ГБПОУ МО «Ступинский техникум им. А.Т. Туманова»

**Курсовая работа**

по МДК 01.01 «Разработка программных модулей»

Тема: Разработка информационной системы городской аптеки.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Специальность: 09.02.07 «Информационные системы и программирование»

Выполнил(а):

Студент(ка) 3 курса группы ИСП-3

Стремоусов Илья Сергеевич

Проверил(а):

Жильцов Алексей Анатольевич

Ступино, 2021

**Содержание**

Введение3

Аналитическая часть5

Описание предметной области5

Описание входной и выходной информации7

Постановка задачи7

Проектная часть9

Описание средств разработки9

Проектирование и разработка базы данных10

Проектирование концептуальной модели БД10

Проектирование логической модели БД11

Проектирование физической модели БД13

Реализация проекта в среде конкретной СУБД13

Создание клиентской части информационной системы16

Руководство по установке информационной системы18

Руководство пользователя18

**Введение**

В современности использование информационных систем значительно упрощает работу, например с помощью оцифровывания таблиц с информации о товарах, рабочих, заказов и т. п. все может умещаться на одном компьютере, а не храниться в большом помещении на бумаге. Упрощается учёт, обработка поиск и вывод информации. Требуется меньше рабочих, а в некоторых случаях даже один рабочий может работать с информационной системой. Данные «гибкие», то есть эти данные могут подвергаться удалению или обновлению без всякого труда.

Из выше сказанного выходит, что информационная система – связанная совокупность средств, методов и персонала, которая используется для обработки, сохранения и вывода информации с решением конкретной задачи.

Современное понимание информационной системы подразумевает использование компьютера как основного средства обработки информации.

Объектом исследования является городская аптека. Предметом исследования является деятельность работников аптеки по отслеживанию заказов и продажи товаров покупателю.

Актуальность темы исследования обусловлена тем, что необходимо упростить процесс работы сотрудников, работающих в городской аптеки, для этого требуется приложение, которое позволит хранить имеющие данные, а именно:

1. Первичные данные работников такие как, ФИО и логин входа с паролем;
2. Информацию о товарах;
3. Первичную информацию о складах такие как, адрес склада и логин работника заведующего складом;
4. Данные о поставщиках.
5. Данные о заказах.
6. Данные о транзакциях.

Цель данной курсовой работы - разработать информационную систему для городской аптеки, используя современные программы и методы.

Для реализации цели требуется выполнить следующие задачи:

1. Исследование предметной области;
2. Анализ требований к информационной системе;
3. Выбор технологий и средств разработки программного обеспечения;
4. Создание серверной части приложения базы данных;
5. Создание клиентской части приложения;
6. Создание руководства программиста и пользователя;

**1 Аналитическая часть**

**1.1 Описание предметной области и функции решаемых задач**

Аптека – система, нацеленная на хранение, изготовление, фасовку и продажей лекарственных средств и медицинских приборов.

В аптеке могут работать люди, которые могут взаимодействовать с базой данных:

1. Фармацевт.
2. Директор.
3. Заведующий складом.
4. Бухгалтер.
5. Администратор.

Фармацевт – человек с медицинским образованием, знающий многие лекарства и как они поведут себя при возможных случаях. Данному работнику будут доступны таблицы о заказах и товарах.

Директор – человек заведующий аптекой, может принимать на работу, так и увольнять рабочих. Данному работнику будут доступны таблицы о заказах, товарах, транзакциях, складах и поставщиках.

Заведующий складом – человек который занимается менеджментом лекарств и медицинских приборов на складе. Данному работнику будут доступны таблицы о заказах и поставщиках.

Бухгалтер – человек который занимается финансовыми вопросами в аптеке. Данному работнику будут доступны таблицы о транзакциях, чтобы совершать отчет.

Администратор – этим человеком зачастую является сам создатель информационной системы. Данный работник имеет доступ к всей базе данных, в соображениях быстрой отладки и улучшения программы.

За каждым работником будет прикреплен собственный аккаунт, под которым этот работник будет входить в программу. Эта функция создана в соображениях круга доступных таблиц работнику. Для этого будет создана отдельная таблица, которая будет содержать названия таблиц, и ключ “роли”, который будет идентифицировать работника в программе. Это сделано в целях более простого и быстрого обновления данных в программе без входа в исходный код. Таблицы транзакций и заказов будут иметь “гибкие” отчеты. Например, в заказах можно будет вывести отчет по еще не пришедшим товарам.

Чтобы продать товар, необходимо войти в программу под соответствующем к данной таблице доступом аккаунтом, затем перейдя в окно товаров выбрать в таблице товар и нажать кнопку добавить в корзину, если этого товара требуется больше повторить действие добавления в корзину товара, после если вас удовлетворяет ваш набор покупок, выполняется покупка, в противном случае имеется кнопка очистить корзину.

Чтобы заказать товар, необходимо войти в программу под аккаунтом, имеющего доступ к данной таблице, затем перейдя в окно заказов можно выбрать имеющиеся товары и поставщиков, чтобы добавить новые товары или поставщиков на окне будут находиться соответствующие кнопки перенаправляющие на окна добавления.

