**ДИПЛОМНАЯ РАБОТА**

**ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ЙОШКАРОЛИНСКОГО ТЕХНОЛОГИЧЕСКОГО КОЛЛЕДЖА “ЙОТК”**

Факультет информатики и вычислительной техники

Специальность: Информационные системы и программирование

Дипломная работа

**ТЕМА:** Разработка ИС магазина цифровой техники «AppleStore»

|  |
| --- |
| Выполнил студент группы:  И-41 Егошин А.А.  Научный руководитель:  Глазырин А.М.  Йошкар-Ола 2023 |

# Содержание:

[Введение 3](#_Toc106144859)

[1. Аналитическая часть 4](#_Toc106144860)

[1.1. Постановка цели и задач 4](#_Toc106144861)

[1.2. Актуальность выбранной темы 4](#_Toc106144862)

[1.3. Основные понятия и определения 6](#_Toc106144863)

[1.4. Описание предметной области 6](#_Toc106144864)

[1.5. Анализ предметной области 7](#_Toc106144865)

[1.6. Анализ существующих аналогов 9](#_Toc106144866)

[1.7. Требования и задачи разрабатываемой подсистемы 10](#_Toc106144867)

[1.8. Анализ и выбор СУБД 12](#_Toc106144868)

[1.9. Выбор программно-аппаратной платформы 16](#_Toc106144869)

[2. Проектная часть 20](#_Toc106144870)

[2.1. Проектирование базы данных 20](#_Toc106144871)

[2.1.1. Построение концептуальной (инфологической) модели БД 21](#_Toc106144872)

[2.1.2. Построение логической модели БД 21](#_Toc106144873)

[2.1.3. Построение физической модели БД 22](#_Toc106144874)

[2.2. Таблицы и типы данных 24](#_Toc106144875)

[2.3. Разработка Программного обеспечения 27](#_Toc106144876)

[2.3.1. Регистрация 29](#_Toc106144877)

[2.3.2. Окно админитратора 31](#_Toc106144878)

[2.3.3. Окно продавца 37](#_Toc106144879)

[Заключение 40](#_Toc106144881)

[Список использованных источников 41](#_Toc106144882)

[Приложение A. Структура БД MySQL Workbench 8 42](#_Toc106144883)

[Приложение Б. Программное обеспечение 48](#_Toc106144884)

# **Введение**

В современном мире развитие информационных технологий является важной составляющей любого бизнеса. Именно эта отрасль активно развивается и вносит свой вклад в экономику многих стран. В этом контексте магазины цифровой техники, такие как AppleStore, становятся все более популярными и приносят немалый доход своим владельцам. Однако, для того чтобы стать еще более успешным и эффективным, каждый магазин должен решить проблемы, связанные с автоматизацией бизнес-процессов и улучшить качество обслуживания клиентов.

Целью данной дипломной работы является разработка информационной системы магазина цифровой техники "AppleStore". В рамках работы будет создана система, которая позволит автоматизировать бизнес-процессы, улучшить качество обслуживания клиентов, повысить эффективность работы магазина и, как следствие, увеличить его прибыль.

Данная дипломная работа представляет собой исследование и разработку важной информационной системы, которая может быть использована в различных магазинах цифровой техники. Предлагаемый подход позволяет повысить эффективность работы и качество обслуживания клиентов не только в магазинах, но и в других организациях, требующих автоматизации бизнес-процессов.

# **1. Аналитическая часть**

# **1.1 Постановка цели и задач**

Целью данной дипломной работы является проектирование и разработка информационной системы для магазина цифровой техники с использованием современных технологий программирования, баз данных и организации взаимодействия с клиентами в целях оптимизации бизнес-процессов магазина и повышения качества обслуживания клиентов.

Для достижения этой цели необходимо выполнить следующие задачи:

Изучить предметную область торговли электроникой и бытовой техникой, а также существующие информационные системы для магазинов.

Разработать требования к информационной системе магазина цифровой техники на основе анализа бизнес-процессов и потребностей клиентов.

Разработать базу данных для информационной системы

Проектировать архитектуру информационной системы, включая схемы баз данных, функциональные и нефункциональные требования.

Реализовать информационную систему с использованием современных технологий программирования и баз данных.

Провести тестирование и отладку информационной системы, чтобы гарантировать ее стабильность и безопасность.

Выполнение этих задач позволит успешно реализовать информационную систему для магазина цифровой техники, повысить качество обслуживания и улучшить бизнес-процессы магазина.

# **1.2 Актуальность выбранной темы**

Актуальность данной темы заключается в том, что корпоративная социальная ответственность является одним из ключевых элементов в развитии современных корпораций в условиях жесткой конкуренции. В цифровой индустрии, покупка техники Apple является популярным выбором потребителей по всему миру. Социальная ответственность является важным критерием для клиентов, который влияет на выбор продукта и развитие доверия к магазину.

В работе можно проанализировать политику Apple Store в области социальной ответственности, их социальный вклад в общество, включая благотворительную деятельность и процедуры ответственного производства, а также проанализировать влияние этой политики на имидж компании. В результате, студент сможет оценить, насколько эффективна социальная ответственность для улучшения репутации магазина цифровой техники "Apple Store".

Кроме того, в работе можно охватить темы связанные со стратегией корпоративной социальной ответственности, описать основные этапы развития и реализации социальной политики, анализировать отзывы пользователей, а также рассмотреть меры повышения эффективности социальной ответственности в магазине Apple Store.

Исследование данной темы позволит студенту получить ценные навыки в области анализа социальной ответственности, анализа репутации бренда, стратегий развития и реализации корпоративной социальной политики. Результаты работы могут быть применены не только в области магазинов цифровой техники, но и в любой другой компании, ориентированной на развитие и укрепление своей репутации в глазах потребителей и общества в целом.

Таким образом, тема дипломной работы, связанной с анализом социальной ответственности AppleStore, является актуальной и может представлять интерес не только для студентов, но и для бизнес-сообщества. В результате выполнения такой работы можно получить ценный опыт в области анализа социальной ответственности и репутации бренда, а также выработать рекомендации по повышению эффективности политики корпоративной социальной ответственности в магазине цифровой техники AppleStore и других компаниях.

# **1.3 Основные понятия и определения**

Информационная система (ИС) - система, главными задачами которой являются хранение, распространение, поиск и обработка информации.

Автоматизация - применение технических средств, экономико-математических методов и систем управления, освобождающих человека частично или полностью от непосредственного участия в процессах получения, преобразования, передачи и использования энергии, материалов или информации.

Автоматизированная система управления - это система «человек-машина», которая должна обеспечивать автоматизированный сбор и обработку информации для оптимизации процесса управления. В отличие от автоматических систем, где человек полностью удалён из контура управления, АСУ предполагает активное участие человека в контуре управления.

# **1.4 Описание предметной области**

Объектом исследования и разработки в данной дипломной работе является информационная система для магазина цифровой техники. Предметной областью является сфера торговли электронными и бытовыми приборами, а также всевозможной цифровой техникой.

Магазин цифровой техники является объектом розничной торговли, где покупатели могут приобретать различные устройства, такие как мобильные телефоны, планшеты, смарт-часы, ноутбуки, компьютеры. Эта сфера активно развивается и становится все более важной, так как все больше людей используют цифровую технику в повседневной жизни.

Развитие технологий и все большая конкуренция приводят к тому, что магазины цифровой техники должны оперативно адаптироваться к изменениям в рынке, управлять своими бизнес-процессами более эффективно и предоставлять высококачественный сервис клиентам. Для решения этих задач необходимо использование информационной системы, которая позволит увеличить эффективность управления магазином, сократить затраты на процессы коммерческого учета товаров и управления складом, а также повысить уровень обслуживания клиентов.

В данной дипломной работе будет разработана информационная система для магазина цифровой техники. Она будет использоваться сотрудниками магазина и клиентами, и будет основана на современных технологиях программирования, баз данных и организации взаимодействия с клиентами. Разработанная система будет включать следующие функции: управление товарами, учет товаров, операции продажи, управление клиентами. Эта информационная система будет помогать магазину улучшать качество обслуживания клиентов, увеличивать продажи и конкурентоспособность на рынке.

# **1.5 Анализ предметной области**

Для анализа предметной области мною были учтены особенности работы магазина и требования заказчика.

Сфера торговли электронной и бытовой техникой является одной из наиболее быстрорастущих и развивающихся в мире. По данным исследовательских компаний, объем рынка цифровой техники продолжает расти с каждым годом и ожидается, что он достигнет 2,1 триллиона долларов США к 2025 году.

Сфера торговли электроникой и бытовой техникой характеризуется значительной конкуренцией. На рынке постоянно появляются новые бренды, продукты и технологии, что требует от магазинов постоянного обновления ассортимента и улучшения сервиса для клиентов.

Одной из главных проблем в данной сфере является управление складом и логистикой. В связи с быстрым оборотом новых позиций на рынке и потребностью в оперативном реагировании на изменения спроса, магазины сталкиваются с проблемой эффективного управления складом и оптимизации логистики.

Магазин цифровой техники "AppleStore" может затрагивать ряд социальных проблем. Например:

1. Экологическая проблема. Производство и потребление цифровой техники, в том числе продукции компании Apple, приводят к массовому накоплению электронных отходов, которые могут нанести серьезный вред окружающей среде.

2. Экономические проблемы. Цифровая техника от Apple является достаточно дорогой, что может привести к тому, что значительная часть населения не сможет позволить себе приобретение продукции этой компании. Это может привести к неравенству в доступе к современным технологиям.

3. Этические проблемы. Среди проблем, которые могут быть связаны с продажей техники от Apple, можно отметить незаконные условия производства, использование детского труда, нарушение прав работников, а также многие другие моральные и этические проблемы, связанные с ресурсами и производством.

Кроме того, продажа цифровой техники может затрагивать и другие социальные проблемы, такие как угрозы кибербезопасности, киберпреступности, утери личной информации и т.д.

Магазин цифровой техники "Apple Store" имеет несколько плюсов, которые тоже стоит учитывать:

1. Инновационные решения и высокое качество продукции. Apple славится своими инновационными решениями и технологиями, которые позволяют создавать высококачественные продукты, удобные в использовании и современные.

2. Широкий спектр продукции. Apple Store предлагает широкий ассортимент продукции, начиная от смартфонов и планшетов, заканчивая ноутбуками и аксессуарами.

3. Уникальный дизайн. Продукция от Apple отличается уникальным и стильным дизайном, что делает ее предпочтительной для тех, кто уделяет важность внешнему виду техники.

4. Бренд и репутация. Бренд Apple уже давно занимает лидирующие позиции в области цифровой техники, благодаря своей репутации и стабильному качеству продукции.

5. Число сервисных центров и дополнительных услуг. Apple Store также предлагает широкий спектр дополнительных услуг и сервисных центров, что делает удобным обслуживание и настройку продуктов, а также оперативное решение проблем при их наличии.

Важно учитывать как преимущества, так и недостатки данного магазина цифровой техники при оценке его значения и социальной значимости.

ИС предполагает две роли пользователей: администратор и продавец.

Администратор занимается работой с сайтом, товарами, клиентской базой, а также имеет возможность просматривать отчетность работы магазина.

Продавец может работать с таблицами БД, продавать товары.

# **1.6 Анализ существующих аналогов**

Обзор аналогов проектируемой системы производиться для формирования массива функциональных требований новой проектируемой системы. Анализ прототипа необходим для выявления перечня существующих недостатков по трем направлениям: недостатки функциональной структуры, перечень недостатков отдельных функций системы прототипа и конструктивные недостатки функциональных элементов. Для проектирования функциональной структуры был произведен обзор существующих аналогов и прототипов программного продукта. В рамках курсовой работы мной был проведен анализ существующих информационных систем, осуществляющих подобную деятельность. В целом, они схожи, так как предприятия подобного типа осуществляют одинаковую деятельность. Различия имеются в имеющемся товаре, тарифах на перевозки, системах обслуживания. При анализе других информационных систем были выявлены следующие недостатки:

- загроможденный интерфейс;

- отсутствие кнопок для часто используемых функций;

- обилие полей для малоиспользуемых функций;

- ошибки в работе ИС;

# **1.7** **Требования и задачи разрабатываемой подсистемы**

Процесс первичного сбора требований является первым шагом в процессе создания требований. Образ продукта описывает, что сейчас представляет собой продукт и каким он станет впоследствии. Образ продукта подразумевает определение долгосрочных перспектив развития проекта, в то время как границы проекта могут быть определены для конкретной итерации. Следует очень внимательно отнестись к формированию образа и границ проекта. Это позволит избежать или, по крайней мере, запланировать на более позднее время реализацию менее критичных для успеха проекта возможностей ПО.

