set; }
 }

public class OrderItem
{
   public int Id { get; set; }
   [Display(Name = "Идентификационный номер заказа")]

   public int OrderId { get; set; }
   [Display(Name = "Идентификационный номер продукта")]

   public int ProductId { get; set; }
   [Display(Name = "Количество порицй")]

   public int Quantity { get; set; }
   [Display(Name = "Стоимость")]
   public double Cost { get; set; }
```

```
}

public class Product
 {
    [Display(Name = "Идентификационный номер")]
    public int Id { get; set; }
    [Display(Name = "Название")]
    public string Name { get; set; }
    [Display(Name = "Описание")]
    public string Description { get; set; }
    [Display(Name = "Категория")]
    public int Id_category { get; set; }
    [Display(Name = "Стоимость")]
    public double Price { get; set; }
 }

public class Recipe
 {
    [Display(Name = "Идентификационный номер")]
    public int Id { get; set; }
    [Display(Name = "Номер продукта")]

    public int ProductId { get; set; }
    [Display(Name = "Инструкция")]
    public string Instructions { get; set; }
 }

public class RecipeIngredient
{
   [Display(Name = "Идентификационный номер")]
   public int Id { get; set; }
   [Display(Name = "Номер рецепта")]

   public int RecipeId { get; set; }
   [Display(Name = "Номер ингредиента")]

   public int IngredientId { get; set; }
   [Display(Name = "Необходимое количество")]
   public double Quantity { get; set; }
}
```

```
public class CategoryContext
{
   private string connectionString;



   public CategoryContext(string dbPath)
   {
      connectionString = dbPath;
   }

   public List<Category> GetAllCategories()
   {
      List<Category> categories = new List<Category>();

      using (SQLiteConnection connection = new SQLiteConnection(connectionString))
      {
         connection.Open();

         string query = "SELECT * FROM categories";

         using (SQLiteCommand command = new SQLiteCommand(query, connection))
         {
            using (SQLiteDataReader reader = command.ExecuteReader())
            {
               while (reader.Read())
               {
                  Category category = new Category
                  {
                     Id = Convert.ToInt32(reader["id"]),
                     Name = reader["name"].ToString()
                  };
                  categories.Add(category);
               }
            }
         }
      }

      return categories;
   }

   public void AddCategory(Category category)
   {
      using (SQLiteConnection connection = new SQLiteConnection(connectionString))
      {
         connection.Open();

         string query = "INSERT INTO categories (name) VALUES (@Name)";

         using (SQLiteCommand command = new SQLiteCommand(query, connection))
         {
            command.Parameters.AddWithValue("@Name", category.Name);
```

```
            command.ExecuteNonQuery();
        }
    }
}

public void UpdateCategory(Category category)
{
    using (SQLiteConnection connection = new SQLiteConnection(connectionString))
    {
        connection.Open();

        string query = "UPDATE categories SET name = @Name WHERE id = @Id";

        using (SQLiteCommand command = new SQLiteCommand(query, connection))
        {
            command.Parameters.AddWithValue("@Name", category.Name);
            command.Parameters.AddWithValue("@Id", category.Id);
            command.ExecuteNonQuery();
        }
    }
}

public void DeleteCategory(int id)
{
    using (SQLiteConnection connection = new SQLiteConnection(connectionString))
    {
        connection.Open();

        string query = "DELETE FROM categories WHERE id = @Id";

        using (SQLiteCommand command = new SQLiteCommand(query, connection))
        {
            command.Parameters.AddWithValue("@Id", id);
            command.ExecuteNonQuery();
        }
    }
}
}



public class IngredientDbContext
{
    private string _connectionString;



    public IngredientDbContext(string connectionString)
    {
        _connectionString = connectionString;
        using (var connection = new SQLiteConnection(_connectionString))
        {
            connection.Open();
            var createTableCommand = connection.CreateCommand();
```

```
            createTableCommand.CommandText = @"
                CREATE TABLE IF NOT EXISTS ingredients (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    ingredient_name TEXT NOT NULL UNIQUE,
                    quantity_in_stock REAL
                )";
            createTableCommand.ExecuteNonQuery();
        }
    }

    public void AddIngredient(Ingredient ingredient)
    {
        using (var connection = new SQLiteConnection(_connectionString))
        {
            connection.Open();
            var addIngredientCommand = connection.CreateCommand();
            addIngredientCommand.CommandText = @"
        INSERT INTO ingredients (ingredient_name, quantity_in_stock)
        VALUES ($ingredientName, $quantityInStock)";
            addIngredientCommand.Parameters.AddWithValue("$ingredientName",
ingredient.IngredientName);
            addIngredientCommand.Parameters.AddWithValue("$quantityInStock",
ingredient.QuantityInStock);
            addIngredientCommand.ExecuteNonQuery();
        }
    }
    public Ingredient GetIngredient(int ID)
    {
        var sql = "SELECT * FROM ingredients WHERE id = @Id";
        using (var connection = new SQLiteConnection(_connectionString))
        using (var command = new SQLiteCommand(sql, connection))
        {
            command.Parameters.AddWithValue("@Id", ID);
            connection.Open();
            using (var reader = command.ExecuteReader())
            {
                if (reader.Read())
                {
                    return new Ingredient
                    {
                        Id = reader.GetInt32(0),
                        IngredientName = reader.GetString(1),
                        QuantityInStock = reader.GetDouble(2)
                    };
                }
            }
        }
        return null;
    }
    public Product ReadProduct(int id)
    {
        var sql = "SELECT * FROM products WHERE id = @Id";
        using (var connection = new SQLiteConnection(_connectionString))
        using (var command = new SQLiteCommand(sql, connection))
```