Чтобы совершить отчеты, необходимо войти в программу под аккаунтом, имеющим доступ к таблицам транзакций и заказов, отчеты по заказам будут находиться в окне заказов товара, транзакции будут сразу иметь функцию отчетов, так как изменять и удалять транзакции запрещено.

**1.2. Описание входной и выходной информации**

Входная информация – совокупность информационных объектов, такие как первичные данные.

Выходная информация – результат обработки входной информации.

Под входной информацией подразумевается:

1. Информация о товаре.
2. Информация о работнике(пользователе).
3. Информация о поставщике.
4. Информация о складе.

Под выходной информацией подразумевается:

1. отчеты о заказах и транзакциях.

**1.3. Постановка задачи**

Требования к разработке базы данных:

* База данных должна содержать только нужные таблицы и данные для вышеописанной предметной области.
* Проведение поэтапного проектирования базы данных

Требования к программному обеспечению:

* Стабильная работа и без ошибок
* Интуитивно понятный интерфейс
* Все объекты системы должны быть логически сгруппированы.

Минимальные системные требования к аппаратному и программному обеспечениям:

* Процессор Intel Core I3-540 или выше;
* Свободная оперативная память не менее 2 Гб;
* Накопитель на жестком диске со свободным объемом памяти не менее 100 МБ;
* 101-клавишная клавиатура;
* Манипулятор «мышь»;
* Наличие одной из Windows:7/8/10;
* Наличие .Net Framework 4.7.2 и более поздней версии.

**2. Проектная часть**

**2.1. Описание средств разработки**

Для решения поставленной задачи была взята среда разработки «Visual Studio 2019». Данная среда разработки программного обеспечения обладает такими преимуществами как:

Удобное управление серверами и их базами данных;

Удобное управление проектами в обозревателе объектов ввиду присутствия всех программных компонентов ПО: от прикладных ресурсов до файлов манифестов и конфигураций, от автоматически задаваемого кода, до программируемых вручную документов с программным кодом;

Наличие удобных инструментов редактирования кода и исправления ошибок;

Наличие удобных инструментов маркировки графического пространства при проектировке файлов графической разметки xaml.

Язык программирования для написания всего кода программы – «C#». Данные средства разработки предоставляют удобные инструменты для реализации программного обеспечения.

База данных «AptekaDB» была реализована в «SQL Server Management Studio 19». Данная СУБД предоставляет удобные инструменты для проектирования и создания будущей базы данных такие как:

Инструменты создания и изменения диаграмм баз данных, с последующим воплощением диаграммы в физическую модель БД.

Инструменты для создания и редактирования таблиц, их ключей, проверочных ограничений и связей.

Инструменты для создания представлений со использованием одной и более таблиц, встроенных и задаваемых процедур и наглядной демонстрации результатов создаваемых запросов.

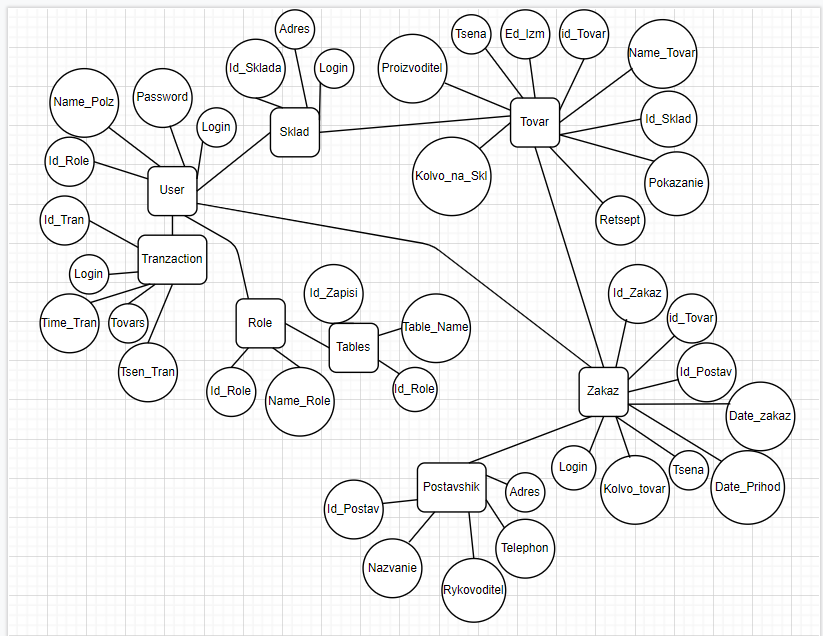
**2.2 Проектирование и разработка базы данных**

Проектирование базы данных — это важный шаг на начальном этапе работы, так как на этом этапе будет строиться вся работа. Для начала нужно понять какие данные должна содержать БД. Сначала создадим модели БД:

1. Концептуальную
2. Логическую
3. Физическую

**2.2.1. Проектирование концептуальной модели БД**

Концептуальной моделью данных называется описание, выполненное с использованием естественного языка, математических формул, таблиц, графиков и других средств, понятных всем людям, работающих над проектированием базы данных. Целью концептуального моделирования является обеспечение наиболее естественных для человека способов сбора и представления той информации, которую предполагается хранить в создаваемой базе данных.



