МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра «Прикладная математика и информатика»
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки / специальности)

Разработка программного обеспечения
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Разработка программного обеспечения информационной системы управления заказами службы доставки продуктов питания

Обучающийся А.О. Иванов _____
(Инициалы Фамилия) (личная подпись)

Руководитель д.т.н., доцент, С.В. Мкртычев
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2024

Аннотация

Работа посвящена разработке программного обеспечения информационной системы управления заказами службы доставки продуктов питания.

Объект исследования – информационная система управления заказами службы доставки продуктов питания.

Предмет исследования – программное обеспечение информационной системы управления заказами службы доставки продуктов питания.

Целью работы является разработка программного обеспечения информационной системы управления заказами службы доставки продуктов питания.

Для достижения поставленной цели необходимо решить следующие задачи:

- выполнить постановку задачи на разработку программного обеспечения информационной системы управления заказами службы доставки продуктов питания;
- спроектировать программное обеспечение информационной системы управления заказами службы доставки продуктов питания;
- выполнить реализацию и тестирование программного обеспечения информационной системы управления заказами службы доставки продуктов питания.

В ходе работы применялись различные программы для проектирования и моделирования. ля проектирования организационной структуры был использован бесплатный сервис для моделирования draw.io, программный продукт StarUML для проектирования диаграмм на основе объектно-ориентированного подхода.

Работа содержит три главы в которых описываются все задачи, которые были поставлены для выполнения цели по разработке программного продукта информационной системы управления службой доставки продуктов питания.

В первой главе описывается постановка задачи на разработку программного обеспечения информационной системы управления службой доставки, которая включает функциональные и архитектурные особенности информационных систем управления доставкой. Также включает обзор и анализ аналогов программного обеспечения информационной системы управления службой доставки.

Вторая глава включает проведённое проектирование программного обеспечения, на основе выделенного процесса для автоматизации. В главе проводится описание проектирования базы данных, основных объектов с которыми будет работать менеджер при управлении службой доставки продуктов питания.

На основе спроектированного программного обеспечения в третьей главе осуществляется его физическая разработка. Разработка включает представление всех основных форм для управления службой доставки, отчеты с информаций которая подвергается обработке и графики для представления эффективности процесса.

В заключении работы описываются все задачи, которые выполнялись в ходе работы.

Выпускная квалификационная работа состоит из 55 страниц печатного текста, 20 рисунков, 10 таблиц, 26 источников информации и 1 приложения.

Оглавление

Введение

Тема выпускной квалификационной работы актуальна в компании, где занимаются доставкой продуктов клиентам.

В условиях жесткой конкуренции компании по доставке продуктов стремятся усовершенствовать процесс доставки путем автоматизации.

С учетом увеличения поступления заказов на доставку продуктов, также увеличивается информация о клиентах, заказах, доставках, которые нуждаются в хранении, возникает потребность в интеграции программного обеспечения для управления доставками продуктов.

Разработка и внедрение программного обеспечения информационной системы управления заказами службы доставки продуктов питания позволит снизить нагрузку на менеджеров приемки заказов. Информационная система минимизирует количество допустимых ошибок на этапе обработки заказов. Обеспечит сохранение и отслеживание заказов от начала и до получения его клиентом, сделавшим заказ.

Объект исследования – информационная система управления заказами службы доставки продуктов питания.

Предмет исследования – программное обеспечение информационной системы управления заказами службы доставки продуктов питания.

Целью работы является разработка программного обеспечения информационной системы управления заказами службы доставки продуктов питания.

Для достижения поставленной цели необходимо решить следующие задачи:

- выполнить постановку задачи на разработку программного обеспечения информационной системы управления заказами службы доставки продуктов питания;
- спроектировать программное обеспечение информационной системы управления заказами службы доставки продуктов питания;

– выполнить реализацию и тестирование программного обеспечения информационной системы управления заказами службы доставки продуктов питания.

В ходе работы применялись различные программы для проектирования и моделирования, для проектирования организационной структуры был использован бесплатный сервис для моделирования draw.io, программный продукт StarUML для проектирования диаграмм на основе объектно-ориентированного подхода.

Работа содержит три главы в которых описываются все задачи, которые были поставлены для выполнения цели по разработке программного продукта информационной системы управления службой доставки продуктов питания.

В первой главе описывается постановка задачи на разработку программного обеспечения информационной системы управления службой доставки, которая включает функциональные и архитектурные особенности информационных систем управления доставкой. Также включает обзор и анализ аналогов программного обеспечения информационной системы управления службой доставки.

Вторая глава включает проведённое проектирование программного обеспечения, на основе выделенного процесса для автоматизации. В главе проводится описание проектирования базы данных, основных объектов с которыми будет работать менеджер при управлении службой доставки продуктов питания.

На основе спроектированного программного обеспечения в третьей главе осуществляется его физическая разработка. Разработка включает представление всех основных форм для управления службой доставки, отчеты с информаций которая подвергается обработке и графики для представления эффективности процесса.

В заключении работы описываются все задачи, которые выполнялись в ходе работы.

Глава 1 Постановка задачи на разработку программного обеспечения информационной системы управления службой доставки

1.1 Функциональные и архитектурные особенности информационных систем управления службой доставки предприятий сферы обслуживания

Программное обеспечение (ПО) информационной системы управления (ИСУ) службой доставки продуктов питания должно быть основано на обеспечении прозрачности, оперативности и удобства управления доставками заказов [1].

Функциональность ПО ИСУ должна позволять управлять процессом службы доставки. Получать, назначать заказы курьерам доставки, обновлять информацию о доставленных заказах и отслеживать заказы от начала его поступления до момента доставки и получения статуса «доставлено».

Информационная система также должна иметь функции для регистрации нового сотрудника, позволять ограничивать доступ к данным путем разграничения ролей.

В сферах обслуживания типовая архитектура ПО ИСУ доставки заказов клиентам имеет пользовательский интерфейс, позволяющий получать информацию и вести поиск по данным, которые уже имеются в системе. Также система должна иметь базу данных для хранения информации о всех заказах (принятых и обработанных), сотрудниках, работающих в службе доставки, которые занимаются непосредственной доставкой продуктов питания клиентам. ИСУ для службы доставки продуктов питания имеет архитектуру с модулями для работы пользователя и модуль хранения данных, между которыми должно быть налажено взаимодействие.

Однако также необходимо добавить компоненты управления назначением заказов курьерам доставки, учета полученных заказов и заказов,

которые уже были доставлены за заданный период, функционал отслеживания загрузки сотрудников доставки, функционал учета заказов в определённые зоны доставки в пиковые часы получения заказов [3].

Архитектура ПО ИСУ направлена на оптимизацию процесса управления службой доставки продуктов питания, повышение эффективности и обеспечение точного учёта и отслеживания данных в соответствующих сферах деятельности.

1.2 Разработка требований к программному обеспечению информационной системы управления службой доставки

ПО ИСУ создается для автоматизации процессов управления службой доставки. Целью создания программного обеспечения является упрощение работы оператора службы, выполняющего работу вручную. ПО ИСУ позволит отслеживать поступающие заказы с реального времени, только те заказы, которые уже были оплачены ранее клиентом.

С помощью программного обеспечения оператор службы доставки сможет решать задачи распределения заказов быстрее в два раза, с учетом представления всей информации о загрузке сотрудников службы доставки и количества актуальных заказов на данный момент.

В системе будут обрабатываться данные распределения заказов курьерам, исходя из этого будет вестись статистика по всем данным доставки продуктов питания.

Перед началом работы в системе будут вводиться данные:

– полученных заказов;

– данные клиента и адреса доставки;

– время доставки.

После обработки входящих данных, будут получены:

– список заказов, назначенных курьеру.

– список выполненных заказов.

Информационная обеспечит выполнение:

– получение данных о заявках;

– получение информации о клиенте;

– вести данные о загрузке сотрудника доставки;

– распределять заказы, выбранному сотруднику;

– вести анализ пиковых часов заказов;

− вести данные о районах, где больше всего заказов на доставку.

Меню автоматизированной информационной системы должно быть понятным для сотрудников и оператора службы доставки. Вход в систему должен осуществляться после прохождения аутентификации данных, введенных пользователем.

В таблице 1 приведены требования к ПО FURPS+.

Таблица 1 – Требования к ПО

Требование	Статус	Полезность	Риск
Функциональные требования			
Учет заказов на доставку продуктов	Активное	Высокая	Низкий
Назначение заказов курьерам	Активное	Высокая	Средний
Сбор и хранение логов событий	Активное	Средняя	Средний
Генерация списка назначенных заказов	Активное	Средняя	Низкий
Управление заказами на доставку продуктов	Активное	Высокая	Средний
Удобство использования			
Интуитивно понятный интерфейс	Активное	Высокая	Низкий
Настраиваемые дашборды	Активное	Высокая	Низкий
Поддержка нескольких языков	Запланировано	Средняя	Средний

Продолжение таблицы 1

Требование	Статус	Полезность	Риск
Надежность			
Требование	Статус	Полезность	Риск
Высокая доступность системы	Активное	Высокая	Высокий
Защита от потери данных	Активное	Высокая	Высокий
Механизмы восстановления после сбоев	Запланировано	Высокая	Высокий
Производительность			
Обработка данных в реальном времени	Активное	Высокая	Высокий
Минимальная задержка уведомлений	Активное	Высокая	Средний
Высокая масштабируемость	Запланировано	Высокая	Высокий
Поддерживаемость			
Легкость обновления и развертывания системы	Активное	Средняя	Средний
Подробная документация	Активное	Высокая	Низкий
Инструменты для отладки и диагностики	Запланировано	Высокая	Средний
Ограничения			
Безопасность данных	Активное	Высокая	Высокий
Соответствие нормативным требованиям	Запланировано	Высокая	Высокий
Интеграция с внешними системами	Активное	Средняя	Средний
Поддержка резервного копирования	Запланировано	Высокая	Высокий

Меню автоматизированной информационной системы должно быть понятным при первом входе в систему и иметь графики с данными по статистике на текущую дату. Личный кабинет сотрудника должен содержать всю необходимую информацию и иметь все инструменты для выполнения поставленных задач.

1.3 Обзор и анализ аналогов программного обеспечения информационной системы управления службой доставки

Проведем поиск и сравнение аналогичных программ для автоматизации службы доставки.

«Мегалогист встраивается в вашу программу 1С и позволяет логистам и водителям работать в единой среде.

Возможности системы Мегалогист для логистов:

− автоматическое и ручное планирование рейсов;

− работа в единой среде 1С;

– контроль выполнения рейсов в онлайн-режиме;

− массовая печать всех документов для курьера;

− закрытие рейса и взаиморасчеты с курьером.

Для руководителя:

− более эффективное использование транспорта;

− контроль затрат на доставку и расчет себестоимости;

− анализ работы службы доставки» [4].

«Стоимость безлимитной лицензии на программу 360 000 рублей. Лицензия на Мегалогист оплачивается один раз и не имеет ограничения на количество пользователей, количество заказов и срок действия. Стоимость лицензий (единоразовый платеж): 150 000 рублей» [4].

Интерфейс управления заказами для курьера, рисунок 1.

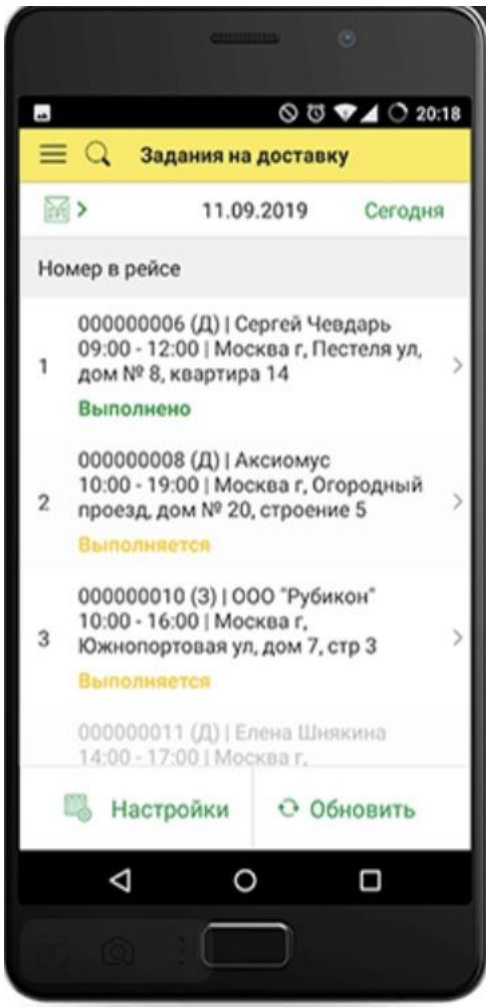

Рисунок 1 – Интерфейс программы «Мегалогист»

«MobiDel — Программа для автоматизации службы доставки. Поможет быстро оформлять заказы, распределять их по курьерам, взаимодействовать с клиентами и контролировать работу предприятия. Расширение бизнеса, требует оптимизации его основных процессов. Основным помощником в этом, может стать программа для автоматизации службы доставки. Информационная система Mobidel отвечает всем критериям этого сложного рабочего процесса. Ее внедрение рекомендуется в первую очередь ресторанам доставки и интернет-магазинам, кафе, барам, программа для доставки суши, программа для доставки воды. Любая компания, репутация которой зависит от скорости и качества предоставления продукта до потребителя, должна уделить особое внимание подразделению курьерской доставки.

