Пермский филиал федерального государственного
автономного образовательного учреждения высшего
образования
«Национальный исследовательский университет
«Высшая школа экономики»

Факультет социально-экономических и компьютерных наук

Можегова Анна Сергеевна

Информационная система зоомагазина

Курсовая работа

студента образовательной программы «Программная инженерия» по направлению подготовки *09.03.04 Программная инженерия*

Руководитель
Приглашенный
преподаватель
кафедры
информационных
технологий в бизнесе

_____

Г.И. Рустамханова

Пермь, 2023 год

Оглавление

Введение

Мы живем в мире, где существует большое количество информации, которое невозможно хранить и анализировать в бумажном виде. Многие организации стараются развиться, поэтому у них есть потребность в приложениях для ведения учёта продуктов, персонала, данных о заказах и многое другое.

Сейчас существуют некоторые решения, связанных с торговлей. Однако не все из них являются доработанными, зачастую для автоматизации рабочего места необходимо несколько приложений, которые требуют обслуживание и определенный персонал в лице техподдержки.

Задачами информационной системы являются предоставление информации об клиентах, персонале, продукции, заказах и итогах, также можно просмотреть информацию о закрытых сменах. В зависимости от роли, под которой авторизуется человек он сможет либо просматривать, либо редактировать, удалять и добавлять те или иные данные. Подразумевается, что разработанная система будет использоваться сотрудниками и клиентами зоомагазина.

Объект исследования – это процесс реализации информационной системы. Предмет исследования – система для автоматизации рабочего места персонала зоомагазина и информирования пользователя.

Целью работы является создание информационной системы для автоматизации рабочего места управляющего и продавца зоомагазина, а также для предоставления актуальной информации пользователю.

Задачами, которые необходимо выполнить для достижения поставленной цели, являются:

– поиск и обзор существующих решений;

– проектирование базы данных;

– разработка программного кода Windows-приложения на языке программирования C#;

– тестирование и отладка приложения.

Реализация системы выполняется в среде программирования Microsoft Visual Studio на языке программирования C#. Сама база данных хранится в СУБД PhpMyAdmin. Тестирование приложения происходит по критериям черного ящика.

Глава 1 Изучение существующих решений

Прежде чем перейти к анализу существующих решений необходимо провести описание предметной области.

Торговая точка ведет свою деятельность для получения прибыли. В магазине существует целый спектр различных товаров, которые могут заинтересовать клиента.

В основные обязанности продавца входит информирование клиентов об товарах, которые имеются в наличии, ведение учета товаров и формирование итогов смены. Управляющий же обязан формировать из отчетов об имеющихся товарах лист для поставок, вести ведомости с личными данными продавцов, подводить итоги по продажам. А клиент заинтересован в покупке, но при этом ему важно, чтобы его корректно проинформировали об товарах. Владелец торговой точки всегда нацелен на клиентов, это значит, что для него важен комфорт, а в последние годы у многих компаний и магазинов появились функции для формирования заказа дистанционно с последующей доставкой.

Для увеличения прибыли и клиентов, а также для автоматизации некоторых процессов можно использовать информационную систему, в которую будут включены такие акторы как управляющий, продавец и клиент. Функционал системы будет нацелен на все основные обязанности и интересы акторов, например, у продавца всегда будет актуальная информация о товарах это поможет не только упростить работу сотруднику, но и улучшить взаимодействие с клиентами, так же формирование поставок у управляющего значительно облегчатся.

1.1. Описание аналогов

В данном пункте детально рассматриваются такие существующие информационные системы, как «АРМ Кассир», «АРМ Менеджер», продукты «1С» и сайт «Четыре лапы». Далее приведены основные преимущества и недостатки, которые затем сравниваются между собой, для того чтобы выдвинуть требования.

1.1.1. Обзор «АРМ Кассир» и «АРМ Менеджер»

АРМ Кассир и АРМ Менеджер являются комплексным программным обеспечением, в котором представлена автоматизация рабочего места соответственно кассира и менеджера как на предприятии, так и на других уровнях торговли.

Для менеджера программное обеспечение идеально подходит для приема и учетом товарных остатков, потому что имеются функции для ведения отчетов и печатных форм. Кассир получается большой спектр пакетов автоматизации процессов, таких как сканеры штрихкода, банковские терминалы, фискальные регистраторы и многое другое.

Имеется и ряд общих преимуществ данных разработок. Самое главное это удобный, минималистичный и интуитивно понятный интерфейс. Это означает, что у нового персонала обучение не займет много времени, кроме того, оно может происходить уже на первых днях на предприятии, потому что нет необходимости проводить какой-либо теоретический инструктаж. Присутствует возможность отображения всех товаров, также остатков и другой необходимой информации в зависимости от должности. Так же можно учесть все виды оплаты и сделать возврат товара. Программа предусматривает возможность рассылки необходимой информации на электронную почту и производит выгрузку статистики по продажам за период.

Недостатками можно выделить отсутствие закрытия смены для продавцов, т.е. это производиться в отдельном приложении, что задерживает сотрудников и затрудняет выгрузки данных из базы. В самом обеспечении предусмотрена печать документов различных товаров, но формирования файла для экспорта не предусмотрено. Нельзя так же не отменить, что данные аналоги не застрахованы от потери данных, все хранится в электронной форме, а восстановления или резервного авто сохранения не предусмотрено, а это означает, что другие вредоносного ПО могут сделать изменения в данных или удалить все. Еще имеются сбои в операционных системах, имеющиеся программы и функционал компьютера не способен состыковаться с данным комплексом программного обеспечения. Это приводит к фатальным ошибкам и выбросам из системы или приложения.

От всех аналогов данная разработка отличается простой использования, здесь имеется хороший функционал для начала свое предпринимательского дела, но с развитием предприятия или бизнеса лучше выбирать более продвинутые модели.

1.1.2. Обзор продуктов «1С»

Самый известный пример автоматизации рабочего процесса и информационной системы являются разработки компании 1С. Большинство программных продуктов компании 1С представляют из себя сложные системы, которые могут развиваться в процессе работы с ней.

Рассмотрим несколько приложений, а именно «1С: Розница», «1С: Предприятие» и «1С: Управление торговлей».

Почему именно они? Дело в том, что 3 этих программных обеспечения удовлетворяют одним и тем же требованиям:

– интеграция с распространенными CMS;

– работа с кассовым оборудованием, сканерами, терминалами сбора данных, принтерами и др.;

– взаимодействие с ЕГАИС и системой Маркировка.

Несмотря на это, эти программные обеспечения отличаются друг от друга, поэтому я проведу краткий обзор на каждое из них.

1С: Розница является третьей по популярности конфигурацией, которая можно сказать является «ребенком» программного продукта 1С: Управление торговлей, так как представляет из себя производную программу от вышеупомянутой. Разработано обеспечение было в целях автоматизации розничной торговли и используется как кассовая программа, как не странно, в розничных магазинах. Программа имеет два направления: товарно-складской учет и учет обращения наличных денежных средств, именно это делает ее универсальным решением для управления розничной торговлей.

Достоинством, которое нужно выделить в первую очередь, является то, что этот продукт может использоваться как в больших розничных, как и в маленьких магазинах, кроме того, его можно внедрять и в пункты выдачи заказов.

В крупных магазинах 1С: Розница помогает организовать рабочее место для кассиров и способна выполнять базовые операции для обслуживания клиентов,

например, открывать и закрывать кассовые смены, принимать оплату и осуществлять возвраты. Существует и дополнительный бонус, с помощью которого конфигурация способна заменять маркетинговые программы для стимулирования сбыта и проводить анализ их эффективности.

В маленьких розничных точках также происходит автоматизация процессов учета на кассе, что помогает вести закупки и управлять складом. Так же из бонусов можно заметить, что система способна осуществлять проведение маркетинговых акций и формирование аналитики продаж.

В пунктах выдачи заказов система позволяет автоматизировать отгрузку заказов клиентам.

Недостатками пользователи выделяют отсутствие формирования сложный отчетов через запросы, отсутствие техподдержки, не имеется широкого функционала в режиме кассира, система предъявляет высокие требования к ПО, и сама система управления взаимоотношениями с клиентами не доработана из-за чего находится на низком уровне. Получается, что потенциальный потребители системы отлично могут автоматизировать свой магазин и склад, но работу продавца и взаимодействие с клиентами будет низкое.

«1С: Управление торговлей» занимает второе место по популярности, представляет из себя вспомогательный инструмент для повышения эффективности бизнеса любого торгового предприятия. Система на предприятии полезна тем, что она в комплексе помогает автоматизировать задачи оперативного и управленческого учета, можно проводить анализ и планирования торговых операций.

Преимущества программного обеспечения заключается в следующем. Во-первых, в отличии от «1С: Розница» оно поддерживает не один конкретный вид торговли, а несколько основных видов: розничную, оптовую, в кредит, по предварительному заказу, комиссионную, это делает программу более подходящей большинству. Во-вторых, имеется функция для оформления и формирования практически всех первичных документов торгового и складского учета, а также документов движения денежных средств. В-третьих, «1С: Управления торговлей» может подключаться к другим системам, например, к «1С: Бухгалтерию», это позволяет делать выгрузку данных бухгалтерского учета. В-четвертых, в настройках

можно отключить модули, которые не нужны, допустим, если ваше предприятие мало, то расширенный функционал может лишь путать сотрудников.

В общих чертах можно выделить еще управления товародвижением и ценообразованием, приема заказов и контроля их исполнения, оптимизации складских запасов, анализа товарооборота, планирования закупок и поставок.

Недостатки заключаются в том, что в программу «1С:УТ» не добавляли функционал для учета производственных операций, громоздкий функционал требует большого времени для изучения, для кого-то возможность выгрузки или расчет заработной платы, учет больничных в дополнительных приложениях является удобным функционалом, некоторые же предпочли все делать в одной системе.

«1С: Предприятие» не имеет большой популярности, но у системы есть интересный и полезный функционал. Данная программа подключается с информационными базами данных организации. Из функционала в ней можно вводить данные, заполнять справочники, формировать отчеты и запускать обработки, но нельзя менять структуру конфигурации.

Основными плюсами является быстрый запуск, простота обновлений, которые выполняются автоматически, высокая скорость работы за счёт мощного сервера, возможность работать удалённо с данными, а также существует разграничение прав доступа, что позволяет работать нескольким должностям в одном приложение и при этом у них будет свой функционал в зависимости от роли.

Минусами является то, что ваше предприятие или торговая точка зависима от подключения к интернету. Обновления хоть и автоматические, но платные. Потребуется длительное время для установки и настройки компонентов, существует множество технических ошибок.

Подводя итог из всего вышесказанного продукты от компании 1С популярны и в зависимости от приложения имеют различный функционал, но, в целом, у них есть одни и те же проблемы, например, для более широкого функционала необходимо скачивать дополнительные приложения, маленький функционал для отдельных должностей, при внедрении Вы будете нуждаться в обучении и в техподдержке со специалистами, которых нужно будет иметь в своем штате, это может ударить по бюджету вашему предприятию.

1.1.3. Обзор интернет-магазин «Четыре лапы»

Интернет-магазины также являются информационными системы, как правило, они создаются исключительно для клиентов и представляют из себя самую продвинутую форму информационных систем. Любой качественно выполненный сайт является информационным ресурсом компании, он помогает передать всю необходимую информацию о товарах и услугах, предоставляемых организацией, таким образом поддерживается постоянный контакт с клиентом. Кроме того, красивый сайт может разрекламировать свою торговую точку, товары, услуги. Если руководитель фирмы или торговой точки заинтересован в привлечении внимания клиентов, то сайты значительно помогают ему в этому.

«Четыре лапы» — это один из самых известных и популярных онлайн-магазинов, рассмотрим преимущества и недостатки.

Сайт, спроектирован и создан профессионалами предоставляет легкость в освоении и поиске необходимой информации. Возможность постоянного контакта с клиентами дает шанс оперативно реагировать на потребности пользователей и проводить коррекцию. Так как «Четыре лапы» содержит в себе справочную информацию, он предоставляют развернутый ответ почти по каждому запросу. Для пользователя сейчас это привлекательно, поскольку клиентам удобнее узнать всю необходимую информацию, не выходя из дома. Еще плюсом является то, что на сайте не только можно сделать заказ из этого магазина, но и можно записаться в ветцентры или на груминг.

Как недостатки можно выделить то, что для новых пользователей сайт может показаться сложным поскольку на главном окне размещено большое количество информации из-за чего глаза разбегаются. А владельцы маленьких фирм могут столкнуться с неприятностями стоимости создания и сопровождения сайта, этот вопрос можно решить, разместив рекламу на своем сайте, но как правило это лишь уменьшает количество потребителей, раздражает и не приносит большое количество прибыли от рекламы.

1.2. Сравнение существующий решений

Сформируем все вышесказанное в один итог, для этого необходимо провести сравнение рассмотренных аналогов. Курсовая работа нацелена на реализацию информационной системы с широким функционалом как для работников торговой точки, так и для клиентов. Должна быть реализована выгрузка важных данных в файл Excel, а также возможность делать закрытие смены. Большим плюсом будет простой и понятный интерфейс, не требующий долгого обучения.

На основе описанных критериев сделана таблица 1, в которой запечатлена оценка того или иного критерия по пятибалльной шкале, где 5 обозначает, что в данной программе или сайте существует полностью реализованный функционал, а 0 это полное отсутствие. Аналоги отмечены цифрами от 1 до 5, где порядок таков: АРМ Кассир и АРМ Менеджер, 1С: Розница», «1С: Предприятие», «1С: Управление торговлей», «Четыре лапы».

Таблица 1 – Сравнительная таблица существующих решений

Решение	1	2	3	4	5
Несколько прав доступа	0/5	0/5	2/5	3/5	0/5
Выгрузка данных	2/5	1/5	3/5	0/5	2/5
Закрытие смены	2/5	4/5	0/5	0/5	0/5
Простой интерфейс	4/5	4/5	2/5	3/5	4/5
Формирование итогов	1/5	2/5	4/5	4/5	0/5

Таким образом, стоит ориентироваться на работки компании 1С, которые необходимо дополнить возможность работы нескольких должностей в одном приложении и закрытием смены. Положительные стороны других существующих решений тоже будут учитываться.

Глава 2 Проектирование базы данных

В данной главе описываются все этапы проектирования базы данных для информационной системы, а именно концептуальный этап, на котором необходимо создать концептуальную модель, основываясь на выбранную предметную область, а именно нужно представить все основные сущности и связи между ними, на данном этапе нет необходимости учитывать никакие особенности СУБД; логический этап, на котором описывается развитие предыдущего этапа с учетом принимаемой модели БД; физический этап проектирования, на нем делают преобразование логической схемы, учитывая особенность, семантику и синтаксис выбранной СУБД.