Совокупность организационного, программного и технического обеспечения составляют информационную систему, позволяющую автоматизировать деятельность предприятия.

Информационная система должна отвечать функциональным требованиям, а также соответствовать требованиям к удобству пользования, надёжности, производительности и поддержке, при этом соответствовать современным стандартам.

Требования, описывающие потребность предприятия относительно будущей подсистемы, рассматриваются с двух сторон.

Со стороны потребителя – выгодность применения. Разрабатываемая система должна стать одной из составляющих уже используемой, поэтому предприятию не придется затрачивать дополнительные средства на приобретение базового ПО.

Со стороны субъектов ИС – безопасность. Система должна отвечать современным требованиям безопасности. Любая информационная система обязана гарантировать защиту используемых персональных данных всех физических лиц согласно статье 86 ТК РФ.

Требования к системе в целом:

– требования к режиму функционирования системы. Подсистема должна функционировать в локальной сети в виде клиент-серверного взаимодействия. Разрабатываемая подсистема должна поддерживать архитектуру «клиент-сервер». перспективы развития и модернизации. Подсистема должна реализовывать возможность дальнейшей модернизации, как программного обеспечения, так и комплекса технических средств.

Требования пользователей, исходящие от персонала, который непосредственно будет использовать подсистему по её назначению:

– быстрое добавление новых товаров;

– возможность редактирования информации всех разделов;

– простота внесения данных о новых совершенных торговых операциях;

Основными ошибками, допускаемыми этапе первичного сбора требований, являются:

• Нечеткое определение границ проекта, что приводит к затягиванию сроков проекта.

• Недостаточно четко определенные группы пользователей продукта. •

• Попытка с первого раза максимально детализировать и проанализировать все требования.

Задачи:

- выполнить проектирование информационного и программного обеспечения системы;

- разработать скрипты описания базы данных и программные коды приложения;

- провести тестирование программного продукта.

При определении области действия программного продукта эффективнее всего воспользоваться методикой «будет/не будет». Ниже определены рамки проекта.

Относительно данного проекта можно сказать, что он будет:

- внутренним;

- применяться в операционных системах Windows;

- предназначен для автоматизации процессов предприятия.

Проект не будет:

- предназначенным для обеспечения доступа к информации не из предприятия;

- использоваться в системах отличных от Windows.

# **1.8 Анализ и выбор СУБД**

База данных — совокупность данных хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.

Поэтому нужно произвести обзор существующих СУБД. По требованиям заказчика в обзоре будет участвовать только бесплатное ПО.

Существуют десятки СУБД, написанных для самых разных платформ, такие как Oracle, IBM DB2, SQL Server, Teradata, Sybase, Ingres, Informix, Empress, MySQL, mSQL, PostgreSQL, LEAP RDBMS, FirstBase, Ocelot, Progress, Typhoon, SQL/DS, Daffodil DB, Cloudscape, Compad Non-Stop SQL MX и SQL/MP, Linter RDBMS SQL, Interbase, Universe, GNU SQL Server.

Рассмотрим некоторые из них:

MS SQL Server

Исходный код MS SQL Server (до версии 7.0) основывался на коде Sybase SQL Server, и это позволило Microsoft выйти на рынок баз данных для предприятий, где конкурировали Oracle, IBM, и, позже, сама Sybase. Microsoft, Sybase и Ashton-Tate первоначально объединились для создания и выпуска на рынок первой версии программы, получившей название SQL Server 1.0 для OS/2 (около 1989 года), которая фактически была эквивалентом Sybase SQL Server 3.0 для Unix, VMS и др. Microsoft SQL Server 4.2 был выпущен в 1992 году и входил в состав операционной системы Microsoft OS/2 версии 1.3. Официальный релиз Microsoft SQL Server версии 4.21 для ОС Windows NT состоялся одновременно с релизом самой Windows NT (версии 3.1). Microsoft SQL Server 6.0 был первой версией SQL Server, созданной исключительно для архитектуры NT и без участия в процессе разработки Sybase.

Сервер баз данных Microsoft SQL Server в качестве языка запросов использует версию языка SQL, получившую название Transact-SQL (сокращённо T-SQL). Язык T-SQL является реализацией SQL-92 (стандарт ISO для языка SQL) с множественными расширениями. T-SQL позволяет использовать дополнительный синтаксис для хранимых процедур и обеспечивает поддержку транзакций (взаимодействие базы данных с управляющим приложением).

При взаимодействии с сетью Microsoft SQL Server и Sybase ASE используют протокол уровня приложения под названием Tabular Data Stream (TDS, протокол передачи табличных данных). Протокол TDS также был реализован в проекте FreeTDS с целью обеспечить различным приложениям возможность взаимодействия с базами данных Microsoft SQL Server и Sybase.

Для обеспечения доступа к данным Microsoft SQL Server поддерживает Open Database Connectivity (ODBC) -- интерфейс взаимодействия приложений с СУБД. Версия SQL Server 2005 обеспечивает возможность подключения пользователей через веб-сервисы, использующие протокол SOAP. Это позволяет клиентским программам, не предназначенным для Windows, кроссплатформенно соединяться с SQL Server. Компания Microsoft также выпустила сертифицированный драйвер JDBC, позволяющий приложениям под управлением Java (таким как BEA и IBM WebSphere) соединяться с Microsoft SQL Server 2000 и 2005.

Также SQL Server поддерживает зеркалирование и кластеризацию баз данных. Кластер сервера SQL - это совокупность одинаково конфигурированных серверов; такая схема помогает распределить рабочую нагрузку между несколькими серверами. Все сервера имеют одно виртуальное имя, и данные распределяются по IP-адресам машин кластера в течение рабочего цикла. Также в случае отказа или сбоя на одном из серверов кластера доступен автоматический перенос нагрузки на другой сервер.

В SQL Server 2005 встроена поддержка.NET Framework. Благодаря этому хранимые процедуры БД могут быть написаны на любом языке платформы.NET, используя полный набор библиотек, доступных для.NET Framework, включая Common Type System (система обращения с типами данных в Microsoft.NET Framework). Однако, в отличие от других процессов,.NET Framework, будучи базисной системой для SQL Server 2005, выделяет дополнительную память и выстраивает средства управления SQL Server вместо того, чтобы использовать встроенные средства Windows. Это повышает производительность в сравнении с общими алгоритмами Windows, так как алгоритмы распределения ресурсов специально настроены для использования в структурах SQL Server.

MySQL — система управления базами данных

MySQL — это реляционная система управления базами данных с открытым исходным кодом. В настоящее время эта СУБД одна из наиболее популярных в веб-приложениях — подавляющее большинство CMS использует именно MySQL (часто только её, без альтернатив), а почти все веб-фреймворки поддерживают MySQL уже на уровне базовой конфигурации (без дополнительных модулей).

Из преимуществ СУБД MySQL стоит отметить простоту использования, гибкость, низкую стоимость владения (относительно платных СУБД), а также масштабируемость и производительность.

MySQL позволяет хранить целочисленные значения со знаком и беззнаковые, длиной в 1, 2, 3, 4 и 8 байтов, работает со строковыми и текстовыми данными фиксированной и переменной длины, позволяет осуществлять SQL-команды SELECT, DELETE, INSERT, REPLACE и UPDATE, обеспечивает полную поддержку операторов и функций в SELECT- и WHERE- частях запросов, работает с GROUP BY и ORDER BY, поддерживает групповые функции COUNT(), AVG(), STD(), SUM(), MAX() и MIN(), позволяет использовать JOIN в запросах, в т.ч. LEFT OUTER JOIN и RIGHT OUTER JOIN, поддерживает репликацию, транзакции, работу с внешними ключами и каскадные изменения на их основе, а также обеспечивает многие другие функциональные возможности.

Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей. Есть и другие типы таблиц, разработанные сообществом.

СУБД MySQL появилась в 1995. Написана на C и C++, протестирована на множестве различных компиляторов и работает на различных платформах. C 2010 года разработку и поддержку MySQL осуществляет корпорация Oracle. Продукт распространяется как под GNU GPL, так и под собственной коммерческой лицензией. Однако по условиям GPL, если какая-либо программа включает исходные коды MySQL, то и эта программа тоже должна распространяться по лицензии GPL. Для нежелающих открывать исходные тексты своих программ как раз предусмотрена коммерческая лицензия, которая, в дополнение к возможности разработки под «закрытой» лицензией, обеспечивает качественную сервисную поддержку.

Для разработки базы данных я выбрал СУБД MySQL так как выявил следующие плюсы для работы с ним:

* многопоточность, поддержка нескольких одновременных запросов;
* оптимизация связей с присоединением многих данных за один проход;
* записи фиксированной и переменной длины;
* ODBC драйвер;
* гибкая система привилегий и паролей;
* гибкая поддержка форматов чисел, строк переменной длины и меток времени;
* интерфейс с языками C и [Perl](https://www.methodlab.ru/technology/perl.shtml), PHP;
* быстрая работа, масштабируемость;
* совместимость с ANSI SQL;
* бесплатна в большинстве случаев;
* хорошая поддержка со стороны провайдеров услуг хостинга;
* быстрая поддержка транзакций через механизм InnoDB.

# **1.9** **Выбор программно-аппаратной платформы**

Microsoft Visual Studio — линейка продуктов компании [Microsoft](https://ru.wikipedia.org/wiki/Microsoft), включающих [интегрированную среду разработки](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) программного обеспечения и ряд других инструментов. Данные продукты позволяют разрабатывать как [консольные](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F) [приложения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), так и игры и приложения с [графическим интерфейсом](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F), в том числе с поддержкой технологии [Windows Forms](https://ru.wikipedia.org/wiki/Windows_Forms), а также [веб-сайты](https://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%B9%D1%82), [веб-приложения](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5), [веб-службы](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%BB%D1%83%D0%B6%D0%B1%D0%B0) как в [родном](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4), так и в [управляемом](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%8F%D0%B5%D0%BC%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) кодах для всех платформ, поддерживаемых [Windows](https://ru.wikipedia.org/wiki/Windows), [Windows Mobile](https://ru.wikipedia.org/wiki/Windows_Mobile), [Windows CE](https://ru.wikipedia.org/wiki/Windows_CE), [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework), [Xbox](https://ru.wikipedia.org/wiki/Xbox), [Windows Phone](https://ru.wikipedia.org/wiki/Windows_Phone) [.NET Compact Framework](https://ru.wikipedia.org/wiki/.NET_Compact_Framework) и [Silverlight](https://ru.wikipedia.org/wiki/Silverlight" \o "Silverlight).

В Visual Studio можно создавать приложения, которые подключаются к данным практически в любом продукте или службе базы данных в любом формате, где угодно — на локальном компьютере, в локальной сети или в общедоступном, частном или гибридном облаке.

Visual Studio включает в себя [редактор исходного кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80_%D0%B8%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BE%D0%B4%D0%B0) с поддержкой технологии [IntelliSense](https://ru.wikipedia.org/wiki/IntelliSense" \o "IntelliSense) и возможностью простейшего [рефакторинга кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3). Встроенный [отладчик](https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio_Debugger) может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер [классов](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) и дизайнер [схемы базы данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Visual Studio позволяет создавать и подключать сторонние дополнения ([плагины](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD)) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем [контроля версий исходного кода](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%D0%BC%D0%B8) (как, например, [Subversion](https://ru.wikipedia.org/wiki/Subversion" \o "Subversion) и [Visual SourceSafe](https://ru.wikipedia.org/wiki/Microsoft_Visual_SourceSafe)), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на [предметно-ориентированных языках программирования](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%B4%D0%BC%D0%B5%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA)) или инструментов для прочих аспектов [процесса разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) (например, клиент Team Explorer для работы с [Team Foundation Server](https://ru.wikipedia.org/wiki/Team_Foundation_Server)).