Инструменты, которые помогут организовать работу:

− телефония и клиентская база;

− распределение заказов;

− мобильное приложение для курьеров;

− смс оповещения о статусе заказа;

− отслеживание заказа на вашем сайте» [19].

На рисунке 2 представлен интерфейс управления заказами в программе MobiDel.

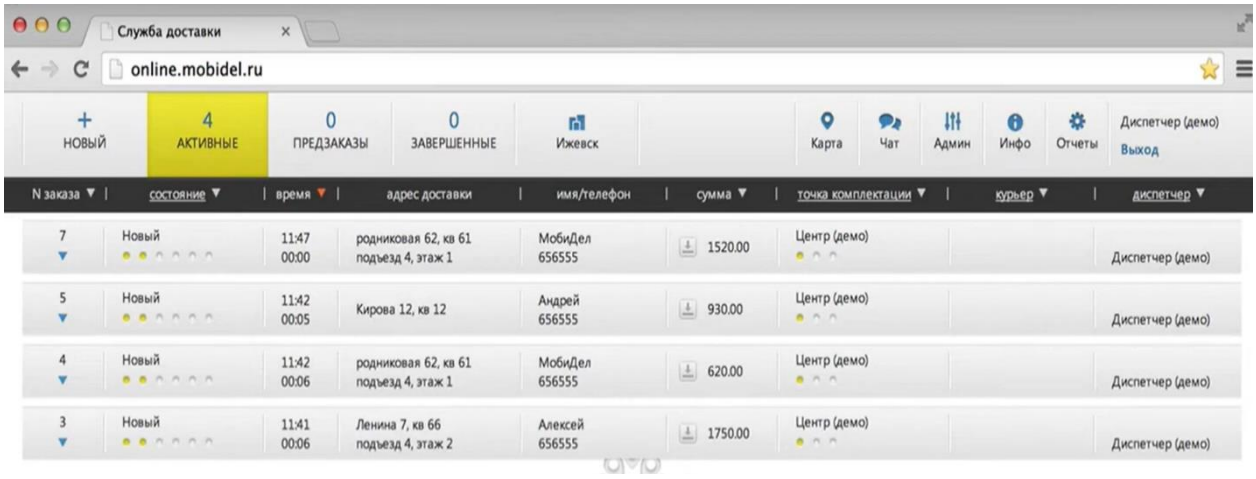

Рисунок 2 – Интерфейс управления заказами в программе MobiDel

«MeaSoft — система управления доставкой MeaSoft является комплексным бизнес-решением для курьерских служб.

Функции системы:

− интеграция с клиентами и подрядчиками;

− инструменты логистики, маршрутизация для курьеров;

− учет и сортировка отправлений;

− учет курьеров, мобильное приложение, расчет зарплаты;

− личный кабинет клиента;

− фулфилмент;

− личный кабинет получателя» [18].

На рисунке 3 представлен интерфейс управления заказами в программе MeaSoft.

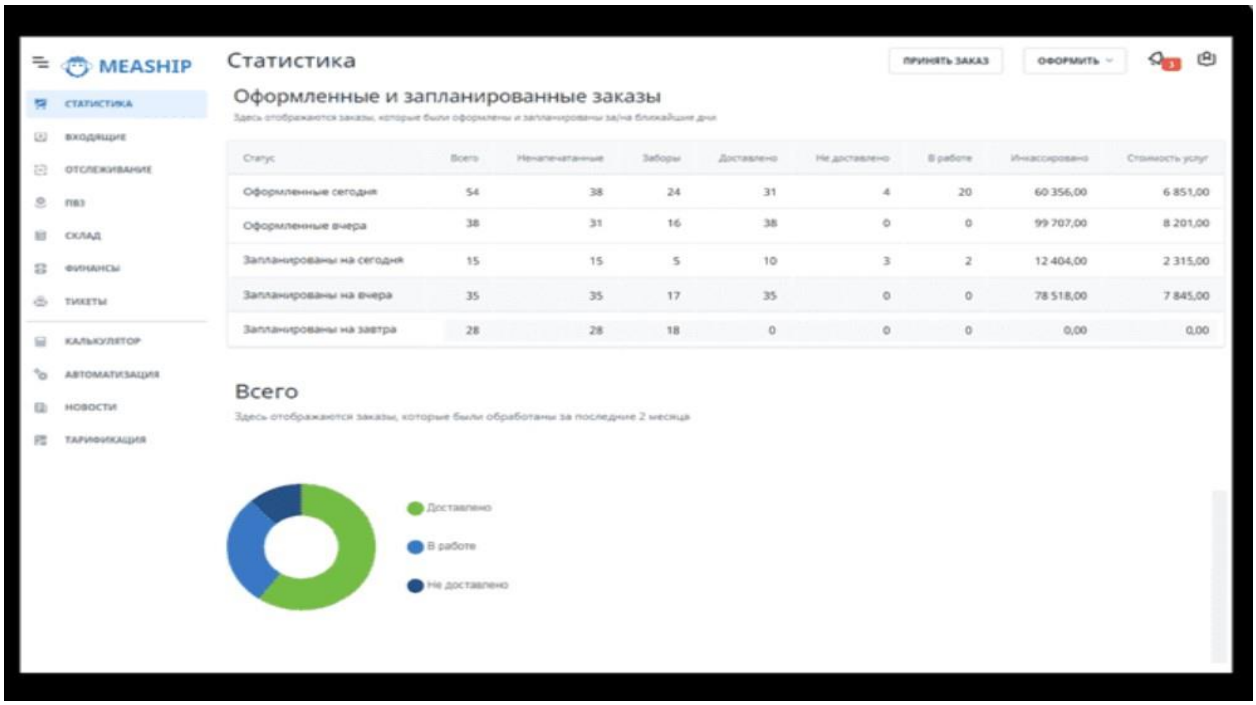

Рисунок 3 – Интерфейс управления заказами в программе MeaSoft

Проведем сравнение программ в таблице 2.

Таблица 2 – Сравнительная таблица программ

Требования к ПО	Мегалогист	MobiDel	MeaSoft
Наличие мобильного приложения	+	-	-
Разграничение прав доступа	-	+	+
Личный кабинет клиента	-	-	+
Статистика работы курьеров	-	-	-
Распределение заказов	+	+	-
Получение уведомлений	+	-	+

На основе рассмотренных программ для службы доставки, которые

были рассмотрены по одним и тем же критериям, показали, что стоимость программ при покупке достаточно высокая. При внедрении программы необходимо также учесть затраты на дальнейшее ее обслуживание.

Вывод по первой главе

Первая глава ориентирована на постановку задачи на разработку программного обеспечения информационной системы управления службой доставки.

Постановка задачи включала рассмотрение функциональных и архитектурных особенностей информационных систем управления службой доставки. На основе поставленной задачи был проведен анализ аналогичных программных продуктов, представленных на рынке, что сподвигло к собственной разработке.

Глава 2 Проектирование программного обеспечения информационной системы управления службой доставки продуктов питания

2.1 Разработка диаграммы вариантов использования программного обеспечения

Проектирование программного продукта на концептуальном уровне предполагает разработку и согласование концепции продукта. В концепции идет подробное описание целей и задач программного продукта, также идет планирование временных, ресурсных и стоимостных затрат на реализацию продукта [5].

Концепция создается и согласовывается с заказчиком, на основе которой группа разработчиков реализует принятую концепцию в реальный программный продукт.

Воспользуемся инструментами моделирования объектно-ориентированного программирования для описания на концептуальном уровне информационную систему управления службой доставки. Для моделирования используем диаграмму вариантов использования, которая наглядно показывает взаимодействие системы и пользователей.

Цель диаграммы показать функционирование системы разработчику и пользователю, для дальнейшей корректировки или внесения изменения в основную логику системы.

Диаграмма вариантов использования с двумя акторами показана на рисунке 4.

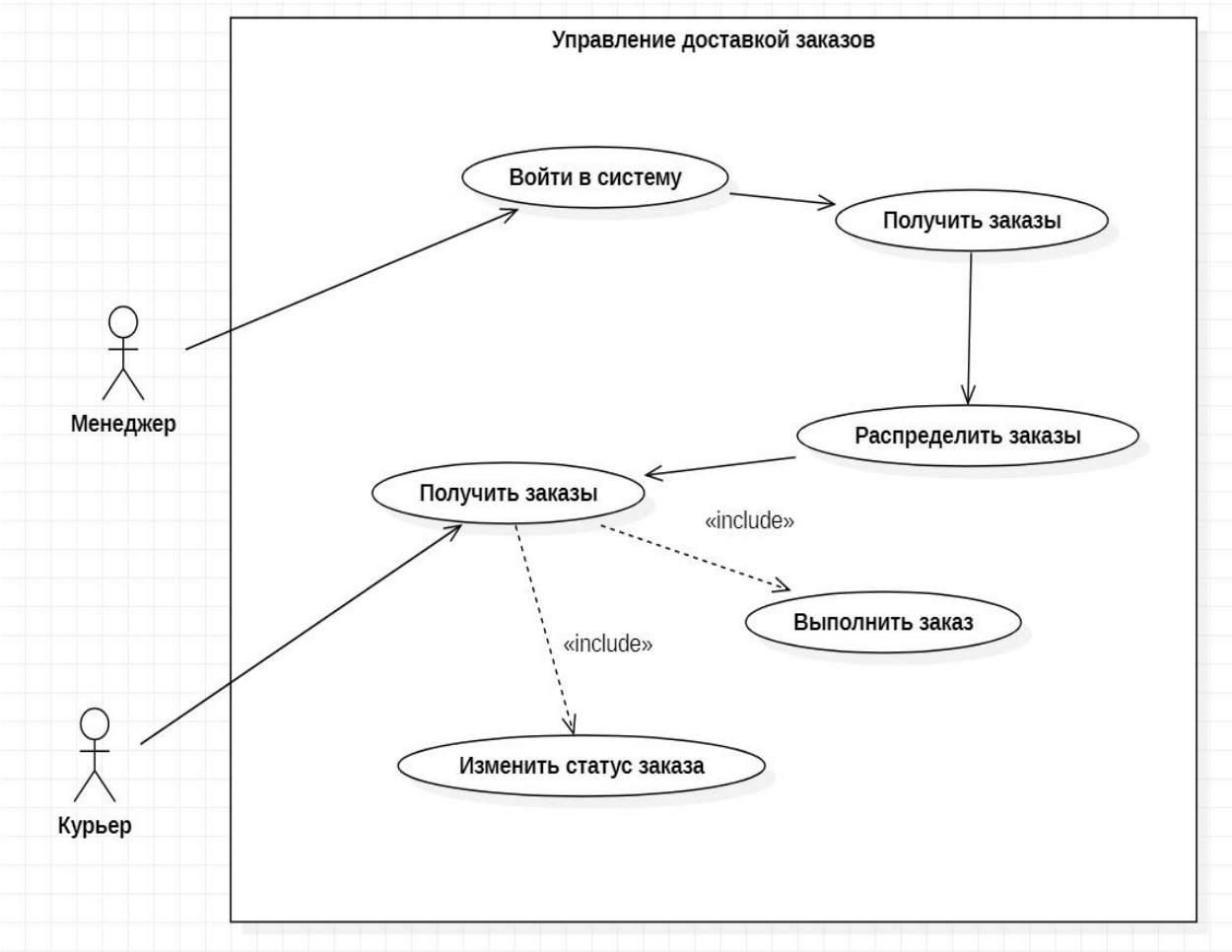

Рисунок 4 – Диаграмма вариантов использования ПО управлению заказами

Функции системы должны покрывать все потребности чтобы обеспечивать управление доставкой заказов клиентов.

2.2 Описание вариантов использования

Описание вариантов использования поможет более подробно получить данные о вариантах использования функций при наступлении ошибки, предусмотреть постусловия для использования сценариев диалога в системе и применения альтернативного потока.

В системе конечными пользователями будут менеджер и курьер. Опишем варианты использования в таблице 3.