2.1.Концептуальный этап

Разрабатываемая база данных должна включать в себя информацию о магазине, об владельце, данные сотрудников, а также данные клиентов, нужно выделить отдельные таблицы для товаров, закрытия смены, особо важной информации сотрудников/клиентов и заказов. Исходя из данной информации можно выделить следующие объекты, их атрибуты и первичные ключи. Выделенные объекты и атрибуты представлены в таблице 2

Таблица 2 – Объекты и атрибуты

Объект	Атрибут	Первичный ключ
Магазин	Код магазина Контактный телефон Адрес Код директора	Код магазина
Директор	Код директора Фамилия Имя Отчество	Код директора
Должности	Код должности Название должности	Код должности

Продавцы	Код продавца Фамилия Имя Отчество Пол Код должности Контактный номер телефона Дата рождения Текущий сотрудник или бывший Зарплата Премия	Код продавца
Управляющие	Код управляющего Фамилия Имя Отчество Пол Код должности Стаж продавца Контактный номер телефона Дата рождения Текущий сотрудник или бывший Зарплата Премия	Код управляющего
Клиент	Код клиента Фамилия Имя Отчество Пол Контактный номер телефона Адрес электронной почты	Код клиента
Товары	Код товара Производитель Наименования Категория Наличие Цена	Код товара
Заказ(новый)	Код заказа Код покупателя Код клиента Описание заказа Итого	Код заказа

Заказ(завершенный)	Код заказа Код покупателя Код клиента Описание заказа Итого	Код заказа
Данные управляющего	Код управляющего Паспорт Логин Пароль	Код управляющего
Данные продавца	Код продавца Паспорт Логин Пароль	Код продавца
Данные покупателя	Код покупателя Паспорт Логин Пароль	Код покупателя
Закрытие смены	Код смены Наличный расчет 1 Наличный расчет 2 Итог по наличному расчету Сбербанк 1 Сбербанк 2 Итого по безналичному Инкассация Количество5000 Количество1000 Количество500 Количество100 Количество50 Мелочь	Код смены
График работы	Код продавца Описание	Код продавца

Проведем спецификацию объектов. (см. таблицу 3)

Таблица 3 – Спецификация

Объект	Атрибут	Первичный ключ
Магазин	Код магазина	Идентификационный атрибут
	Название	описательный атрибут
	Адрес	описательный атрибут
	Контактный телефон	описательный атрибут
	Код директора	описательный атрибут

Должности	Код должности	Идентификационный атрибут
	Название должности	описательный атрибут
Директор	Код директора	Идентификационный атрибут
	Фамилия	описательный атрибут
	Имя	описательный атрибут
	Отчество	описательный атрибут
Продавцы	Код продавца	Идентификационный атрибут
	Фамилия	описательный атрибут
	Имя	описательный атрибут
	Отчество	описательный атрибут
	Пол	описательный атрибут
	Код должности	Идентификационный атрибут
	Контактный номер телефона	описательный атрибут
	Текущий сотрудник или бывший	описательный атрибут
	Зарплата	описательный атрибут
	Премия	описательный атрибут
Данные покупателя	Код покупателя	Идентификационный атрибут
	Паспорт	описательный атрибут
	Логин	описательный атрибут
	Пароль	описательный атрибут
Данные управляющего	Код управляющего	Идентификационный атрибут
	Паспорт	описательный атрибут
	Логин	описательный атрибут
	Пароль	описательный атрибут

Клиент	Код клиента	Идентификационный атрибут
	Фамилия	описательный атрибут
	Имя	описательный атрибут
	Отчество	описательный атрибут
	Пол	описательный атрибут
	Контактный номер телефона	описательный атрибут

	Адрес электронной почты	описательный атрибут
Товары	Код товара	Идентификационный атрибут
	Производитель	описательный атрибут
	Наименование	описательный атрибут
	Категория	описательный атрибут
	Наличие	описательный атрибут
Данные продавца	Код продавца	Идентификационный атрибут
	Паспорт	описательный атрибут
	Логин	описательный атрибут
	Пароль	описательный атрибут
Управляющие	Код управляющего	Идентификационный атрибут
	Фамилия	описательный атрибут
	Имя	описательный атрибут
	Отчество	описательный атрибут
	Пол	описательный атрибут
	Код должности	Идентификационный атрибут
	Стаж продавца	описательный атрибут
	Контактный номер телефона	описательный атрибут
	Текущий сотрудник или бывший	описательный атрибут
	Зарплата	описательный атрибут
	Премия	описательный атрибут
Заказ	Код заказа	Идентификационный атрибут
	Код покупателя	Идентификационный атрибут
	Описание заказа	описательный атрибут
	Итого	описательный атрибут
Закрытие смены	Код смены	Идентификационный атрибут
	Наличный расчет 1	описательный атрибут
	Наличный расчет 2	описательный атрибут
	Итого по безналичному	описательный атрибут
	Сбербанк 1	описательный атрибут
	Сбербанк 2	описательный атрибут
	Итого по безналичному	описательный атрибут
	Купюры5000	описательный атрибут
	Купюры1000	описательный атрибут
	Купюры500	описательный атрибут
	Купюры100	описательный атрибут
	Купюры50	описательный атрибут
	Мелочь	описательный атрибут
График работы	Код продавца	Идентификационный атрибут
	Описание	описательный атрибут
Новый заказ	Код нового заказа	Идентификационный атрибут
	Название1	описательный атрибут
	Производитель1	описательный атрибут
	Количество1	описательный атрибут

	Название2	описательный атрибут
	Производитель2	описательный атрибут
	Количество2	описательный атрибут
	Название3	описательный атрибут
	Производитель3	описательный атрибут
	Количество3	описательный атрибут
	Номер клиент	описательный атрибут

Следующим этапом является представление связи между таблицами. Они обозначены в таблице 4.

Таблица 4 – Спецификация связей

Наименование связи	**Объекты**	**Показатель кардинальности**	**Степень участия**
ВЛАДЕЕТ	Директор Магазин	1:1	Полная Полная
НАНИМАЕТ	Магазин Продавец	1:M	Полная Полная
ВЫБИРАЕТ	Магазин Управляющий	1:1	Полная Полная
ВКЛЮЧАЕТ В СЕБЯ	Магазин Товар	1:M	Полная Полная
ИМЕЕТ	Продавец Данные продавца	1:1	Полная Полная
ОБСЛУЖИВАЕТ	Продавец Клиент	1:1	Полная Полная
ПРИСВАИВАЕТСЯ	Продавец Должность	1:1	Полная Полная
ЕСТЬ	Клиент Данные клиента	1:1	Полная Полная
ИМЕЕТ	Управляющий Данные управляющего	1:1	Полная Полная

ЗАПОЛНЯЕТ	Продавец Конец смены	1:1	Полная Полная
ФОРМИРУЕТ	Продавец Завершенный заказ	1:M	Частичная Полная
ПРОСМАТРИВАЕТ	Продавец Новые заказы	1:M	Частичная Частичная
ДОБАВЛЯЕТ	Клиент Новый заказ	1:1	Частичная Частичная
СОЗДАЕТ	Управляющий График	1:M	Частичная Полная

2.2.Логический этап

На данном этапе необходимо рассмотреть все функциональные зависимости и привести все таблицы к трем нормальным формам.

При установлении ***функциональных зависимостей*** при проектировании БД необходимо учесть следующие связи между атрибутами:

– **1ФЗ:** По коду магазина можно однозначно определить его название, адрес, телефон и код директора.

– **2ФЗ:** По коду директора можно однозначно определить его фамилию, имя и отчество.

– **3ФЗ:** По коду продавца можно однозначно определить фамилию, имя, отчество, пол, номер телефона, дату рождения, текущий ли это сотрудник, код должности, зарплату и премию.

– **4ФЗ:** По коду управляющего однозначно определить фамилию, имя, отчество, пол, номер телефона, дату рождения, текущий ли это сотрудник, код должности, зарплату и премию и его стаж работы.

– **5ФЗ:** По коду должности можно однозначно определить название должности.

– **6ФЗ:** По коду клиента можно однозначно определить его фамилию, имя, отчество, пол, адрес электронной почты, номер телефона и дату регистрации.

– **7ФЗ:** По коду товара можно однозначно определить производителя, цену на товар, название товара, категорию, наличие.

– **8ФЗ:** По коду нового заказа можно определить продукты, производителей и количество, которые заказал клиент, а также его номер телефона

– **9ФЗ:** По коду завершенного заказа можно однозначно определить код покупателя, код продавца, который принял заказ, описание и прибыль.

– **11ФЗ:** По коду продавца можно однозначно определить его данные такие как: паспорт, логин и пароль.

– **12ФЗ:** По коду управляющего можно однозначно определить его данные такие как: паспорт, логин и пароль.

– **13ФЗ:** По коду клиента можно однозначно определить его данные такие как: паспорт, логин и пароль.

– **14ФЗ:** По коду продавца можно определить описание его рабочей смены.

– **15ФЗ:** По коду смены можно определить инкассацию, суммы по кассам при наличном и безналичном расчете, отдельно покупюрник и количество денежных средств на той или иной кассе.

Отношение находится в 1НФ, если все его атрибуты являются простыми, все используемые домены должны содержать только скалярные значения. Не должно быть повторений строк в таблице. Исходя из всех атрибутов, которые у меня имеются, чтобы не было нарушения первой нормальной формы такие значения как ФИО покупателей/продавцов/управляющего должны храниться как отдельные атрибуты (нет никаких группировок данных): фамилия, имя, отчество. Кроме того, кортежи не могут повторяться, поэтому нельзя допустить таких ситуаций, как дублирование значений кортежей.

В соответствии с описанными выше функциональными зависимостями формируем все

первичный ключ, которые включает следующие атрибуты:

– Код магазина

– Код директора

– Код продавца

– Код управляющего

– Код должности
– Код клиента
– Код товара
– Код нового заказа
– Код завершенного заказа
– Код управляющего
– Код клиента
– Код смены

Значения атрибутов будут уникальными для каждой строки и не будут сгруппированы, тем самым не нарушат 1 нормальную форму, кроме того, они позволяют однозначно идентифицировать запись.

К большой избыточности данных и следующим аномалиям при выполнении операций приводит использование универсального отношения (Отношение, представленное в корректной форме, включающее все необходимые при эксплуатации БД атрибуты):

1. Аномалии ввода (Данные практически всех столбцов многократно повторяются. Повторяются и некоторые наборы данных). Такая ситуация может произойти, если, например, у меня будет управляющий, который не проработал 1 года, а в строке стаж необходимо что-то написать. Поэтому я даю возможность вводить нулевое значение.

2. Аномалии редактирования и аномалии удаления. Первое связано с избыточностью данных, что приводит к проблемам при их изменении. При изменении повторяющихся данных придется многократно изменять их значения, однако, если изменения будут внесены не во все кортежи, возникнет несоответствие информации, которое называется аномалией обновления, а второе возникает, когда происходит удаление не всех дублированных кортежей. Если соблюдать 1 НФ, то мы избежим повторяющиеся кортежи и эти аномалии.

Таблица (отношение) соответствует второй нормальной форме, если таблица приведена к первой нормальной форме 1НФ и в таблице все неключевые столбцы должны зависеть от полного ключа (в случае если он составной).

В моем случае нет составных ключей, поэтому нет необходимости проверять атрибуты

на 2НФ.

Отношение находится в 3НФ, когда находится во 2НФ и каждый не ключевой атрибут не транзитивно зависит от первичного ключа. Проще говоря, второе правило требует выносить все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы в отдельные таблицы.

В описанных выше отношениях отсутствуют зависимости между неключевыми атрибутами, следовательно, они находятся в 3НФ.

Например, если бы в таблице «Товары» существовали вместе атрибуты «Цена», «Код товара» и «Название товара», то атрибуты «Цена» и «Название товара» были бы зависимы не только от ключа, но и от друг друга. Поэтому данные значения находятся в различных таблицах (путем декомпозиции).

Примеры по следующим таблицам:

1. Магазин (Код магазина, Код директора, Адрес, Телефон)

Код магазина- первичный ключ, Код директора- это внешний ключ, который позволяющим определить директора выделенной таблице «Директор»

2. Директор (Код директора, Фамилия, Имя, Отчество)

Код директора- первичный ключ. Фамилия, имя и отчество не зависят друг от друга, а зависят ток от первичного ключа.

3. Управляющий (Код управляющего, Фамилия, Имя, Отчество, Пол, Дата рождения, Номер телефона, Текущий сотрудник, Зарплата, Премия, Стаж)

У управляющего первичный ключ Код управляющего. Так же все неключевые атрибуты не зависимы друг от друга. Например, Дата рождения не зависима от Фамилии.

4. Продавец (Код продавца, Фамилия, Имя, Отчество, Пол, Дата рождения, Номер телефона, Текущий сотрудник, Код должности, Зарплата, Премия)

Код продавца- первичный ключ, код должности- внешний ключ, который ссылается на таблицу «Должности».3НФ выполняется потому, что неключевые атрибуты независимы друг от друга, например, по номеру телефона я не могу определить фамилию или имя.

5. Клиент (Код клиента, Фамилия, Имя, Отчество, Адрес электронной почты, Телефон, Дата регистрации)

Код клиента- первичный ключ. От этого кода зависимы все неключевые

атрибуты. По адресу электронной почты или по какому-либо другому неключевому атрибуту я не могу определить другой неключевой атрибут- приведено к 3НФ.

6. Данные управляющего (Код управляющего, Паспорт, Логин, Пароль)

Код управляющего- первичный код. Паспорт не зависит от логина или пароля.

7. Данные продавца (Код продавца, Паспорт, Логин, Пароль)

Первичный ключ- код продавца, остальные неключевые атрибуты не зависимы друг от друга, то есть только по первичному ключу я могу определить логин, пароль или паспорт.

8. Данные клиента (Код клиента, Паспорт, Логин, Пароль)

Код клиента- первичный ключ. Аналогично с верхними двумя примерами все приведено к 3НФ.

9. Товар (Код товара, Производитель, Наличие, Категория, Наименование)

Товар имеет первичный ключ остальные неключевые атрибуты не зависимы друг от друга, то есть только по первичному ключу я могу определить другие атрибуты, поэтому приведено к 3НФ.

10. Завершенный заказ (Код заказа, Код клиента, Код продавца, Описание, Сумма)

Ключевым атрибутом является код заказа. Код клиента и код продавца внешние ключи, которые никак не зависят от друг друга или других неключевых атрибутов, значит, 3НФ соблюдена.

11. Новый заказ (Код заказа, Наименование 1 товара, Производитель 1 товара, Количество1, Наименование 2 товара, Производитель 2 товара, Количество2, Наименование 3 товара, Производитель 3 товара, Количество3, Номер телефона)

Неключевые атрибуты никак не зависимы друг от друга, поэтому требования выполняются.

12. Должность (Код должности, Название должности)

Код должности- первичный ключ, а название товара очевидно зависит лишь от первичного ключа- 3НФ соблюдена.