**Рисунок 1 «Концептуальная модель БД»**

**2.2.2. Построение логической модели БД**

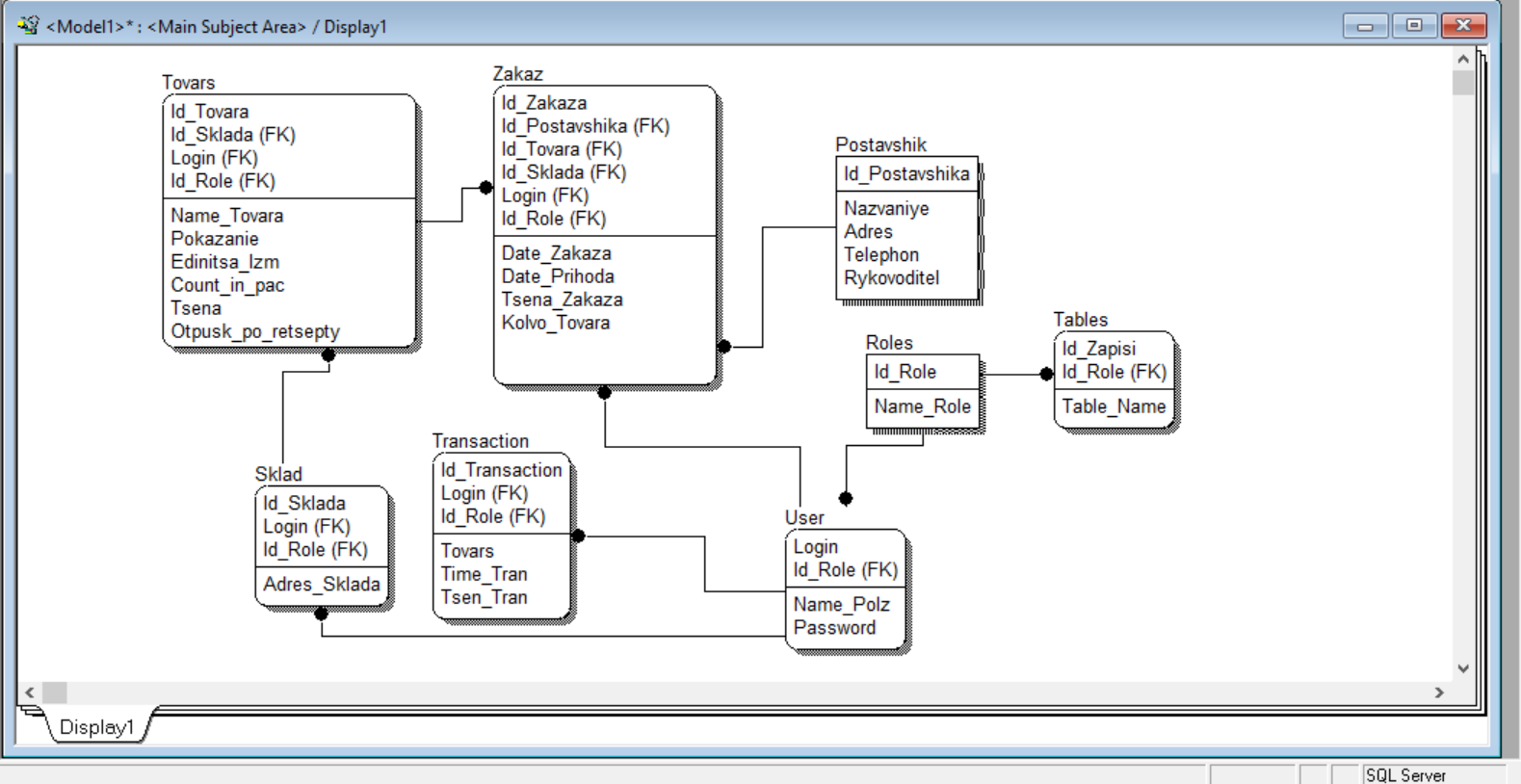
В результате построения концептуальной модели базы данных стало возможным построение логической структуры базы данных (рисунок 1).

Логическая модель данных является визуальным графическим представлением структур данных, их атрибутов и связей. Логическая модель представляет данные таким образом, чтобы они легко воспринимались бизнес-пользователями. Проектирование логической модели должно быть свободно от требований платформы и языка реализации или способа дальнейшего использования данных.

Логическая структура базы данных из 8 таблиц: «Товары», «Склад», «Транзакции», «Роли», «Таблицы», «Пользователи», «Поставщики», «Заказы».

Поля из каждой таблицы:

1. «Товары»: «Номер товара», «Имя товара», «Показания», «Единица измерения», «Количество в упаковке», «Цена товара», «Номер склада», «Отпуск по рецепту», «Количество на складе», «Производитель».
2. «Склад»: «Номер склада», «Адрес склада», «Логин заведующего складом».
3. «Транзакции»: «Номер транзакции», «Товары», «Логин совершившего транзакцию», «Время транзакции», «Цена транзакции».
4. «Роли»: «Идентификатор роли», «Название роли».
5. «Таблицы»: «Номер записи таблицы», «Название таблицы», «Идентификатор роли».
6. «Пользователи»: «Логин пользователя», «Имя пользователя», «Пароль», «Идентификатор роли».
7. «Поставщики»: «Номер поставщика», «Название компании», «Адрес компании», «Телефон», «Руководитель».
8. «Заказы»: «Номер товара», «Номер поставщика», «Дата заказа товара», «Дата прибытия товара», «Цена заказа», «Количество товара», «Номер заказа», «Логин заказчика».