Таблица 3 – Описание вариантов использования

Вариант использования	Краткое описание	Основной поток событий	Альтернативные потоки	Предусловия	Постусловия
Войти в систему	Вариант использования предусматривает вход в систему пользователя (Менеджер)	Система запрашивает логин пользователя и пароль. Пользователь выбирает свою логин и вводит пароль. Система проверяет логин и пароль, после чего дает доступ входа в систему.	Пользователь ввел неправильно логин и/или пароль, система выводит сообщение об ошибке. Пользователь может вернуться к началу основного потока или отказаться от входа в систему, при этом выполнение варианта использования завершается.	Отсутствуют	Если вариант использования выполнен успешно, пользователь входит в систему.
Получить заказы	Вариант использования предусматривает получения заказов пользователем (Менеджер)	Пользователь входит в систему под своими данными. Система открывает список с данными заказов.	Система не загрузила список новых заказов. Пользователь может перезапустить систему и войти заново, после чего открыть список заказов.	Пользователь должен быть зарегистрирован в системе.	Если система имеет новые заказы, при обновлении и будет получен новый список с данными заказов.

Продолжение таблицы 3

Вариант использования	Краткое описание	Основной поток событий	Альтернативные потоки	Предусловия	Постусловия
Распределить заказы	Вариант использования предусматривает назначение заказов пользователем (Менеджер)	Пользователь входит в систему под своими данными. Пользователь открывает список курьеров. Система открывает список курьеров доступных к назначению заказов. Пользователь выбирает сотрудника и назначает ему выбранный заказ.		Пользователь должен быть зарегистрирован в системе. Система должна содержать данные о заказах и сотрудниках.	Если система имеет новые заказы, менеджер может их назначить курьерам.
Изменить статус заказа	Вариант использования предусматривает выполнение заказов пользователем (Курьер)	Пользователь входит в систему под своими данными. Система открывает список назначенных заказов. Пользователь доставляет заказы и меняет статус «доставлено».	Система не загрузила список назначенных заказов. Пользователь может перезапустить систему и войти заново, после чего открыть список заказов.	Пользователь должен быть зарегистрирован в системе. Система должна содержать данные о заказах.	Система правильно распределила заказы, курьер может выполнить заказы.

На основе построенной диаграммы вариантов использования процесса управления доставкой заказами целесообразно будет смоделировать процесс в диаграмме деятельности.

Диаграмма позволяет посмотреть, как процессы осуществляются в системе.

Рисунок 5 демонстрирует деятельность пользователей в системе.

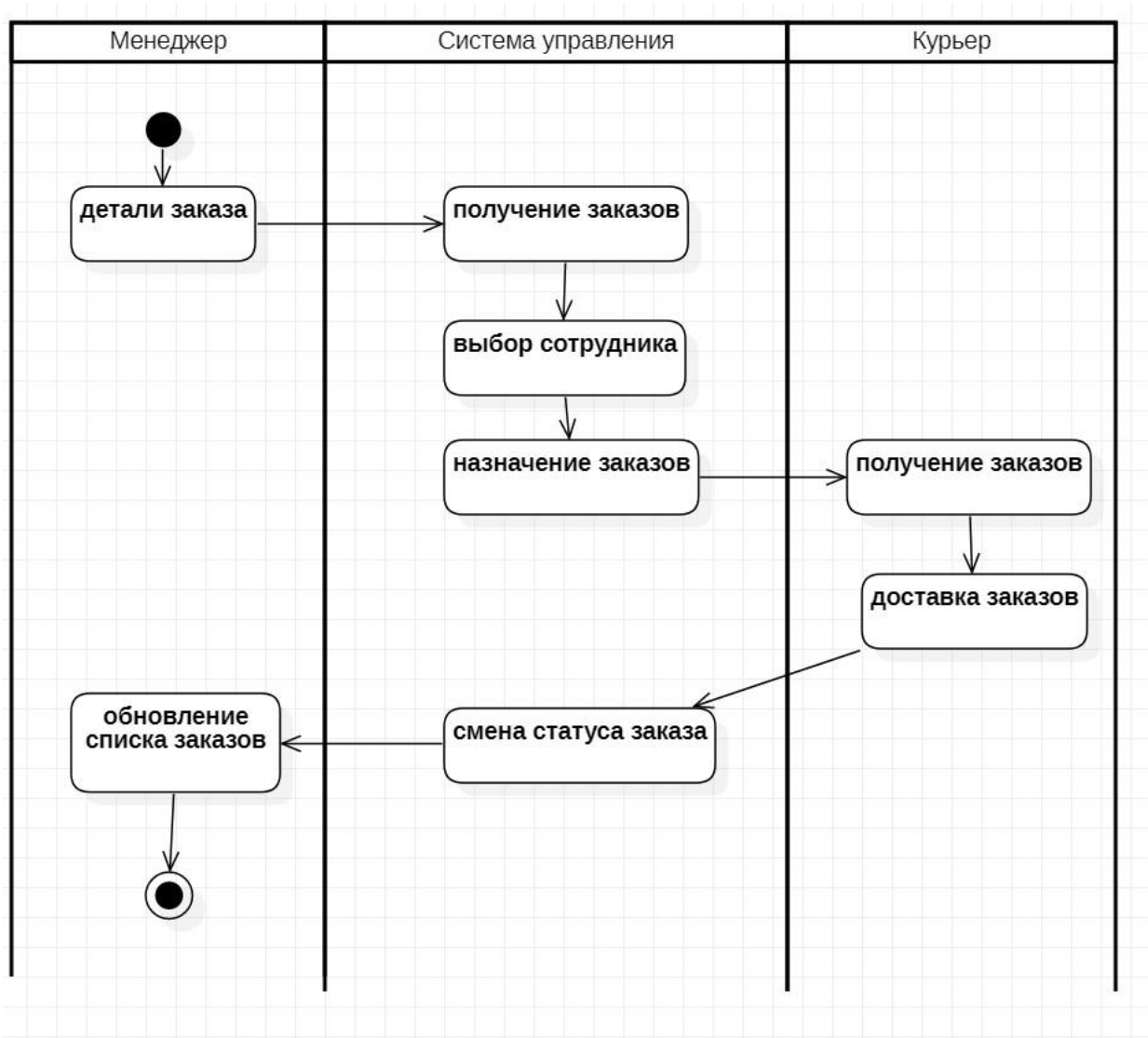

Рисунок 5 – Диаграмма деятельности пользователей при управлении доставкой заказов

Менеджер ведет весь процесс управления доставкой заказов продуктов питания. При получении заказов от клиентов менеджер планирует назначение

заказов курьерам. Основывается в первую очередь на загруженность курьера, тем самым назначает заказы таким образом, чтобы все курьеры могли доставить заказы в ближайшее время.

Диаграммы, смоделированные ранее, позволили рассмотреть процесс управления заказами подробнее. Дальнейшее проектирование программного обеспечения требует выбрать методологию.

2.3 Выбор методологии проектирования программного обеспечения

Выбор методологии проектирования и разработки программного продукта позволит правильно спланировать проект по реализации программного продукта и обеспечить группе разработчиков инструменты для реагирования на возникающие риски на всем жизненном цикле программного продукта [6].

«Гибкая методология разработки (Agile software development) – манифест, содержащий основные ценности и принципы, на которых базируются подходы к управлению проектами, который решает проблемы традиционного проектного менеджмента. Agile подходит для инновационных проектов. Гораздо меньше он подходит для процессной деятельности. Эти подходы подразумевают интерактивную разработку, с периодическими обновлениями требований от заказчика и их реализацию посредством самоорганизующихся команд, сформированных из экспертов разного профиля. Под термином «гибкая методология разработки» следует понимать подходы на основе данного манифеста, или фреймворки» [12].

«Работа с применением гибкой методологии состоит из серии коротких циклов (итераций), длительностью 2-3 недели. Каждая итерация включает в себя этапы планирования, анализа требований, проектирование, разработку, тестирование и документирование. По завершению каждой итерации команда предъявляет заказчику «осязаемые» результаты работы, например, первичную версию продукта или часть функционала, которую можно посмотреть,

оценить, протестировать, а потом доработать или скорректировать» [12].

«Scrum — это методика, помогающая командам вести совместную работу. Как спортивная команда готовится к решающей игре (к слову, scrum — англ. «схватка», элемент игры в регби), так и команда сотрудников компании должна извлекать уроки из полученного опыта, осваивать принципы самоорганизации, работая над решением проблемы, и анализировать свои успехи и провалы, чтобы постоянно совершенствоваться. Scrum содействует этому» [11].

«Методику Scrum чаще всего применяют команды разработчиков приложений, но принципы и опыт ее использования можно применить к командной работе любого рода. Это одна из причин такой популярности методики. Scrum часто представляют как платформу для управления проектами по методике Agile. Участники команды Scrum проводят собрания, используют специальные инструменты и принимают на себя особые роли, чтобы организовать работу и управлять ею.

Несмотря на то, что первоначально метод Scrum был рассчитан на разработку IT-проектов, сегодня он применяется и в других областях» [11].

«Бережливое производство (оно же Lean Management) – это не просто какая-то отдельная методика, инструмент или схема бизнес-процесса.

Бережливый подход сейчас — это один из кирпичиков, который составляет здоровенную стену методологий гибкой разработки, наряду с Agile, Scrum, Kanban, Feature driven development и остальными.

Фактически, смысл бережливого производства сводится к простой идее: если бизнес-процесс можно сделать дешевле, качественнее и быстрее, то это нужно делать. Формулировка очевидная, но сам процесс перехода на новую модель работы очевидным не будет.

Прежде всего: бережливое производство никогда не бывает «полностью законченным» – постоянно работаете над тем, чтобы бизнес-процессы становились всё совершеннее и совершеннее. И эта работа всегда включает в себя ровно два ключевых шага:

– анализ. Оценка текущей ситуации в компании;
– изменения. После того, как анализ проведён, нужно предложить решение для каждой выявленной проблемы. Оно не должно быть заведомо идеальным и не обязательно будет менять весь порядок работы, но шаг вперёд должен быть сделан» [1].

«В основе Kanban’а лежит простая мысль: объем незавершенной работы надо ограничивать. Любую новую задачу можно начинать не ранее, чем выполнена одна из начатых. Это не значит, что в работе должна быть только одна задача, — их может быть несколько. Принципиально, чтобы это количество было ограничено» [5].

«Все задачи, которые поступают от заказчика, записываются на стикеры и размещаются в графе «В ожидании». Им можно назначить приоритет: более важные размещают выше и принимают в работу в первую очередь, а второстепенные — только когда будут выполнены приоритетные. Как только разработчик приступает к работе над очередной задачей, соответствующий листок переносится в графу «В работе». Когда она выполнена, стикер отправляется в последнюю графу» [5].

Для разработки ПО ИСУ службой доставки продуктов питания подойдет методология Канбан [7]. В методологии есть большой ассортимент инструментов, позволяющих выполнять поставленные задачи. Также методология позволяет при необходимости вернуться на шаг назад и исправить допущенные ошибки в программном обеспечении.

2.4 Логическое проектирование программного обеспечения

Структура ПО ИСУ доставки заказов клиентов состоит компонентов таких как модули и классы, которые взаимодействуют друг с другом.

Взаимодействие между компонентами может происходить разными способами, например по средствам API или запросов на сервер.

Для проектирования современной системы существует множество

инструментов и паттернов.

В монолитной архитектуре каждый слой отвечает за отдельную функциональность, за доступ к базе данных, взаимодействие с пользовательским интерфейсом. Такая архитектура достаточно легко разворачивается.

Рассмотрим шаблон проектирования, который наиболее подходит для разработки ПО ИСУ службой доставки продуктов питания.

«MVVM – это паттерн разработки, позволяющий разделить приложение на три функциональные части:

Model – основная логика программы (работа с данными, вычисления, запросы и так далее).

View – вид или представление (пользовательский интерфейс).

ViewModel – модель представления, которая служит прослойкой между View и Model.

Такое разделение позволяет ускорить разработку и поддерживаемость программы – можно менять один компонент, не затрагивая код другого.

Отдельными уровнями кода MVVM являются:

Модель: Этот уровень отвечает за абстракцию источников данных. Модель и ViewModel работают вместе для получения и сохранения данных.

Представление: Цель этого уровня — информировать ViewModel о действиях пользователя. Этот уровень наблюдает за ViewModel и не содержит никакой логики приложения.

ViewModel: предоставляет доступ к тем потокам данных, которые важны для представления. Кроме того, она служит связующим звеном между моделью и представлением» [20].

На рисунке 6 – показана схема взаимодействия данных.