Разработка приложения была выполнена на языке программирования C#.

Название «Си шарп» (от англ. sharp — диез) происходит от [буквенной музыкальной нотации](https://ru.wikipedia.org/wiki/%D0%91%D1%83%D0%BA%D0%B2%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BD%D0%BE%D1%82%D0%B0%D1%86%D0%B8%D1%8F), где латинской букве C соответствует нота [До](https://ru.wikipedia.org/wiki/%D0%94%D0%BE), а знак [диез](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B5%D0%B7) (англ. sharp) означает повышение соответствующего ноте [звука](https://ru.wikipedia.org/wiki/%D0%97%D0%B2%D1%83%D0%BA) на [полутон](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%83%D1%82%D0%BE%D0%BD)[[8]](https://ru.wikipedia.org/wiki/C_Sharp#cite_note-8), что аналогично названию языка [C++](https://ru.wikipedia.org/wiki/C%2B%2B), где «++» обозначает [инкремент](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%BA%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D1%82) [переменной](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)). Название также является игрой с цепочкой [C](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) → [C++](https://ru.wikipedia.org/wiki/C%2B%2B) → C++++(C#), так как символ «#» можно представить состоящим из 4 знаков «+»[[9]](https://ru.wikipedia.org/wiki/C_Sharp#cite_note-9).

Из-за технических ограничений на отображение (стандартные шрифты, браузеры и т. д.), а также из-за того, что знак диеза ♯ не представлен на стандартной клавиатуре компьютера, при записи имени языка программирования используют [знак решётки](https://ru.wikipedia.org/wiki/%D0%97%D0%BD%D0%B0%D0%BA_%D1%80%D0%B5%D1%88%D1%91%D1%82%D0%BA%D0%B8) (#)[[10]](https://ru.wikipedia.org/wiki/C_Sharp#cite_note-MS_CSharp_FAQ-10). Это соглашение отражено в Спецификации языка C# ECMA-334[[11]](https://ru.wikipedia.org/wiki/C_Sharp#cite_note-ECMA-334-11). Тем не менее, на практике (например, при размещении рекламы и коробочном дизайне[[12]](https://ru.wikipedia.org/wiki/C_Sharp#cite_note-12)), «Майкрософт» использует знак диеза.

Названия языков программирования не принято переводить, поэтому язык называют, используя транскрипцию, — «Си шарп».

C# — [объектно-ориентированный](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) [язык программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F). Разработан в [1998](https://ru.wikipedia.org/wiki/1998)—[2001 годах](https://ru.wikipedia.org/wiki/2001_%D0%B3%D0%BE%D0%B4) группой инженеров компании [Microsoft](https://ru.wikipedia.org/wiki/Microsoft) под руководством [Андерса Хейлсберга](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D0%B9%D0%BB%D1%81%D0%B1%D0%B5%D1%80%D0%B3,_%D0%90%D0%BD%D0%B4%D0%B5%D1%80%D1%81) и Скотта Вильтаумота[[7]](https://ru.wikipedia.org/wiki/C_Sharp" \l "cite_note-7) как язык разработки приложений для платформы [Microsoft .NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework). Впоследствии был стандартизирован как [ECMA](https://ru.wikipedia.org/wiki/ECMA)-334 и [ISO](https://ru.wikipedia.org/wiki/ISO)/[IEC](https://ru.wikipedia.org/wiki/IEC) 23270.

C# относится к семье языков с [C-подобным синтаксисом](https://ru.wikipedia.org/wiki/C-%D0%BF%D0%BE%D0%B4%D0%BE%D0%B1%D0%BD%D1%8B%D0%B9_%D1%81%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81), из них его синтаксис наиболее близок к [C++](https://ru.wikipedia.org/wiki/C%2B%2B) и [Java](https://ru.wikipedia.org/wiki/Java). Язык имеет [статическую типизацию](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), поддерживает [полиморфизм](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), [перегрузку операторов](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B0_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BE%D0%B2) (в том числе операторов явного и неявного приведения типа), [делегаты](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BB%D0%B5%D0%B3%D0%B0%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), атрибуты, [события](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D0%B9%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [переменные](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [свойства](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [обобщённые](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D1%91%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) типы и методы, [итераторы](https://ru.wikipedia.org/wiki/%D0%98%D1%82%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80), [анонимные функции](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%BE%D0%BD%D0%B8%D0%BC%D0%BD%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) с поддержкой [замыканий](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%BC%D1%8B%D0%BA%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [LINQ](https://ru.wikipedia.org/wiki/Language_Integrated_Query), [исключения](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%B8%D1%81%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B9), [комментарии](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%80%D0%B8%D0%B8_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) в формате [XML](https://ru.wikipedia.org/wiki/XML).

Переняв многое от своих предшественников — языков [C++](https://ru.wikipedia.org/wiki/C%2B%2B), [Delphi](https://ru.wikipedia.org/wiki/Delphi_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)), [Модула](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D0%B0-2), [Smalltalk](https://ru.wikipedia.org/wiki/Smalltalk" \o "Smalltalk) и, в особенности, [Java](https://ru.wikipedia.org/wiki/Java) — С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает [множественное наследование](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BD%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) классов (между тем допускается [множественная реализация интерфейсов](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_(%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)#%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BD%D0%B0%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%B8_%D1%80%D0%B5%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%8)).

Особенности данного языка программирования.

C# разрабатывался как язык программирования прикладного уровня для [CLR](https://ru.wikipedia.org/wiki/Common_Language_Runtime) и, как таковой, зависит, прежде всего, от возможностей самой CLR. Это касается, прежде всего, системы типов C#, которая отражает [BCL](https://ru.wikipedia.org/wiki/Base_Class_Library). Присутствие или отсутствие тех или иных выразительных особенностей языка диктуется тем, может ли конкретная языковая особенность быть транслирована в соответствующие конструкции CLR. Так, с развитием CLR от версии 1.1 к 2.0 значительно обогатился и сам C#; подобного взаимодействия следует ожидать и в дальнейшем (однако, эта закономерность была нарушена с выходом C# 3.0, представляющего собой расширения языка, не опирающиеся на расширения платформы .NET). CLR предоставляет C#, как и всем другим [.NET](https://ru.wikipedia.org/wiki/.NET_Framework)-ориентированным языкам, многие возможности, которых

лишены «классические» языки программирования. Например, [сборка мусора](https://ru.wikipedia.org/wiki/%D0%A1%D0%B1%D0%BE%D1%80%D0%BA%D0%B0_%D0%BC%D1%83%D1%81%D0%BE%D1%80%D0%B0) не реализована в самом C#, а производится CLR для программ, написанных на C# точно так же, как это делается для программ на [VB.NET](https://ru.wikipedia.org/wiki/Visual_Basic_.NET), [J#](https://ru.wikipedia.org/wiki/Visual_J%E2%99%AF) и др.

# **2. Проектная часть**

# **2.1 Проектирование базы данных**

Проектирование базы данных — это процесс создания модели данных, которая определяет, как будут храниться, обрабатываться и организовываться данные в базе данных. Ниже приведены возможные аспекты, которые можно включить в проектирование базы данных:

1. Анализ требований: изучение потребностей пользователей, определение функциональности системы, определение основных свойств, которыми должна обладать база данных.

2. Создание концептуальной модели данных: разработка концептуальной модели базы данных, которая определяет структуру информации и связи между сущностями.

3. Создание логической модели данных: создание логической модели данных, которая фиксирует сущности, атрибуты и связи между ними, с учетом бизнес-правил и требований к базе данных.

4. Создание физической модели данных: создание физической модели данных, которая определяет структуру и хранение данных, учитывая специфические требования к программному обеспечению, на которое будет установлена база данных.

5. Установление правил для доступа к данным: определение прав доступа к базе данных для различных групп пользователей, определение уровней безопасности и контроля.

6. Создание схемы базы данных: создание конкретной схемы базы данных, включающей в себя таблицы, поля и ключи, учитывая особенности программного обеспечения для работы с базой данных.

7. Разработка скриптов для создания базы данных: написание скриптов для создания базы данных и всех ее компонентов, которые можно использовать для установки и настройки базы данных и т.д.

# **2.1.1 Построение концептуальной (инфологической) модели БД**

Концептуальная модель данных является высокоуровневым описанием данных для решения бизнес-задач и выступает в качестве концепции, которую впоследствии может быть использована для создания физических и логических моделей данных. Она помогает команде проектирования понимать, какая информация необходима и как взаимосвязаны сущности в рамках системы, что может быть полезно для анализа процесса бизнеса и определения бизнес-требований.

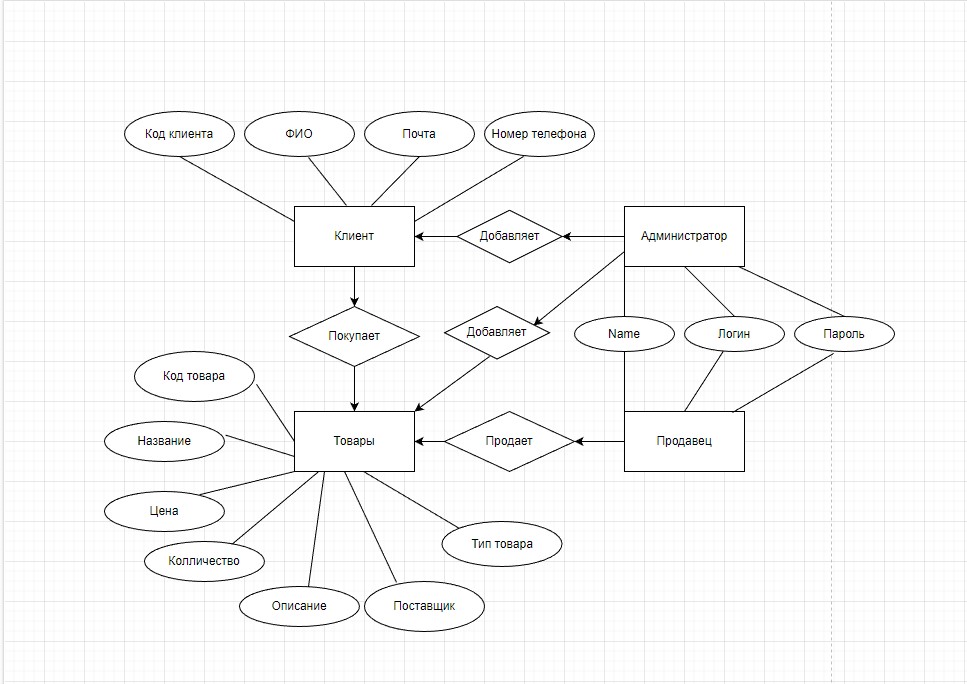


Рисунок 1. Концептуальная модель БД

# **2.1.2 Построение логической модели БД**

Логическая модель данных - это следующий уровень после концептуальной модели данных и описывает структуру данных в терминах таблиц и их связей. В отличие от концептуальной модели данных, логическая модель данных уже описывает технические детали, такие как типы данных, свойства таблиц и связей между ними, ключи и индексы.

Цель логической модели данных - описать данные в терминах их хранения и доступа, включая структуры таблиц и связи между ними, а также обеспечение выполнения бизнес-правил и ограничений целостности данных.