13. Закрытие смены (Код смены, Наличный расчет 1, Наличный расчет 2, Итого по наличному расчету, Сбербанк 1, Сбербанк 2, Итого по безналичному, Возврат, Инкассация, Купюры5000, Купюры1000, Купюры500, Купюры100,

Купюры50, Мелочь)

Код смены- первичный ключ, от которого зависят все неключевые атрибуты и при этом у них нет зависимости между собой.

14. График работы (Код продавца, Описание)

Код продавца- первичный ключ, а описание очевидно зависит лишь от первичного ключа- 3НФ соблюдена.

2.3.Физический этап

Теперь необходимо логическую модель перенести в выбранную СУБД, для разрабатываемой системы выбран PhpMyAdmin, но перед этим была проведена проверка, что СУБД поддерживает требуемые типы данных и способна адекватного их хранения. На стадии переноса важно соблюдать модификацию логической схемы с учетом семантики и синтаксиса, это значит, что нужно соблюдать правила наименования таблиц, атрибутов, типов данных. Важным этапом является определение прав доступа. На данный момент у меня имеется единственный пользователь, который имеет возможность делать все операции с БД, а именно, удалять, добавлять и редактировать как саму БД, так и отдельные таблицы.

Таким образом получается база данных, изображённая на рисунке 2.3.1.

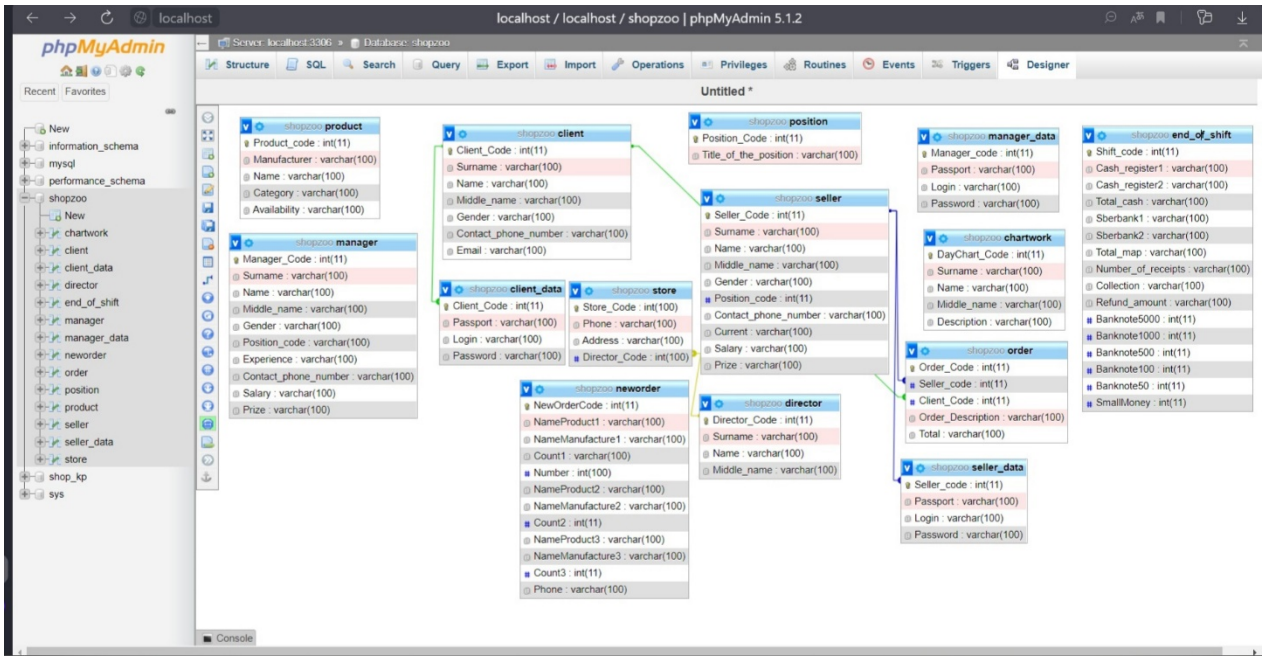

Рисунок 2.3.1 – База данных зоомагазина

Глава 3 Реализация информационной системы

В данной главе рассматривается все реализованные классы, описывается их функционал и из чего они состоят, подробно с кодом предоставляется описание функционала, а так же предоставлено краткое описание практическую значимость использования подключенных библиотек и пакетов.

3.1.Используемые библиотеки и пространства имен

Для того, чтобы достичь поставленной цели, необходимо реализовать работающее приложение с помощью Windows Forms. Навыки разработки Windows Forms приложений развиваются непосредственно в данный момент благодаря дисциплине «Конструирование программного обеспечения», однако дополнительно приходилось изучать некоторые пространства имен, которые будут описаны в данной главе.

3.1.1. System.Windows.Forms, System. Drawing, System. Collections.Generic

Пространство имен System.Windows.Forms содержит классы для создания приложений Windows, которые позволяют наиболее эффективно использовать расширенные возможности пользовательского интерфейса, доступные в операционной системе Microsoft Windows .

В данной курсовой работе используются разнообразные элементы управления, такие как PictureBox, Button, Label, GroupBox, dataGridView и TextBox, и события класса Control, в частности Click, MouseMove, Load, TextChanged, KeyPress, Validated.

Элементы управления PictureBox используются для отображения графических файлов для реализации каких-либо действий с помощью событий Click. Последнее так же актуально и для Button.

Элементы управления Label применяются для отображения какой-либо информации: например, размещены сверху TextBox, чтобы показывать какую информацию необходимо ввести.

GroupBox используется для того, чтобы поместить туда TextBox и Label, относящиеся к одной таблице.

Событие Click реагирует на клик пользователем по какому-либо элементу управления (в основном, это PictureBox и Button).

Событие MouseMove проверяет, навел ли пользователь указатель мыши на элемент управления (PictureBox или Button). Cursor элемента управления становится равным Cursors.Hand. Это дает понять пользователю, что элемент можно кликнуть.

Событие Load определяет, что будет выполняться при каждой загрузке формы.

Событие TextChanged срабатывает при изменении значения свойства Text, например, с помощью его я определяю как должна вести себя программа, когда вводятся разрешенные или запрещенные значения.

Событие KeyPress происходит, когда пользователь нажимает клавишу, которая вводит печатаемый знак, в данном событии так же можно прописать ограничения на ввод.

Событие Validated возникает при проверке действительности элемента управления., чаще всего я использую его в форме «Закрытие смены», где нужно проверять точно ли инкассация, введенная продавцам сходится со значениями итогов и возврата.

Помимо событий, так же активно используется создание и изменение элементов управления в программном коде с помощью свойств Controls, таких как Control.Size (задание размера), Control.BackColor (настройка фонового цвета), Control.Visible (виден ли объект на форме) и другие.

Пространство имен System.Drawing обеспечивает доступ к функциональным возможностям графического интерфейса. Классы данного пространства способны работать с графическими изображениями и цветами.

С помощью структуры Color, я работала с выделением необходимых для меня Buttom и пользовательских элементов для реализации календаря.

Последнее пространство имен содержит интерфейсы и классы, определяющие универсальные коллекции, которые позволяют пользователям создавать строго типизированные коллекции, обеспечивающие повышенную производительность и безопасность типов по сравнению с не универсальными строго типизированными коллекциями. Представляет строго типизированный список объектов, доступных по

индексу. Поддерживает методы для поиска по списку, выполнения сортировки и других операций со списками.

3.1.2. System. Text.RegularExpressions

Предоставляет функциональные возможности регулярных выражений, которые могут быть использованы из любой платформы или языка, работающих в рамках платформы .NET. В дополнение к типам, содержащимся в данном пространстве имен, класс RegexStringValidator позволяет определить, совпадает ли определенная строка с шаблоном регулярного выражения. Представляет постоянное регулярное выражение.

3.1.3. Пакет MySql.Data.MySqlClient

Пакет необходим для операции с базой данных MySQL на C #. Нам нужно импортировать пакет MySql.Data.MySqlClient для подключения к базе данных MySQL в C #. Пакет содержит в себе некоторые фунции и классы, например, MySqlConnection- это класс, который представляет открытое соединение с базой данных MySQL, MySqlConnection- класса для инициализации нового экземпляра, MySqlConnection.Open()- функция открывает соединение для выполнения любой операции, MySqlConnection.Close()- функция закрывает ранее открытое соединение, MySqlCommand нужен для проведения запросов к базе MySQL, MySqlDataAdapter содержит ряд команд данных и соединения с базой данных, нужен для обновления.

3.1.4. Пакет Microsoft.Office.Interop.Excel и FontAwesome.Sharp

Используется для выгрузки данных в Excel и подключается через ссылки к WinForms.

Библиотека для встраивания значков Font Awesome в приложения WPF и Windows Forms через NuGet. Использовалось для оформления меню.

3.2.Разработанные классы

Для выполнения всех требований были реализованы следующие классы и пользовательские элементы управления. Диаграмма классов представлена на рисунке 3.2.

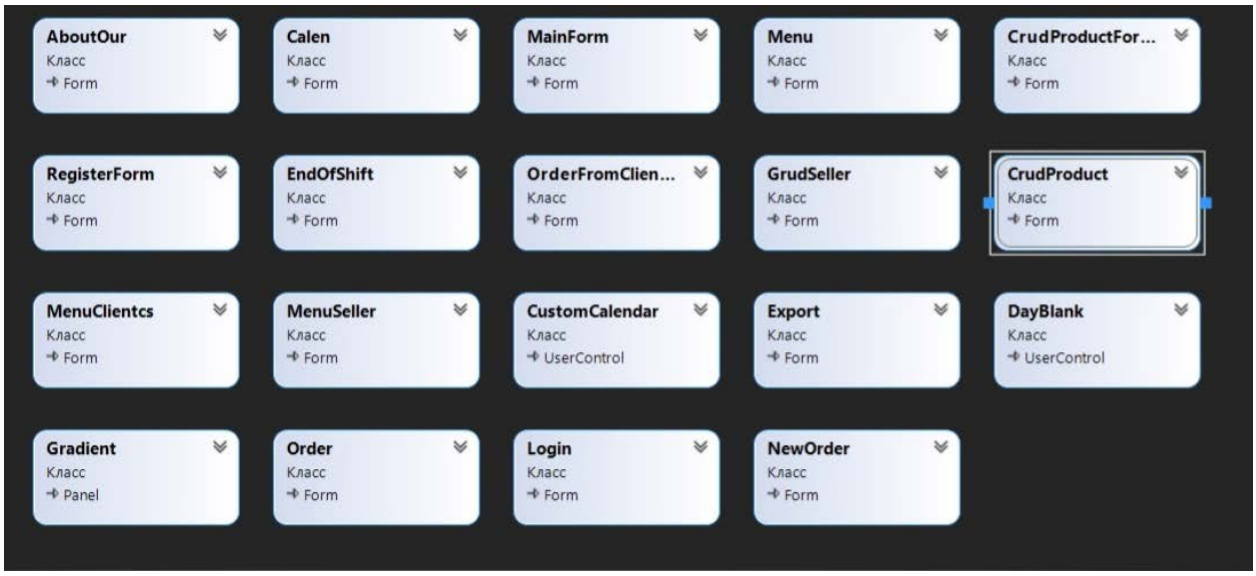

Рисунок 3.2 – Диаграмма разработанных классов

3.2.1. Класс AboutOur

Форма, созданная для размещения на ней основной информации об магазине, вызывается как дочернее окно к классу MenuClients. На форме размещен адрес и контактный номер, так же присутствует ссылка на приложение 2гис, чтобы пользователь мог понять, как ему добраться. (рис 3.2.1)

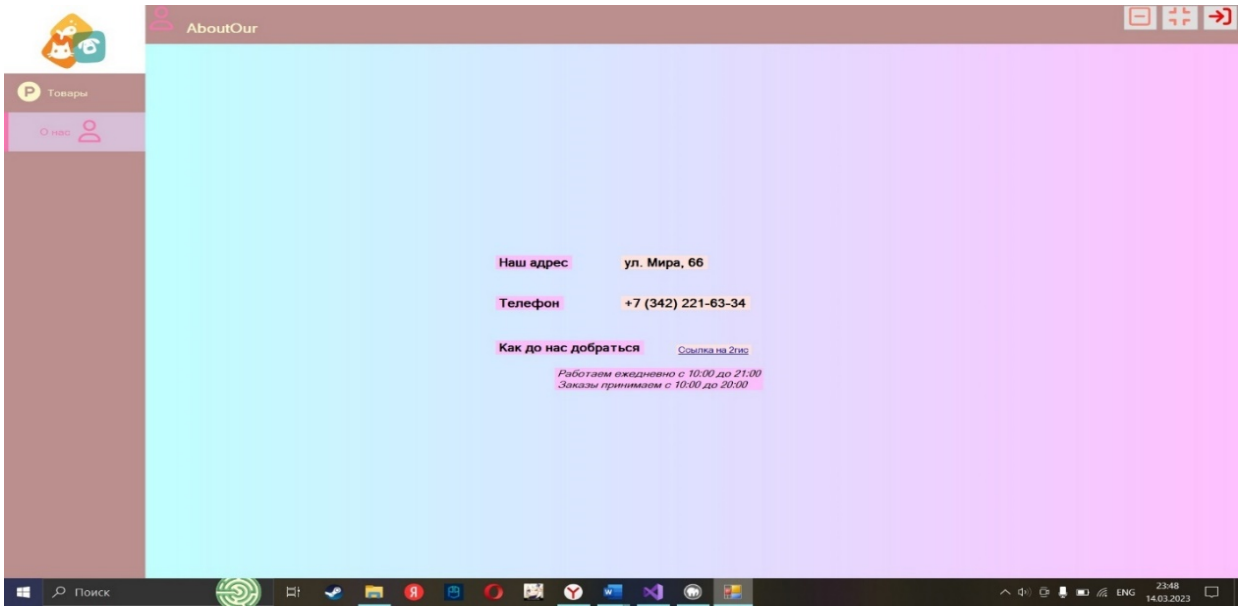

Рисунок 3.2.1 – Дочерняя форма «О нас»

3.2.2. Класс RegisterForm и Login

В данном классе реализуется форма для самостоятельной регистрации пользователя в приложении, перейти в нее можно через форму Login, которая запускается в начале работы приложения. Клиенту необходимо ввести свои личные данные, которые далее записываются в 2 таблицу БД. RegisterForm содержит в себе Label и TextBox. Для всех TextBox сделаны соответствующие проверки, например, в строке с фамилией нельзя вводить цифры, кроме того, после нажатия кнопки «Зарегистрироваться» срабатывает проверка на пустоту всех TextBox, если найдется пустой, то данные не будут добавлены и клиенту будет выведен MessegBox о том, что необходимо заполнить все имеющиеся строки. (рис 3.2.2.1)