```
    {
        command.Parameters.AddWithValue("@Id", id);
        connection.Open();
        using (var reader = command.ExecuteReader())
        {
            if (reader.Read())
            {
                return new Product
                {
                    Id = reader.GetInt32(0),
                    Name = reader.GetString(1),
                    Description = reader.IsDBNull(2) ? null : reader.GetString(2),
                    Id_category = reader.GetInt32(3),
                    Price = reader.GetDouble(4)
                };
            }
        }
    }
    return null; // Если элемент не найден
}

public List<Ingredient> GetIAllngredients()
{
    List<Ingredient> ingredients = new List<Ingredient>();
    using (var connection = new SQLiteConnection(_connectionString))
    {
        connection.Open();
        var getIngredientsCommand = connection.CreateCommand();
        getIngredientsCommand.CommandText = "SELECT * FROM ingredients";
        var reader = getIngredientsCommand.ExecuteReader();
        while (reader.Read())
        {
            Ingredient ingredient = new Ingredient
            {
                Id = reader.GetInt32(0),
                IngredientName = reader.GetString(1),
                QuantityInStock = reader.GetDouble(2),
            };
            ingredients.Add(ingredient);
        }
    }
    return ingredients;
}

public void UpdateIngredient(Ingredient ingredient)
{
    using (var connection = new SQLiteConnection(_connectionString))
    {
        connection.Open();
        var updateIngredientCommand = connection.CreateCommand();
        updateIngredientCommand.CommandText = @"
    UPDATE ingredients
    SET ingredient_name = $ingredientName, quantity_in_stock = $quantityInStock
    WHERE id = $id";
```

```
            updateIngredientCommand.Parameters.AddWithValue("$ingredientName",
ingredient.IngredientName);
            updateIngredientCommand.Parameters.AddWithValue("$quantityInStock",
ingredient.QuantityInStock);
            updateIngredientCommand.Parameters.AddWithValue("$id", ingredient.Id);
            updateIngredientCommand.ExecuteNonQuery();
        }
    }

    public void DeleteIngredient(int id)
    {
        using (var connection = new SQLiteConnection(_connectionString))
        {
            connection.Open();
            var deleteIngredientCommand = connection.CreateCommand();
            deleteIngredientCommand.CommandText = "DELETE FROM ingredients WHERE id = $id";
            deleteIngredientCommand.Parameters.AddWithValue("$id", id);
            deleteIngredientCommand.ExecuteNonQuery();
        }
    }
}

public class ProductContext
{
    string _connectionString;

    public ProductContext(string connectionString)
    {
        _connectionString = connectionString;
    }

    public void CreateProduct(Product product)
    {
        var sql = "INSERT INTO products (name, description, id_category,price) VALUES (@ProductTypeName,
@Description, @Id_category, @Price)";
        try
        {
            using (var connection = new SQLiteConnection(_connectionString))
            using (var command = new SQLiteCommand(sql, connection))
            {
                command.Parameters.AddWithValue("@ProductTypeName", product.Name);
                command.Parameters.AddWithValue("@Description", product.Description);
                command.Parameters.AddWithValue("@Id_category", product.Id_category);
                command.Parameters.AddWithValue("@Price", product.Price);
                connection.Open();
                command.ExecuteNonQuery();
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
```

```
    public void DeleteProduct(int id)
    {
        var sql = "DELETE FROM products WHERE id = @Id";
        try
        {
            using (var connection = new SQLiteConnection(_connectionString))
            using (var command = new SQLiteCommand(sql, connection))
            {
                command.Parameters.AddWithValue("@Id", id);

                connection.Open();
                command.ExecuteNonQuery();
            }
        }
        catch (Exception e)
        {
            MessageBox.Show("При работе с СУБД \n возникли ошибки: " + e.Message);
        }
    }

    public Product ReadProduct(int id)
    {
        var sql = "SELECT * FROM products WHERE id = @Id";
        using (var connection = new SQLiteConnection(_connectionString))
        using (var command = new SQLiteCommand(sql, connection))
        {
            command.Parameters.AddWithValue("@Id", id);
            connection.Open();
            using (var reader = command.ExecuteReader())
            {
                if (reader.Read())
                {
                    return new Product
                    {
                        Id = reader.GetInt32(0),
                        Name = reader.GetString(1),
                        Description = reader.IsDBNull(2) ? null : reader.GetString(2),
                        Id_category = reader.GetInt32(3),
                        Price = reader.GetDouble(4)
                    };
                }
            }
        }
        return null;
    }

    public void UpdateProduct(Product prodType)
    {
        var sql = "UPDATE products SET name = @ProductTypeName, description = @Description, id_category
= @Id_category, price = @Price WHERE id = @Id";
        using (var connection = new SQLiteConnection(_connectionString))
        using (var command = new SQLiteCommand(sql, connection))
        {
            command.Parameters.AddWithValue("@Id", prodType.Id);
```