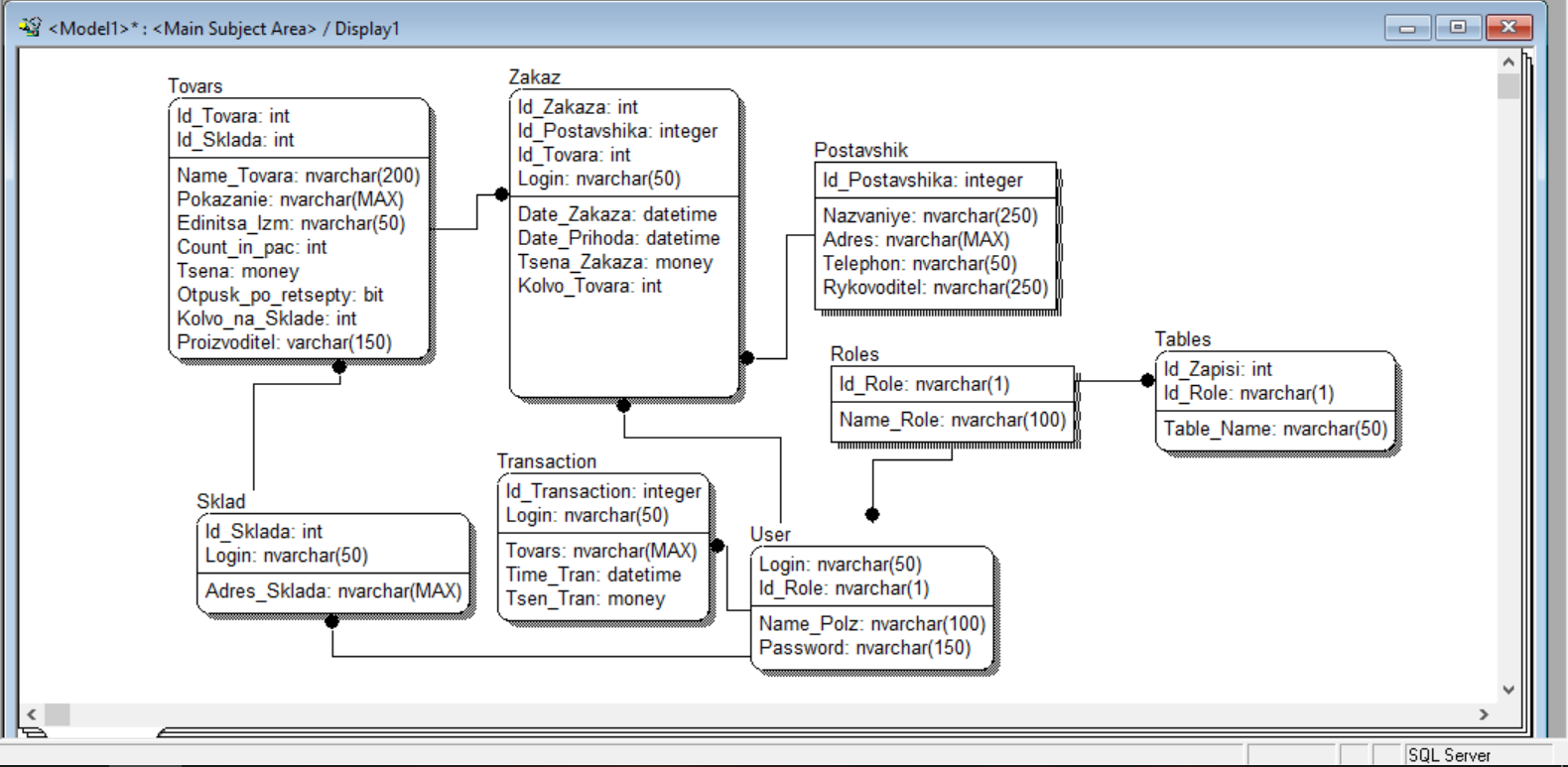
**Рисунок 2 «Логическая модель БД»**

**2.2.3. Построение физической модели БД**

Физическая модель БД определяет способ размещения данных в среде хранения и способы доступа к этим данным, которые поддерживаются на физическом уровне.

**Физическая модель** базы данных содержит все детали, необходимые конкретной СУБД для создания базы: наименования таблиц и столбцов, типы данных, определения первичных и внешних ключей и т. п.

На данном этапе необходимо определиться с типами данных полей и проверочными ограничениями.



**Рисунок 3 «Физическая модель БД»**

**2.2.4. Реализация проекта в среде конкретной СУБД**

Проект был реализован в СУБД Microsoft SQL Server 2019 с использованием утилиты Microsoft SQL Server Management Studio 19, позволяющей конфигурировать, управлять и администрировать все компоненты Microsoft SQL Server.