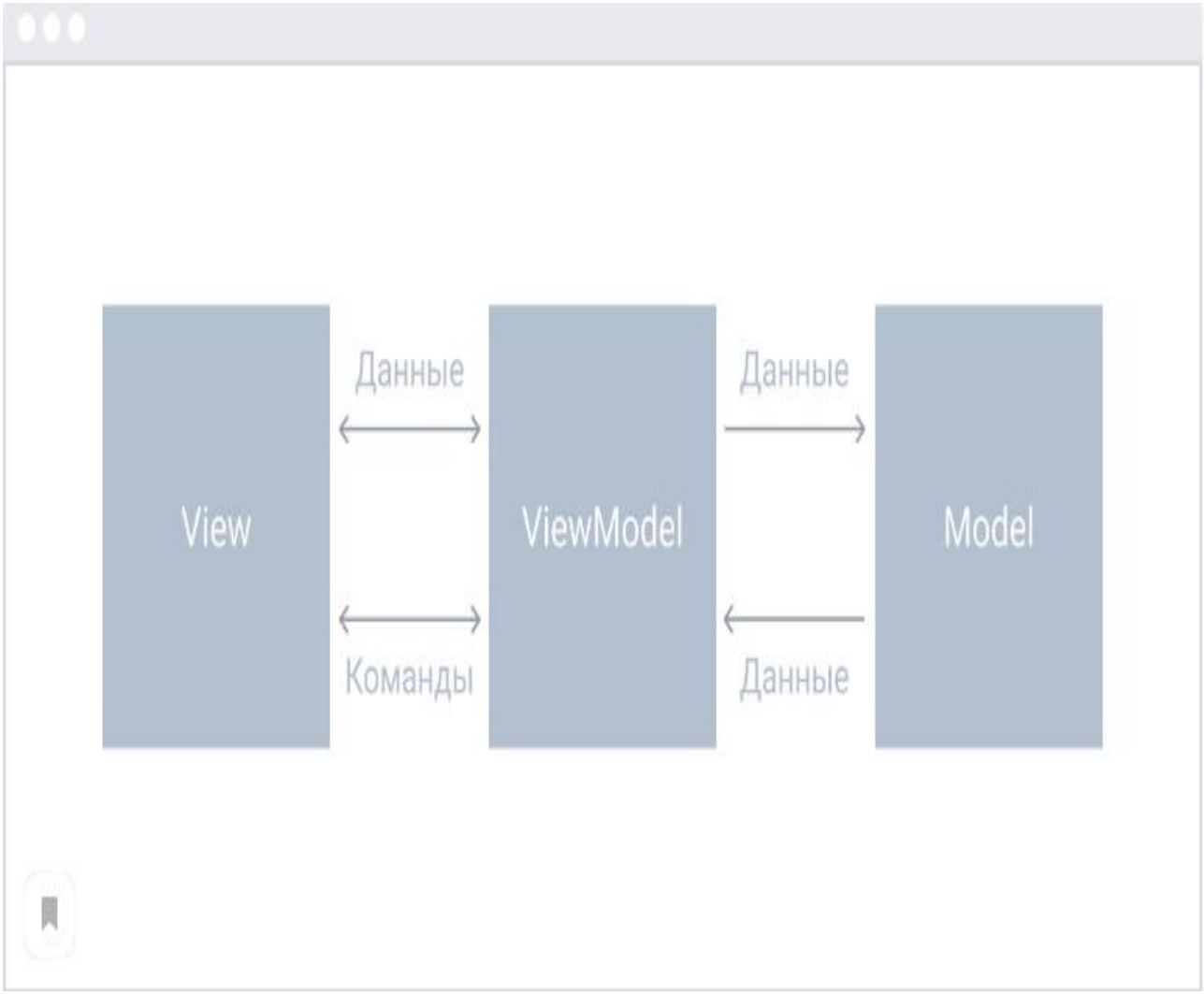

Рисунок 6 – Взаимодействие данных при использовании MVVM

На рисунке 7 представлена архитектура программного обеспечения информационной системы управления службой доставки продуктов питания.

Модуль Model содержит логику для базы данных, модуль View содержит представления программного обеспечения, которые будут получены из модуля ViewModel, где будет находиться вся бизнес-логика ПО.

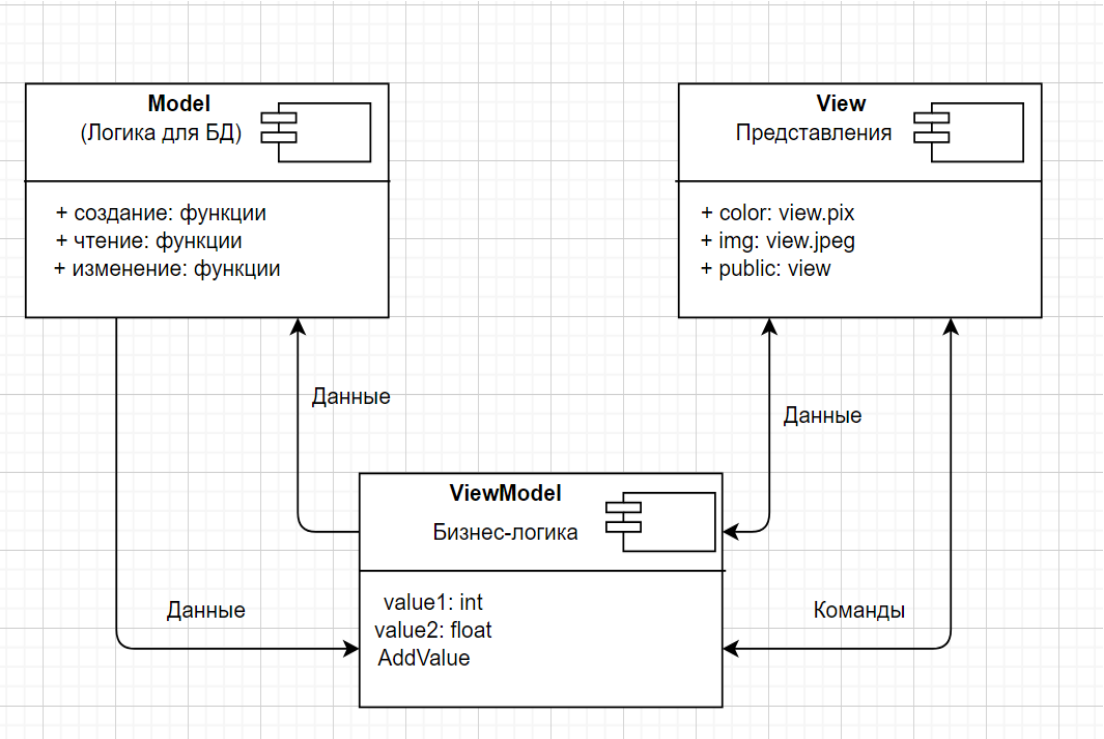

Рисунок 7 – Диаграмма компонентов ПО ИСУ доставкой заказов

На основе выбранного шаблона проектирования можно построить диаграмму компонентов ПО ИСУ, рисунок 8.

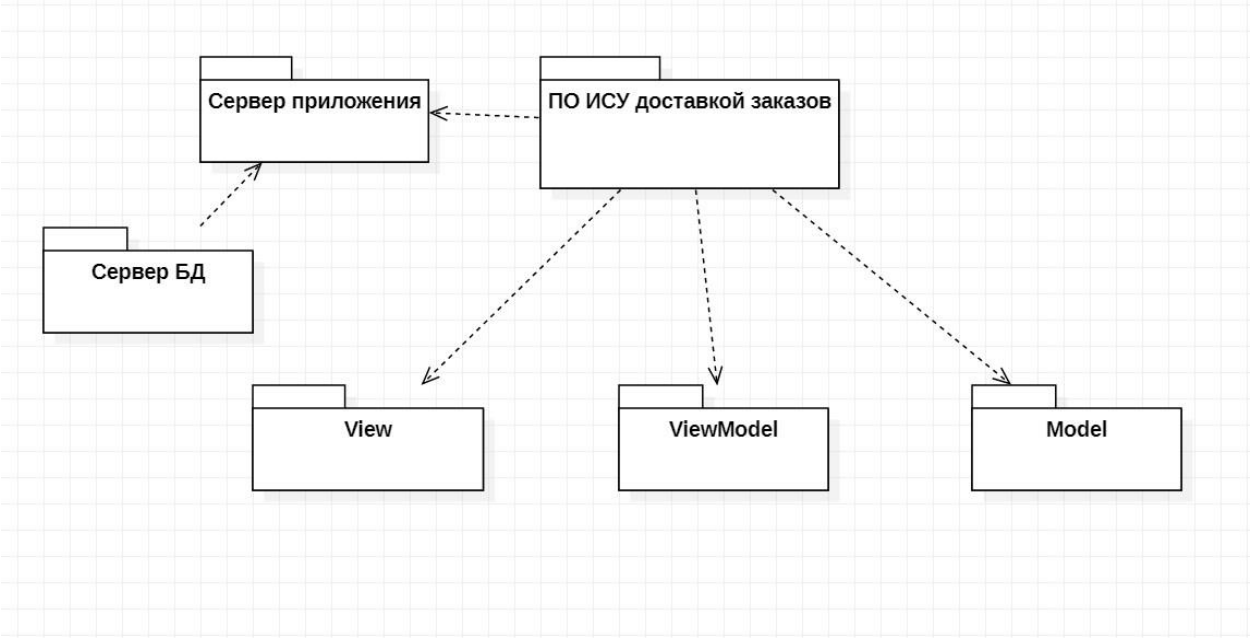

Рисунок 8 – Диаграмма пакетов по шаблону MVVM

Функции, отвечающие за логику программы на различных уровнях, не заключены в классы, поэтому диаграмма классов отсутствует.

Архитектура ПО является значимой при проектировании и разработке, она определяет ее структуру и гибкость.

2.5 Разработка логической структуры базы данных

Разработка логической структуры должна осуществляться на основе потребностей и требований компании оказывающей услуги доставки продуктов питания.

Структура базы данных ПО ИСУ доставки заказов будет содержать несколько таблиц, показана на рисунке 9.

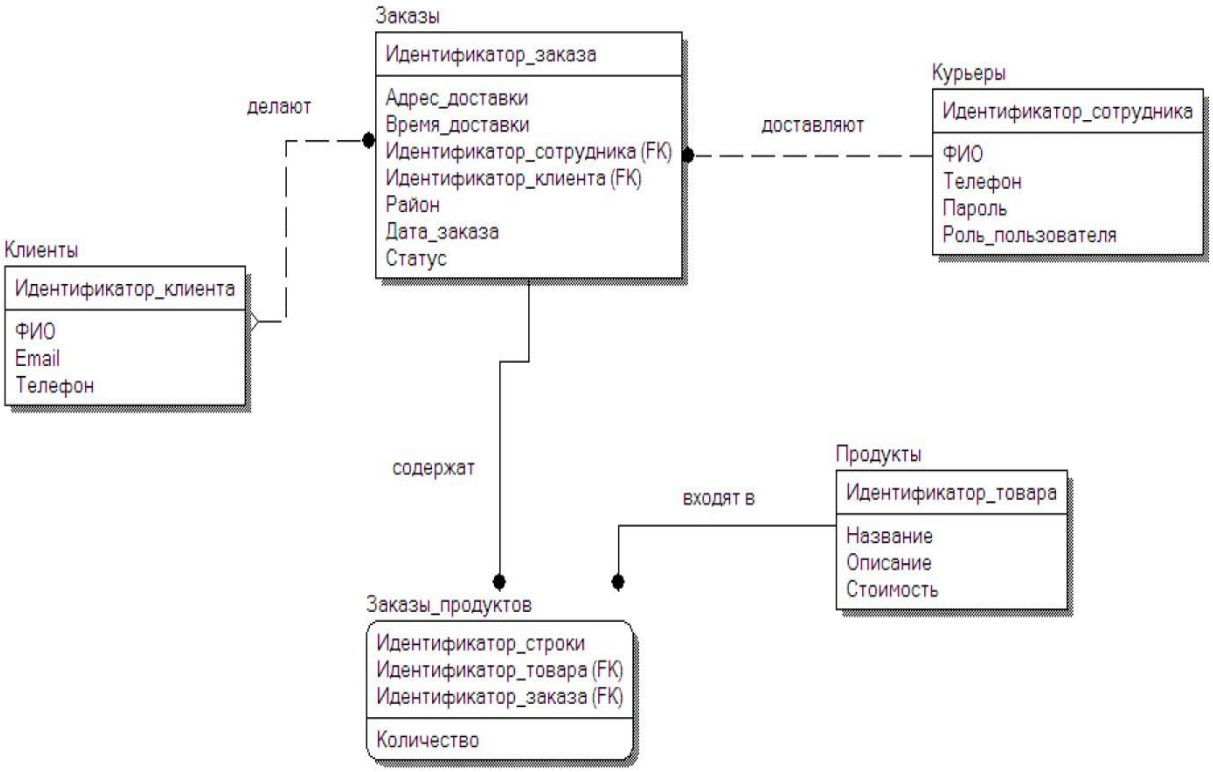

Рисунок 9 – Логическая модель данных

Таблица «Заказы продуктов» (OrderItems) имеет идентификатор заказа продуктов, который не может быть нулевым значением, количество заказанного продукта, название продукта и номер заказа.

Таблица «Курьеры» (Couriers) имеет идентификатор курьера, который не может быть нулевым значением, фамилию, имя, отчество курьера, телефон (строковое значение), значение показывающее свободный курьер или нет.

Таблица «Продукты» (Products) имеет идентификатор продуктов, который не может быть нулевым значением, название продукта, описание продукта и стоимость продукта.