```
            command.Parameters.AddWithValue("@PastryTypeName", prodType.Name);
            command.Parameters.AddWithValue("@Description", prodType.Description);
            command.Parameters.AddWithValue("@Id_category", prodType.Id_category);
            command.Parameters.AddWithValue("@Price", prodType.Price);
            command.ExecuteNonQuery();
        }
    }



    public List<Product> GetAllProduct()
    {
        var productTypes = new List<Product>();
        var sql = "SELECT * FROM products";
        try
        {
            using (var connection = new SQLiteConnection(_connectionString))
            using (var command = new SQLiteCommand(sql, connection))
            {
                connection.Open();
                using (var reader = command.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        var productType = new Product
                        {
                            Id = reader.GetInt32(0),
                            Name = reader.GetString(1),
                            Description = reader.IsDBNull(2) ? null : reader.GetString(2),
                            Id_category = reader.GetInt32(3),
                            Price = reader.GetDouble(4)
                        };
                        productTypes.Add(productType);
                    }
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Ошибка при получении данных: " + ex.Message);
        }
        return productTypes;
    }

}
```

```
public partial class Form1 : Form
 {
    string conn;
    public Form1()
    {
       InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
       OpenFileDialog openFileDialog = new OpenFileDialog();
       openFileDialog.Filter = "SQLite Database Files (*.db;*.sqlite)|*.db;*.sqlite|All Files (*.*)|*.*";
       openFileDialog.Title = "Select SQLite Database File";

       if (openFileDialog.ShowDialog() == DialogResult.OK)
       {
          conn = "Data Source="+ openFileDialog.FileName;
       }

    }

    private void button2_Click(object sender, EventArgs e)
    {
       if (conn != null)
       {
          Views.Orders.OrdersFormList list = new Views.Orders.OrdersFormList(conn);
          list.ShowDialog();
       }
    }

    private void button3_Click(object sender, EventArgs e)
    {
       if (conn != null)
       {
          ProductTypeFormList list = new ProductTypeFormList(conn);
          list.ShowDialog();
       }
    }

    private void button4_Click(object sender, EventArgs e)
    {
       IngredientListForm ingredientListForm = new IngredientListForm(conn);
       ingredientListForm.ShowDialog();
    }
 }

public partial class Create : Form
{
   OrdersFormList parent;
   public int cnt;
   public string cur_index;
   string connection_string;
```

```
    Order order;
    List<Product> products;
    List<Category> categories;
    List<ProductViewModel> productViews;
    List<OrderItem> selectedItems;


    public Create(OrdersFormList p, string c)
    {
        InitializeComponent();
        connection_string = c;
        parent = p;
        order = new Order();
        order.OrderDate = dateTimePicker1.Value.Date;
        order.OrderTime = dateTimePicker1.Value.TimeOfDay;
        OrdersContext ordersContext = new OrdersContext(connection_string);
        ordersContext.Create(order);
        int Id = ordersContext.GetAllOrders().FirstOrDefault(n => n.OrderTime == order.OrderTime &&
n.OrderDate == order.OrderDate).Id;
        order.Id = Id;
        selectedItems = new List<OrderItem>();
        dateTimePicker1.Format = DateTimePickerFormat.Custom;
        dateTimePicker1.CustomFormat = "dd/MM/yyyy HH:mm";
        Init();

    }

    private void button1_Click(object sender, EventArgs e)
    {
        RecipeContext recipeContext = new RecipeContext(connection_string);
        RecipeIngredientDbContext recipeIngredientDbContext = new
RecipeIngredientDbContext(connection_string);
        IngredientDbContext ingredientDbContext = new IngredientDbContext(connection_string);
        OrderItemContext orderItemContext = new OrderItemContext(connection_string);
        ProductContext productContext = new ProductContext(connection_string);


        foreach(OrderItem item in selectedItems)
        {
            int count_items = 1;
            orderItemContext.CreateOrderItem(item);
            count_items = item.Quantity;
            Product product = productContext.ReadProduct(item.ProductId);
            Recipe recipe = recipeContext.GetAllRecipes().FirstOrDefault(n => n.ProductId == product.Id);
            var recipeIngredients = recipeIngredientDbContext.GetAllRecipeIngredients().Where(n => n.RecipeId
== recipe.Id).ToList();
            foreach (RecipeIngredient recipeIngredient in recipeIngredients)
            {
                Ingredient ingredient = ingredientDbContext.GetIngredient(recipeIngredient.IngredientId);
                ingredient.QuantityInStock= ingredient.QuantityInStock - recipeIngredient.Quantity*count_items;
                ingredientDbContext.UpdateIngredient(ingredient);
            }
        }
```

```
    double ttcst = 0;
    foreach (OrderItem orderItem in selectedItems)
    {
        ttcst += orderItem.Cost;
    }
    order.TotalCost = ttcst;
    OrdersContext ordersContext = new OrdersContext(connection_string);
    ordersContext.Update(order);
    parent.Init();
    this.Close();

}

private void addItmOrderBtn_Click(object sender, EventArgs e)
{
    int num = 0;
    bool ok = int.TryParse(cur_index, out num);
    if (ok)
    {
        CountForm countForm = new CountForm(this);
        countForm.ShowDialog();
        OrderItem item = new OrderItem();
        var prod = products.FirstOrDefault(p => p.Id == num);
        item.OrderId = order.Id;
        item.Quantity = cnt;
        item.ProductId = num;
        item.Cost = Math.Round(cnt*prod.Price,2);
        string info = prod.Name + " Кол-во : " + cnt + " цена : " + item.Cost;
        listBox1.Items.Add(info);
        selectedItems.Add(item);

        textBox1.Text = "";
        double ttcst = 0;
        foreach(OrderItem orderItem in selectedItems)
        {
            ttcst += orderItem.Cost;
        }
        textBox1.Text  = ttcst.ToString();