Так как уже созданы три таблицы модели базы данных, можно переходить к созданию самой базы данных. Первым делом, для создания базы данных в Microsoft SQL Server нужно создать подключение к серверу, на котором мы уже и будем работать.

После подключения переходим к созданию базы данных, это можно сделать через запрос написав Create DataBase, либо через обычное создание путем нажатия кнопки New DataBase. Данная база будет называться AptekaDB.

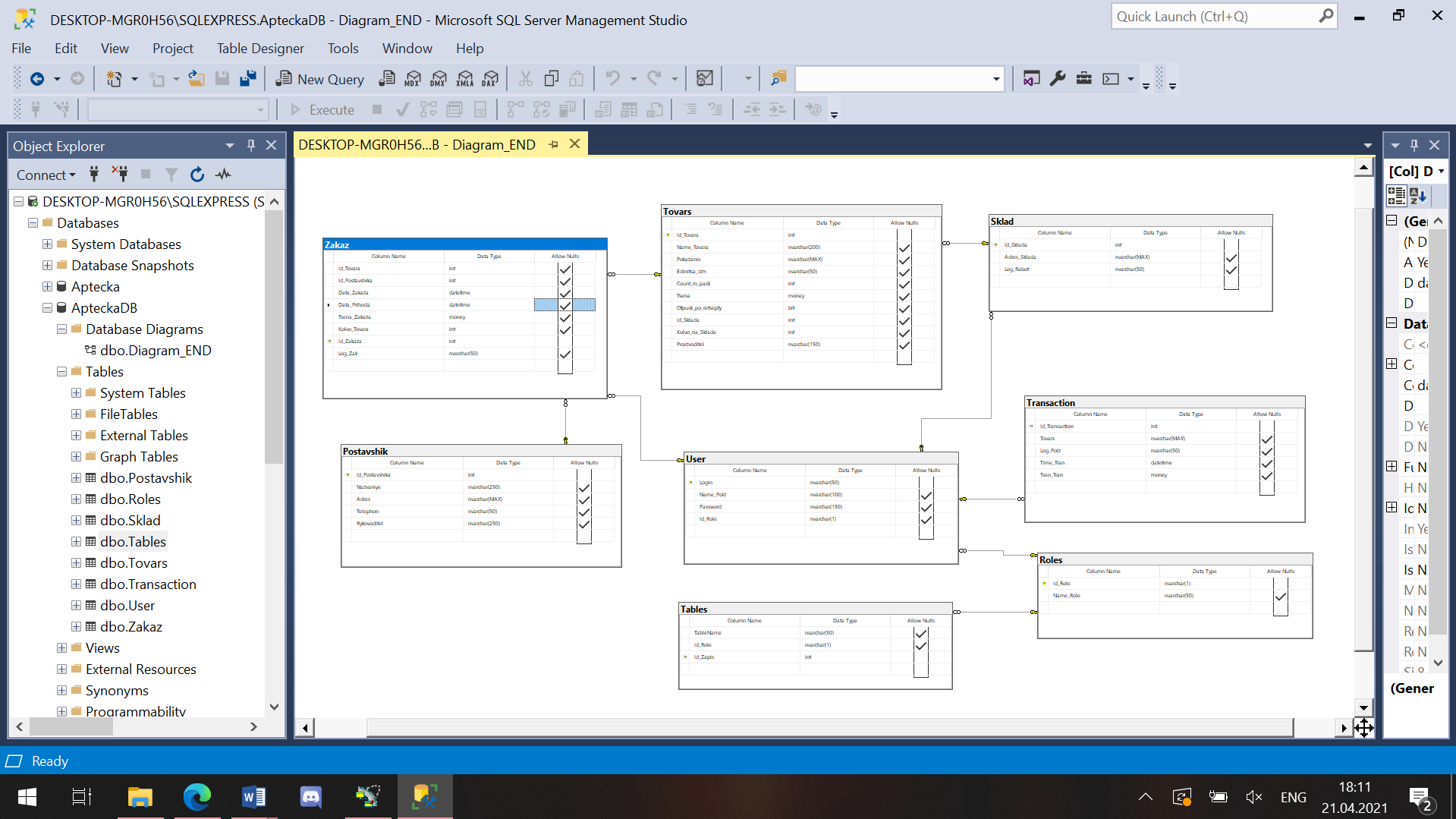
Когда создали базу, то можно переходить к созданию таблиц в ней. Таблицы также можно сделать через запрос написав Create Table, либо через обычное нажатие кнопки New Table. В итоге у нас должно получиться восемь таблиц:

1. Postavshik;
2. Roles;
3. Sklad;
4. Tables;
5. Tovars;
6. Transaction;
7. User;
8. Zakaz.

Также в каждую таблицу обязательно нужно добавить поля, которые нужно заполнить данными.

После создания таблиц нужно перейти к их заполнению. Заполнять таблицы также можно двумя способами через Edit Top 200 Rows или же через запрос при, время создание таблиц.

Так же следует создать диаграмму БД, показанную на рисунке 4.