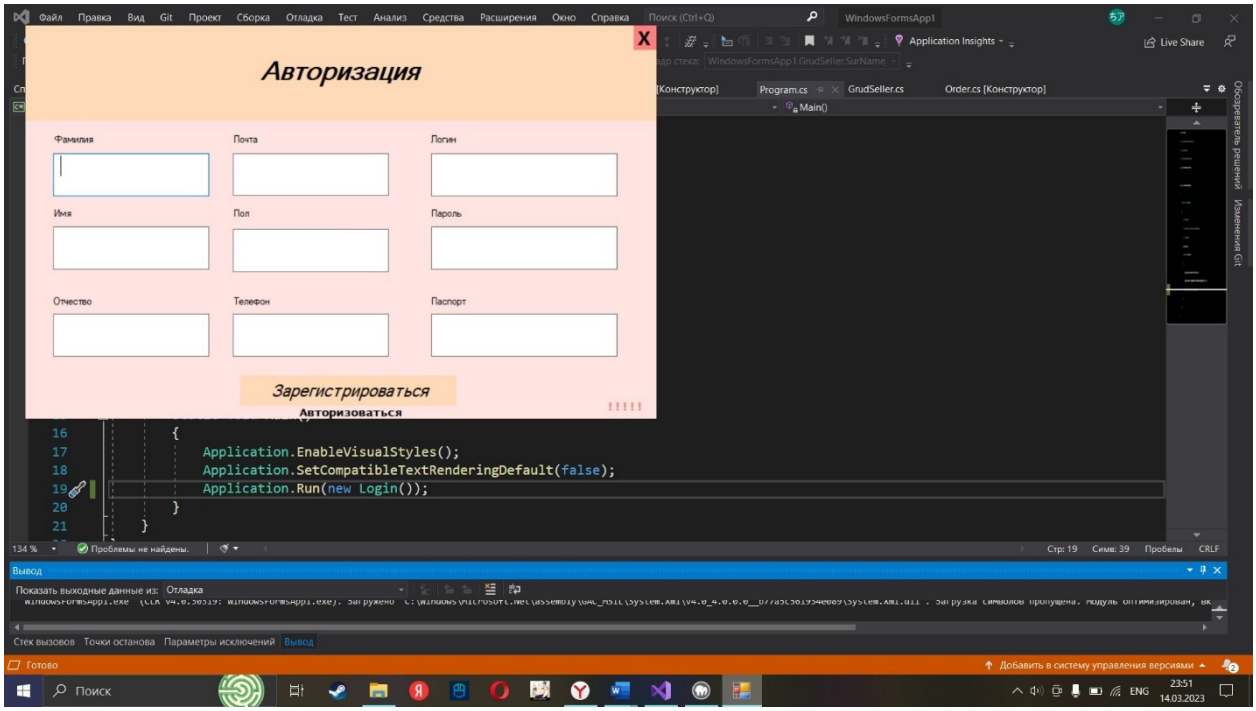

Рисунок 3.2.2.1 – Форма регистрации

Login-это класс содержащий весь функционал для формы, которая появляется при запуске приложения. При нажатии кнопки «Войти» происходит проверка, если пользователь будет найдет в одной из таблиц (управляющий, продавец или клиент), то автоматически произойдет перед к соответствующему ему меню, иначе пользователя уведомят, что авторизироваться не удалось. На этой форме так же присутствуют проверки ввода. (рис 3.2.2.2)

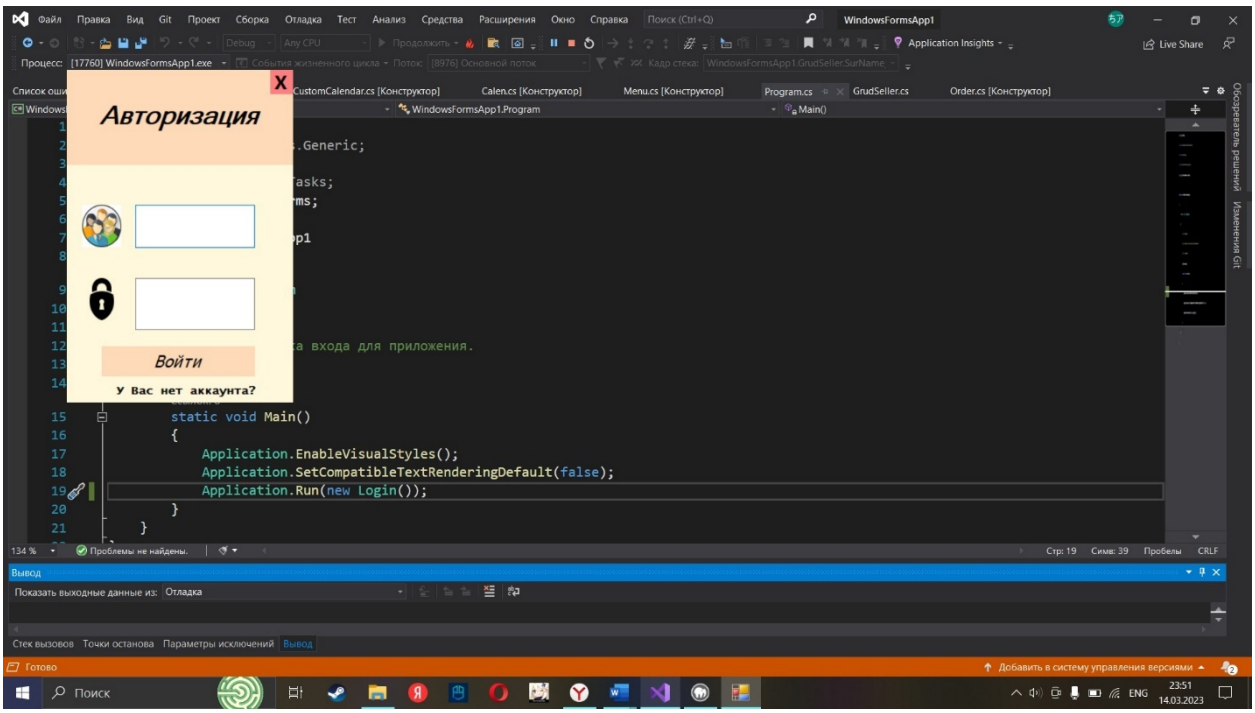

Рисунок 3.2.2.2 – Форма для входа

3.2.3. Класс MenuClients, MenuSeller, Menu

С помощью этого класса реализована форма для клиента, она представляет из себя родительскую форму, поскольку при нажатии интересующей кнопки, например, «О нас» часть изначальной формы замениться на дочернюю форму AboutOur. Пользователь может просмотреть 2 дочерних формы и через одну из них вызвать всплывающую форму. Для данной формы нет необходимости делать проверки на ввод, существует лишь функционал закрытия приложения, свертывания и развертывания окна. (рис 3.2.3.1)

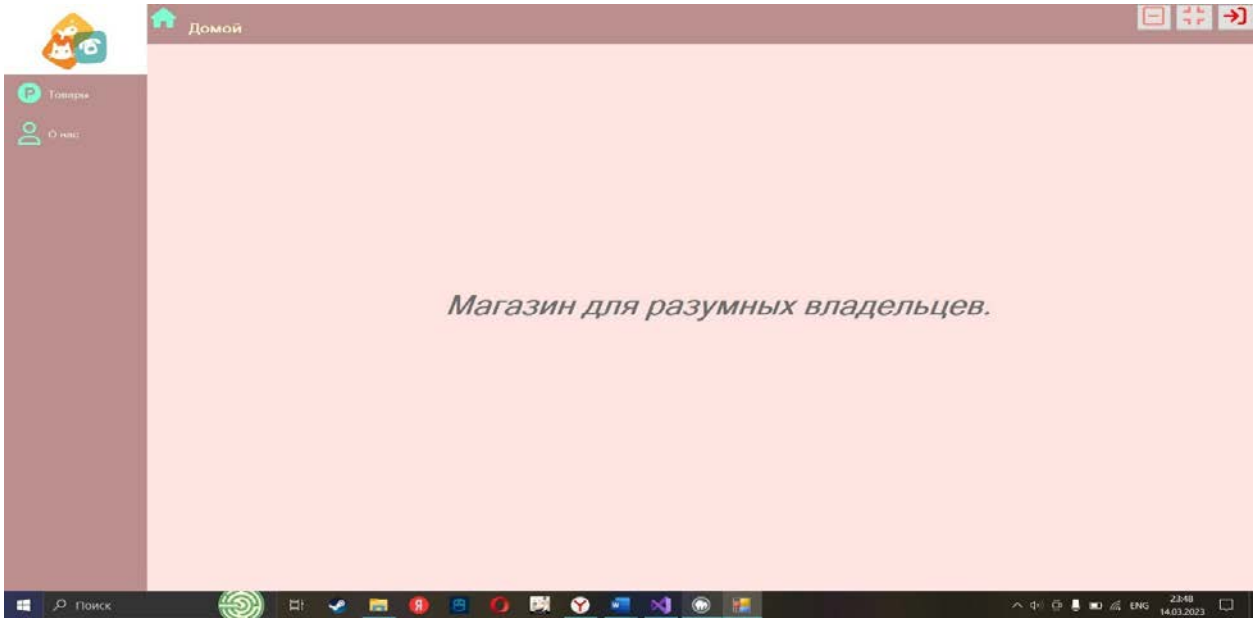

Рисунок 3.2.3.1 – Меню для клиента

Класс аналогичный MenuClients, нужен для отображения формы, отвечающей за меню для продавца. Дизайн был сделан точно таким же как и для вышеупомянутой формы, отличается лишь функционалом, а именно количеством дочерних форм. В данном случае реализовано 5 дочерних форм, которые отвечают за товары, клиентов, закрытие смены, новые и завершенные заказы. (рис 3.2.3.2)

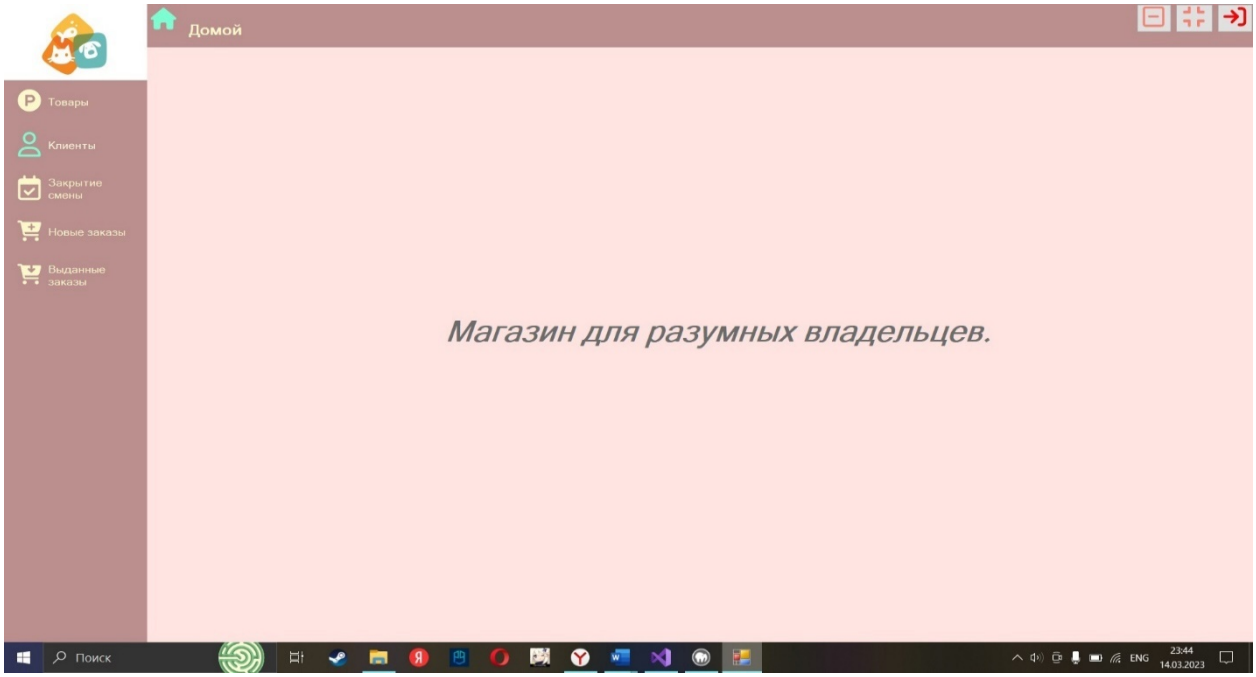

Рисунок 3.2.3.2 – Меню продавца

Класс необходимый для реализации меню для управляющего, сделан в той же стилистике, что и остальные, количество дочерних форм 6, а именно, товары, клиент, продавцы, календарь, экспорт и итоги. (рис 3.2.3.3)

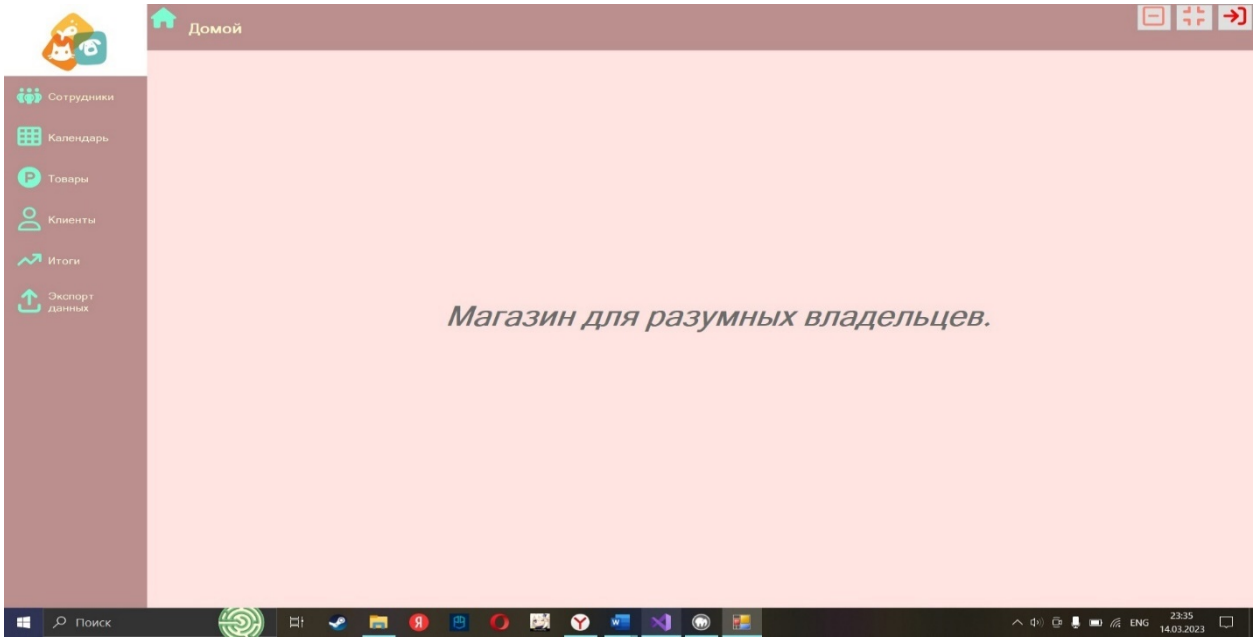

Рисунок 3.2.3.3 – Меню управляющего

3.2.4. Класс Gradient, DayBlank, CustomCalendar. Calen

Gradient- пользовательский интерфейс управления, созданный для того, чтобы делать фон для форм в виде градиента. Чтобы получить желаемый результат необходимо задать значения для BackColor и TopColor.