    }
}
void Init()
{
    CategoryContext categoryContext = new CategoryContext(connection_string);
    ProductContext productContext = new ProductContext(connection_string);
    productViews = new List<ProductViewModel>();
    products = productContext.GetAllProduct();
    categories = categoryContext.GetAllCategories();
    foreach (Product product in products)
    {
      string cat_name = categories.First(n => n.Id == product.Id_category).Name;
       ProductViewModel productViewModel = new ProductViewModel();
       productViewModel.Id = product.Id;
```

```
            productViewModel.Name = product.Name;
            productViewModel.Price = product.Price;
            productViewModel.Category = cat_name;
            productViews.Add(productViewModel);
        }
        dataGridView1.DataSource = null;

        DataTable dataTable = new DataTable();
        var properties = typeof(ProductViewModel).GetProperties();

        foreach (var prop in properties)
        {
            var displayAttribute = prop.GetCustomAttribute<DisplayAttribute>();
            string columnName = displayAttribute != null ? displayAttribute.Name : prop.Name;
            dataTable.Columns.Add(columnName, prop.PropertyType);
        }

        foreach (var item in productViews)
        {
            var row = new object[properties.Length];
            for (int i = 0; i < properties.Length; i++)
            {
                row[i] = properties[i].GetValue(item);
            }
            dataTable.Rows.Add(row);
        }

        dataGridView1.DataSource = dataTable;
    }

    private void dataGridView1_MouseClick(object sender, MouseEventArgs e)
    {

    }

    private void dataGridView1_CellMouseClick(object sender, DataGridViewCellMouseEventArgs e)
    {
        if (e.RowIndex >= 0)
        {
            var firstCellValue = dataGridView1.Rows[e.RowIndex].Cells[0].Value;
            if (firstCellValue != null)
            {
                cur_index = firstCellValue?.ToString();
            }
        }
    }
}



public partial class ProductTypeFormList : Form
 {
    public string cur_index;
    public string connection_string;
```

```
public ProductContext context;
List<Product> pastryTypes;

public ProductTypeFormList()
{
   InitializeComponent();
}
public ProductTypeFormList(string conn)
{
   connection_string = conn;
   InitializeComponent();
   context = new ProductContext(connection_string);
   pastryTypes = context.GetAllProduct();
   Init();

}
public void Init()
{
   dataGridView1.DataSource = null;
   pastryTypes = context.GetAllProduct();


   DataTable dataTable = new DataTable();
   var properties = typeof(Product).GetProperties();

   foreach (var prop in properties)
   {
      var displayAttribute = prop.GetCustomAttribute<DisplayAttribute>();
      string columnName = displayAttribute != null ? displayAttribute.Name : prop.Name;
      dataTable.Columns.Add(columnName, prop.PropertyType);
   }

   foreach (var item in pastryTypes)
   {
      var row = new object[properties.Length];
      for (int i = 0; i < properties.Length; i++)
      {
         row[i] = properties[i].GetValue(item);
      }
      dataTable.Rows.Add(row);
   }

   // Установка DataSource
   dataGridView1.DataSource = dataTable;
}

private void button1_Click(object sender, EventArgs e)
{
   int ID = int.Parse(cur_index);
   context.DeleteProduct(ID);
   Init();
}

private void dataGridView1_Click(object sender, EventArgs e)
```

```
    {

    }


    private void dataGridView1_CellMouseClick(object sender, DataGridViewCellMouseEventArgs e)
    {
        if (e.RowIndex >= 0)
        {
            var firstCellValue = dataGridView1.Rows[e.RowIndex].Cells[0].Value;
            if (firstCellValue != null)
            {
                cur_index = firstCellValue?.ToString();
            }
        }

    }

    private void button2_Click(object sender, EventArgs e)
    {
        Views.ProductTypes.ProductTypeCreateForm create = new ProductTypeCreateForm(this);
        create.ShowDialog();
    }

    private void button3_Click(object sender, EventArgs e)
    {
        int num = 0;
        bool ok = int.TryParse(cur_index, out num);
        if (ok)
        {
            ProductTypes.ProductTypeEditForm pastryTypeEditForm = new ProductTypeEditForm(this);
            pastryTypeEditForm.ShowDialog();
        }
    }
}
```

Приложение Б.

Листинг кода шаблонов

```
BEGIN TRANSACTION;
CREATE TABLE IF NOT EXISTS "categories" (
	"id"	INTEGER NOT NULL,
	"name" TEXT NOT NULL,
	PRIMARY KEY("id" AUTOINCREMENT)
);
CREATE TABLE IF NOT EXISTS "ingredients" (
	"id"	INTEGER,
	"ingredient_name"	TEXT NOT NULL UNIQUE,
	"quantity_in_stock"	REAL NOT NULL,
	PRIMARY KEY("id" AUTOINCREMENT)
);
CREATE TABLE IF NOT EXISTS "order_items" (
	"id"	INTEGER,
	"order_id"	INTEGER NOT NULL,
	"product_type_id"	INTEGER NOT NULL,
	"quantity"	INTEGER NOT NULL,
	"cost"	REAL,
	PRIMARY KEY("id" AUTOINCREMENT),
	FOREIGN KEY("order_id") REFERENCES "orders"("id") ON DELETE CASCADE,
	FOREIGN KEY("product_type_id") REFERENCES "products"("id") ON DELETE CASCADE
);
CREATE TABLE IF NOT EXISTS "orders" (
	"id"	INTEGER,
	"order_date"	TEXT NOT NULL DEFAULT CURRENT_DATE,
	"order_time"	TEXT NOT NULL DEFAULT CURRENT_TIME,
```