**Рисунок 4 «Диаграмма БД»**

**2.3. Создание клиентской части информационной системы**

После постройки базы данных в Microsoft SQL Server переходим к следующему этапу, а именно к построение программного приложения в Visual Studio 2019, но при создании приложения нужно помнить, что программа должна быть информативной и простой в использовании.

Первое с чего следует начать, это с того, что нужно создать проект и связать его с Microsoft SQL Server, чтобы можно было пользоваться БД.

После того, как выполнили соединение, переходим к созданию форм и заполнения их инструментами, а после и кодом. Для начала нужно создать двенадцать форм:

1. MainForm
2. AddTovar
3. LogForm
4. PostForm
5. RegForm
6. SldForm
7. TablesForm
8. TovarForm
9. TranForm
10. UserForm
11. ZakForm
12. ZakViewForm

Следующим шагом нужно добавить элементы на эти формы. Для этого приложения потребуются следующие элементы:

1. Label;
2. Button;
3. TextBox;
4. DataGridView;
5. ComboBox
6. RadioButton
7. CheckButton
8. ListBox
9. GroupBox
10. DateTimePicker

Label - предназначен для создания подписей к другим элементам управления или для вывода сообщений прямо на поверхности формы.

Button - позволяет пользователю щелкнуть его для выполнения действия. На элементе управления Button могут отображаться текст и изображения. При щелчке кнопки мышью элемент управления выглядит так, как будто его нажимают и отпускают.

TextBox – используется для ввода и редактирования текста.

DataGridView — это элемент управления, позволяющий отображать данные в табличном формате, в настраиваемой сетке.

ComboBox – позволяет занести в него коллекцию данных и выбрать их.

RadioButton – при нажатии активируется, если при этом присутствую несколько, данных кнопок деактивирует их.

CheckBox – при нажатии активируется, но работает не как RadioButton. То есть при наличии несколько таких же элементов не деактивирует их.

ListBox – позволяет выводить данные в виде списка.

GroupBox – элемент позволяющий добиться чтобы элементы визуально выглядели сгруппировано.

DateTimePicker – элемент позволяющий вывести или выбрать дату уже в отформатированном варианте даты.

После того как все инструменты были добавлены, размещены и подписаны, переходить к написанию кода.

Для написания данного приложения были использованы следующие библиотеки и классы:

Using System;

Using System.Data;

Using System.Linq;

Using System.Windows.Forms

Using Microsoft.Office.Interop.Excel;

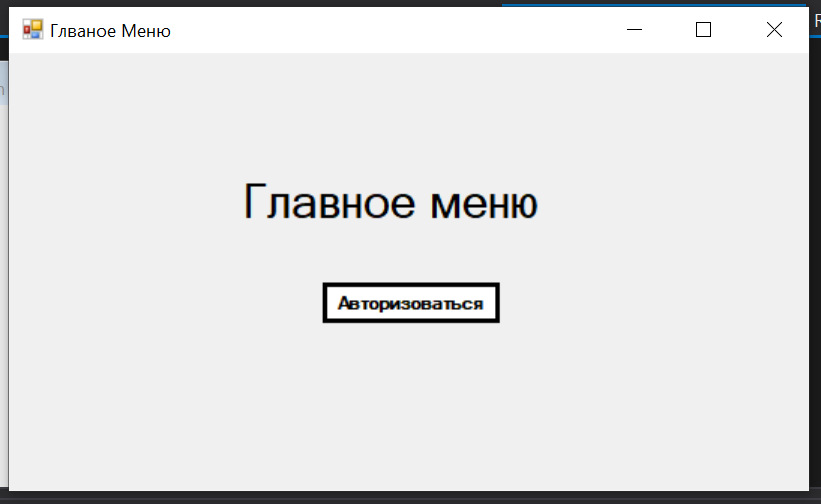
Class GetLogin – созданный класс для получения логина идентификатора и составления отчета для уменьшения общего количества кода.

**2.4. Руководство по установке информационной системы**

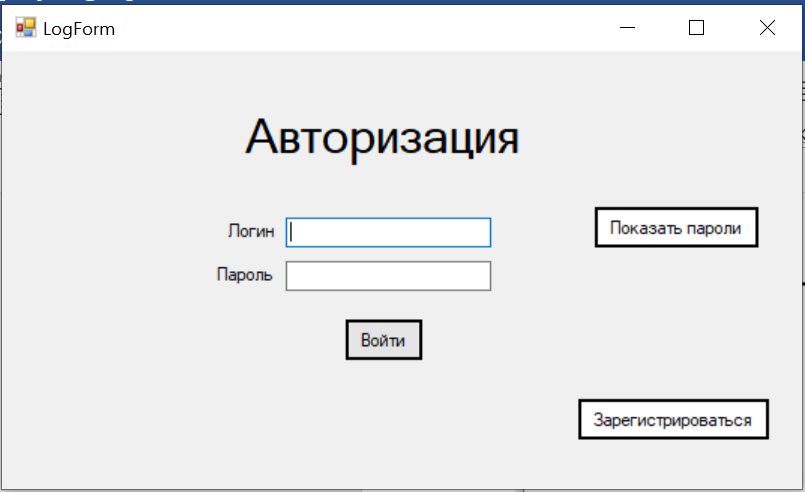
Программное обеспечение будет распространено в стенах городской аптеки, поэтому перенос на CD-ROM диск не обязательна, можно воспользоваться флеш-картой памятью не менее 150 Мб. Будет производиться перенос программного обеспечения и по требованию установка Framework.