DayBlank- пользовательский элемент управления, созданный для формирования календаря. Представляет из себя TextBox с цифрами 01, в дальнейшем этот элемент вставляется в ячейки календаря и для каждой из них просчитывается свое значение.

CustomCalendar- пользовательский элемент управления, состоящий из 42х DayBlank, нужен для дальнейшего использования в классе Calen.

Форма, в которую добавлен CustomCalendar вместе с несколькими дополнительными Panel, благодаря которым можно перемещаться по месяцам и годам. Не требует никаких проверок. (рис 3.2.4.1)

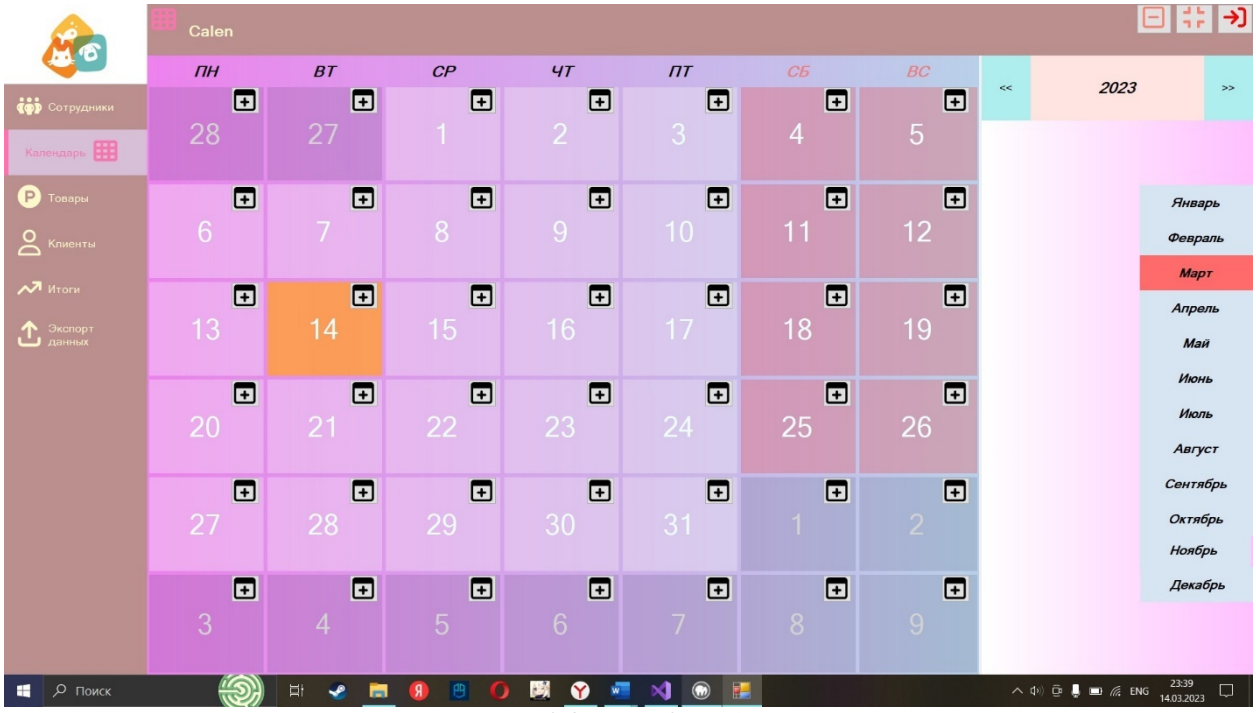

Рисунок 3.2.4.1 – Календарь

3.2.5. Класс Order, OrderFromClient, NewOrder

Описывает функционал для формы с CRUD операциями для продавца, связанные с завершенными заказами. Продавец может добавить, удалить, отредактировать, обновить заказ или же найти необходимый, тогда он отобразиться в нижнем окне dataGridView. (рис 3.2.5.1)

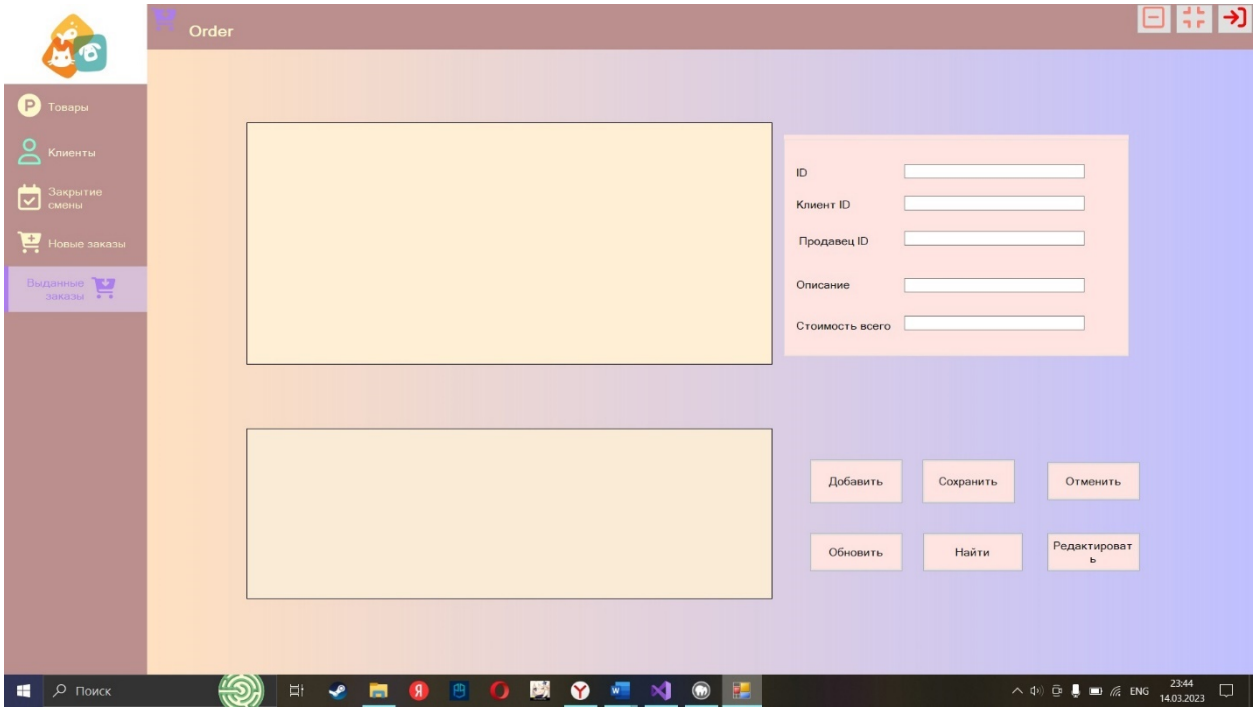

Рисунок 3.2.5.1 – Форма заказа

Данный класс предназначен для клиента, что следует из названия, здесь содержится функционал для формирования заказа от пользователя, для продавца это же новый заказ. В данной форме пользователю необходимо ввести всю информацию о своем заказе в нужные TextBox, а также выбрать количество различных товаров, которые он приобретает это делается через ComboBox. В классе реализованы проверки на ввод, а также проверки на соответствие выбранного количества в ComboBox и заполненных TextBox. Когда клиент выбирает кнопку «Оставить запрос», то он автоматически записывается в БД, если выполнены все проверки. (рис 3.2.5.2)

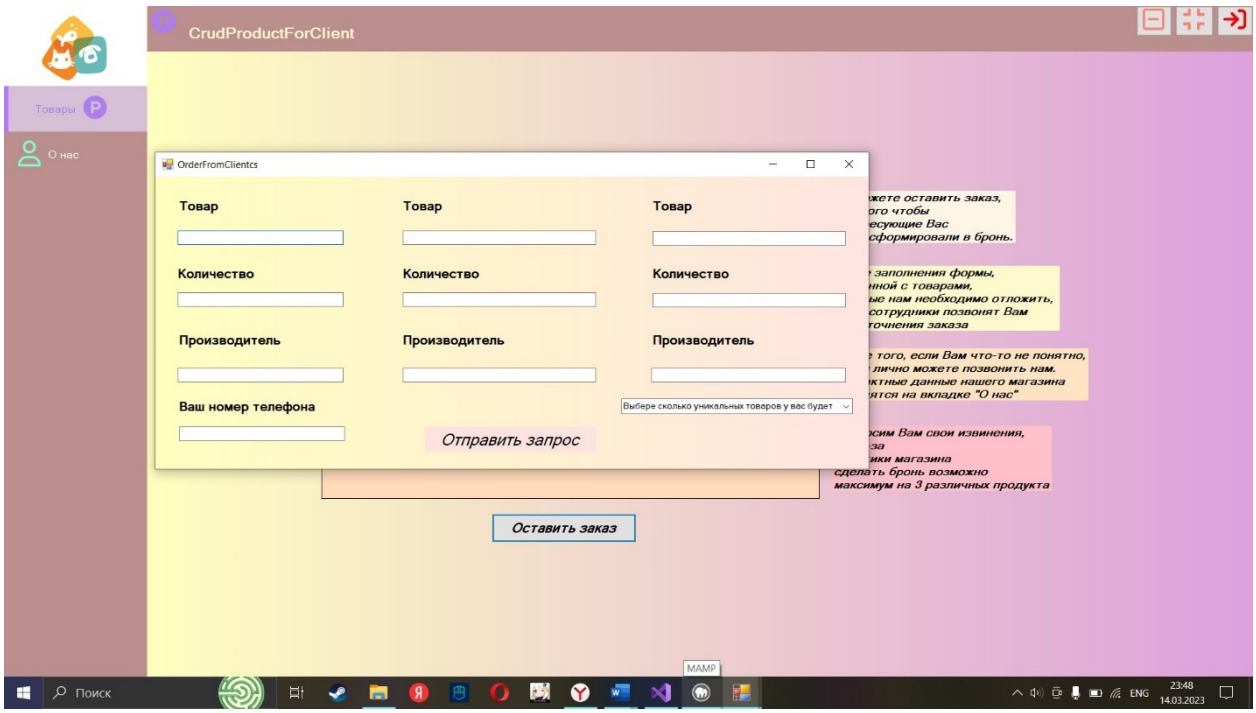

Рисунок 3.2.5.2 – Форма для отправления заказа

Форма, которая отображается в качестве дочерней для продавца, в ней можно увидеть новые заказы, когда заказ приобретает статус «Завершенный», то продавец должен его удалить из текущей таблицы и вписать в новую. Присутствуют проверки на ввод. (рис 3.2.5.3)

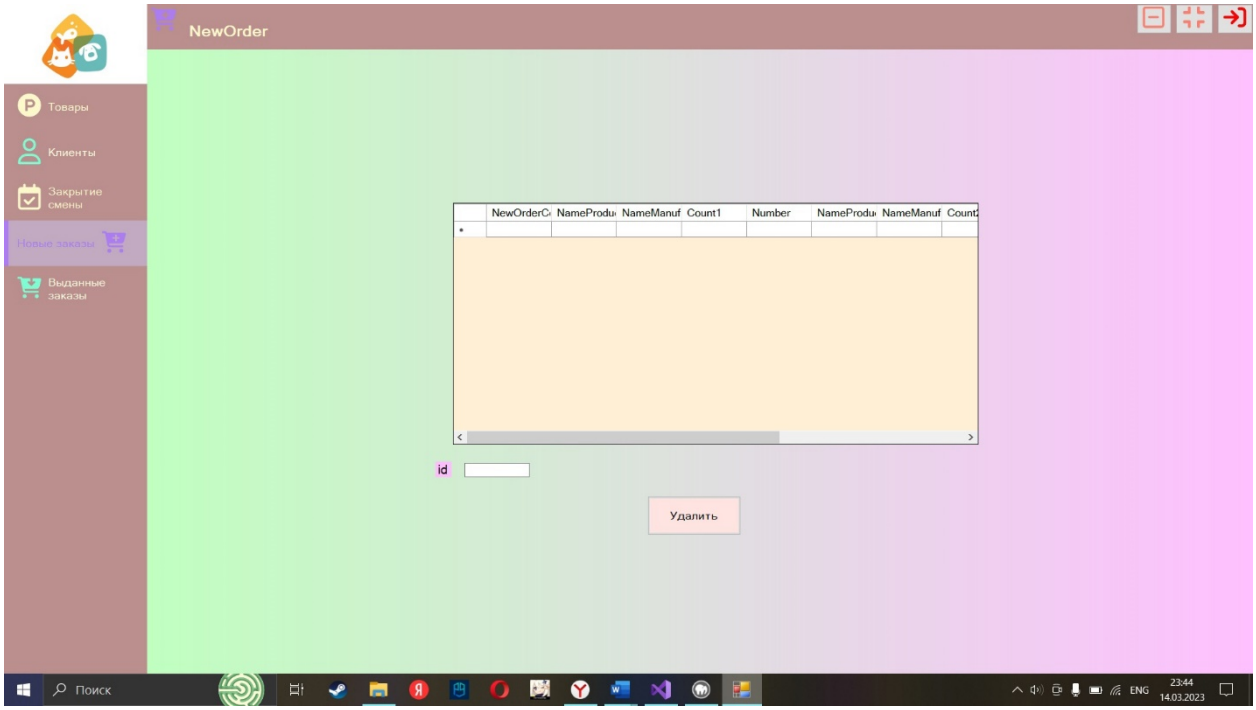

Рисунок 3.2.5.3 – Новые заказы от клиентов

3.2.6. Класс EndOfShift

Класс нужный для реализации функционала для закрытия смены. В форме соответствующей этому классу продавец вводит все данные по наличному и безналичному расчету на обоих кассах, инкассацию и возврат, а также покупюрник. Здесь прописаны проверки чуть сложнее, чем у других форм, поскольку нужно не только проверять точно ли в TextBox записаны числа, нужно сравнивать значения инкассации с данными, которые введены в покупюрнике и в итогах по наличному, безналичному расчёту и возвратах, если продавец допустил ошибку, то система уведомит его об этом. Если же все данные были верными, то все запишется в базу. (рис 3.2.6.1)