```
	"total_cost"	REAL,
	PRIMARY KEY("id" AUTOINCREMENT)
);
CREATE TABLE IF NOT EXISTS "products" (
	"id"	INTEGER NOT NULL,
	"name"	TEXT NOT NULL,
	"description"	TEXT NOT NULL,
	"id_category"	INTEGER NOT NULL,
	"price"	REAL,
	PRIMARY KEY("id" AUTOINCREMENT),
	FOREIGN KEY("id_category") REFERENCES "categories"("id") ON DELETE CASCADE
);
CREATE TABLE IF NOT EXISTS "recipe_ingredients" (
	"id"	INTEGER NOT NULL,
	"recipe_id"	INTEGER NOT NULL,
	"ingredient_id"	INTEGER NOT NULL,
	"quantity"	REAL NOT NULL,
	PRIMARY KEY("id" AUTOINCREMENT),
	FOREIGN KEY("ingredient_id") REFERENCES "ingredients"("id") ON DELETE CASCADE,
	FOREIGN KEY("recipe_id") REFERENCES "recipes"("id") ON DELETE CASCADE
);
CREATE TABLE IF NOT EXISTS "recipes" (
	"id"	INTEGER,
	"product_id"	INTEGER NOT NULL,
	"instructions"	TEXT NOT NULL,
	PRIMARY KEY("id" AUTOINCREMENT),
	FOREIGN KEY("product_id") REFERENCES "products"("id") ON DELETE CASCADE
);
CREATE TABLE IF NOT EXISTS "roles" (
```

```
	"id"	INTEGER NOT NULL,
	"name"	TEXT NOT NULL,
	PRIMARY KEY("id" AUTOINCREMENT)
);
CREATE TABLE IF NOT EXISTS "users" (
	"id"	INTEGER NOT NULL,
	"login"	TEXT NOT NULL,
	"password"	TEXT NOT NULL,
	"id_role"	INTEGER NOT NULL,
	PRIMARY KEY("id" AUTOINCREMENT),
	FOREIGN KEY("id_role") REFERENCES "roles"("id") ON DELETE CASCADE
);
INSERT INTO "categories" VALUES (1,'Кофе');
INSERT INTO "categories" VALUES (2,'Выпечка');
INSERT INTO "ingredients" VALUES (4,'Кофейные зерна',4570.0);
INSERT INTO "ingredients" VALUES (5,'Вода',7490.0);
INSERT INTO "ingredients" VALUES (6,'Шоколад',1800.0);
INSERT INTO "ingredients" VALUES (7,'Сливки',2500.0);
INSERT INTO "ingredients" VALUES (8,'Молоко',5390.0);
INSERT INTO "ingredients" VALUES (9,'Масло',900.0);
INSERT INTO "ingredients" VALUES (10,'Мука (пшеничная)',5700.0);
INSERT INTO "ingredients" VALUES (11,'Сахар',4330.0);
INSERT INTO "ingredients" VALUES (12,'Яйца',3984.0);
INSERT INTO "ingredients" VALUES (13,'Разрыхлитель',1500.0);
INSERT INTO "ingredients" VALUES (14,'Дрожжи (сухие)',1190.0);
INSERT INTO "ingredients" VALUES (15,'Ванильный экстракт или сахар',3000.0);
INSERT INTO "ingredients" VALUES (16,'Какао-порошок',2500.0);
INSERT INTO "ingredients" VALUES (17,'Орехи',3100.0);
INSERT INTO "ingredients" VALUES (18,'Ликер',1000.0);
```

```
INSERT INTO "ingredients" VALUES (19,'Специи',2000.0);
INSERT INTO "ingredients" VALUES (20,'Песочное печенье',3000.0);
INSERT INTO "order_items" VALUES (1,23,5,3,420.0);
INSERT INTO "order_items" VALUES (2,23,12,2,260.0);
INSERT INTO "order_items" VALUES (3,25,1,3,300.0);
INSERT INTO "order_items" VALUES (4,27,7,3,450.0);
INSERT INTO "order_items" VALUES (5,29,3,3,360.0);
INSERT INTO "order_items" VALUES (6,31,9,3,480.0);
INSERT INTO "order_items" VALUES (7,33,12,3,390.0);
INSERT INTO "order_items" VALUES (8,33,1,3,300.0);
INSERT INTO "order_items" VALUES (9,33,11,3,330.0);
INSERT INTO "order_items" VALUES (10,35,6,4,480.0);
INSERT INTO "order_items" VALUES (11,36,14,1,310.0);
INSERT INTO "order_items" VALUES (12,37,12,1,130.0);
INSERT INTO "order_items" VALUES (13,38,17,3,120.0);
INSERT INTO "order_items" VALUES (14,38,1,3,300.0);
INSERT INTO "order_items" VALUES (15,43,9,8,1280.0);
INSERT INTO "order_items" VALUES (16,45,1,6,600.0);
INSERT INTO "order_items" VALUES (17,46,1,3,300.0);
INSERT INTO "order_items" VALUES (18,47,1,4,400.0);
INSERT INTO "order_items" VALUES (19,48,9,2,320.0);
INSERT INTO "order_items" VALUES (20,48,11,4,440.0);
INSERT INTO "order_items" VALUES (21,48,1,6,600.0);
INSERT INTO "order_items" VALUES (22,51,7,3,450.0);
INSERT INTO "order_items" VALUES (23,51,11,4,440.0);
INSERT INTO "order_items" VALUES (24,52,1,7,700.0);
INSERT INTO "orders" VALUES (3,'2025-01-29','12:11:34',0.0);
INSERT INTO "orders" VALUES (5,'2025-01-13','10:35:10.4993752',0.0);
INSERT INTO "orders" VALUES (7,'2025-01-13','13:25:22.7598108',0.0);
```