Таблица «Заказы» (Orders) имеет идентификатор заказа, который не может быть нулевым значением, данные клиента, адрес доставки и время доставки, район, дата заказа, статус заказа, данные курьера, который будет назначен на заказ.

Рассмотрим процесс проектирования и разработки интерфейса ПО ИСУ.

Выводы по второй главе

Благодаря проведенному проектированию программного обеспечения информационной системы управления службой доставки разработчики смогут грамотно реализовать программное обеспечение для выполнения основных функций обеспечивающих доставку заказов клиентам. Также было проведено логическое проектирование, в котором была представлена логика ПО ИСУ и проектирование модели данных, где были определены таблицы необходимые для хранения информации.

Глава 3 Реализация и тестирование программного обеспечения ИС

3.1 Выбор средств разработки ПО

Процесс выбора средств для разработки ПО ИСУ службой доставки является неотъемлемой частью для реализации.

Возьмем к рассмотрению несколько языков программирования и систем управления базой данных.

Рассмотрим C++, C# и Java, наиболее распространенные [16].

«C# (произносится «си-шарп») — это современный язык программирования, разработанный компанией Microsoft. Он широко используется для разработки разнообразных приложений, в том числе для создания Windows-приложений, веб-приложений, мобильных приложений под платформу Xamarin, игр на платформе Unity и многих других приложений» [14].

«К 2000 году у Microsoft были готовы промышленные версии новых технологий и решений для обмена сообщениями и данными, а также для создания Internet-приложений. Была выпущена и новая платформа для разработки под новые решения — .NET. В ней объединились сразу несколько языков программирования, что было в новинку для того времени» [14].

«Ещё одним новшеством платформы .NET была технология активных серверных страниц ASP.NET (Active Server Page). С её помощью можно было относительно быстро разработать веб-приложения, взаимодействующие с базами данных. Специально для ASP.NET был создан язык программирования C#. Да и сама ASP.NET была полностью написана на нём» [14].

«C++ — чрезвычайно мощный язык, содержащий средства создания эффективных программ практически любого назначения, от низкоуровневых утилит и драйверов до сложных программных комплексов самого различного назначения. Высокая совместимость с языком C, позволяющая использовать весь существующий C-код (код C может быть с минимальными переделками

скомпилирован компилятором C++; библиотеки, написанные на C, обычно могут быть вызваны из C++ непосредственно без каких-либо дополнительных затрат, в том числе и на уровне функций обратного вызова, позволяя библиотекам, написанным на C, вызывать код, написанный на C++)» [15].

«Поддерживаются различные стили и технологии программирования, включая традиционное директивное программирование, ООП, обобщенное программирование, мета программирование (шаблоны, макросы). Имеется возможность работы на низком уровне с памятью, адресами, портами. Возможность создания обобщённых контейнеров и алгоритмов для разных типов данных, их специализация и вычисления на этапе компиляции, используя шаблоны. Кроссплатформенность. Доступны компиляторы для большого количества платформ, на языке C++ разрабатывают программы для самых различных платформ и систем» [15].

Сравнительная таблица позволит определить плюсы и минусы языков программирования.

Таблица 4 – Результаты сравнения языков программирования

Основные отличия	C++	C#	Java
Понятный синтаксис	-	+	+
Интеграция с базой данных	+	+	+
Скорость обработки данных	+	+	-
Функциональность и гибкость языка	-	+	-
Интеграция с фрэймворками	-	+	+

Разработка ПО ИСУ службой доставки заказов должна иметь интерфейс пользователя и базу данных, где будет храниться информация необходимая диспетчерам и курьерам [24]. Для хранения данных требуется выбрать СУБД.

Сравниваться по функциональности будут СУБД: SQL Server, PostgreSQL, MySQL.

«SQL Server является надежной базой данных для любых целей, может продолжать расширяться по мере наполнения информацией, без заметного уменьшения быстродействия операций с записями в многопользовательском режиме. Пользователи могут быть добавлены путем модернизации оборудования. В последнем тесте поддерживалось до 4600 пользователей базы данных» [21].

«Обеспечивается максимальная безопасность. Ваши данные защищены от несанкционированного доступа за счет интеграции сетевой безопасности с сервером безопасности. Поскольку безопасность на уровне пользователя, пользователи могут иметь ограниченный доступ к записи данных, тем самым защищая их от модификации или поиска, указав доступ на уровне пользовательских привилегией. Кроме того, с данными, хранящимися на отдельном сервере, сервер работает как шлюз, который ограничивает несанкционированный доступ» [21].

«SQL Server обрабатывает запросы от пользователей и только отправляет пользователю результаты запроса. Таким образом, минимальная информация передается по сети. Это улучшает время отклика и устраняет узкие места в сети. Это также позволяет использовать SQL Server в качестве идеальной базы данных для интернет» [21].

«PostgreSQL — это мощная объектно-реляционная система баз данных с открытым исходным кодом, более 35 лет активно разрабатываемая, которая заслужила прочную репутацию за надежность, функциональность и производительность» [8].

«В официальной документации можно найти огромное количество информации, описывающей, как установить и использовать PostgreSQL. Сообщество с открытым исходным кодом предоставляет множество полезных мест, где можно ознакомиться с PostgreSQL, узнать, как это работает, и найти возможности для карьерного роста. Узнайте больше о том, как взаимодействовать с сообществом» [8].

Ниже представлены несколько функциональных возможностей СУБД:

– вложенные запросы;

– представления;

– ссылочная целостность – внешние ключи.

Однако есть и минусы, как и любой другой системы управления данными. Главным минусом является плохая генерация с языками программирования [12].

«СУБД SQL Server разработана Microsoft и имеет хорошую интеграцию с разными языками высокого уровня. Объем хранения данных, может расширяться по мере наполнения информацией, без заметного уменьшения быстродействия операций с записями в многопользовательском режиме» [22].

«Код взаимодействия с базой данных может быть очень громоздким, однако его можно сократить, воспользовавшись Entity Framework. Entity Framework — это решение для работы с базами данных, которое используется в программировании на языках семейства .NET. Оно позволяет взаимодействовать с СУБД с помощью сущностей (entity), а не таблиц. Также код с использованием EF пишется гораздо быстрее. Подключить Entity Framework можно к любому проекту — от Xamarin до ASP.NET» [2].

Система управления базами данных MySQL очень универсальная и работает на многих операционных системах. Также СУБД имеет хорошую производительность, что показывает ее работу с небольшими базами данных быстрее, чем другие программы.

СУБД легко настраивается под особенности программного обеспечения и обеспечивает высокий уровень безопасности хранения и передачи данных. Из минусов можно выделить недостаток функций и медленное развитие по сравнению с другими СУБД.

Представлены системы управления базой данных в сравнительной таблице 5.

Таблица 5 – Результаты сравнения баз данных

Основные отличия	MySQL	MS SQL	PostgreSQL
Доступность	-	+	+
Хранение большого объема данных	+	+	+
Скорость обработки операций с большими данными	+	+	-
Защита от несанкционированного доступа	+	+	+
Простота использования	-	-	-

Сравнительный анализ СУБД показал, что с применением языка программирования C# подходящей будет СУБД Microsoft SQL Server 2022 [23].

3.2 Описание системной архитектуры ПО

Реализация программного обеспечения системы управления службой доставки будет основана на клиент-серверной архитектуре.

«Трехуровневая архитектура будет иметь три уровня:

– клиент;

– сервер приложений;

– сервер базы данных.

Клиент будет взаимодействовать с сервером приложений. Сервер приложений в свою очередь будет обрабатывать всю бизнес-логику системы и обращаться к серверу базы данных при поступлении запроса от клиента» [6].

Сервер базы данных при получении запроса отвечает серверу приложений, который в свою очередь отправляет информацию клиенту.

Архитектура ПО показана на рисунке 10.

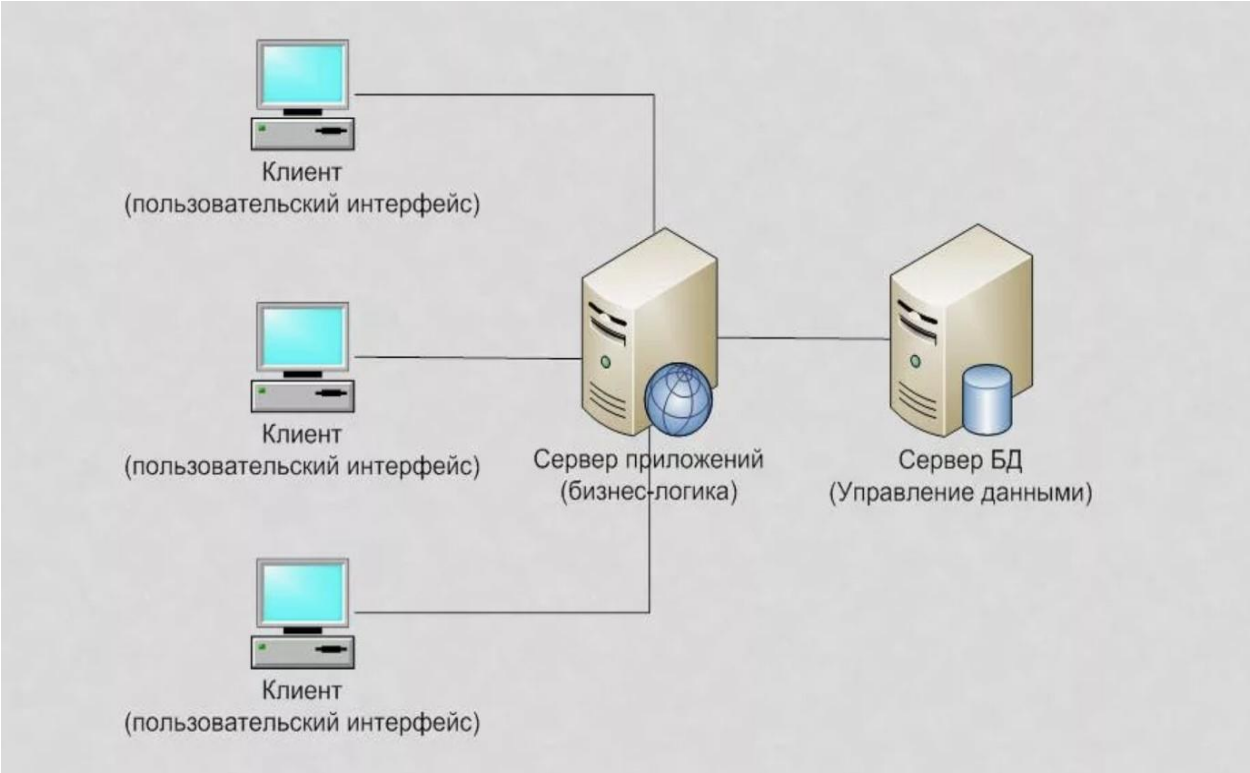

Рисунок 10 – Архитектура ПО

Трехуровневая архитектура является проще расширяемой и модифицируется без влияния на конечных пользователей.

3.3 Проектирование модели данных

Хранение информации о заказах обеспечит база данных. В базе данных информация может храниться долгое время и по средствам запросов к ней предоставлять пользователю данные, запрашиваемые пользователем [9].

В базе данных должна содержаться информация о клиентах, которые оформляют заказ на доставку продуктов питания. Данные о заказах, которые были созданы должны храниться с самого начала и до завершения его, после чего заказ должен храниться также в базе данных.

Назначенные заказы должны быть сотрудникам доставки, данные о которых также сохраняются в базу данных.

Защита информации в системе будет осуществляться путем

разграничения прав доступа. Доступ к данным будут иметь сотрудники, которые будут принимать непосредственное участие в управлении доставкой заказов [13].

Информация будет храниться в таблицах, которые необходимо выделить для создания базы данных. Все данные должны храниться в реляционной базе данных.

База данных построена на данных представленных в таблицах 6 - 10. В таблицах указаны атрибуты и типы данных. Также показаны первичные ключи (PK), которые будут иметь связи с таблицами в таблицах отмечены как внешние ключи (FK).

Таблица 6 – Заказы продуктов

	Атрибут	Тип
PK	Идентификатор строки	int
FK	Идентификатор продукта (внешний ключ)	int
	Количество	int
FK	Идентификатор заказа (внешний ключ)	datetime

Таблица 7 – Заказы

	Атрибут	Тип поля
PK	Идентификатор заказа	int
FK	Идентификатор клиента (внешний ключ)	int
	Адрес доставки	varchar (250)
	Время доставки	datetime
	Район	varchar (50)
	Дата заказа	datetime
	Статус	varchar (30)
FK	Идентификатор курьера (внешний ключ)	int

Таблица 8 – Курьеры

	Атрибут	Тип
PK	Идентификатор сотрудника	int
	ФИО	varchar (50)
	Телефон	varchar (30)
	Пароль	varchar (30)
	Роль пользователя	int

Таблица 9 – Продукты

	Атрибут	Тип
PK	Идентификатор товара	int
	Название	varchar (50)
	Описание	varchar (250)
	Стоимость	decimal

Таблица 10 – Клиенты

	Атрибут	Тип
PK	Идентификатор клиента	int
	ФИО	varchar (250)
	Email	varchar (250)
	Телефон	varchar (11)

Диаграмма БД SQL Server показана на рисунке 11.