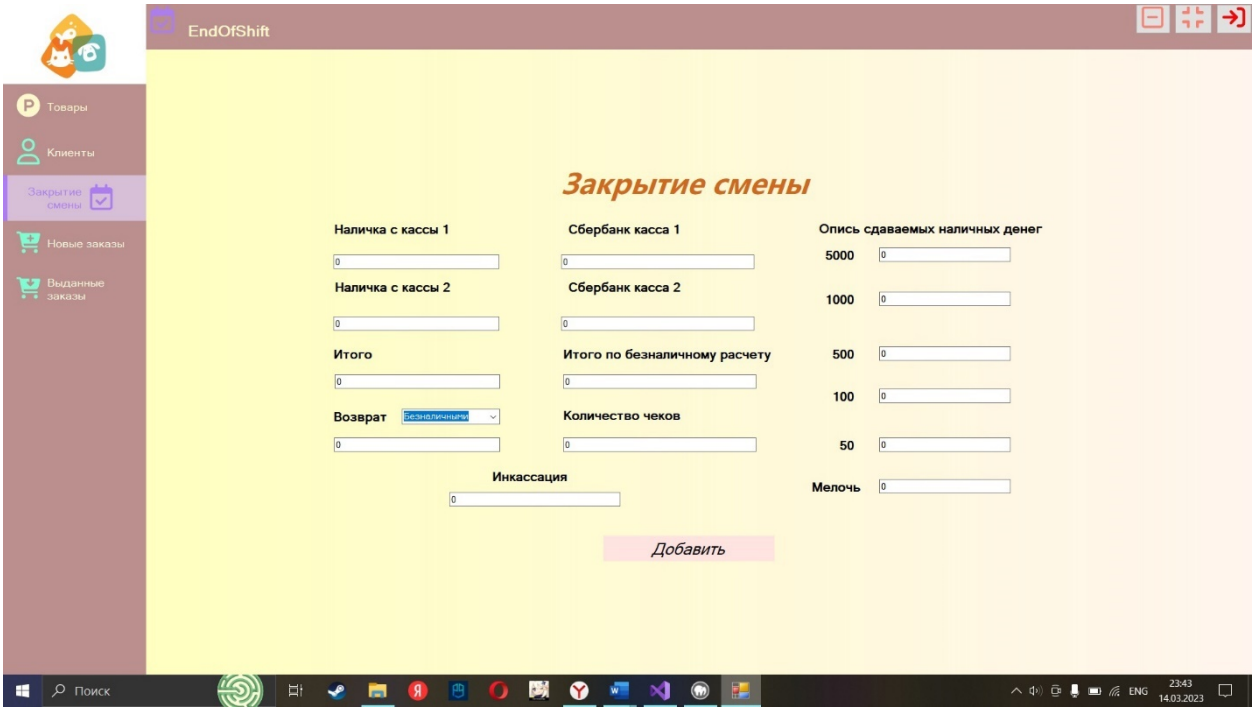

Рисунок 3.2.6.1 – Форма закрытия смены

3.2.7. Класс CrudSeller,MainForm, CrudProductForClient, CrudProduct

CRUD операции по продавцам необходимы для управляющих, для того чтобы удалить или добавить нового сотрудника, также бывают ситуации когда нужно найти сотрудника и изменить некоторые его данные. У всех TextBox прописаны проверки на ввод. (рис 3.2.7.1)

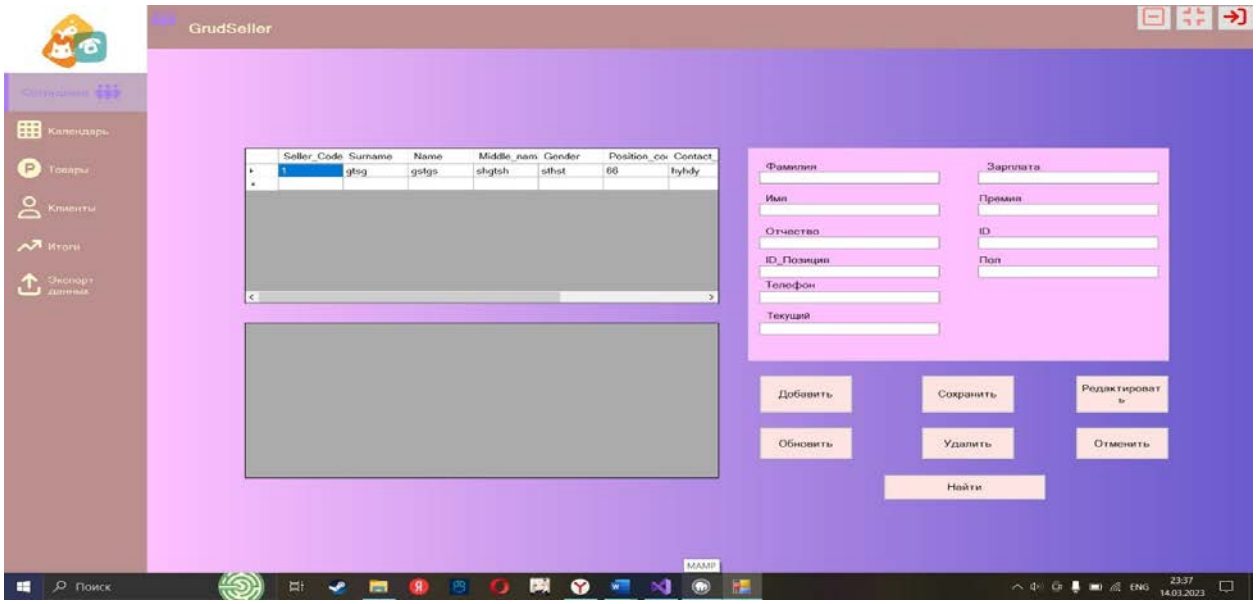

Рисунок 3.2.7.1 – Операции Crud по продавцам

Описывает функционал для формы с CRUD операциями по клиентам для продавца и управляющего. Здесь можно просмотреть таблицу из базы данных в dataGridView, добавить, удалить, изменить данные или же найти необходимый товар, он будет отображен во второй dataGridView. (рис 3.2.7.2)

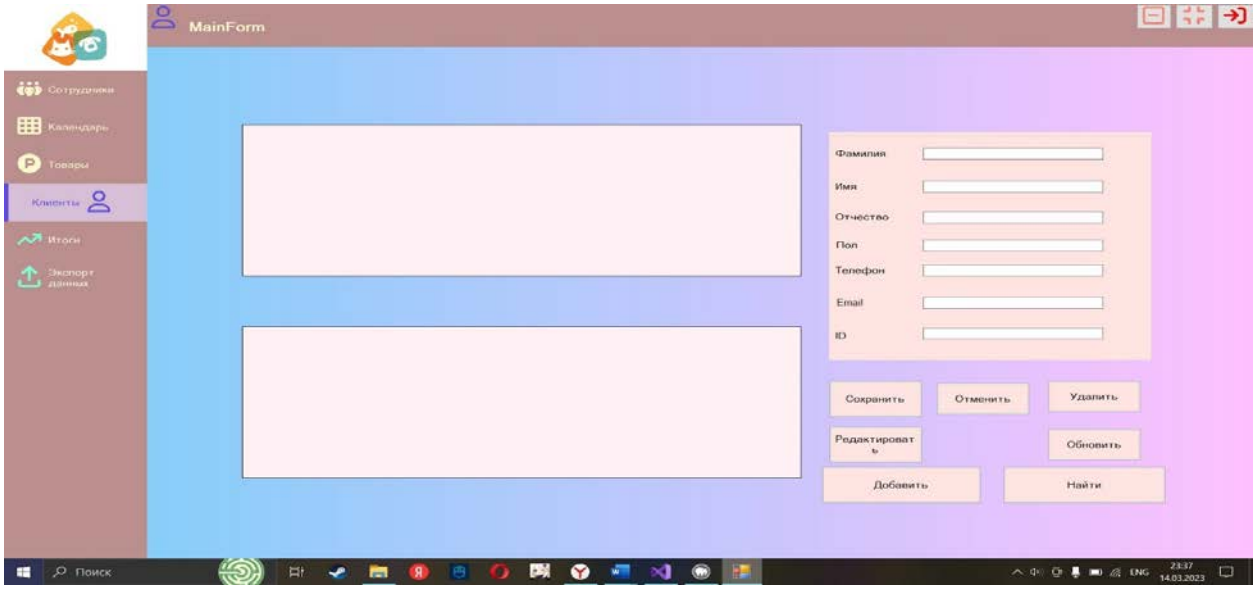

Рисунок 3.2.7.2 – Операции Crud по сотрудникам

CRUD операции по товарам для клиента отличаются от тех, что представлены для продавца или управляющего. Клиент не может ничего изменять или редактировать, в его функционале есть лишь просмотр и поиск товаров, также учтены все проверки. (рис 3.2.7.3)

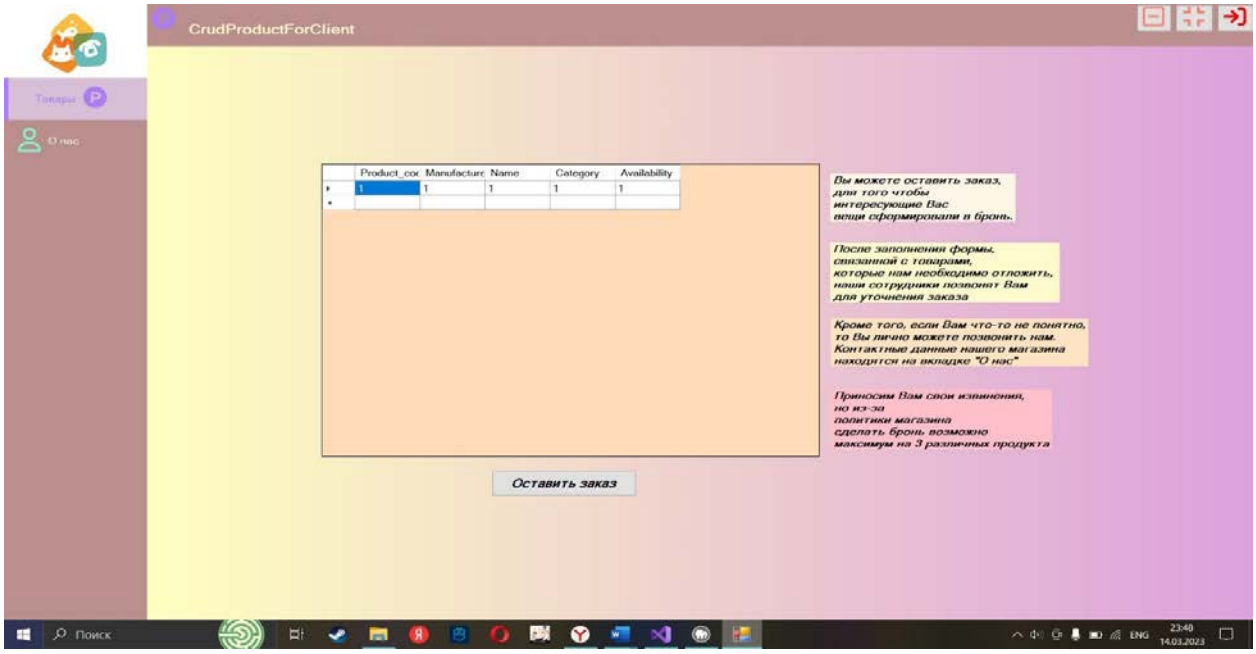

Рисунок 3.2.7.3 – Товары для клиента

Реализация CRUD операция для продавца и управляющего, где у обоих есть права на удаление, редактирование, добавление и изменение товаров, также организован поиск товара. Присутствуют проверки на ввод. (рис 3.2.7.4)

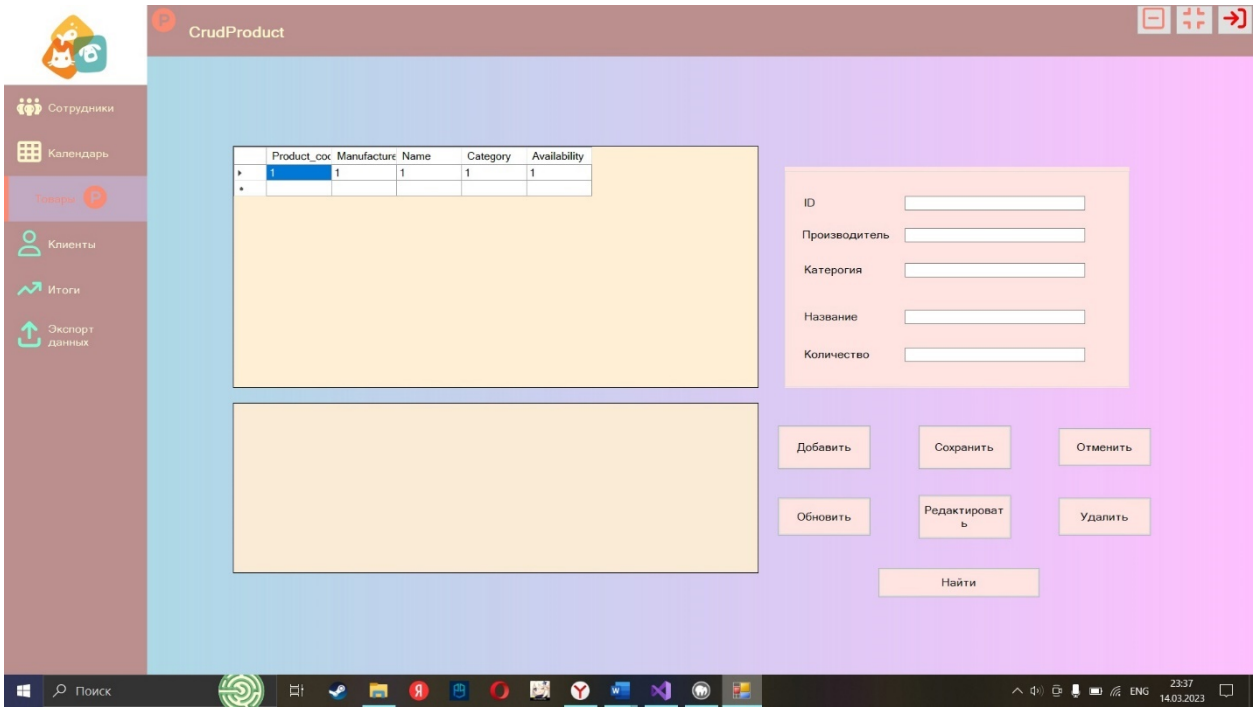

Рисунок 3.2.7.4 – Продукты для продавца и управляющего

3.2.8. Класс Export

Нужен для формирования отчетов в файл формата Excel. Никакие проверки для данной формы не нужны. (рис 3.2.8.1)