```
INSERT INTO "orders" VALUES (8,'2025-01-13','13:25:49.5946633',0.0);
INSERT INTO "orders" VALUES (9,'2025-01-13','13:25:49.6181894',0.0);
INSERT INTO "orders" VALUES (10,'2025-01-13','13:27:09.0349424',0.0);
INSERT INTO "orders" VALUES (11,'2025-01-13','13:27:09.0579514',0.0);
INSERT INTO "orders" VALUES (12,'2025-01-13','13:27:46.8152139',0.0);
INSERT INTO "orders" VALUES (13,'2025-01-13','13:27:46.8395546',0.0);
INSERT INTO "orders" VALUES (14,'2025-01-13','13:52:15.6752591',0.0);
INSERT INTO "orders" VALUES (15,'2025-01-13','13:52:15.7014958',0.0);
INSERT INTO "orders" VALUES (16,'2025-01-13','13:58:24.4936317',0.0);
INSERT INTO "orders" VALUES (17,'2025-01-13','13:58:24.5202581',0.0);
INSERT INTO "orders" VALUES (18,'2025-01-13','14:03:55.2056354',0.0);
INSERT INTO "orders" VALUES (19,'2025-01-13','14:03:55.2296695',0.0);
INSERT INTO "orders" VALUES (20,'2025-01-13','14:08:40.7875214',0.0);
INSERT INTO "orders" VALUES (21,'2025-01-13','14:08:40.8121832',0.0);
INSERT INTO "orders" VALUES (22,'2025-01-13','14:15:42.2978482',0.0);
INSERT INTO "orders" VALUES (23,'2025-01-13','14:15:42.3228656',0.0);
INSERT INTO "orders" VALUES (24,'2025-01-13','14:21:50.4799530',0.0);
INSERT INTO "orders" VALUES (25,'2025-01-13','14:21:50.4984828',300.0);
INSERT INTO "orders" VALUES (26,'2025-01-13','14:23:47.7884228',0.0);
INSERT INTO "orders" VALUES (27,'2025-01-13','14:23:47.8038885',450.0);
INSERT INTO "orders" VALUES (29,'2025-01-13','14:24:06.2200719',360.0);
INSERT INTO "orders" VALUES (31,'2025-01-13','14:25:09.6161828',480.0);
INSERT INTO "orders" VALUES (33,'2025-01-13','14:31:32.6948457',1020.0);
INSERT INTO "orders" VALUES (35,'2025-01-13','15:12:04.3303529',480.0);
INSERT INTO "orders" VALUES (37,'2025-01-13','15:13:00.5848250',130.0);
INSERT INTO "orders" VALUES (38,'2025-01-13','15:20:28.9511960',420.0);
INSERT INTO "orders" VALUES (39,'2025-01-13','15:25:58.7467599',0.0);
INSERT INTO "orders" VALUES (40,'2025-01-13','15:31:10.5315489',0.0);
INSERT INTO "orders" VALUES (41,'2025-01-13','15:31:23.6826984',0.0);
```

INSERT INTO "orders" VALUES (42,'2025-01-13','15:31:40.5084939',0.0);

INSERT INTO "orders" VALUES (43,'2025-01-14','10:33:10.6472287',0.0);

INSERT INTO "orders" VALUES (44,'2025-01-14','10:37:03.3760275',0.0);

INSERT INTO "orders" VALUES (45,'2025-01-14','10:37:14.9464283',0.0);

INSERT INTO "orders" VALUES (46,'2025-01-14','10:37:51.1401387',0.0);

INSERT INTO "orders" VALUES (47,'2025-01-14','10:39:59.2561922',400.0);

INSERT INTO "orders" VALUES (48,'2025-01-14','10:47:02.3514514',1360.0);

INSERT INTO "orders" VALUES (51,'2025-01-14','14:49:53.3285845',890.0);

INSERT INTO "orders" VALUES (52,'2025-01-14','15:00:54.2788270',700.0);

INSERT INTO "products" VALUES (1,'Эспрессо','Классический итальянский кофе, приготовленный с помощью давления.',1,100.0);

INSERT INTO "products" VALUES (2,'Американо','Эспрессо с добавлением горячей воды.',1,100.0);