**2.4.1 Руководство пользователя**

При запуске программы будет запущено главное меню программы, содержащее кнопку авторизации, перейдя по которой появится форма авторизации, на которой можно авторизоваться или произвести регистрацию.

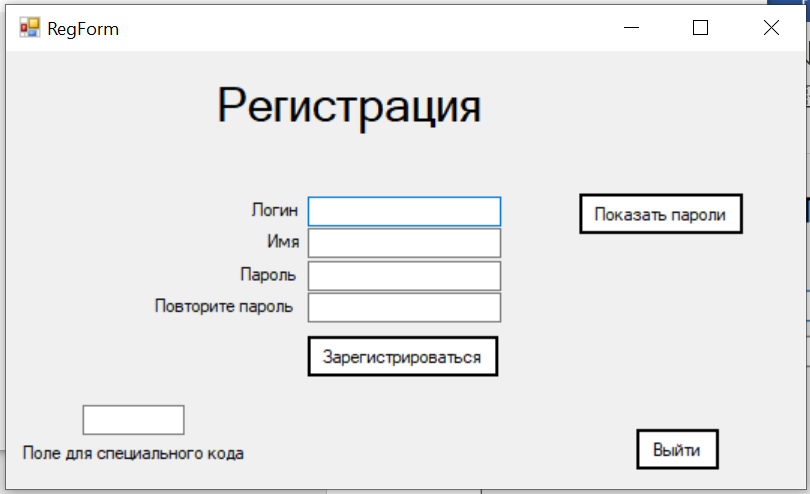


**Рисунок 5 «Главное меню программы»**



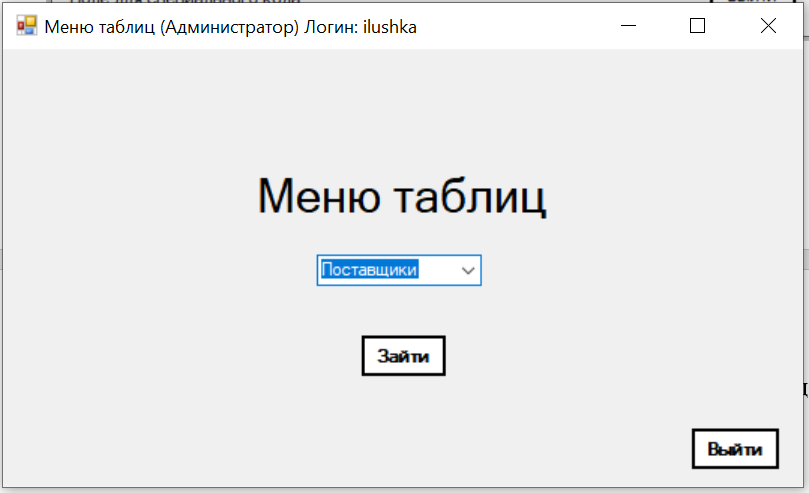
**Рисунок 6 «Меню авторизации»**

Во время регистрации заполняется бланк, содержащий: Логин, имя, пароль и специальный код, выдаваемый администратором для дальнейшей идентификации работника аптеки.