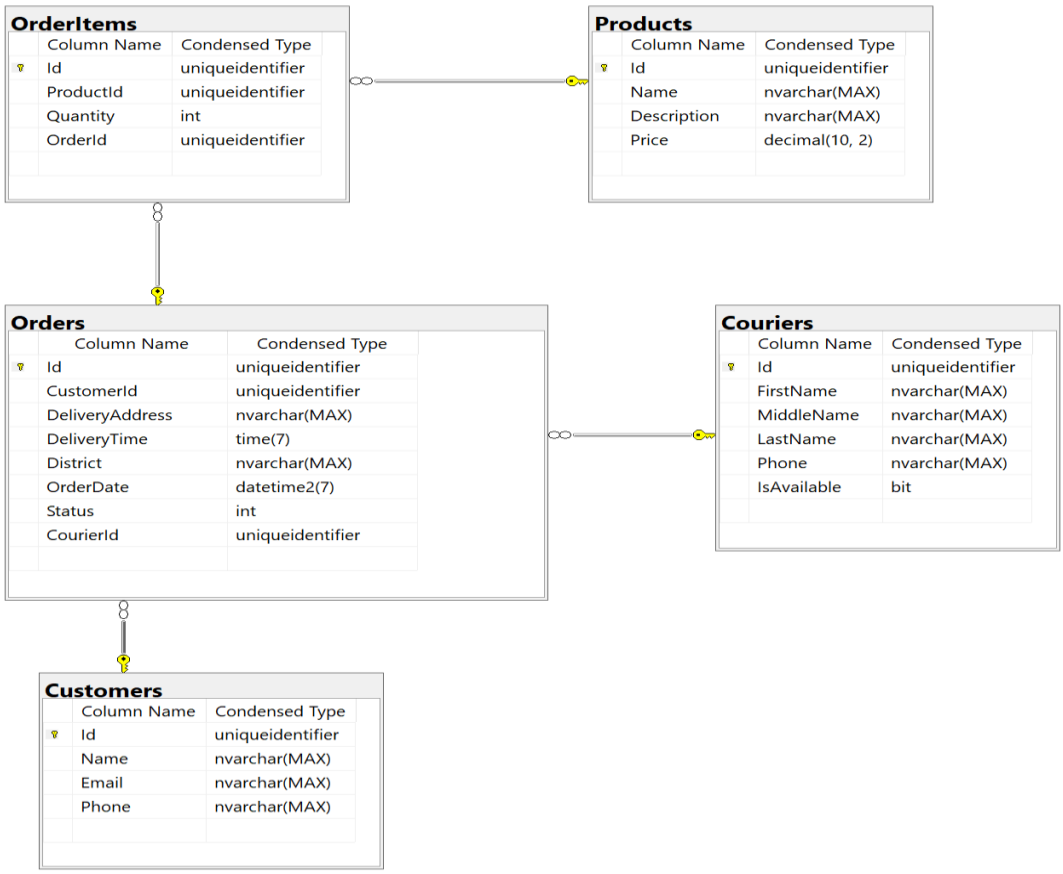

Рисунок 11 – Диаграмма БД SQL Server

Схема базы данных состоит из пяти таблиц, каждая из которых содержит информацию необходимую для управления процессом доставки продуктов питания клиентам.

3.4 Проектирование и разработка пользовательского интерфейса

При проектировании интерфейса пользователя необходимо учесть удобство использования интерфейса, размещение навигации на главном меню пользователя.

Интерфейс пользователя позволяет пользователям компании обращаться к данным в базе данных и получать представления в виде информации в

интерфейсе пользователя [10].

Реализованный интерфейс ПО ИСУ службой доставки должен отвечать требованиям пользователя. Интерфейс будет понятным для пользователя, иметь приятный фон и иметь навигацию [17].

На рисунке 12 показан макет формы с заказами клиентов.

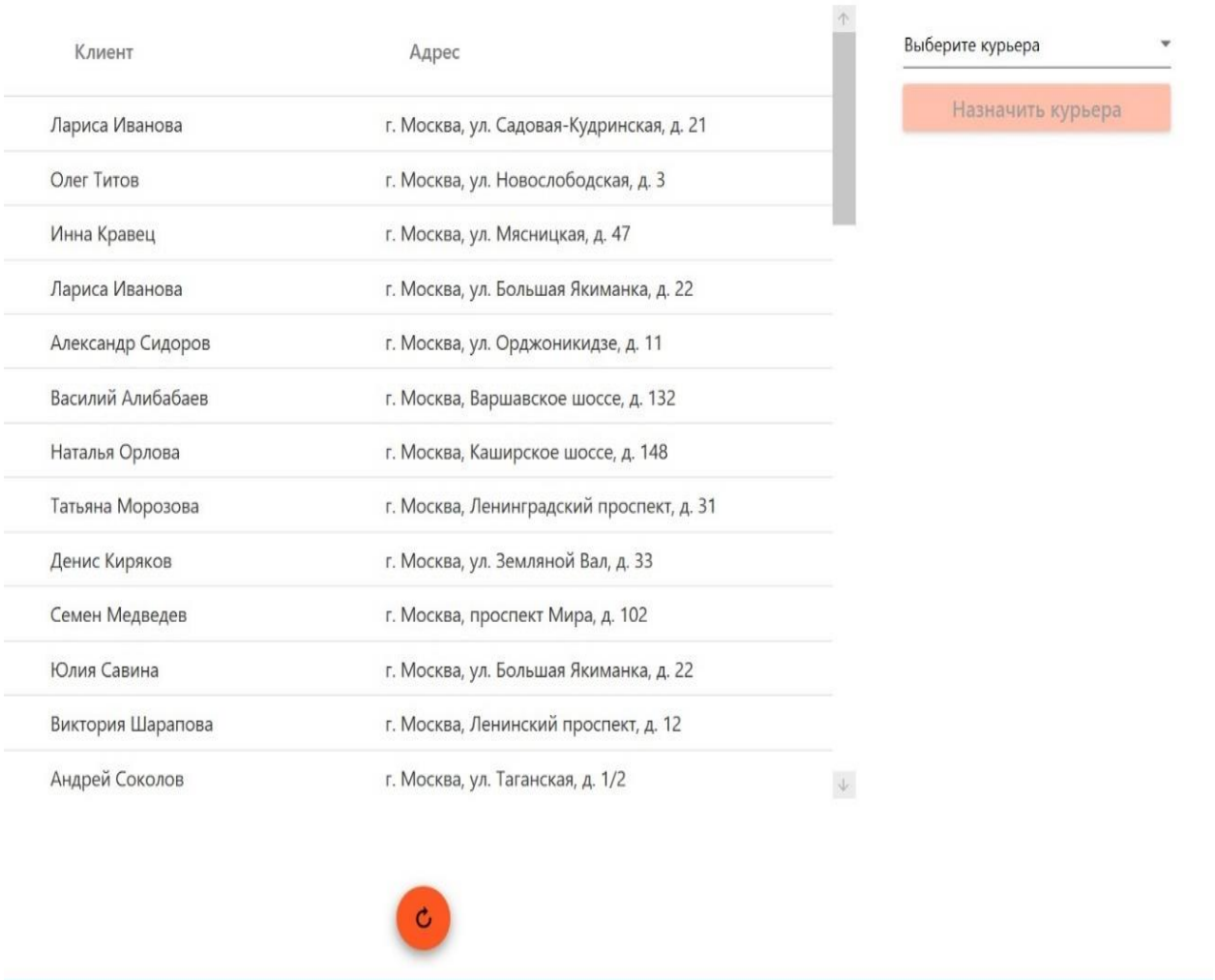

Рисунок 12 – Макет формы с данными заказов клиентов

Интерфейс пользователя обеспечит взаимодействие пользователя с функционалом системы, определяя удобство использования ПО ИСУ службы доставки заказов клиентам.

Листинг программы представлен в приложении А.

3.5 Описание реализации ПО ИСУ службы доставки заказов

Пользователь сможет начать работу в системе после того как пройдет авторизацию. Авторизация позволит разграничить доступ к данным, согласно роли сотрудника [25].

Информация, представленная на главном меню пользователя:

– заказы для назначения курьерам;

– заказы по районам в процентах;

– диаграмма распределения заказов по сотрудникам;

– показатели невыполненных заказов;

– данные районов с наибольшим количеством заказов;

– показатели заказов с распределением по времени получения;

– показатели времени доставки.

Управлением доставки заказов клиентам и распределением их на курьеров должен заниматься сотрудники у которого будут доступы к получению заказов.

Система должна обеспечивать получение всех данных, их обработку и возможность пользователю, который управляет заказами доставкой их клиентам получать актуальную информацию о нагрузке каждого курьера и пиковом времени получения заказов.

Определить загрузку курьера на заказах появится возможность с помощью построенной диаграммы, которая будет отображаться на интерфейсе сотрудника (менеджера).

Круговая диаграмма показана на рисунке 13.

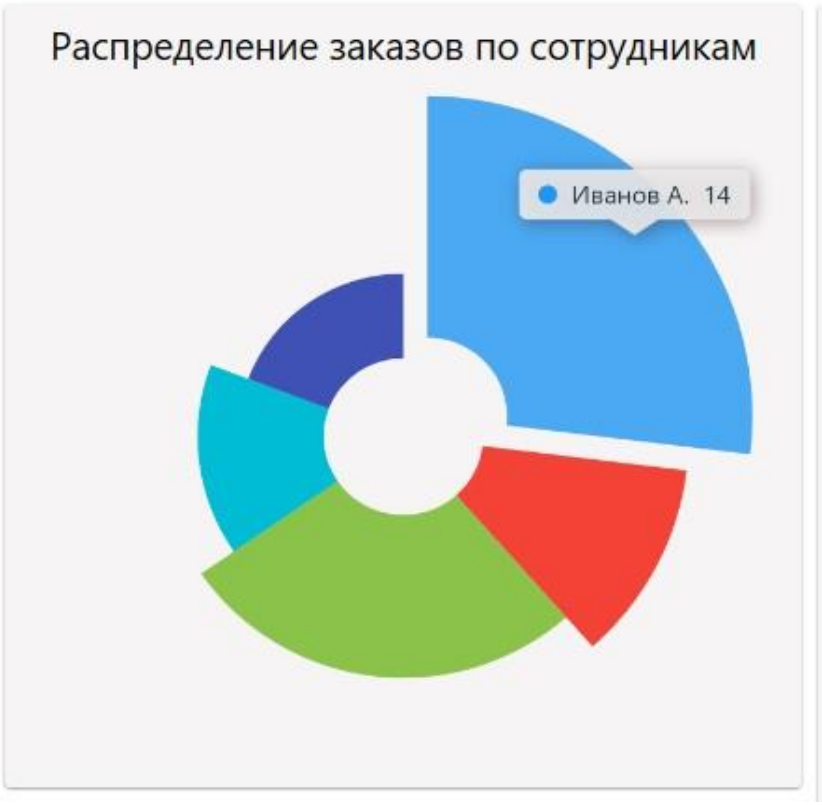

Рисунок 13 – Круговая диаграмма, показывающая заказы сотрудников (курьеров)

Благодаря полученной информации о загрузке заказами курьеров менеджер сможет правильно распределить новые заказы.

После обновления списка заказов, менеджер также будет иметь количество полученных заказов, которые еще не назначены на доставку, рисунок 14.

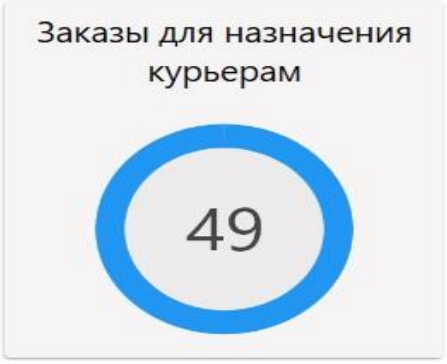

Рисунок 14 – Диаграмма, показывающая количество новых заказов

Все полученные заказы от клиентов должны быть назначены курьерам, чтобы это стало возможным должна быть форма с данными для назначения заказа.

При выборе заказа, который поступил в компанию, должен быть возможность открыть и посмотреть информацию о клиенте и адресе, куда необходимо доставить заказ. Для Назначения заказа также необходимы данные курьера, которые можно будет выбирать из списка всех курьеров, которые будут доступны на заданную дату, рисунок 15.