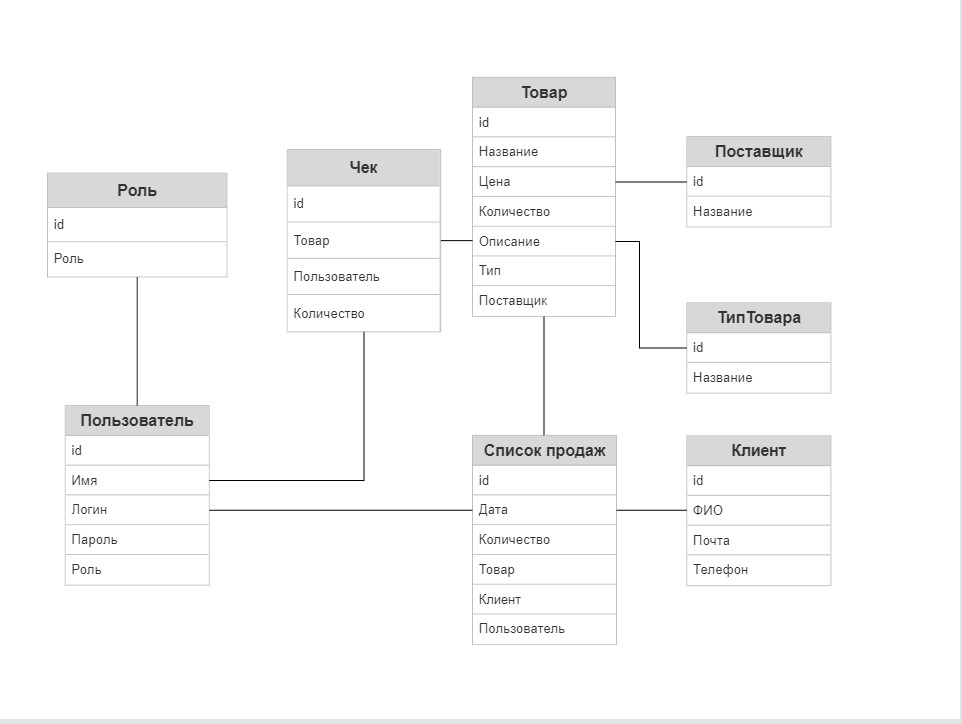


Рисунок 2. Логическая модель БД

# **2.1.3 Построение физической модели БД**

Физическая модель данных - это уровень моделирования данных, который описывает технические детали реализации базы данных. Эта модель предоставляет детальную информацию об объектах базы данных и их отношениях, используемых для хранения данных, таких как таблицы, представления, индексы, хранимые процедуры и т.д.

Физическая модель данных определяет технические детали, такие как наименование таблиц, типы данных, ограничения на поля, индексы, связи между таблицами и использование триггеров. Она описывает способы, которыми эти объекты хранят данные в реальном мире и как они могут быть связаны между собой.

Физическая модель данных, как правило, является технически-ориентированной и основывается на логической модели данных. Это значит, что она фокусируется на том, как данные будут храниться в базе данных и как они будут использоваться, а также каким образом доступ к данным будет производиться.

Цель физической модели данных - обеспечить наилучшую производительность базы данных и удовлетворение бизнес-требований. Она определяет, как база данных будет реализована и интегрирована в приложения, работающие с базой данных, что помогает оптимизировать производительность и минимизировать ошибки в работе с базой данных.

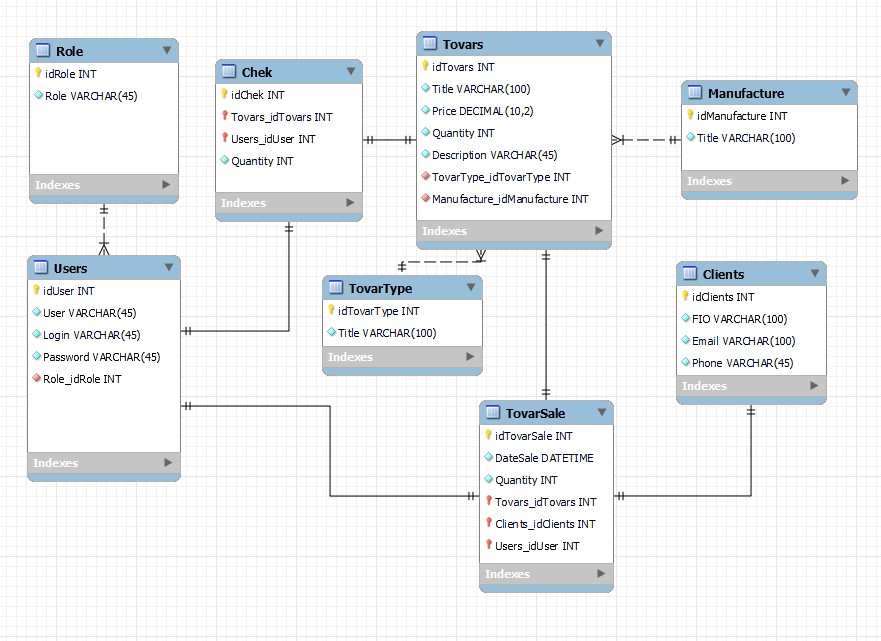


Рисунок 3. Физическая модель БД

# **2.2 Таблицы и типы данных**

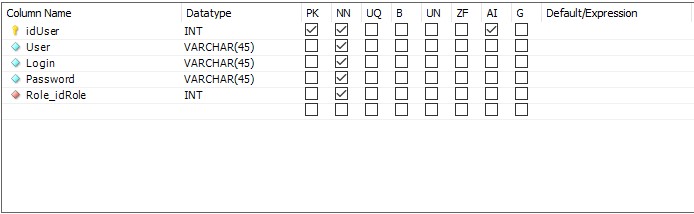


Рисунок 4 “Таблица Пользователь”

В Рисунке 4 описана Таблица Пользователь, в ней указано имя пользователя, логин, пароль.

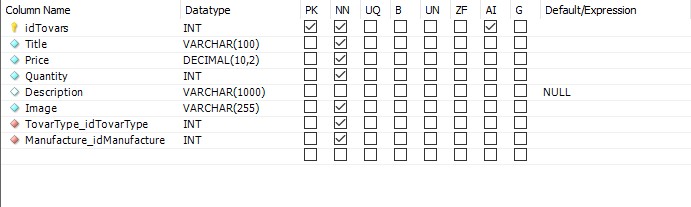


Рисунок 5 “Таблица Товар”

В Рисунке 5 описана Таблица Товар, в ней указано название товара, цена товара, количество товара, его описание, изображение, тип товара и поставщик.

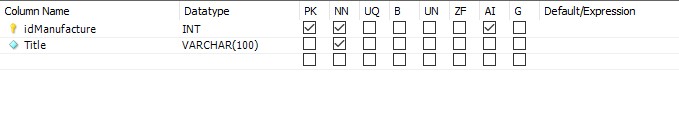


Рисунок 6 “Таблица Поставщик”

В Рисунке 6 описано название поставщика.

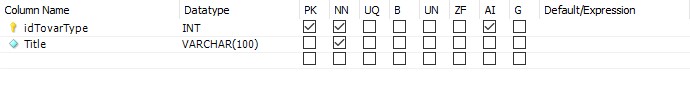


Рисунок 7 “Таблица Тип товара”

В Рисунке 7 описано название типа товара.

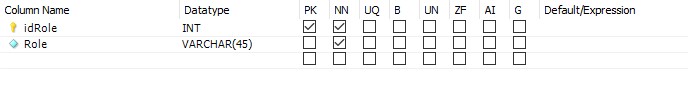


Рисунок 8 “Таблица Роль”

В Рисунке 8 описано название роли пользователя.

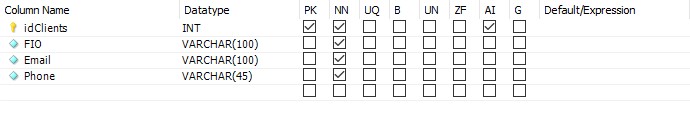


Рисунок 9 “Таблица Клиент”

В Рисунке 9 описано ФИО клиента, электронная почта клиента, мобильный телефон клиента.

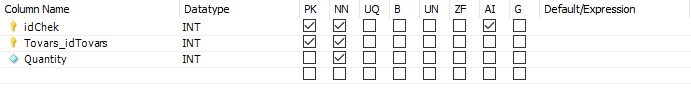


Рисунок 10 “Таблица Чек”

В Рисунке 10 описаны товар и его количество.

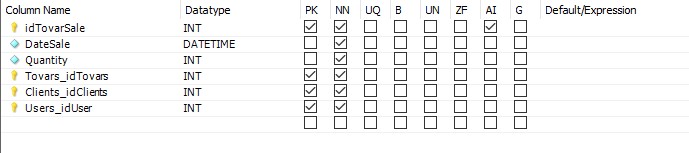


Рисунок 11 “Таблица Список продаж”

В Рисунке 11 описана дата продажи товара, количество проданного товара, сам товар, покупатель и продавец.

# **2.3 Разработка Программного обеспечения**

Приложение выполнено на языке C#. Для разработки базы данных использовалась СУБД MySQL Workbench.

Основная задача любого успешного проекта заключается в том, чтобы на момент запуска системы и в течение всего времени ее эксплуатации можно было обеспечить:

* требуемую функциональность системы и степень адаптации к изменяющимся условиям ее функционирования;
* требуемую пропускную способность системы;
* требуемое время реакции системы на запрос;
* безотказную работу системы в требуемом режиме, иными словами — готовность и доступность системы для обработки запросов пользователей;
* простоту эксплуатации и поддержки системы;

Проектирование информационных систем охватывает три основные области:

* проектирование объектов данных, которые будут реализованы в базе данных;

В реальных условиях проектирование — это поиск способа, который удовлетворяет требованиям функциональности системы средствами имеющихся технологий с учетом заданных ограничений.

К любому проекту предъявляется ряд абсолютных требований, например максимальное время разработки проекта, максимальные денежные вложения в проект и т.д. Одна из сложностей проектирования состоит в том, что оно не является такой структурированной задачей, как анализ требований к проекту или реализация того или иного проектного решения.

Считается, что сложную систему невозможно описать в принципе. Это, в частности, касается систем управления предприятием. Одним из основных аргументов является изменение условий функционирования системы, например директивное изменение тех или иных потоков информации новым руководством. Еще один аргумент — объемы технического задания, которые для крупного проекта могут составлять сотни страниц, в то время как технический проект может содержать ошибки. Возникает вопрос: а может, лучше вообще не проводить обследования и не делать никакого технического проекта, а писать систему «с чистого листа» в надежде на то, что произойдет некое чудесное совпадение желания заказчика с тем, что написали программисты, а также на то, что все это будет стабильно работать?

Если разобраться, то так ли уж непредсказуемо развитие системы и действительно ли получить информацию о ней невозможно? Вероятно, представление о системе в целом и о предполагаемых путях ее развития можно получить посредством семинаров. После этого разбить сложную систему на более простые компоненты, упростить связи между компонентами, предусмотреть независимость компонентов и описать интерфейсы между ними (чтобы изменение одного компонента автоматически не влекло за собой существенного изменения другого компонента), а также возможности расширения системы и «заглушки» для нереализуемых в той или иной версии системы функций. Исходя из подобных элементарных соображений описание того, что предполагается реализовать в информационной системе, уже не кажется столь нереальным. Можно придерживаться классических подходов к разработке информационных систем, один из которых — схема «водопада» . Кратко будут рассмотрены и некоторые другие подходы к разработке информационных систем, где использование элементов, описанных в схеме «водопада», также допустимо. Какого подхода из описываемых ниже придерживаться — в какой-то мере дело вкуса и обстоятельств.

Жизненный цикл программного обеспечения представляет собой модель его создания и использования. Модель отражает его различные состояния, начиная с момента возникновения необходимости в данном ПО и заканчивая моментом его полного выхода из употребления у всех пользователей.

# **2.3.1 Регистрация**

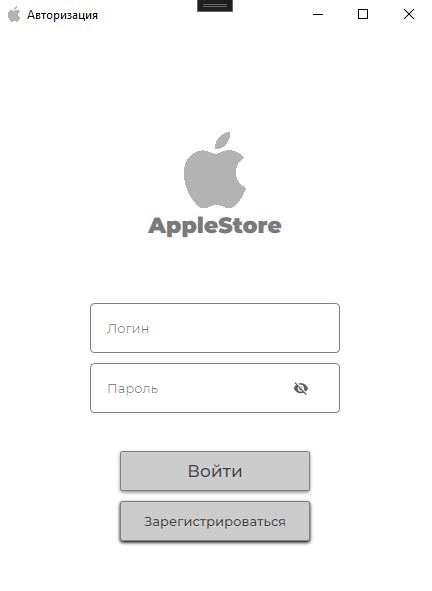


Рисунок 12 “Окно авторизации”

В окне авторизации можно войти под администратором или под продавцом. Если нет аккаунта то нужно зарегистрироваться. У нас есть два варианта регистрации администратором или продавцом.