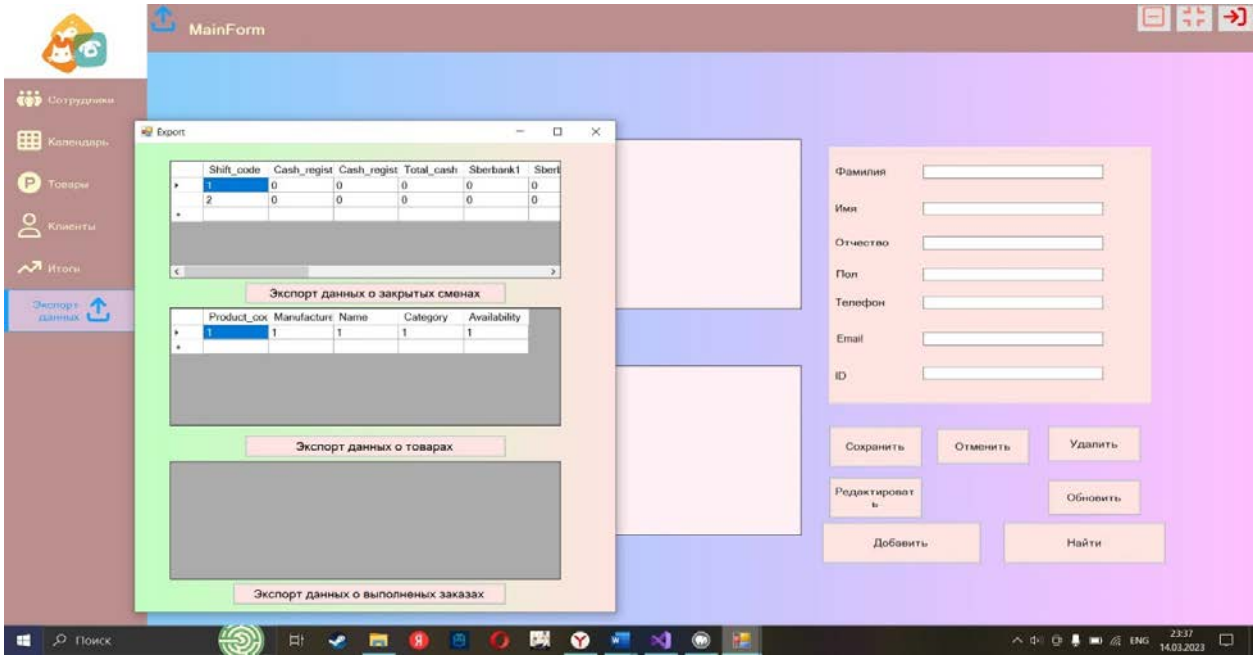

Рисунок 3.2.8.1 – Экспорт

3.2.9. Класс Chart

Класс, в котором создается форма для изображения графика, это необходимо для того, чтобы управляющий мог смотреть итоги по закрытым сменам, по оси X рассматривается код смены, а по Y инкассация. Проверки для данной формы не нужны. (рис 3.2.9.1)

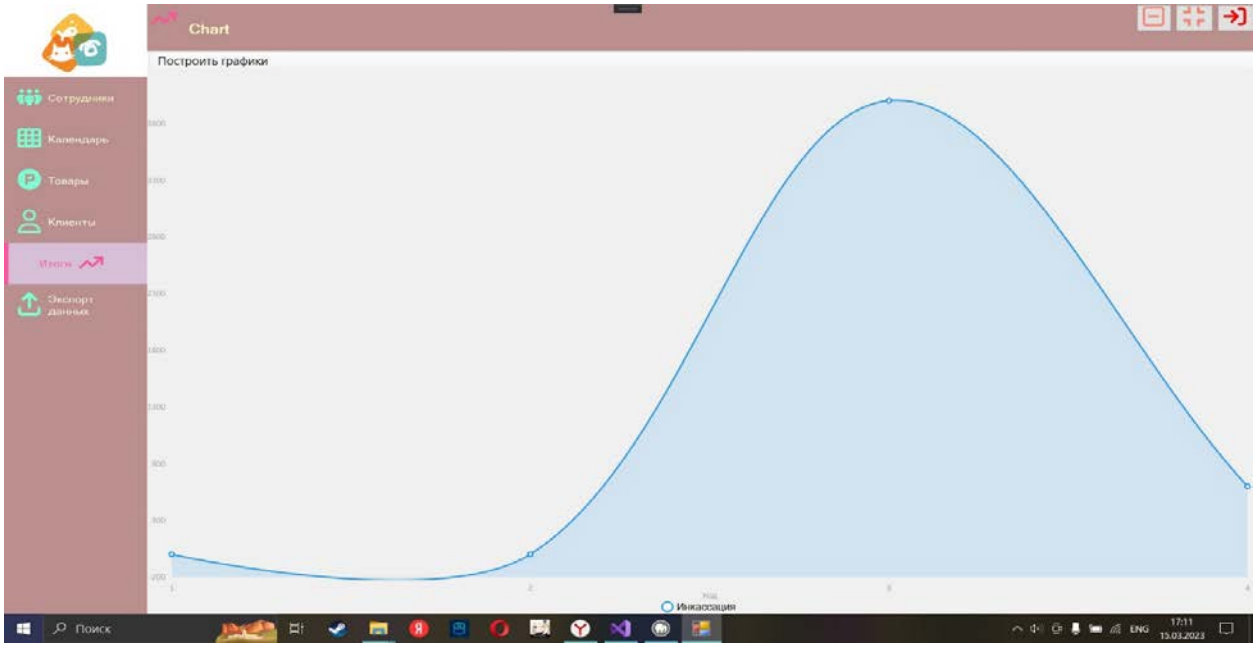

Рисунок 3.2.9.1 – Итоги

3.3.Реализация основных функций

В данном подпункте описываются основной функционал, который был разработан для информационной системы.

3.3.1. Переход между формами и месяцами/годами в календаре

В разработанной информационной системе существует два вида перехода между формами. Первый это переход между родительской и дочерней формой. Для реализации такого перехода необходимо было объявить приватную форму (рис. 3.3.1.1)

```
Ссылок: 4
public partial class Menu : Form
{

    private Panel leftBorderBtn;
    private IconButton currentBtn;
    private Form currentChildForm;
    ссылка: 1
    public Menu()
    {
        InitializeComponent();
        leftBorderBtn = new Panel();
        leftBorderBtn.Size = new Size(7, 70);
        panelMenu.Controls.Add(leftBorderBtn);
        WindowState = FormWindowState.Maximized;
        this.Text = string.Empty;
        this.ControlBox = false;
        this.DoubleBuffered = true;
        this.MaximizedBounds = Screen.FromHandle(this.Handle).WorkingArea;
    }
```

Рисунок 3.3.1.1 – Объявление необходимых переменных

Далее была создан метод для открытия дочерней формы. Изначально необходимо проверить не является ли текущая форма нулевой, если равна, то мы текущей родительской форме присваиваем значение переданной формы, настраиваем некоторые параметры и вызываем метод Show() для ее отображения. Далее этот метод вызывается в событии Click(рис.3.3.1.2)

```
Ссылок: 4
private void OpenChildForm(Form childForm)
{
    if (currentChildForm!=null)
    {
        currentChildForm.Close();
    }
    currentChildForm = childForm;
    childForm.TopLevel = false;
    childForm.FormBorderStyle = FormBorderStyle.None;
    childForm.Dock = DockStyle.Fill;
    panelDesktop.Controls.Add(childForm);
    panelDesktop.Tag = childForm;
    childForm.BringToFront();
    childForm.Show();
    IbITitle.Text = childForm.Text;
}
ссылка: 1
private void iconButton1_Click(object sender, EventArgs e)
{
    ActivateButton(sender, RGBColors.color1);
    OpenChildForm(new GrudSeller());
}
```

Рисунок 3.3.1.2 – Метод для вызова формы и событие

Также в событии Click вызывается метод для определения активной кнопки, он нужен для отображения определенного значка сверху дочерней формы, а так же для отображения определенного цвета, который показывает, на какой вкладке вы находитесь(рис.3.3.1.3)

```
Ссылок: 6
private void ActivateButton(object senderBtn, Color color)
{
    if (senderBtn!=null)
    {
        DisableButton();
        currentBtn = (IconButton)senderBtn;
        currentBtn.BackColor = Color.Thistle;
        currentBtn.ForeColor = color;
        currentBtn.TextAlign = ContentAlignment.MiddleCenter;
        currentBtn.IconColor = color;
        currentBtn.TextImageRelation = TextImageRelation.TextBeforeImage;
        currentBtn.TextAlign = ContentAlignment.MiddleRight;
        leftBorderBtn.BackColor = color;
        leftBorderBtn.Location = new Point(0, currentBtn.Location.Y);
        leftBorderBtn.Visible = true;
        leftBorderBtn.BringToFront();
        iconCurrent.IconChar = currentBtn.IconChar;
        iconCurrent.IconColor = color;
    }
}
```

Рисунок 3.3.1.3 – Метод для отображения цвета и значка

Но также существует и отображение форм как плавающих окон, для этого так же используется событие Click, но записано оно чуть иначе. Вместо того, чтобы вызывать метод OpenChildForm, мы вызываем саму форму, присваиваем ей переменную и уже эту переменную транслируем через Show (). (рис.3.3.1.4)

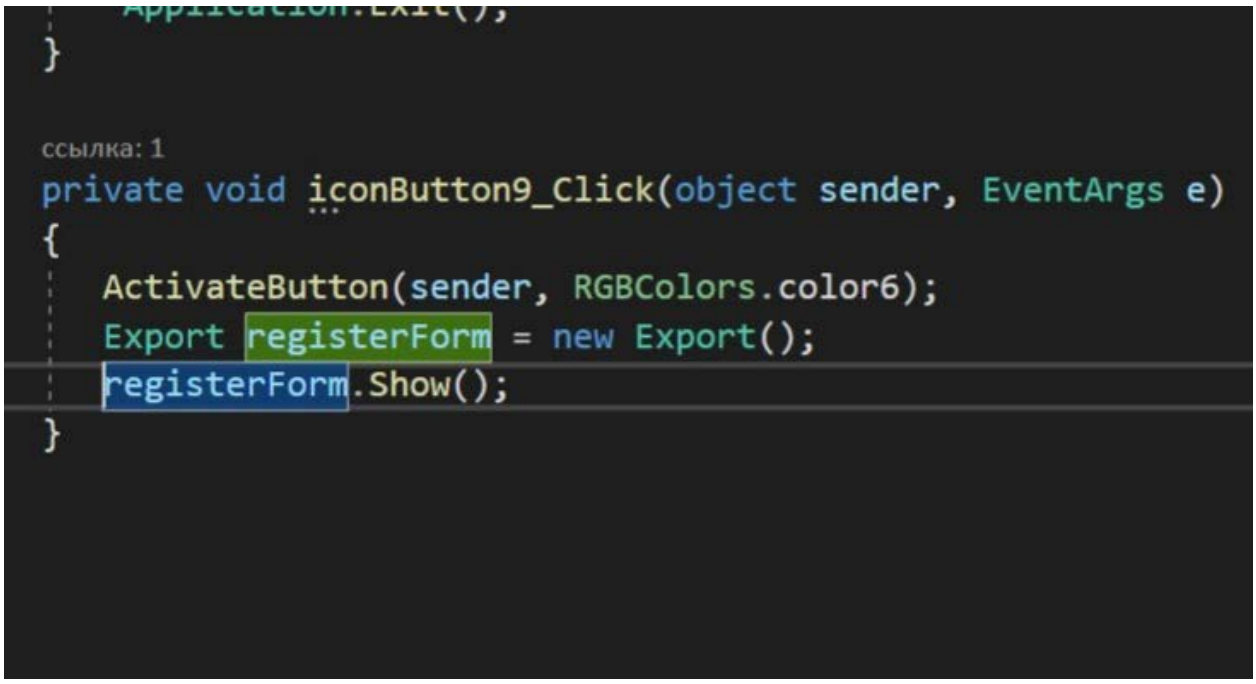

Рисунок 3.3.1.4 – Вызов плавающих окон

Для того, чтобы оптимально написать реализацию переходов я ввела переменную типа целочисленное значение с именем выбранный месяц, как можно заметить у меня есть событие загрузки календаря, в нем я создаю новую форму CustomCalendar, заполняю ее днями, а также выделяю соответствующий месяц, все это происходит также с помощью DateTime.Now.Year и DateTime.Now.Month , который синхронизируется с компьютером и определяет какой сегодня год и месяц. . (рис 3.3.1.5).

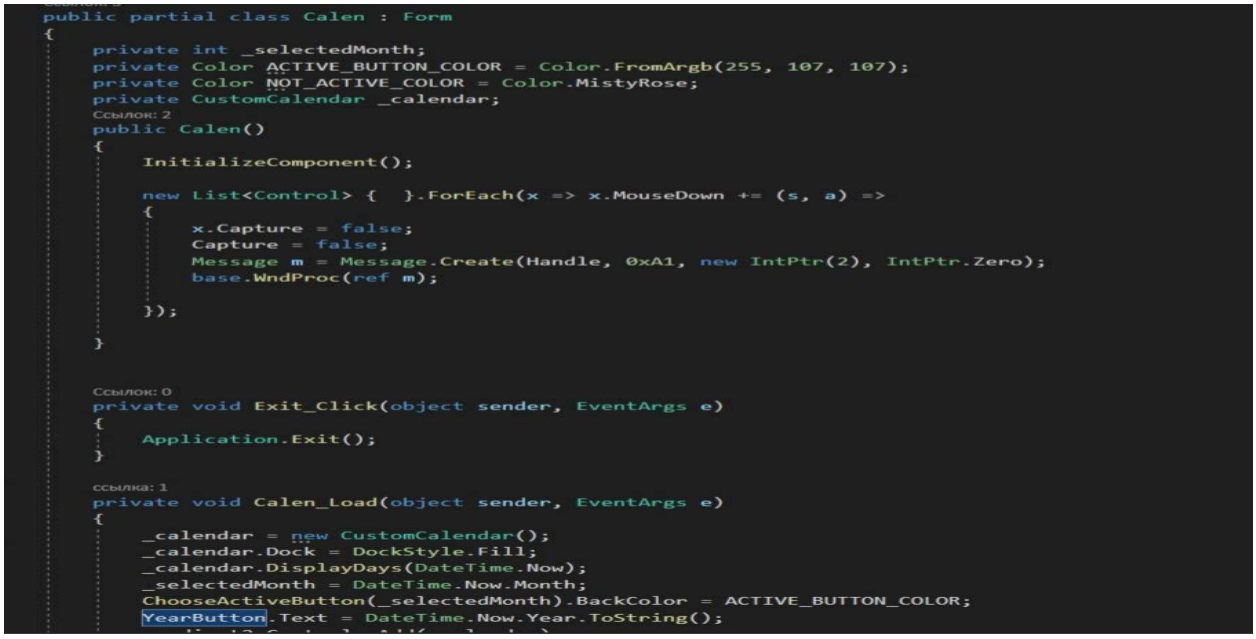

Рисунок 3.3.1.5 – Объявление переменный и цвета для активных и неактивных кнопок

Также мы создаем лист со всеми месяцами и добавляем тупа событие Click. В данном событии определяется выбранный меняц и срабатывает метод для обновления вида календаря, в котором мы подгоняем значения дней под месяц и год. (рис 3.3.1.6)

```
ссылка: 1
private void Calen_Load(object sender, EventArgs e)
{
    _calendar = new CustomCalendar();
    _calendar.Dock = DockStyle.Fill;
    _calendar.DisplayDays(DateTime.Now);
    _selectedMonth = DateTime.Now.Month;
    ChooseActiveButton(_selectedMonth).BackColor = ACTIVE_BUTTON_COLOR;
    YearButton.Text = DateTime.Now.Year.ToString();
    gradient2.Controls.Add(_calendar);
    new List<Control> { JanuaryButton, FebruaryButton, MarchButton, AprilButton, MayButton, JuneButton, JulyButton, AugustButton,
        SeptemberButton, OctoberButton, NovemberButton,DecemberButton }.ForEach((Action<Control>)(x =>
        {
            x.Click += MonthButtonClick;
        }));
}

ссылка: 1
private void MonthButtonClick(object sender, EventArgs e)
{
    _selectedMonth = (sender as Button).TabIndex;
    RefreshCalendar(int.Parse(YearButton.Text), _selectedMonth);
}
Ссылок: 3
private void RefreshCalendar(int year, int month)
{
    var date = new DateTime(year, month, 1);
    _calendar.DisplayDays(date);
    SetMonthButtonActive(ChooseActiveButton(month));
}
```