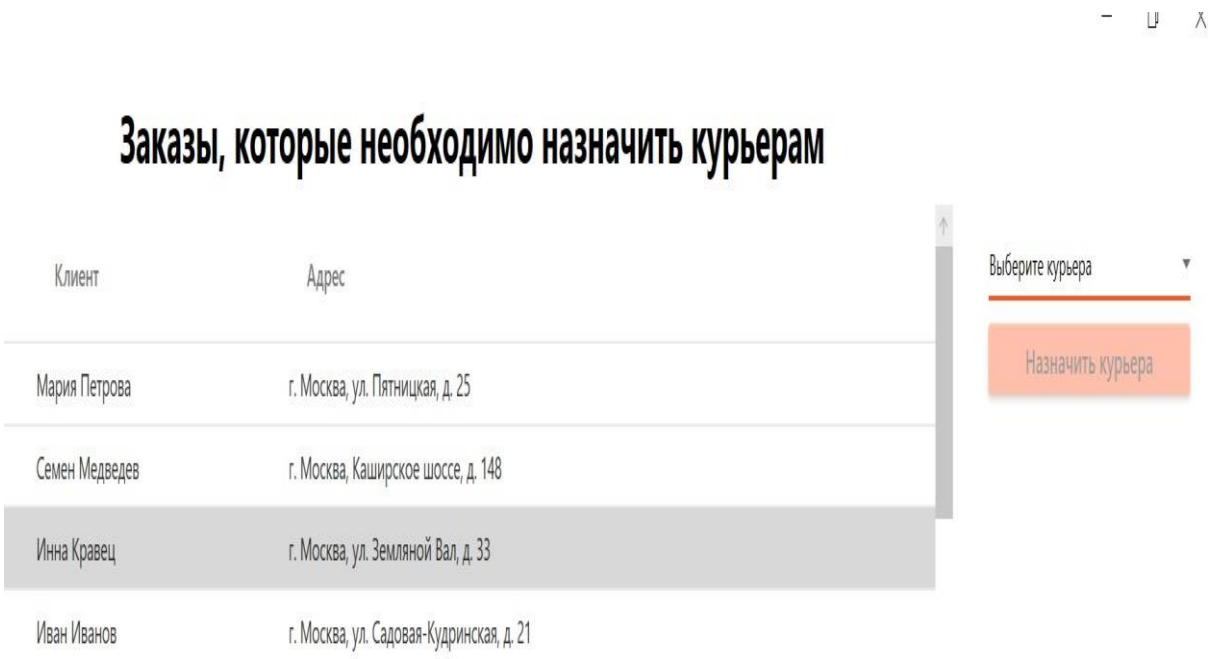

Рисунок 15 – Форма для назначения заказов курьеру

После того как данный курьера выбраны согласно загруженности, которую также контролировать должен менеджер, подтверждается назначение заказа данному курьеру, рисунок 16.

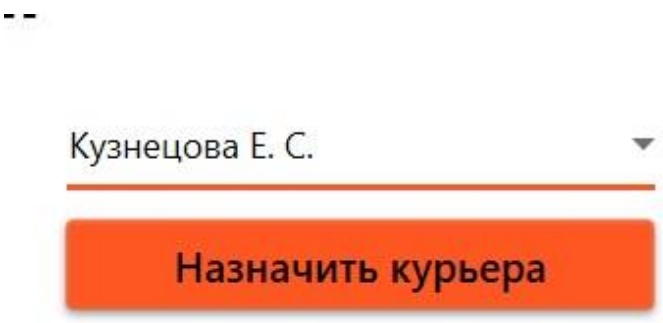

Рисунок 16 – Кнопка, подтверждающая назначение заказа курьеру

Все заказы, полученные менеджером в течении рабочего дня, должны равномерно распределяться курьерам для доставки клиентам. Готовые формы обеспечат быстрый доступ к данным и повысят эффективность выполнения необходимых функций, которые нужны при управлении доставкой заказов клиентам.

3.6 Тестирование программного обеспечения информационной системы управления службой доставки

«Тестирование – это проверка программы на соответствие требований. Во время процесса тестирования идёт поиск разницы между ожидаемым и фактическим поведением продукта.

Целью тестирования является предоставление актуальной информации о продукте на данный момент и поиск дефектов на ранних этапах разработки.

Тестировщик проводит необходимое тестирование для выявления недочетов в работе программного продукта» [11].

Существует много видов и типов тестирования:

– функциональное тестирование;

– нагрузочное тестирование;

– конфигурационное тестирование;

– юзабилити-тестирование;

– тестирование черного ящика.

Для программного обеспечения ИСУ службой доставки будет использоваться функциональное тестирование, которое позволяет проверять функциональность системы без глубоко ознакомления кода и архитектуры системы.

Проведем тестирование прохождения авторизации при входе в систему. Если пользователь вводит неправильные данные, система должна после обработки данных оповестить пользователя о неправильных введенных данных, рисунок 17.

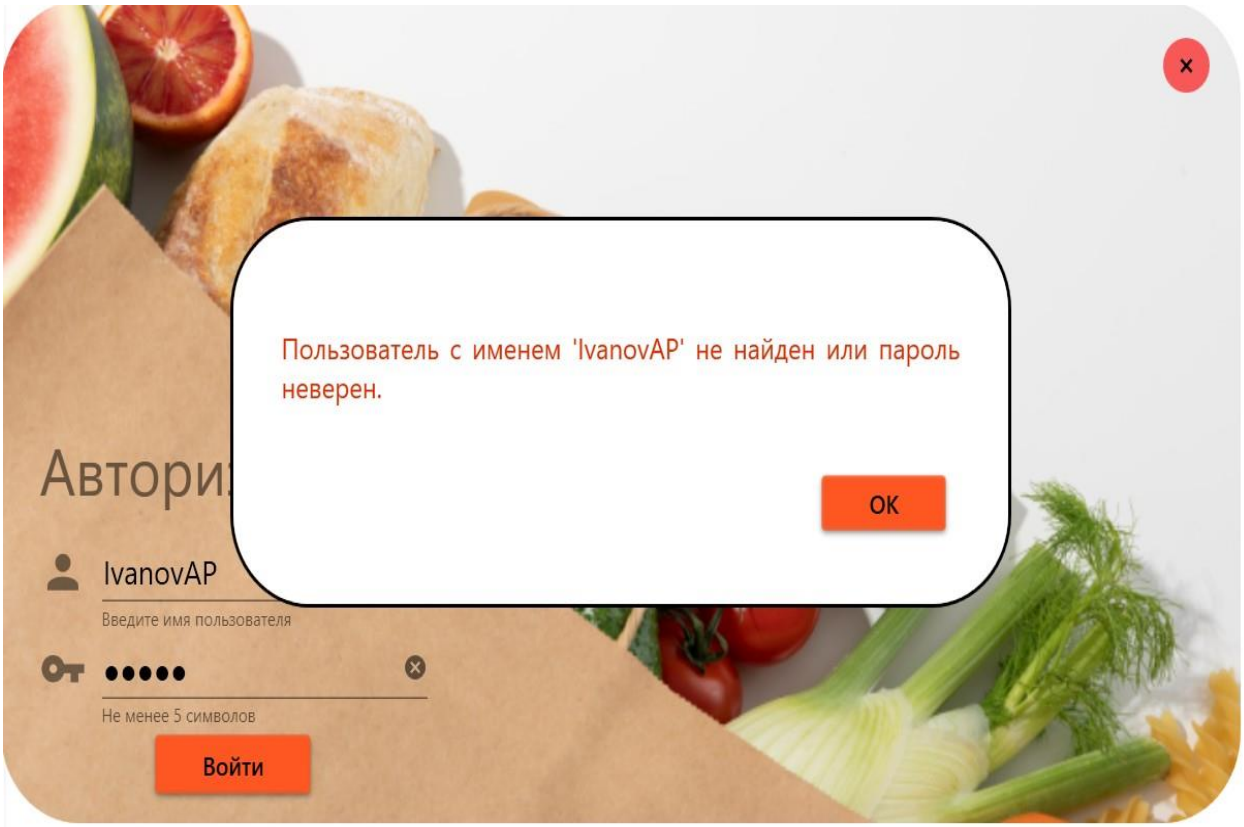

Рисунок 17 – Тестирование прохождения авторизации

После прохождения авторизации пользователь получает данные заказов,

которые нужно назначить курьерам. Проверка корректности обработки входящих данных будет осуществляться на основе отсутствия выбранного курьера. Система при отсутствии данных курьера должна закрыть доступ к активности кнопки «Назначить курьера», рисунок 18.

Рисунок 18 – Тестирование активности кнопки «Назначение курьера»

После того как будет выбран курьер, кнопка должна быть активна для возможности пользователя подтвердить выбранного курьера, рисунок 19.

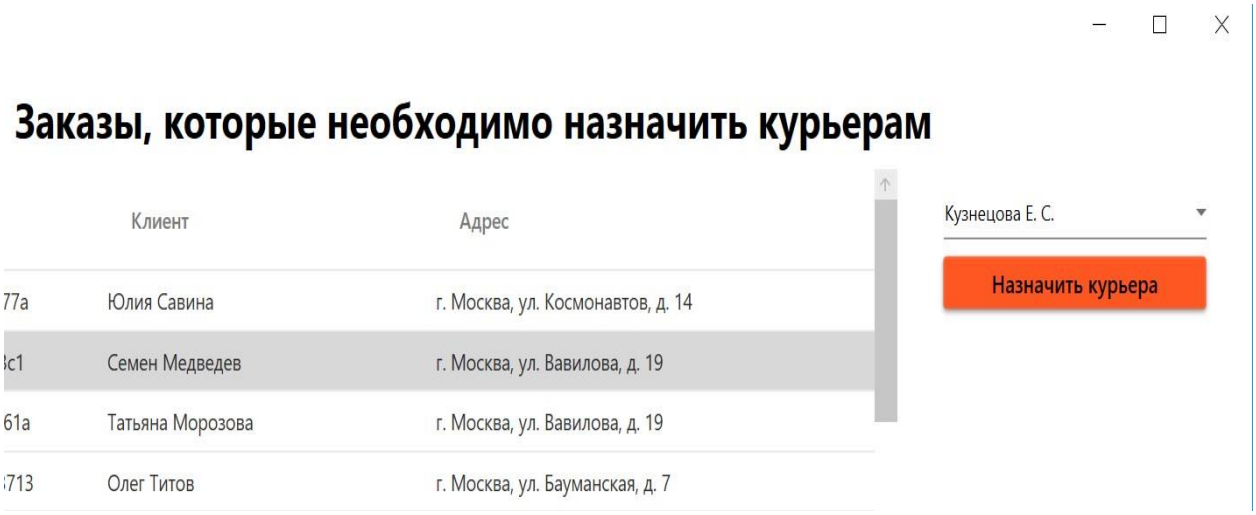

Рисунок 19 – Тестирование активности кнопки «Назначение курьера»

Подтверждением назначения заказа система должна оповещать пользователя сообщением, рисунок 20.

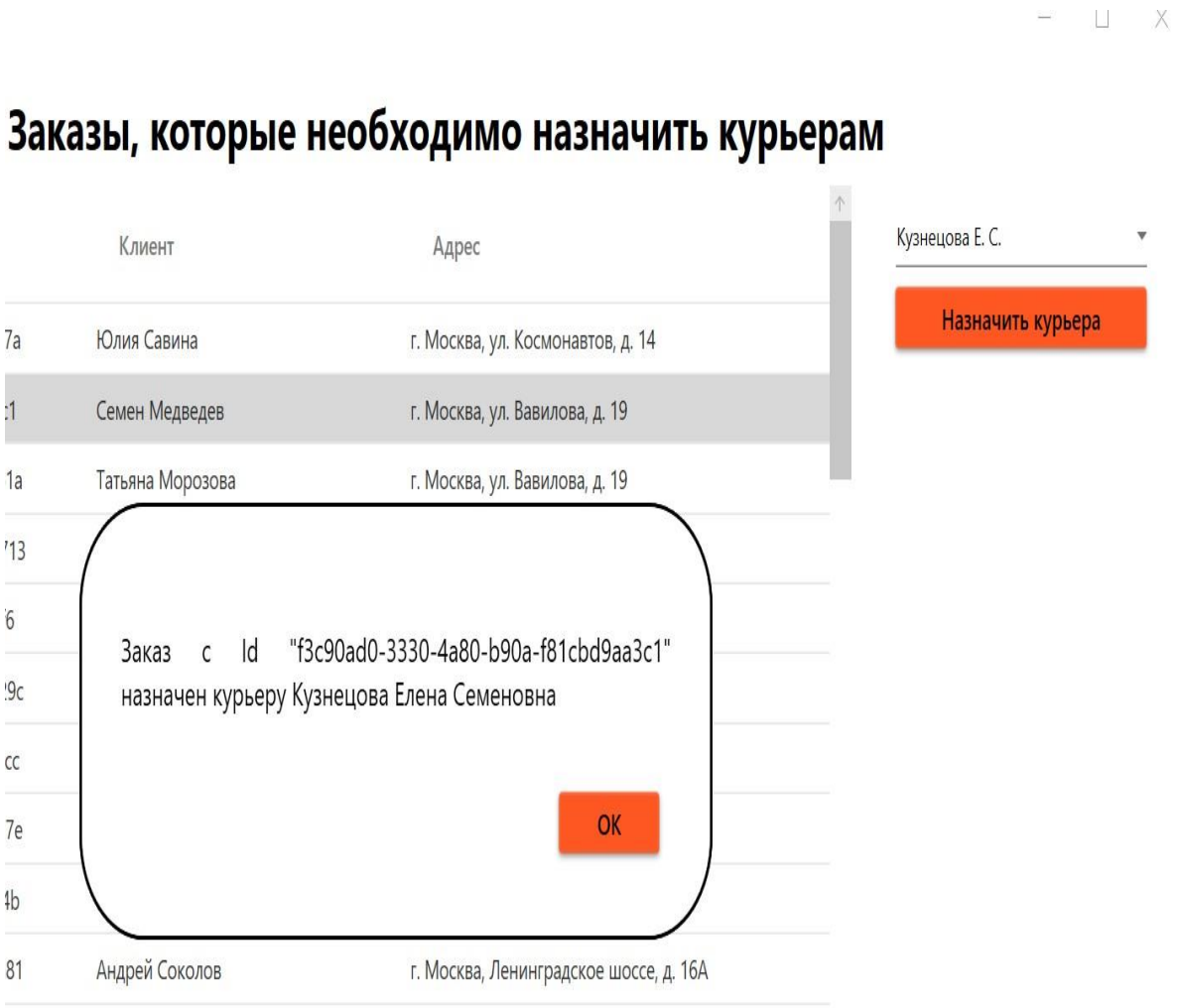

Рисунок 20 – Сообщение пользователю о назначенном заказе

Проведённое тестирование позволили проверить корректность выполнения основных функций программного обеспечения информационной системы [26].