INSERT INTO "products" VALUES (3,'Капучино','Эспрессо с паровым молоком и молочной пеной.',1,120.0);

INSERT INTO "products" VALUES (4,'Латте','Эспрессо с паровым молоком и небольшим количеством пенки.',1,130.0);

INSERT INTO "products" VALUES (5,'Мокко','Латте с добавлением шоколадного сиропа.',1,140.0);

INSERT INTO "products" VALUES (6,'Далгоне','Кофе, взбитый до кремовой консистенции с сахаром.',1,120.0);

INSERT INTO "products" VALUES (7,'Флэт Уайт','Латте с меньшим количеством молока и более крепким вкусом.',1,150.0);

INSERT INTO "products" VALUES (8,'Кофе по-восточному','Кофе, приготовленный с кардамоном и сахаром.',1,100.0);

INSERT INTO "products" VALUES (9,'Круассан','Традиционная французская выпечка с нежным тестом.',2,160.0);

INSERT INTO "products" VALUES (10,'Мафин','Пышный сладкий пирожок, часто с фруктами.',2,210.0);

INSERT INTO "products" VALUES (11,'Брауни','Шоколадный десерт с влажной текстурой.',2,110.0);

INSERT INTO "products" VALUES (12,'Тартуф','Миниатюрный десерт с шоколадной начинкой.',2,130.0);

INSERT INTO "products" VALUES (13,'Чизкейк','Десерт на основе сыра с хрустящей основой.',2,260.0);

INSERT INTO "products" VALUES (14,'Пирожное Наполеон','Слоеное пирожное с кремом.',2,310.0);

INSERT INTO "products" VALUES (15,'Пирожное Мадлен','Мягкое пирожное в форме раковины.',2,150.0);

INSERT INTO "products" VALUES (16,'Кекс','Сладкая выпечка с ягодами или орехами.',2,70.0);

INSERT INTO "products" VALUES (17,'Песочное печенье','Овощное печенье с богатым масляным вкусом.',2,40.0);

INSERT INTO "products" VALUES (18,'Простой кофе с молоком','Обычный кофе три в одном',1,0.0);

INSERT INTO "products" VALUES (19,'Деготь','Самое крепкое кофе в области',1,200.0);

INSERT INTO "products" VALUES (20,'Тестовый элемент','Тестовый элемент для демонстрации

работы приложения',1,0.0);

INSERT INTO "recipe_ingredients" VALUES (1,1,4,20.0);

INSERT INTO "recipe_ingredients" VALUES (2,1,5,30.0);

INSERT INTO "recipe_ingredients" VALUES (3,2,4,30.0);

INSERT INTO "recipe_ingredients" VALUES (4,2,5,90.0);

INSERT INTO "recipe_ingredients" VALUES (5,3,4,30.0);

INSERT INTO "recipe_ingredients" VALUES (6,3,8,150.0);

INSERT INTO "recipe_ingredients" VALUES (7,3,8,30.0);

INSERT INTO "recipe_ingredients" VALUES (8,4,4,30.0);

INSERT INTO "recipe_ingredients" VALUES (9,4,8,200.0);

INSERT INTO "recipe_ingredients" VALUES (10,4,8,10.0);

INSERT INTO "recipe_ingredients" VALUES (11,5,4,30.0);

INSERT INTO "recipe_ingredients" VALUES (12,5,8,200.0);

INSERT INTO "recipe_ingredients" VALUES (13,5,11,30.0);

INSERT INTO "recipe_ingredients" VALUES (14,5,7,30.0);

INSERT INTO "recipe_ingredients" VALUES (15,6,4,8.5);

INSERT INTO "recipe_ingredients" VALUES (16,6,11,25.0);

INSERT INTO "recipe_ingredients" VALUES (17,6,5,30.0);

INSERT INTO "recipe_ingredients" VALUES (18,6,8,200.0);

INSERT INTO "recipe_ingredients" VALUES (19,7,4,30.0);

INSERT INTO "recipe_ingredients" VALUES (20,7,8,120.0);

INSERT INTO "recipe_ingredients" VALUES (21,8,10,10.0);