Рисунок 3.3.1.6 – Событие Click и метод для обновления

Также пользователь может выбирать месяц, для этого сделан метод выбора активной кнопки. Для каждой кнопки задан index от 1 до 12. В методе обновления мы используем функции для инициализации выбранного месяца, для которого в CustomCalendar просчитается значение дней (рис 3.3.1.7)

```
Ссылок: 2
private Button ChooseActiveButton(int index)
{
    switch (index)
    {
        case 1:
            return JanuaryButton;
        case 2:
            return FebruaryButton;
        case 3:
            return MarchButton;
        case 4:
            return AprilButton;
        case 5:
            return MayButton;
        case 6:
            return JuneButton;
        case 7:
            return JulyButton;
        case 8:
            return AugustButton;
        case 9:
            return SeptemberButton;
        case 10:
            return OctoberButton;
        case 11:
            return NovemberButton;
        case 12:
            return DecemberButton;
    }
    return null;
}
```

Рисунок 3.3.1.7 – Событие Click и метод для обновления

Последние два метода нужны для определения следующего и предыдущего месяца. Определяется это достаточно легко, либо вычитаем, либо прибавляем единицу, а потом обновляем календарь. (рис 3.3.1.8)

```
    ссылка: 1
    private void SetMonthButtonActive(Button activeButton)
    {
        foreach (Control item in MonthNavigationPanel.Controls)
        {
            if (item.GetType() == typeof(Button))
                item.BackColor = NOT_ACTIVE_COLOR;
        }
        activeButton.BackColor = ACTIVE_BUTTON_COLOR;
    }

    ссылка: 1
    private void PreviosYearButton_Click(object sender, EventArgs e)
    {
        YearButton.Text = (int.Parse(YearButton.Text) - 1).ToString();
        RefreshCalendar(int.Parse(YearButton.Text), _selectedMonth);
    }

    ссылка: 1
    private void NextYearButton_Click(object sender, EventArgs e)
    {
        YearButton.Text = (int.Parse(YearButton.Text) + 1).ToString();
        RefreshCalendar(int.Parse(YearButton.Text), _selectedMonth);
    }
}
```

Рисунок 3.3.1.8 – Событие Click и метод для обновления

3.3.2. CRUD операции

Это самые базовые операции, которые можно делать с БД или же отдельными таблицами, в представленных подпунктах я рассмотрю операциями над одной из таблиц. Все CRUD операции осуществляются через запросы, для того чтобы, во-первых, была возможность писать запросы нужно ввести некоторые переменные и подключить библиотеки, о которых мы говорили ранее, а также прописать строку подключения MySqlConnection, во-вторых, нужно знать над какими данными производится запрос, для этого я использую SQL- запрос Select*from (название таблицы в БД) для отображения информации в dataGridView.(рис 3.3.2.1)

```
DB db = new DB();
MySqlConnection connection = new MySqlConnection("server=localhost;port=3306;username=root; password=root;database=shopzoo");
MySqlCommand command;
MySqlDataAdapter da;
MySqlDataReader reader;
DataTable dt;

Ссылок: 3
public MainForm()
{
    InitializeComponent();
    try
    {
        connection.Close();
        connection.Open();
        command = new MySqlCommand("Select * from client", connection);
        command.ExecuteNonQuery();
        da = new MySqlDataAdapter(command);
        dt = new DataTable();
        da.Fill(dt);
        dataGridView1.DataSource = dt.DefaultView;
        connection.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

Рисунок 3.3.2.1 – Подключение к СУБД и представление таблицы

3.3.2.1. Добавление новых записей

Для того, чтобы осуществить добавление новой записи нужно использовать запрос Insert into (название таблицы) Values, прежде чем запрос начнет обрабатываться нужно сделать блок для проверки на подключение или другие ошибки, в моем случае это try catch. Далее открывается соединение с базой данной, в запросе мы пишем переменные, которые отвечают за каждую ячейку в таблице, после этого добавляем в каждую из них информацию из соответствующего TextBox и закрываем подключение. (рис. 3.3.2.1.1)

```
ссылка: 1
private void btnAdd_Click(object sender, EventArgs e)
{

    groupBox1.Enabled = true;
    try
    {
    connection.Open();
    command = new MySqlCommand("Insert into client VALUES (NULL, @first_Name, @name, @patr,@gender,@phone,@email)", connection);
    command.Parameters.AddWithValue("@first_Name", SurName.Text);
    command.Parameters.AddWithValue("@name", textName.Text);
    command.Parameters.AddWithValue("@patr", textS.Text);
    command.Parameters.AddWithValue("@gender", textBoxGender.Text);
    command.Parameters.AddWithValue("@phone", textPhone.Text);
    command.Parameters.AddWithValue("@email", textEmail.Text);


    command.ExecuteNonQuery();
    connection.Close();
    MessageBox.Show("Ok");
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }

}
```

Рисунок 3.3.2.1.1– SQL-запрос на добавление

3.3.2.2. Поиск необходимой записи

Для поиска реализуется запрос вида Select*from (название таблицы) where. Также присутствует блок try catch, подключение и отключение от базы. В запросе мы пишем по какому коду необходимо найти клиента и какой TextBox соответствует этому коду, если запрос прошел успешно, то информация отображается в dataGridView, который находится ниже основного. (рис. 3.3.2.2.1)

```
ссылка: 1
private void search_Click(object sender, EventArgs e)
{
    groupBox1.Enabled = true;
    command = new MySqlCommand("Select * from Client where Client_Code=@code_Client", connection);
    command.Parameters.AddWithValue("@code_Client", textclientcode.Text);
    MySqlDataAdapter da = new MySqlDataAdapter(command);
    DataTable dt = new DataTable();
    da.Fill(dt);
    dataGridView2.DataSource = dt;

}
```

Рисунок 3.3.2.2.1 – SQL-запрос для поиска

3.3.2.3. Удаление записи

Чтобы удалить запись нам понадобиться запрос Delete, который пишется так: Delete from (название таблицы) where и пишется TextBox, который соответствует первичному ключу таблицы. Обязательно во всех запросах присутствует try catch, соединение и отключение с БД, так же я уведомляю продавца, если операция прошла успешно. (рис. 3.3.2.3.1)

```
ссылка: 1
private void btnDelete_Click_1(object sender, EventArgs e)
{
    try
    {
        connection.Close();
        connection.Open();
        command = new MySqlCommand("Delete from Client where Client_Code= '" + SurName.Text + "' ", connection);
        command.ExecuteNonQuery();
        connection.Close();
        MessageBox.Show("Data successfully deleted");
        cancel();
        LoadData();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

Рисунок 3.3.2.3.1 – SQL-запрос для поиска

3.3.2.4. Обновление таблицы

Для того, чтобы обновить таблицу после удаления или добавления новой записи, нужно написать запрос Update (название таблицы) и через set записать ко всем атрибутам таблицы соответствующие TextBox. (рис. 3.3.2.4.1)

```
ссылка: 1
private void btnUpdate_Click_1(object sender, EventArgs e)
{
    try
    {
        connection.Close();
        connection.Open();
        command = new MySqlCommand("Update Client set Surname ='" + SurName.Text +
        "',First_Name='" + textName.Text + "', Patronymic='" +
        textS.Text + "',Phone_Number='" + textPhone.Text + "',Gender='" +
        textBoxGender.Text+"',Email='" + textEmail.Text  + "' where Product_Code= '" + textclientcode.Text + "' ", connection);
        command.ExecuteNonQuery();
        connection.Close();
        MessageBox.Show("Data successfully update");
        cancel();
        LoadData();


    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

Рисунок 3.3.2.4.1– SQL-запрос для обновления

3.3.3. Построение графиков

Чтобы построить график необходимо использовать запрос Select* from (название таблицы), для этого сначала пропишем подключение к нашей БД. Все данные из таблицы мы загружаем в dataset, далее мы создали таблицу, в которую поместили два параметра dataset и имя таблицы, которую будем заполнять. После этого мы обрабатываем событие Click, где нужно проверить наш dataset, если он не нулевой, то мы отчистим, иначе мы создаем параметры для графика, series отвечает за хранение графика, другие два параметра берутся из нашей БД, а именно

инкассация, как целочисленное значение и код смены, как строка. После этого мы проходимся по строкам всей наши таблицы и выбираем лишь строки, относящиеся к коду смены и инкассации, после чего мы очищаем ось X и задаем ей значения от кода смены. Задаем объект линия, для которой присваиваем значения инкассации и строим график. (рис 3.3.3.1 и рис. 3.3.3.2)

```
ссылка: 1
private void Chart_Load(object sender, EventArgs e)
{
    MySqlConnection connection = new MySqlConnection("server=localhost;port=3306;username=root; password=root;database=shopzoo");
    connection.Open();

    da = new MySqlDataAdapter("SELECT*FROM end_of_shift ", connection);
    dataSet = new DataSet();

    cartesianChart1.LegendLocation = LegendLocation.Bottom;
}

ссылка: 1
private void toolStripButton1_Click(object sender, EventArgs e)
{
    if (dataSet.Tables["end_of_shift"]!=null)
    dataSet.Tables["end_of_shift"].Clear();

    da.Fill(dataSet, "end_of_shift");
    dt = dataSet.Tables["end_of_shift"];
    SeriesCollection series = new SeriesCollection();
    ChartValues<int> ShiftValues = new ChartValues<int>();
    List<string> code = new List<string>();
    foreach (DataRow row in dt.Rows)
    {
        ShiftValues.Add(Convert.ToInt32(row["Collection"]));
        code.Add(Convert.ToInt32(row["Shift_code"]).ToString());
    }
    cartesianChart1.AxisX.Clear();
    cartesianChart1.AxisX.Add(new Axis()
    {
        Title="Код",
        Labels=code
    });
    LineSeries line = new LineSeries();
    line.Title = "Инкассация";
    line.Values = ShiftValues;

    series.Add(line);
```

Рисунок 3.3.3.1 – Код для события Load

```
ссылка: 1
private void toolStripButton1_Click(object sender, EventArgs e)
{
    if (dataSet.Tables["end_of_shift"]!=null)
    dataSet.Tables["end_of_shift"].Clear();

    da.Fill(dataSet, "end_of_shift");
    dt = dataSet.Tables["end_of_shift"];
    SeriesCollection series = new SeriesCollection();
    ChartValues<int> ShiftValues = new ChartValues<int>();
    List<string> code = new List<string>();
    foreach (DataRow row in dt.Rows)
    {
        ShiftValues.Add(Convert.ToInt32(row["Collection"]));
        code.Add(Convert.ToInt32(row["Shift_code"]).ToString());
    }
    cartesianChart1.AxisX.Clear();
    cartesianChart1.AxisX.Add(new Axis()
    {
        Title="Код",
        Labels=code
    });
    LineSeries line = new LineSeries();
    line.Title = "Инкассация";
    line.Values = ShiftValues;

    series.Add(line);
    cartesianChart1.Series = series;

}
}
```

Рисунок 3.3.3.2 – Событие Click

Заключение

В заключении стоит сказать, что задачи, которые были поставлены изначально, были выполнены:

– в открытом доступе был проведен поиск существующих решений, которые были проанализированы как отдельно, так и между собой, это позволило выявить преимущества и недостатки, в результате были получены представления об требованиях к функционалу и оформлению системы;

– перед началом проектирования базы данных были изучены материалы на дисциплине «Базы данных», а также дополнительно были прочитаны некоторые полезные статьи на сайте habr, результатом проектирования является реализованные таблицы, хранимые в СУБД и имеющие свои особенности;

– при написании программы использовались различные пространства имен, классы, методы и знания оптимизации кода это помогло написать все необходимые функции на языке программирования C# и реализовать приложение Windows Forms, в котором имеется 16 различных форм и 2 пользовательских элемента управления;

– качественная работа приложения зависит от того, как тщательно было проведено тестирование, в моем случае я проводила тестирование и отладку двумя способами: черный и белый ящик; в основном все операции связанные с данными из БД проверялись первым методом, а функции связанные с переходами между формами, отклик программы на события проверялись по второму методу, в результате было выявлены все возможные ошибки, которые были устранены.

Отсюда следует, что цель данной курсовой работы была достигнута. Кроме разработанной информационной системы были получены полезные навыки программирования на Windows Form, а знания реализации и проектирования БД были закреплены на практике, большим плюсом было изучение СУБД PhpMyAdmin.

Библиографический список

1. Windows.Forms – пространство имен Form // https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.forms?view=netframework-4.7.2.

2. System.Windows – пространство имен Drawing // https://docs.microsoft.com/ru-ru/dotnet/api/system.drawing?view=netframework-4.7.2

3. Работа с PHPMYAdmin// https://www.researchgate.net/publication/302370556_Creating_Databases_Using_PHPMyAdmin.

4. Проектирование базы данных// https://www.researchgate.net/publication/331437238_NORMALIZATION_IN_DATABASE_DESIGN.

5. Предметная область// https://www.researchgate.net/publication/368823210_The_Impact_of_the_Efficiency_and_Effectiveness_of_Electronic_Accounting_Information_Systems_on_the_Quality_of_Accounting_Information// https://www.researchgate.net/publication/366878864_THE_NEW_DIMENSION_OF_ACCOUNTING_A_DIGITAL_ACCOUNTING_A_STUDY_ON_EXPERT_SYSTEM

Приложение А

Техническое задание на разработку информационной системы по стандарту ГОСТ 19.201-78

Приложение В

Руководство оператора.

Приложение С

Программа и методика испытаний.