В ходе проведенного тестирования были выявлены ошибки в сроке авторизации при входе в систему, что было исправлено на этапе тестирования. Эти меры обеспечили более стабильную и эффективную работу системы, что положительно сказалось на общем пользовательском опыте.

Выводы по третьей главе

В ходе реализации и тестирования ПО ИСУ службой доставки заказов было выполнено ряд задач.

Выбор инструментов для разработки программного обеспечения был основан на требованиях и предпочтениях разработчика, что позволило эффективно приступить к работе над ПО ИСУ.

Разработка логической структуры БД была проведена с учётом требований к хранению и обработке данных в ИСУ, что обеспечило эффективное функционирование системы.

Проектирование и разработка пользовательского интерфейса включали в себя создание удобного и интуитивно понятного интерфейса. Отладка программного обеспечения была проведена для выявления и исправления ошибок и недочётов в работе системы, что обеспечило её стабильную и безопасную работу.

Заключение

В первой главе проводится постановка задачи, которая включала рассмотрение функциональных и архитектурных особенностей информационных систем управления службой доставки. На основе поставленной задачи был проведен анализ аналогичных программных продуктов, представленных на рынке, что сподвигло к собственной разработке.

На базе функциональных и архитектурных особенностей процессов службы доставки заказов построены диаграммы, где показаны как должен осуществляться процесс управления доставки продуктов питания. Очевидным минусом, который был выявлен в ходе анализа оказался не автоматизированный процесс управления доставки, что стало актуальным для разработки программного продукта.

Во второй главе осуществляются проектные решения, на основе построенных диаграмм процесса и предложены меры по автоматизации процесса управления службой доставки продуктов питания.

В ходе выбора архитектуры программного обеспечения и проектирования логики были представлены диаграммы компонентов и пакетов, показывающие взаимодействие модулей системы. Для хранения данных были определены необходимые таблицы, показана схема модели данных и разработаны атрибуты с определёнными типами данных. На основе проведённого анализа языков программирования высокого уровня и систем управления данными были выбраны инструменты разработки необходимые для физической реализации программного обеспечения ИСУ.

В третьей главе была проведена реализация ПО ИСУ службой доставки заказов. Для успешного и безопасного использования системы было проведено тестирования корректной работы системы и обработки входящий данных, выявленные недочеты и ошибки были устранены в ходе тестирования. Программное обеспечение информационной системы управления службой доставки готов к внедрению в компанию.

Список используемой литературы и используемых источников

1. Бережливое производство [Электронный ресурс]. - Режим доступа: https://www.kickidler.com/ru/info/vnedrenie-berezhlivogo-proizvodstva-lean-na-chto-rasschityivat-i-k-chemu-gotovitsya.html (дата обращения: 01.10.2024).

2. Как использовать Entity Framework [Электронный ресурс]. – URL: https://skillbox.ru/media/code/entity_framework/ (дата обращения: 01.10.2024).

3. Концептуальное логическое и физическое моделирование данных [Электронный ресурс]. – URL: https://frameworx.ru/SID/datamodelling.html (дата обращения: 01.10.2024).

4. Мегалогист [Электронный ресурс] – URL: https://mega-logist.ru/ (дата обращения: 03.10.2024).

5. Методологии разработки ПО: Kanban [Электронный ресурс] – URL: https://gb.ru/posts/kanban_howto (дата обращения: 01.10.2024).

6. Моделирование систем с использованием информационных технологий: учебн. пособие / В. Г. Лисиенко, Н. Г. Дружинина, О. Г. Трофимова, С. П. Трофимов. – Екатеринбург: УГТУ-УПИ, 2009. – 440 с.

7. Онлайн курс обучения программированию: методологии разработки [Электронный ресурс]. – URL: https://javarush.ru/groups/posts/647-metodologii-razrabotki-po (дата обращения: 01.10.2024).

8. Отличия, достоинства и недостатки базы данных PostgreSQL: что такое PostgreSQL [Электронный ресурс]. – URL: https://oracle-patches.com/common/3214-что-такое-postgresql (дата обращения: 01.10.2024)

9. Попова-Коварцева, Д. А. Основы проектирования баз данных: учеб. пособие / Д.А. Попова-Коварцева, Е.В. Сопченко. – Самара: Изд-во Самарского университета, 2019. – 112 с.

10. C GUI на C#: Кроссплатформенное приложение [Электронный ресурс]. – URL: https://skillbox.ru/media/code/ne_windows_(дата обращения: 01.10.2024).

11. Что такое Scrum? [Электронный ресурс]. – Режим доступа:

https://www.atlassian.com/ru/agile/scrum (дата обращения: 01.10.2024).

12. Agile. Гибкие методологии. [Электронный ресурс]. - Режим доступа: https://www.e-xecutive.ru/wiki/index.php/Agile._Гибкие_методологии (дата обращения: 01.10.2024).

13. Choy, K. L. Development of an intelligent customer-supplier relationship management system: the application of case-based reasoning / K. L. Choy, Kenny K H Fan, Victor Lo // Industrial Management & Data Systems. – 2003. – Vol. 103, No. 4. – P. 263-274.

14. GeekBrains – Язык программирования C# [Электронный ресурс]. – URL: https://geekbrains.ru/posts/yazyk-programmirovaniya-c-sharp-istoriya-specifika-mesto-na-rynke (дата обращения: 01.10.2024).

15. Helpiks.org: Достоинства и недостатки языка [Электронный ресурс]. - URL: https://helpiks.org/6-21879.html (дата обращения: 01.10.2024).

16. Java-энциклопедия языков программирования: Java. [Электронный ресурс]. – URL: http://progopedia.ru/language/java/ (дата обращения: 01.10.2024).

17. Law, Y. F. D. An Integrated Case-Based Reasoning Approach for Intelligent Help Desk Fault Management / Y. F. D. Law, B. F. Sew, S. E. J. Kwan // Expert Systems with Applications. – 1997. – Vol. 13, No. 4. – P. 265-274.

18. MeaSoft [Электронный ресурс]. – URL: https://measoft.ru/measoftproduct/ (дата обращения: 03.10.2024).

19. Mobidel [Электронный ресурс] – URL: https://mobidel.ru/ (дата обращения: 03.10.2024).

20. Model-View-ViewModel (MVVM). [Электронный ресурс] – URL: https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm (дата обращения: 03.10.2024).

21. SoftClipper: что такое SQL Server. [Электронный ресурс] – URL: https://softclipper.net/foxpro-i-sql/sravnenie-baz-dannykh-microsoft-sql-server-i-microsoft-visual-foxpro.html (дата обращения: 01.10.2024).

22. SQL Server: руководство по SQL Server [Электронный ресурс]. –

URL: https://docs.microsoft.com/ru-ru/sql/sql-server/tutorials-for-sql-server-2016?view=sql-server-ver15 (дата обращения: 01.10.2024).

23. SQL Server: программное обеспечение [Электронный ресурс]. – URL: https://www.microsoft.com/ru-ru/sql-server/sql-server-downloads (дата обращения: 01.10.2024).

24. C# [Электронный ресурс]. – URL: https://dotnet.microsoft.com/en-us/languages/csharp (дата обращения: 01.10.2024).

25. The C Sharp (C#) Beginner’s Guide [Электронный ресурс]. – URL: https://medium.com/c-sharp-language/the-c-beginners-guide-6a14af03ed85 (дата обращения: 01.10.2024).

26. Visual Studio 2022 [Электронный ресурс]. – URL: https://www.techspot.com/downloads/7493-visual-studio-2022.html (дата обращения: 01.10.2024).

Приложение А

Листинг программы

```
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Hosting;
using System;
using System.IO;
using System.Windows;
namespace FoodDelivery.WpfApplication
{
   public static class Program
   {
      [STAThread]
      public static void Main()
      {
         try
         {
            var app = new App();
            app.InitializeComponent();
            app.Run();
         }
         catch (Exception e)
         {
            try
            {
               using var sw = new StreamWriter($"./startupErrors.txt", false,
System.Text.Encoding.UTF8);
               sw.WriteLine(e.Message);
            }
```

```
 catch (Exception ex)
            {
               MessageBox.Show(ex.Message);
            }
            MessageBox.Show(e.Message);
         }
      }
      public static IHostBuilder CreateHostBuilder(string[] arg)
      {
         if (App.CurrentDirectory is null)
         {
            throw new InvalidOperationException("App.CurrentDirectory is null");
         }
         var hostBuilder = Host.CreateDefaultBuilder(arg);
         hostBuilder.UseContentRoot(App.CurrentDirectory);
         hostBuilder.ConfigureAppConfiguration((host, cfg) =>
         {
            if (App.IsDesignMode)
            {
               cfg.SetBasePath(App.CurrentDirectory);
            }
            else
            {
               cfg.SetBasePath(Environment.CurrentDirectory);
            }
            cfg.AddJsonFile("appsettings.json", optional: false, reloadOnChange:
true);
```

```
});
            hostBuilder.ConfigureServices(App.ConfigureServices);
            return hostBuilder;
        }
    }
}
namespace FoodDelivery.Infrastructure
        //public override void Dispose()
        //{
        //    // Печатаем хэш-код в отладочную информацию
        //        Debug.WriteLine($"FoodDeliveryDbContext с хэшем {_hashCode}
уничтожается.");
        //    base.Dispose();
        //}
 protected override void OnModelCreating(ModelBuilder builder)
        {
            base.OnModelCreating(builder);
builder.ApplyConfigurationsFromAssembly(Assembly.GetExecutingAssembly());
        }
    }
}
namespace FoodDelivery.ConsoleApp
{
    internal class Program
    {
        #region Host
        private static object _singletonLocker = new Object();
private static IHost? _host;
```

```
«public static IHost Host
 {
    get
    {
       if (_host == null)
       {
          lock (_singletonLocker)
          {
             if (_host == null)
                _host                                                    =
CreateHostBuilder(Environment.GetCommandLineArgs()).Build();
          }
       }
 return _host;
    }
 }
 public static IServiceProvider Services => Host.Services;
 public static IHostBuilder CreateHostBuilder(string[] arg)
 {
    var                                hostBuilder                       =
Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(arg);
    hostBuilder.UseContentRoot(Environment.CurrentDirectory);
    hostBuilder.ConfigureAppConfiguration((host, cfg) =>
    {
       cfg.SetBasePath(Environment.CurrentDirectory);
cfg.AddJsonFile("appsettings.json", optional: false, reloadOnChange: true);
    });» [25]
    hostBuilder.ConfigureServices(ConfigureMyServices);
```

```
        return hostBuilder;
    }
    public static void ConfigureMyServices(HostBuilderContext host,
IServiceCollection services)
    {
        services.RegisterServices();
        services.AddDatabase(host.Configuration.GetSection("Database"));
    }
 #endregion
    static async Task Main(string[] args)
    {
        try
        {
            using var scope = Host.Services.CreateScope();
            var                                dbInitializer                                =
scope.ServiceProvider.GetRequiredService<DbInitializer>();
            await dbInitializer.Initialize();
        }
        catch (Exception e)
        {
            ConsoleExtensions.WriteLineError(e, "An error occurred");
        }
        finally
        {
            await ConsoleExtensions.WriteFooterAndWaitToEnterAsync();
        }
    }
  }
}
```