```
INSERT INTO "recipe_ingredients" VALUES (22,8,5,125.0);
INSERT INTO "recipe_ingredients" VALUES (23,8,11,5.0);
INSERT INTO "recipe_ingredients" VALUES (24,8,19,2.0);
INSERT INTO "recipe_ingredients" VALUES (25,9,10,250.0);
INSERT INTO "recipe_ingredients" VALUES (26,9,9,150.0);
INSERT INTO "recipe_ingredients" VALUES (27,9,14,5.0);
INSERT INTO "recipe_ingredients" VALUES (28,9,8,125.0);
INSERT INTO "recipe_ingredients" VALUES (29,9,11,5.0);
INSERT INTO "recipe_ingredients" VALUES (30,9,11,30.0);
INSERT INTO "recipe_ingredients" VALUES (31,10,10,250.0);
INSERT INTO "recipe_ingredients" VALUES (32,10,11,150.0);
INSERT INTO "recipe_ingredients" VALUES (33,10,12,2.0);
INSERT INTO "recipe_ingredients" VALUES (34,10,8,125.0);
INSERT INTO "recipe_ingredients" VALUES (35,10,13,10.0);
INSERT INTO "recipe_ingredients" VALUES (36,10,9,80.0);
INSERT INTO "recipe_ingredients" VALUES (37,11,6,150.0);
INSERT INTO "recipe_ingredients" VALUES (38,11,9,100.0);
INSERT INTO "recipe_ingredients" VALUES (39,11,11,200.0);
INSERT INTO "recipe_ingredients" VALUES (40,11,12,2.0);
INSERT INTO "recipe_ingredients" VALUES (41,11,10,100.0);
INSERT INTO "recipe_ingredients" VALUES (42,11,17,50.0);
INSERT INTO "recipe_ingredients" VALUES (43,12,6,200.0);
INSERT INTO "recipe_ingredients" VALUES (44,12,7,100.0);
INSERT INTO "recipe_ingredients" VALUES (45,12,16,30.0);
INSERT INTO "recipe_ingredients" VALUES (46,12,18,30.0);
INSERT INTO "recipe_ingredients" VALUES (47,13,12,500.0);
INSERT INTO "recipe_ingredients" VALUES (48,13,11,150.0);
INSERT INTO "recipe_ingredients" VALUES (49,13,12,3.0);
INSERT INTO "recipe_ingredients" VALUES (50,13,10,200.0);
```

INSERT INTO "recipe_ingredients" VALUES (51,13,7,100.0);

INSERT INTO "recipe_ingredients" VALUES (52,14,10,300.0);

INSERT INTO "recipe_ingredients" VALUES (53,14,7,500.0);

INSERT INTO "recipe_ingredients" VALUES (54,14,11,100.0);

INSERT INTO "recipe_ingredients" VALUES (55,15,10,200.0);

INSERT INTO "recipe_ingredients" VALUES (56,15,11,150.0);

INSERT INTO "recipe_ingredients" VALUES (57,15,12,3.0);

INSERT INTO "recipe_ingredients" VALUES (58,15,9,150.0);

INSERT INTO "recipe_ingredients" VALUES (59,15,13,10.0);

INSERT INTO "recipe_ingredients" VALUES (60,15,15,1.0);

INSERT INTO "recipe_ingredients" VALUES (61,16,10,250.0);

INSERT INTO "recipe_ingredients" VALUES (62,16,11,200.0);

INSERT INTO "recipe_ingredients" VALUES (63,16,12,3.0);

INSERT INTO "recipe_ingredients" VALUES (64,16,8,150.0);

INSERT INTO "recipe_ingredients" VALUES (65,16,13,10.0);

INSERT INTO "recipe_ingredients" VALUES (66,16,9,100.0);

INSERT INTO "recipe_ingredients" VALUES (67,17,10,250.0);

INSERT INTO "recipe_ingredients" VALUES (68,17,9,125.0);

INSERT INTO "recipe_ingredients" VALUES (69,17,11,100.0);

INSERT INTO "recipe_ingredients" VALUES (70,17,12,1.0);

INSERT INTO "recipe_ingredients" VALUES (71,17,15,1.0);

INSERT INTO "recipes" VALUES (1,1,'Сварите 30 мл эспрессо с 18-20 г кофейных зерен с помощью кофемашины.');

INSERT INTO "recipes" VALUES (2,2,'Приготовьте 30 мл эспрессо и добавьте 90-120 мл горячей воды.');

INSERT INTO "recipes" VALUES (3,3,'Сварите 30 мл эспрессо и добавьте 150 мл горячего молока с 30 мл молочной пенки.');

INSERT INTO "recipes" VALUES (4,4,'Приготовьте 30 мл эспрессо, добавьте 200 мл горячего молока и украсите 10-20 мл молочной пенки.');

INSERT INTO "recipes" VALUES (5,5,'Сварите 30 мл эспрессо, добавьте 200 мл горячего молока и 30 г шоколадного сиропа, украсить взбитыми сливками по желанию.');

INSERT INTO "recipes" VALUES (6,6,'Смешайте 2 ст. ложки (8-10 г) Instant кофе, 2 ст. ложки (25 г) сахара и 2 ст. ложки (30 мл) горячей воды до стойкой пены. Подавайте с 200 мл молока по желанию.');

INSERT INTO "recipes" VALUES (7,7,'Приготовьте 30 мл эспрессо и добавьте 120 мл парного молока.');

INSERT INTO "recipes" VALUES (8,8,'Смешайте 10 г молотого кофе с 100-150 мл воды. По желанию добавьте сахар и специи (кардамон).');

INSERT INTO "recipes" VALUES (9,9,'Смешайте 250 г муки, 150 г масла, 5 г дрожжей, 125 мл молока, 5 г соли и 30 г сахара. Замесить тесто, дать подняться и сформировать круассаны. Выпекать при 200°C до золотистой корочки.');

INSERT INTO "recipes" VALUES (10,10,'Смешайте 250 г муки, 150 г сахара, 2 яйца, 125 мл молока, 10 г разрыхлителя и 80 г масла, запекать в форме 20-25 минут при 180°C.');

INSERT INTO "recipes" VALUES (11,11,'Растопите 150 г шоколада с 100 г масла. Добавьте 200 г сахара, 2 яйца и 100 г муки. Добавьте орехи по желанию. Выпекать 30 минут при 180°C.');

INSERT INTO "recipes" VALUES (12,12,'Смешайте 200 г растопленного шоколада с 100 мл сливок. Формируйте шарики и обваляйте в 30 г какао-порошка. По желанию добавьте ликер.');

INSERT INTO "recipes" VALUES (13,13,'Смешайте 500 г творога или сливочного сыра с 150 г сахара и 3 яйцами. Выпекать в основе из 200 г печенья 40 минут при 160°C. Сливки по желанию.');

INSERT INTO "recipes" VALUES (14,14,'Используйте 300 г слоеного теста, выпекайте до золотистого цвета. Приготовьте крем из 500 мл сливок и 100 г сахара. Соберите пирожное.');

INSERT INTO "recipes" VALUES (15,15,'Смешайте 200 г муки, 150 г сахара, 3 яйца, 150 г растопленного масла, 10 г разрыхлителя и ваниль. Выпекать 10-12 минут при 180°C.');

INSERT INTO "recipes" VALUES (16,16,'Смешайте 250 г муки, 200 г сахара, 3 яйца, 150 мл молока, 10 г разрыхлителя и 100 г масла, выпекать до готовности при 180°C.');

INSERT INTO "recipes" VALUES (17,17,'Смешайте 250 г муки, 125 г масла, 100 г сахара, 1 яйцо и ваниль, формируйте печенье и выпекайте при 180°C до золотистого цвета.');

COMMIT;