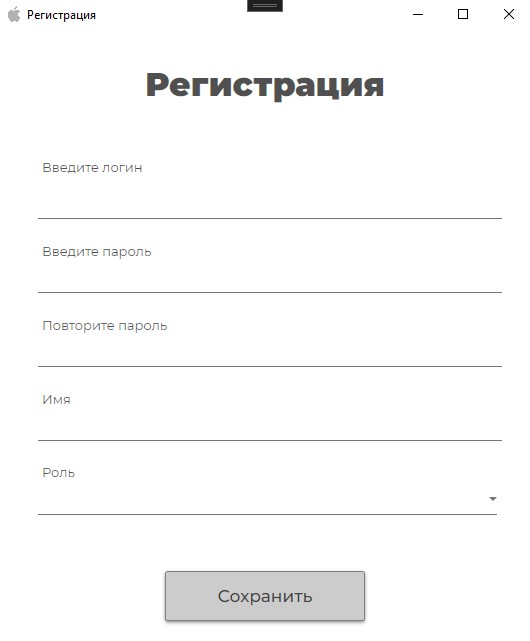


Рисунок 13 “Окно регистрации”

В окне регистрации мы регистрируем пользователя выберая ему роль администратора или продавца.

После авторизации у нас есть 2 исхода событий, если мы выбираем администратора, то попадаем в его окно.

Так же после успешной авторизации появиться сообщение с приветствием этого администратора.

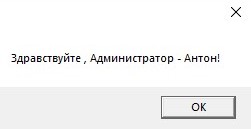


Рисунок 14 “Сообщение при успешной авторизации администратора ”

Если же вы не правильно ввели данные появиться следующее сообщение.

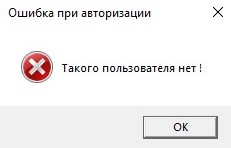


Рисунок 15 “Сообщение при не правильной авторизации”

# **2.3.2 Окно администратора**

После авторизации ва попадете в окно администратора

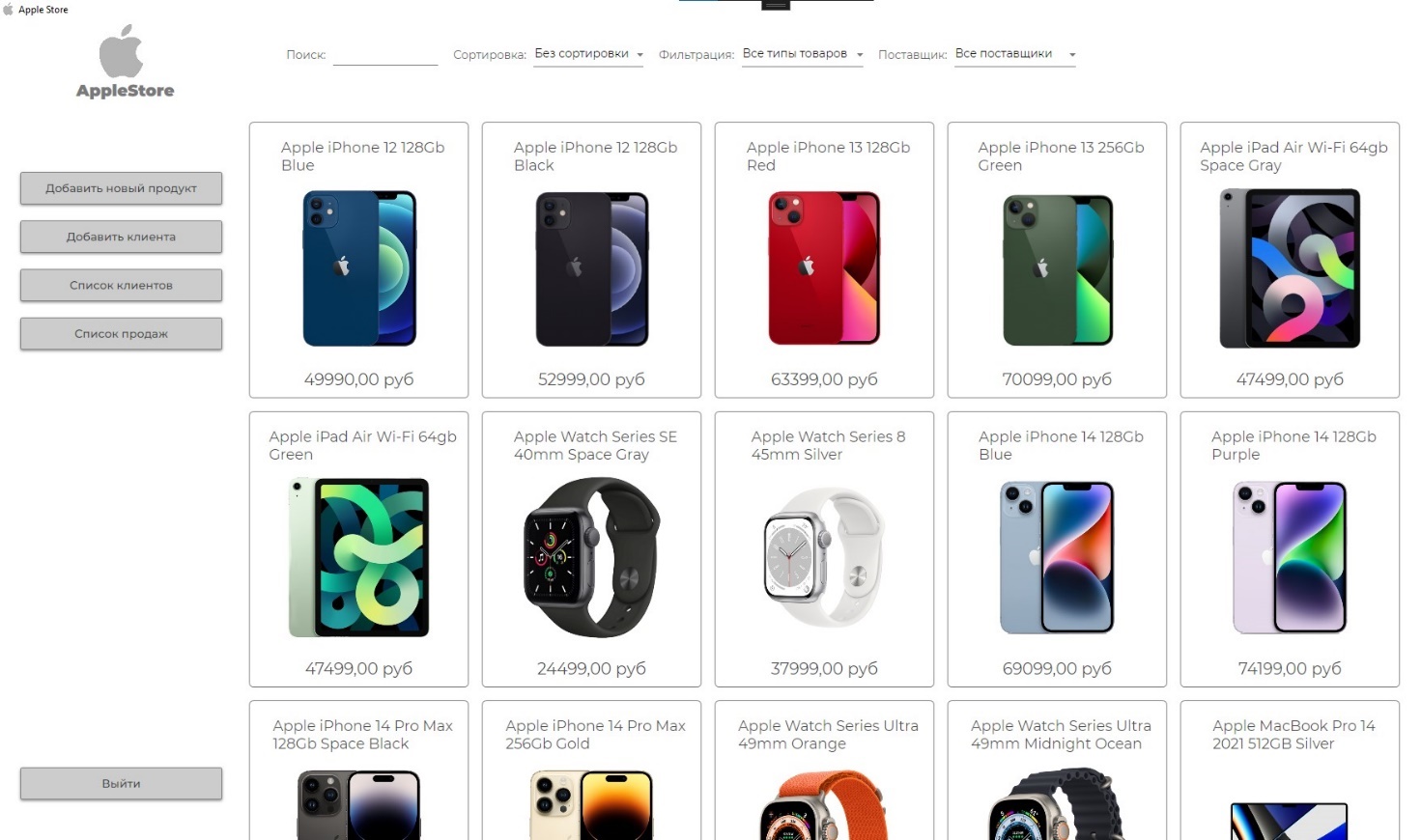


Рисунок 16 “Окно администратора”

В этом окне администратор может:

* выполнить поиск товара
* сортировку товра по названию и по цене
* фильтрацию товара по типу и поставщикам
* добавить новый товар
* добавить нового клиента
* посмотреть список клиентов
* посмотреть список продаж
* изменить текущий товар

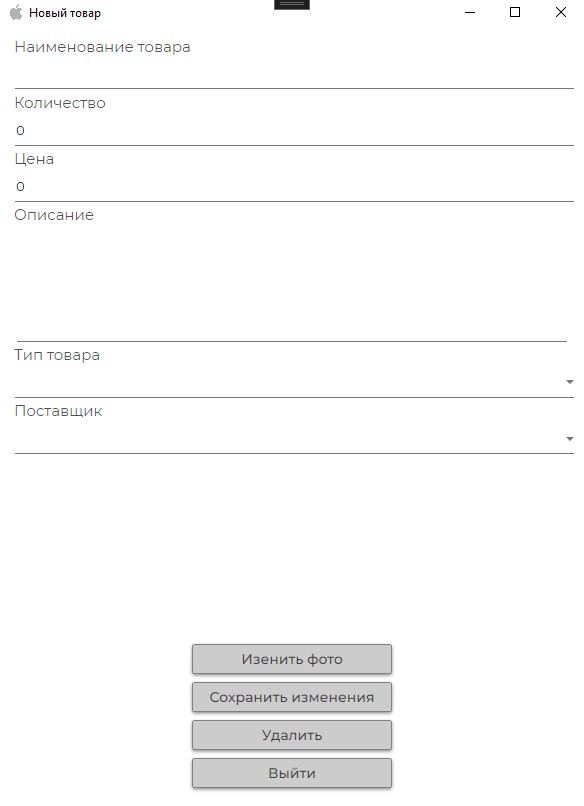


Рисунок 17 “Окно добавления нового товара”

В этом окне администратор заполняет значения нового товара и после чего он появиться в списке товаров.

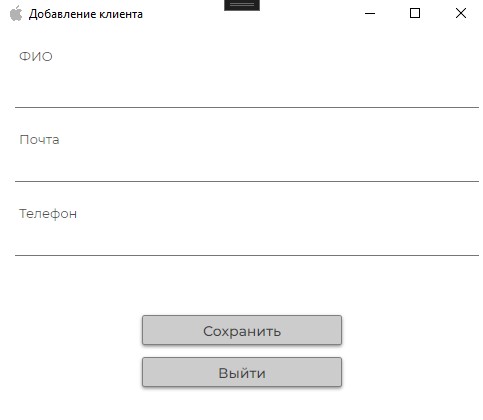


Рисунок 18 “Окно добавления нового клиента”

В этом окне администратор заполняет значения клиента и после чего он клиент добавиться в базе данных.



Рисунок 19 “Окно изменения сущуствующего товара”

В этом окне он может поменять значения сущуствующего товара и сохранить изменения.

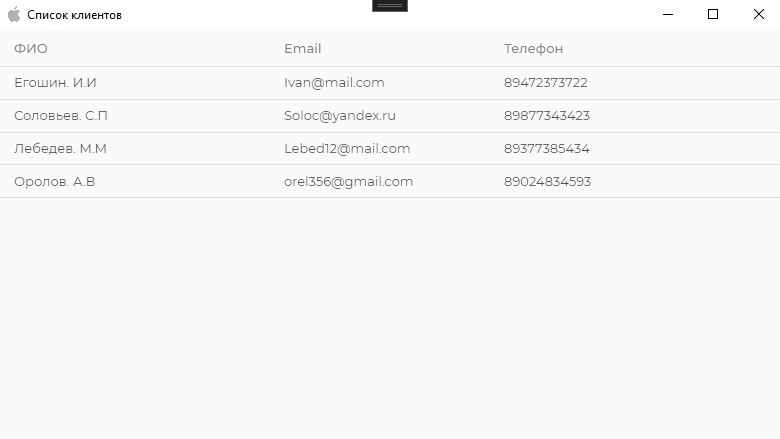


Рисунок 20 “Окно списка клиентов”

В этом окне администратор может посмотреть список всех клиентов.

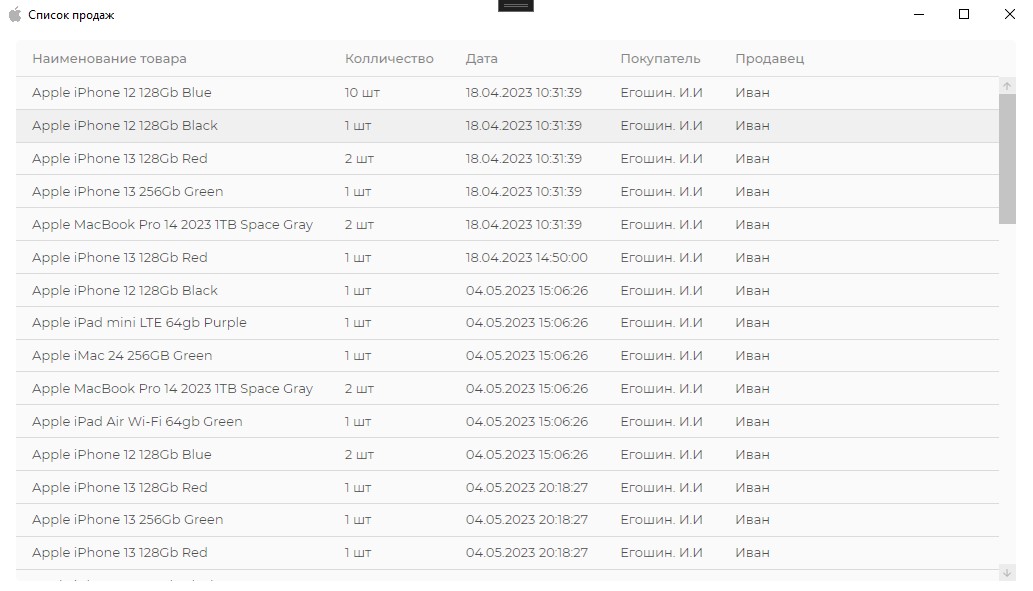


Рисунок 21 “Окно списка продаж”

В этом окне админимтратор может посмотреть список продаж всех товаров.

# **2.3.3 Окно продавца**

Если же мы авторизовались с ролью продавца мы попадем в его окно.

После успешной авторизации появиться сообщение с приветствием продавца

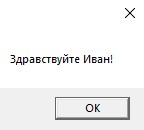


Рисунок 22 “ Сообщение при успешной авторизации продавца”

Далее откроется окно продавца

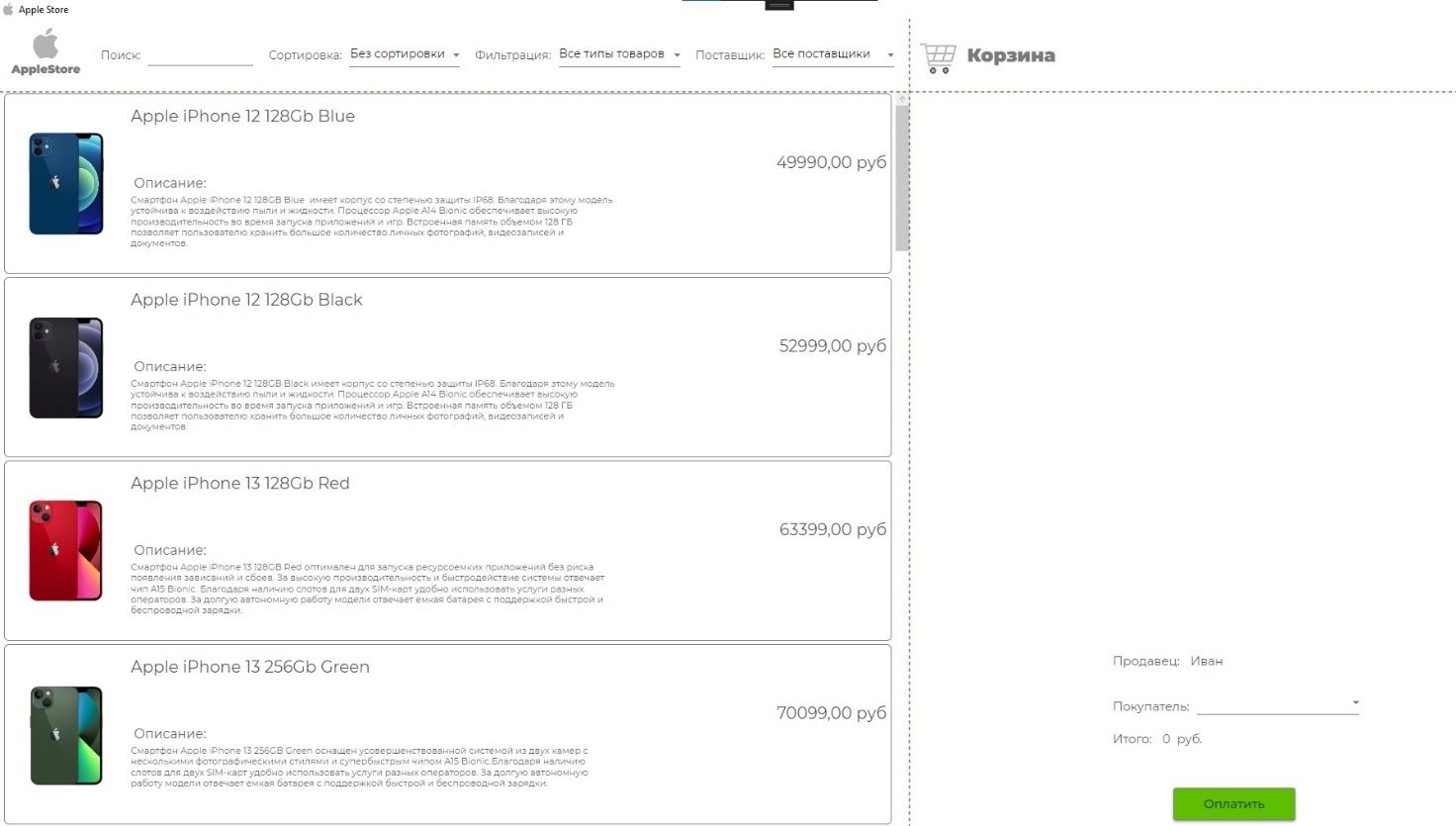


Рисунок 23 “Окно продавца”

В окне продавца выведены товары с его описанием и стоимостью, а так же есть корзина, куда добавляется товар.

В этом окне продавец может:

* выполнить поиск товара
* сортировку товра по названию и по цене
* фильтрацию товара по типу и поставщикам
* добавить товар в корзину
* оплатить

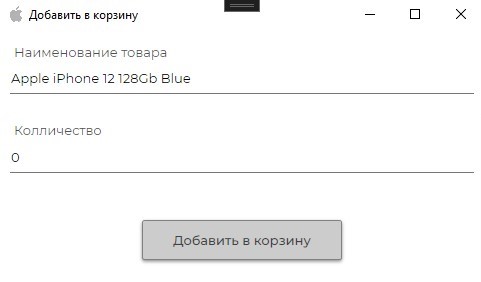


Рисунок 24 “Окно добавления товара в корзину”

Чтобы добавить товар в корзину нужено 2 раза нажать на нужный товар и откроется это окно.

После добавления, этот товар появиться в корзине. Так же он может удалить не нужный товар.

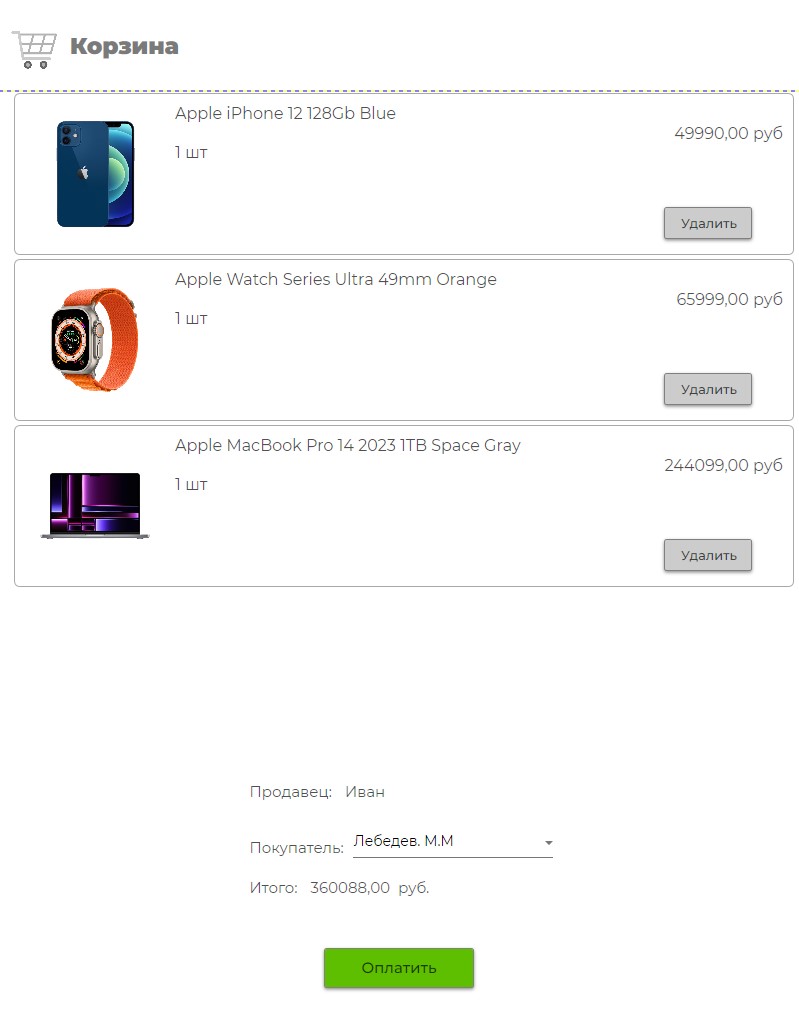


Рисунок 25 “Корзина”

Снизу корзины выводится имя продавца, имя покупателя, итоговая стоимость всех товаров в корзине.

После оплаты выводится сообщение об успешной оплате и корзина отчищается.

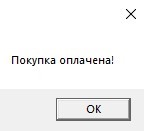


Рисунок 26 “Оплата”

# **Заключение**

В ходе выполнения выпускной квалификационной работы средствами MySQL WORKBENCH 8.0 CE и Microsoft Visual Studio 2022 была разработана информационная система магазина цифровой техники «AppleStore».

При разработке и реализации информационной системы магазина цифровой техники «AppleStore» было проведено исследование предметной области и выявлены основные задачи, требующие решения.

В соответствии с полученной и проанализированной информацией были выбраны средства разработки системы, разработана база данных системы, ее инфологическая, логическая и физическая модели, выбрана СУБД для решения поставленных задач.

Система реализована с помощью языка разработки С# и работает верно во всех предполагаемых ситуациях.

Разработанная ИС отвечает всем техническим требованиям. Таким образом, поставленная нами цель достигнута, задачи выполнены.

В результате выполнения данной работы была получена значительная практическая польза, а также проявленный наукой интерес. Информация, полученная в ходе работы, может быть использована для дальнейших исследований в области управления проектами и разработки программных продуктов.

Наконец, можно сделать вывод, что выполнение данной работы позволило мне расширить свой кругозор в области создания программного обеспечения, использования новых технологий и языков программирования, а также получить ценный практический опыт.

# **Список использованных источников**

* Кудрина, Е. В. Основы алгоритмизации и программирования на языке c# : учеб. пособие для СПО / Е. В. Кудрина, М. В. Огнева. — М. : Издательство Юрайт, 2019. — 322 с.
* Пирогов, В.Ю. Информационные системы и базы данных: организация и проектирование: Учебное пособие / В.Ю. Пирогов. - СПб.: БХВ-Петербург, 2009. - 528 c.
* Реутов, А.П. Автоматизированные информационные системы: методы построения и исследования / А.П. Реутов, М.В. Черняков, С.Н. Замуруев. - М.: Радиотехника, 2010. - 328 c.
* Постников В.М., Основы эксплуатации автоматизированных систем обработки информации и управления. Краткий курс: учеб. пособие - М.: Издательство МГТУ им. Н.Э.Баумана, 2013. - 177 с.
* Вендров А.М. CASE-технологии. Современные методы и средства проектирования информационных систем. — М.: Финансы и статистика, 2004. — 238 с.: ил.
* Кариев Ч.А. Создание Windows-приложений на основе Visual C#. 2-е изд. – М.: НОУ "Интуит", 2016. - 978 с
* Троелсен и Джепикс. Язык программирования C# 7 и платформы .NET и .NET Core

# **Приложение А. Структура БД MySQL Workbench 8**

Скрипт на создание БД:

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

USE `aegoshin` ;

-- -----------------------------------------------------

-- Table `aegoshin`.`Role`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `aegoshin`.`Role` (

`idRole` INT NOT NULL,

`Role` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idRole`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `aegoshin`.`Users`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `aegoshin`.`Users` (

`idUser` INT NOT NULL AUTO\_INCREMENT,

`User` VARCHAR(45) NOT NULL,

`Login` VARCHAR(45) NOT NULL,

`Password` VARCHAR(45) NOT NULL,

`Role\_idRole` INT NOT NULL,

PRIMARY KEY (`idUser`),

INDEX `fk\_Users\_Role\_idx` (`Role\_idRole` ASC) VISIBLE,

CONSTRAINT `fk\_Users\_Role`

FOREIGN KEY (`Role\_idRole`)

REFERENCES `aegoshin`.`Role` (`idRole`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `aegoshin`.`TovarType`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `aegoshin`.`TovarType` (

`idTovarType` INT NOT NULL AUTO\_INCREMENT,

`Title` VARCHAR(100) NOT NULL,

PRIMARY KEY (`idTovarType`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `aegoshin`.`Manufacture`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `aegoshin`.`Manufacture` (

`idManufacture` INT NOT NULL AUTO\_INCREMENT,

`Title` VARCHAR(100) NOT NULL,

PRIMARY KEY (`idManufacture`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `aegoshin`.`Tovars`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `aegoshin`.`Tovars` (

`idTovars` INT NOT NULL AUTO\_INCREMENT,

`Title` VARCHAR(100) NOT NULL,

`Price` DECIMAL(10,2) NOT NULL,

`Quantity` INT NOT NULL,

`Description` VARCHAR(45) NULL,

`Image` VARCHAR(255) NOT NULL,

`TovarType\_idTovarType` INT NOT NULL,

`Manufacture\_idManufacture` INT NOT NULL,

PRIMARY KEY (`idTovars`),

INDEX `fk\_Tovars\_TovarType1\_idx` (`TovarType\_idTovarType` ASC) VISIBLE,

INDEX `fk\_Tovars\_Manufacture1\_idx` (`Manufacture\_idManufacture` ASC) VISIBLE,

CONSTRAINT `fk\_Tovars\_TovarType1`

FOREIGN KEY (`TovarType\_idTovarType`)

REFERENCES `aegoshin`.`TovarType` (`idTovarType`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Tovars\_Manufacture1`

FOREIGN KEY (`Manufacture\_idManufacture`)

REFERENCES `aegoshin`.`Manufacture` (`idManufacture`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `aegoshin`.`Clients`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `aegoshin`.`Clients` (

`idClients` INT NOT NULL AUTO\_INCREMENT,

`FIO` VARCHAR(100) NOT NULL,

`Email` VARCHAR(100) NOT NULL,

`Phone` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idClients`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `aegoshin`.`Chek`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `aegoshin`.`Chek` (

`idChek` INT NOT NULL AUTO\_INCREMENT,

`Tovars\_idTovars` INT NOT NULL,

`Users\_idUser` INT NOT NULL,

`Quantity` INT NOT NULL,

PRIMARY KEY (`idChek`, `Tovars\_idTovars`, `Users\_idUser`),

INDEX `fk\_Chek\_Tovars1\_idx` (`Tovars\_idTovars` ASC) VISIBLE,

INDEX `fk\_Chek\_Users1\_idx` (`Users\_idUser` ASC) VISIBLE,

CONSTRAINT `fk\_Chek\_Tovars1`

FOREIGN KEY (`Tovars\_idTovars`)

REFERENCES `aegoshin`.`Tovars` (`idTovars`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Chek\_Users1`

FOREIGN KEY (`Users\_idUser`)

REFERENCES `aegoshin`.`Users` (`idUser`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `aegoshin`.`TovarSale`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `aegoshin`.`TovarSale` (

`idTovarSale` INT NOT NULL AUTO\_INCREMENT,

`DateSale` DATETIME NOT NULL,

`Quantity` INT NOT NULL,

`Tovars\_idTovars` INT NOT NULL,

`Clients\_idClients` INT NOT NULL,

`Users\_idUser` INT NOT NULL,

PRIMARY KEY (`idTovarSale`, `Tovars\_idTovars`, `Clients\_idClients`, `Users\_idUser`),

INDEX `fk\_TovarSale\_Tovars1\_idx` (`Tovars\_idTovars` ASC) VISIBLE,

INDEX `fk\_TovarSale\_Clients1\_idx` (`Clients\_idClients` ASC) VISIBLE,

INDEX `fk\_TovarSale\_Users1\_idx` (`Users\_idUser` ASC) VISIBLE,

CONSTRAINT `fk\_TovarSale\_Tovars1`

FOREIGN KEY (`Tovars\_idTovars`)

REFERENCES `aegoshin`.`Tovars` (`idTovars`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_TovarSale\_Clients1`

FOREIGN KEY (`Clients\_idClients`)

REFERENCES `aegoshin`.`Clients` (`idClients`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_TovarSale\_Users1`

FOREIGN KEY (`Users\_idUser`)

REFERENCES `aegoshin`.`Users` (`idUser`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

# **Приложение Б. Программное обеспечение**

**Листинг окна авторизации:**

public partial class Auto : Window

{

public int Id { get; }

public string Login { get; }

public Auto()

{

InitializeComponent();

}

private void Entbtn\_Click(object sender, RoutedEventArgs e)

{

try

{

aegoshinContext db = new aegoshinContext();

var uSerBD = db.Users.FirstOrDefault(x => x.User1 == login.Text && x.Password == password.Password);

if (uSerBD == null)

{

MessageBox.Show("Такого пользователя нет !", "Ошибка при авторизации",

MessageBoxButton.OK, MessageBoxImage.Error);

}

else

{

switch (uSerBD.RoleIdRole)

{

case 1:

MessageBox.Show("Здравствуйте , Администратор - " + uSerBD.Login + "!");

MainWindow sda = new MainWindow();

sda.Show();

this.Close();

break;

case 2:

MessageBox.Show("Здравствуйте " + uSerBD.Login + "!");

var id = uSerBD.IdUser;

var name = uSerBD.Login;

SaleWindow sd = new SaleWindow(id, name);

sd.Show();

this.Close();

break;

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void Regbtn\_Click(object sender, RoutedEventArgs e)

{

var NewEditwindow = new Registr(new User());

if ((bool)NewEditwindow.ShowDialog())

{

using (var context = new aegoshinContext()) ;

}

}

}

**Листинг окна регистрации:**

public partial class Registr : Window, INotifyPropertyChanged

{

public User AddUser { get; set; }

public List<Role> RoleList { get; set; }

public Registr(User Registr)

{

InitializeComponent();

DataContext = this;

AddUser = Registr;

using (var context = new aegoshinContext())

{

RoleList = context.Roles.ToList();

role.ItemsSource = RoleList;

role.DisplayMemberPath = "Role1";

}

}

public event PropertyChangedEventHandler? PropertyChanged;

private void savereg\_Click(object sender, RoutedEventArgs e)

{

using (var context = new aegoshinContext())

{

try

{

User user = null;

if (AddUser.IdUser != 0)

user = context.Users.Find(AddUser.IdUser);

else

user = new User();

if (user != null)

{

user.User1 = AddUser.User1;

user.Password = AddUser.Password;

user.Login = AddUser.Login;

user.RoleIdRole = AddUser.RoleIdRoleNavigation.IdRole;

if (user.IdUser == 0)

context.Users.Add(user);

else

context.Users.Update(user);

if (context.SaveChanges() > 0)

{

DialogResult = true;

}

}

}

catch (Exception ex)

{

if (ex.InnerException != null)

MessageBox.Show(ex.InnerException.Message);

else

MessageBox.Show(ex.Message);

}

}

}

}

**Листинг окна администратора:**

public partial class MainWindow : Window, INotifyPropertyChanged

{

public List<TovarType> TovarTypeList { get; set; }

public List<Manufacture> ManufactureList { get; set; }

public MainWindow()

{

InitializeComponent();

DataContext = this;

using (var context = new aegoshinContext())

{

TovarList = context.Tovars

.Include(t => t.TovarTypeIdTovarTypeNavigation)

.Include(m => m.ManufactureIdManufactureNavigation)

.ToList();

TovarTypeList = context.TovarTypes.ToList();

TovarTypeList.Insert(0, new TovarType { Title = "Все типы товаров" });

ManufactureList = context.Manufactures.ToList();

ManufactureList.Insert(0, new Manufacture { Title = "Все поставщики" });

}

}

private IEnumerable<Tovar> \_TovarList;

public IEnumerable<Client> ClientList { get; set; }

public IEnumerable<Tovar> TovarList

{

get

{

var Result = \_TovarList;

if (ManufactureFilterId > 0)

Result = Result.Where(

p => p.ManufactureIdManufacture == ManufactureFilterId);

if (TovarTypeFilterId > 0)

Result = Result.Where(

p => p.TovarTypeIdTovarType == TovarTypeFilterId);

switch (SortType)

{

// сортировка по названию и цене продукции

case 1:

Result = Result.OrderBy(p => p.Title);

break;

case 2:

Result = Result.OrderByDescending(p => p.Title);

break;

case 3:

Result = Result.OrderByDescending(p => p.Price);

break;

case 4:

Result = Result.OrderBy(p => p.Price);

break;

}

if (SearchFilter != "")

Result = Result.Where(

p => p.Title.IndexOf(SearchFilter, StringComparison.OrdinalIgnoreCase) >= 0

);

return Result;

}

set

{

\_TovarList = value;

Invalidate();

}

}

public event PropertyChangedEventHandler? PropertyChanged;

private void Invalidate(string ComponentName = "TovarList")

{

if (PropertyChanged != null)

PropertyChanged(

this,

new PropertyChangedEventArgs(ComponentName));

}

private string SearchFilter = "";

private void SearchFilterTextBox\_KeyUp(object sender, KeyEventArgs e)

{

SearchFilter = SearchFilterTextBox.Text;

Invalidate();

}

public string[] SortList { get; set; } =

{

"Без сортировки",

"название по убыванию",

"название по возрастанию",

"цена по убыванию",

"цена по возрастанию"

};

private int SortType = 0;

private void SortTypeComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

SortType = SortTypeComboBox.SelectedIndex;

Invalidate();

}

private int TovarTypeFilterId = 0;

private void TovarTypeFilter\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

TovarTypeFilterId = (TovarTypeFilter.SelectedItem as TovarType).IdTovarType;

Invalidate();

}

private int ManufactureFilterId = 0;

private void ManufactureFilter\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

ManufactureFilterId = (ManufactureFilter.SelectedItem as Manufacture).IdManufacture;

Invalidate();

}

private void ListView\_MouseDoubleClick(object sender, MouseButtonEventArgs e)

{

var NewEditWindow = new EditTovarWindow(ListView.SelectedItem as Tovar);

if ((bool)NewEditWindow.ShowDialog())

{

// при успешном сохранении продукта перечитываем список продукции

using (var context = new aegoshinContext())

{

TovarList = context.Tovars

.Include(t => t.TovarTypeIdTovarTypeNavigation)

.Include(m => m.ManufactureIdManufactureNavigation)

.ToList();

Invalidate();

}

}

}

private void AddTov\_Click(object sender, RoutedEventArgs e)

{

var NewEditwindow = new EditTovarWindow(new Tovar());

if ((bool)NewEditwindow.ShowDialog())

{

using (var context = new aegoshinContext())

{

TovarList = context.Tovars.ToList();

Invalidate();

}

}

}

private void AddClient\_Click(object sender, RoutedEventArgs e)

{

var NewEditWindow = new EditClient(new Client());

if ((bool)NewEditWindow?.ShowDialog())

{

using (var context = new aegoshinContext())

{

ClientList = context.Clients.ToList();

Invalidate();

}

}

}

private void ClList\_Click(object sender, RoutedEventArgs e)

{

var auto = new ClList();

auto.Show();

}

private void SaleList\_Click(object sender, RoutedEventArgs e)

{

var auto = new TovSale();

auto.Show();

}

private void ex\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

}

**Листинг окна добавления товара:**

public partial class EditTovarWindow : Window, INotifyPropertyChanged

{

public List<Manufacture> ManufactureList { get; set; }

public List<TovarType> TovarTypeList { get; set; }

public Tovar CurrentTovar { get; set; }

public EditTovarWindow(Tovar EditTovar)

{

InitializeComponent();

DataContext = this;

CurrentTovar = EditTovar;

using (var context = new aegoshinContext())

{

ManufactureList = context.Manufactures.ToList();

manuf.ItemsSource = ManufactureList;

TovarTypeList = context.TovarTypes.ToList();

tovtype.ItemsSource = TovarTypeList;

manuf.DisplayMemberPath = "Title";

}

}

public string WindowName

{

get

{

return CurrentTovar.IdTovars == 0 ? "Новый товар" : "Редактирование товара";

}

}

public event PropertyChangedEventHandler? PropertyChanged;

private void Invalidate(string ComponentName = "TovarList")

{

if (PropertyChanged != null)

PropertyChanged(

this,

new PropertyChangedEventArgs(ComponentName));

}

private void PhotoBtn\_Click(object sender, RoutedEventArgs e)

{

OpenFileDialog GetImageDialog = new OpenFileDialog();

GetImageDialog.Filter = "Файлы изображений: (\*.png, \*.jpg)|\*.png;\*.jpg";

GetImageDialog.InitialDirectory = Environment.CurrentDirectory;

if (GetImageDialog.ShowDialog() == true)

{

img.Source = new BitmapImage(new Uri(GetImageDialog.FileName));

CurrentTovar.Image = GetImageDialog.FileName.Substring(Environment.CurrentDirectory.Length);

Invalidate();

}

}

private void SaveBtn\_Click(object sender, RoutedEventArgs e)

{

using (var context = new aegoshinContext())

{

try

{

Tovar tovar = null;

if (CurrentTovar.IdTovars != 0)

tovar = context.Tovars.Find(CurrentTovar.IdTovars);

else

tovar = new Tovar();

if (tovar != null)

{

tovar.Title = CurrentTovar.Title;

tovar.Quantity = CurrentTovar.Quantity;

tovar.Description = CurrentTovar.Description;

tovar.TovarTypeIdTovarType = CurrentTovar.TovarTypeIdTovarTypeNavigation.IdTovarType;

tovar.ManufactureIdManufacture = CurrentTovar.ManufactureIdManufactureNavigation.IdManufacture;

tovar.Price = CurrentTovar.Price;

tovar.Image = CurrentTovar.Image;

if (tovar.IdTovars == 0)

context.Tovars.Add(tovar);

else

context.Tovars.Update(tovar);

if (context.SaveChanges() > 0)

{

DialogResult = true;

}

}

}

catch (Exception ex)

{

if (ex.InnerException != null)

MessageBox.Show(ex.InnerException.Message);

else

MessageBox.Show(ex.Message);

}

}

}

private void DelBtn\_Click(object sender, RoutedEventArgs e)

{

using (var context = new aegoshinContext())

{

try

{

var userCount = context.Users

.Where(ps => ps.IdUser == CurrentTovar.IdTovars).Count();

if (userCount > 0)

throw new Exception("Нельзя удалять продукт с продажами");

var tovar = context.Tovars.Find(CurrentTovar.IdTovars);

context.Tovars.Remove(tovar);

if (context.SaveChanges() > 0)

{

DialogResult = true;

}

}

catch (Exception ex)

{

if (ex.InnerException != null)

MessageBox.Show(ex.InnerException.Message);

else

MessageBox.Show(ex.Message);

}

}

}

private void ExitBtn\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

}

**Листинг окна добавления клиента:**

public partial class EditClient : Window, INotifyPropertyChanged

{

public Client AddClient { get; set; }

public EditClient(Client EditClient)

{

InitializeComponent();

DataContext = this;

AddClient = EditClient;

using (var context = new aegoshinContext()) ;

}

public event PropertyChangedEventHandler? PropertyChanged;

private void Save\_Click(object sender, RoutedEventArgs e)

{

using (var context = new aegoshinContext())

{

try

{

Client client = null;

if (AddClient.IdClients != 0)

client = context.Clients.Find(AddClient.IdClients);

else

client = new Client();

if (client != null)

{

client.Fio = AddClient.Fio;

client.Email = AddClient.Email;

client.Phone = AddClient.Phone;

if (client.IdClients == 0)

context.Clients.Add(client);

else

context.Clients.Update(client);

if (context.SaveChanges() > 0)

{

DialogResult = true;

}

}

}

catch (Exception ex)

{

if (ex.InnerException != null)

MessageBox.Show(ex.InnerException.Message);

else

MessageBox.Show(ex.Message);

}

}

}

private void Exi\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

}

**Листинг окна списка клиентов:**

public partial class ClList : Window, INotifyPropertyChanged

{

public IEnumerable<Client> ClientList { get; set; }

public ClList()

{

InitializeComponent();

DataContext = this;

using (var context = new aegoshinContext())

{

ClientList = context.Clients.ToList();

}

}

public event PropertyChangedEventHandler? PropertyChanged;

}

**Листинг окна списка продаж:**

public partial class TovSale : Window

{

public List<TovarSale> SalesList { get; set; }

public TovSale()

{

InitializeComponent();

DataContext = this;

using (aegoshinContext db = new aegoshinContext())

{

try

{

SalesList = db.TovarSales

.Include(p => p.TovarsIdTovarsNavigation)

.Include(c => c.ClientsIdClientsNavigation)

.Include(c => c.UsersIdUserNavigation)

.ToList();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

}

**Листинг окна продавца:**

public partial class SaleWindow : Window, INotifyPropertyChanged

{

public int Id { get; }

public string Login { get; }

public decimal TotalSum;

private IEnumerable<Tovar> \_TovarList;

public List<TovarType> TovarTypeList { get; set; }

public List<Manufacture> ManufactureList { get; set; }

public IEnumerable<Chek> ChekList { get; set; }

public IEnumerable<Client> ClientList { get; set; }

public IEnumerable<Tovar> TovarList

{

get

{

var Result = \_TovarList;

if (ManufactureFilterId > 0)

Result = Result.Where(

p => p.ManufactureIdManufacture == ManufactureFilterId);

if (TovarTypeFilterId > 0)

Result = Result.Where(

p => p.TovarTypeIdTovarType == TovarTypeFilterId);

switch (SortType)

{

// сортировка по названию и цене продукции

case 1:

Result = Result.OrderBy(p => p.Title);

break;

case 2:

Result = Result.OrderByDescending(p => p.Title);

break;

case 3:

Result = Result.OrderByDescending(p => p.Price);

break;

case 4:

Result = Result.OrderBy(p => p.Price);

break;

}

if (SearchFilter != "")

Result = Result.Where(

p => p.Title.IndexOf(SearchFilter, StringComparison.OrdinalIgnoreCase) >= 0

);

return Result;

}

set

{

\_TovarList = value;

Invalidate();

}

}

public SaleWindow(int id, string name)

{

InitializeComponent();

DataContext = this;

using (var context = new aegoshinContext())

{

ChekList = context.Cheks.Include(tovar => tovar.TovarsIdTovarsNavigation)

.ToList();

}

decimal totalSum = ChekList.Sum(c => c.Quantity \* c.TovarsIdTovarsNavigation.Price);

Itogo.Content = totalSum.ToString();

TotalSum = totalSum;

using (var context = new aegoshinContext())

{

ClientList = context.Clients.ToList();

ChekList = context.Cheks.ToList();

TovarList = context.Tovars.ToList();

TovarTypeList = context.TovarTypes.ToList();

TovarTypeList.Insert(0, new TovarType { Title = "Все типы товаров" });

ManufactureList = context.Manufactures.ToList();

ManufactureList.Insert(0, new Manufacture { Title = "Все поставщики" });

}

Id = id;

Login = name;

Name.Content = Login;

}

public event PropertyChangedEventHandler? PropertyChanged;

private void Invalidate(string ComponentName = "TovarList")

{

if (PropertyChanged != null)

PropertyChanged(

this,

new PropertyChangedEventArgs(ComponentName));

}

private void Invalidate1(string ComponentName = "ChekList")

{

if (PropertyChanged != null)

PropertyChanged(

this,

new PropertyChangedEventArgs(ComponentName));

}

private void Invalidate2(string ComponentName = "TotalSum")

{

if (PropertyChanged != null)

PropertyChanged(

this,

new PropertyChangedEventArgs(ComponentName));

}

private string SearchFilter = "";

private void SearchFilterTextBox\_KeyUp(object sender, KeyEventArgs e)

{

SearchFilter = SearchFilterTextBox.Text;

Invalidate();

}

public string[] SortList { get; set; } =

{

"Без сортировки",

"название по убыванию",

"название по возрастанию",

"цена по убыванию",

"цена по возрастанию"

};

private int SortType = 0;

private void SortTypeComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

SortType = SortTypeComboBox.SelectedIndex;

Invalidate();

}

private int TovarTypeFilterId = 0;

private void TovarTypeFilter\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

TovarTypeFilterId = (TovarTypeFilter.SelectedItem as TovarType).IdTovarType;

Invalidate();

}

private int ManufactureFilterId = 0;

private void ManufactureFilter\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

ManufactureFilterId = (ManufactureFilter.SelectedItem as Manufacture).IdManufacture;

Invalidate();

}

private void listv\_MouseDoubleClick(object sender, MouseButtonEventArgs e)

{

var EditWindow = new Windows.EditSale(listv.SelectedItem as Tovar);

if ((bool)EditWindow.ShowDialog())

{

using (var context = new aegoshinContext())

{

ChekList = context.Cheks.ToList();

Invalidate1();

TovarList = context.Tovars.ToList();

Invalidate();

decimal totalSum = ChekList.Sum(c => c.Quantity \* c.TovarsIdTovarsNavigation.Price);

Itogo.Content = totalSum.ToString();

TotalSum = totalSum;

Invalidate2();

}

}

}

private int ClientFilterId = 0;

private void cln\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

ClientFilterId = (cln.SelectedItem as Client).IdClients;

}

private void btnpay\_Click(object sender, RoutedEventArgs e)

{

using (var context = new aegoshinContext())

{

try

{

TovarSale tovarSale = null;

tovarSale = new TovarSale();

tovarSale.UsersIdUser = Id;

tovarSale.ClientsIdClients = ClientFilterId;

int AddSale = context.Database.ExecuteSqlInterpolated(

$"INSERT INTO `aegoshin`.`TovarSale` (`Tovars\_idTovars`, `DateSale`,`Quantity`,`Clients\_idClients`,`Users\_IdUser`) SELECT `Chek`.`Tovars\_idTovars`, now(), `Chek`.`Quantity`, ({tovarSale.ClientsIdClients}), ({Id}) FROM `aegoshin`.`Chek`; DELETE FROM `aegoshin`.`Chek`;");

context.TovarSales.Add(tovarSale);

MessageBox.Show("Покупка оплачена!");

ChekList = context.Cheks.ToList();

Invalidate1();

TovarList = context.Tovars.ToList();

Invalidate();

ChekList = context.Cheks.Include(tovar => tovar.TovarsIdTovarsNavigation)

.ToList();

decimal totalSum = ChekList.Sum(c => c.Quantity \* c.TovarsIdTovarsNavigation.Price);

Itogo.Content = totalSum.ToString();

TotalSum = totalSum;

Invalidate2();

context.SaveChanges();

}

catch (Exception ex)

{

//MessageBox.Show(ex.Message);

}

}

}

private void DelTov\_Click(object sender, RoutedEventArgs e)

{

var selectedTovar = (sender as Button).Tag as Chek;

// Ищем чек, в котором находится этот товар

using (var context = new aegoshinContext())

{

var chek = context.Cheks.FirstOrDefault(c => c.IdChek == selectedTovar.IdChek);

if (chek != null)

{ //Удаление товара из чека

context.Cheks.Remove(chek);

context.SaveChanges();

ChekList = context.Cheks.ToList();

Invalidate1();

TovarList = context.Tovars.ToList();

Invalidate();

decimal totalSum = ChekList.Sum(c => c.Quantity \* c.TovarsIdTovarsNavigation.Price);

Itogo.Content = totalSum.ToString();

TotalSum = totalSum;

Invalidate2();

}

}

}

}

**Листинг окна добавления товара в корзину:**

public partial class EditSale : Window, INotifyPropertyChanged

{

public User AddUser { get; set; }

public Tovar CurrentTovar { get; set; }

public Chek CurrentSale { get; set; }

public EditSale(Tovar tov)

{

InitializeComponent();

DataContext = this;

CurrentTovar = tov;

CurrentSale = new Chek();

}

public event PropertyChangedEventHandler? PropertyChanged;

private void Invalidate(string ComponentName = "TovarList")

{

if (PropertyChanged != null)

PropertyChanged(

this,

new PropertyChangedEventArgs(ComponentName));

}

private void edbtn\_Click(object sender, RoutedEventArgs e)

{

using (var context = new aegoshinContext())

{

try

{

if (CurrentSale.Quantity < 1)

{ MessageBox.Show("Выберите количество товара"); return; }

Chek chek = null;

chek = new Chek();

if (chek != null)

{

chek.Quantity = CurrentSale.Quantity;

chek.TovarsIdTovars = CurrentTovar.IdTovars;

if (chek.IdChek == 0)

context.Cheks.Add(chek);

else

context.Cheks.Update(chek);

if (context.SaveChanges() > 0)

{

DialogResult = true;

}

}

}

catch (Exception ex)

{

if (ex.InnerException != null)

MessageBox.Show(ex.InnerException.Message);

else

MessageBox.Show(ex.Message);

}

}

}

}