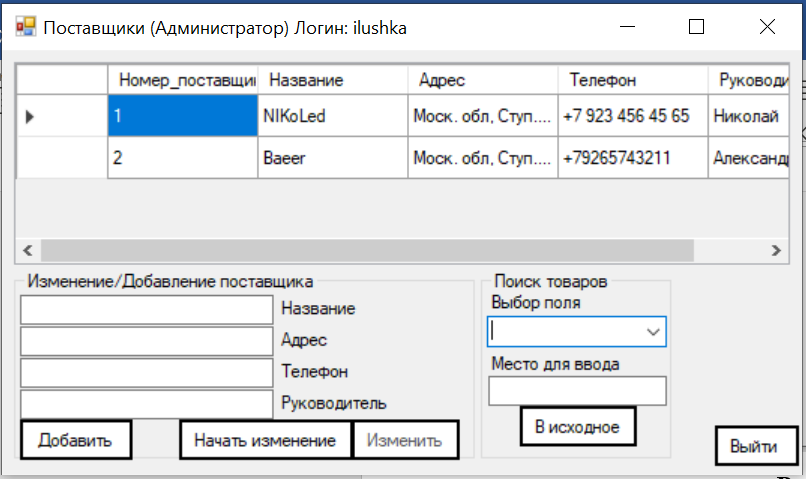


**Рисунок 6 «Форма регистрации»**

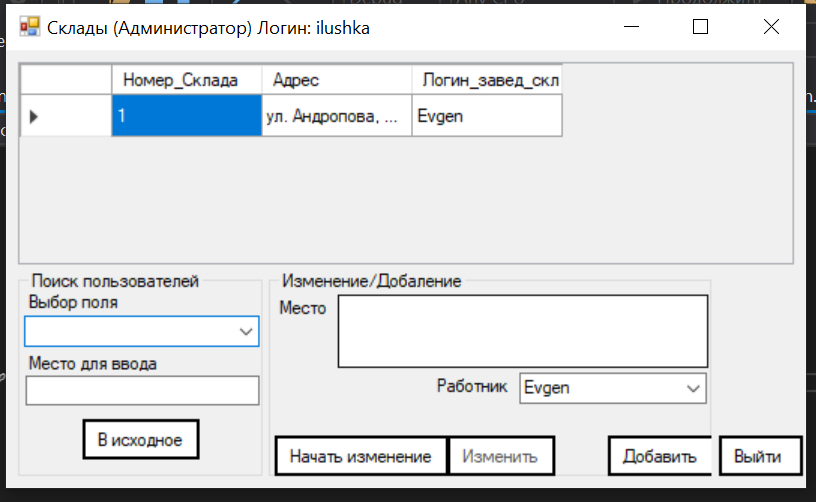
После авторизации появляется окно содержащую список таблиц для дальнейшего их посещения.



**Рисунок 7 «Меню таблиц»**

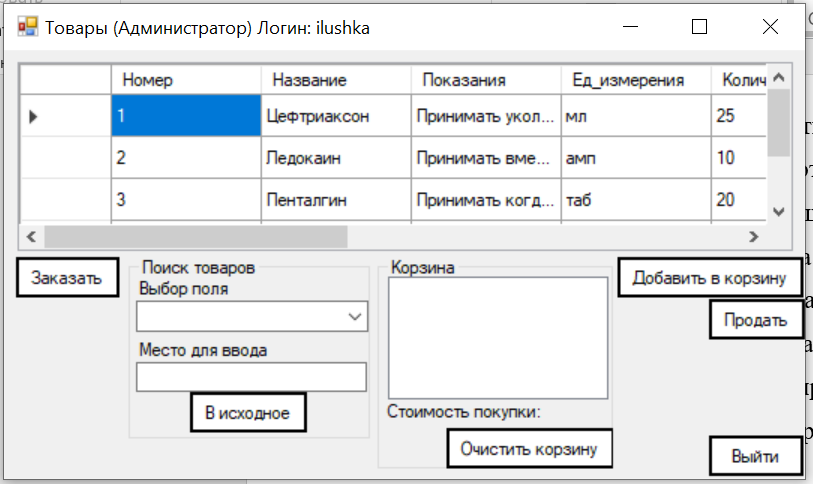


**Рисунок 8 «Окно поставщиков»**

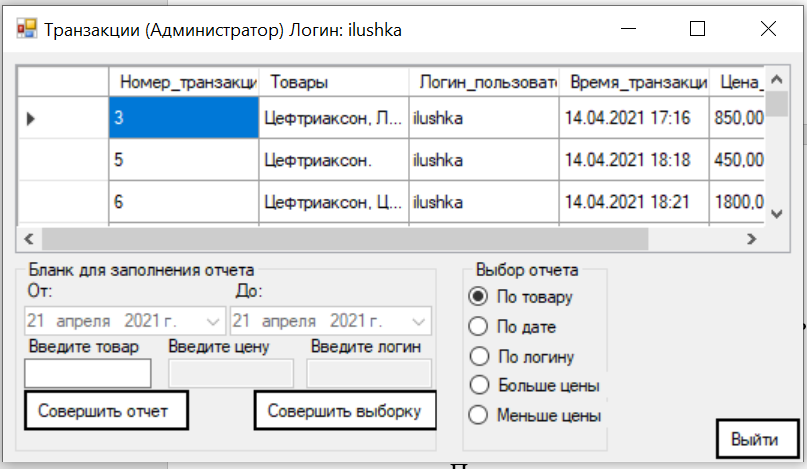


**Рисунок 9 «Окно складов»**

Для того чтобы добавить поставщика нужно заполнить бланк и нажать кнопку «Добавить». Для того чтобы изменить поля поставщика, требуется сначала мышью выбрать в таблице поставщика кликнув по нему, затем нажать кнопку «начать изменение», после нажатия на кнопку произойдет автоматическое заполнение бланка данными выбранного поставщика, после изменения в бланке нужно нажать кнопку «изменить» и произойдет обновление полей таблицы. На окне складов аналогичный функционал. Для того чтобы не было некорректных данных в поле телефон в таблице поставщиков был выключен ввод символов и производится форматирование данных.

**Рисунок 10 «Окно товаров»**

На окне товаров происходит транзакция, для того чтобы совершить продажу товаров требуется выбрать на таблице требуемый товар, затем нажать на кнопку «Добавить в корзину», после выбора требуемого количества товаров требуется нажать на кнопку «Продать», после этого программа начнет проверку наличие товаров в корзине на отпуск только по рецепту врача, в случае нахождения программа выдаст сообщение требующее предъявить рецепт от врача, в противном случае программа отменит транзакцию. Если покупка будет совершена программа запишет в таблицу транзакций цену покупки, отформатированный список проданных товаров, время продажи и логин продавшего товар, затем программа подсчитает проданные товары и сделает обновление поля «Количество\_на\_складе» у товаров на количество купленного товара.



**Рисунок 11 «Окно транзакций»**

На окне транзакций производится только составление отчета, отчет можно составить по данным пунктам:

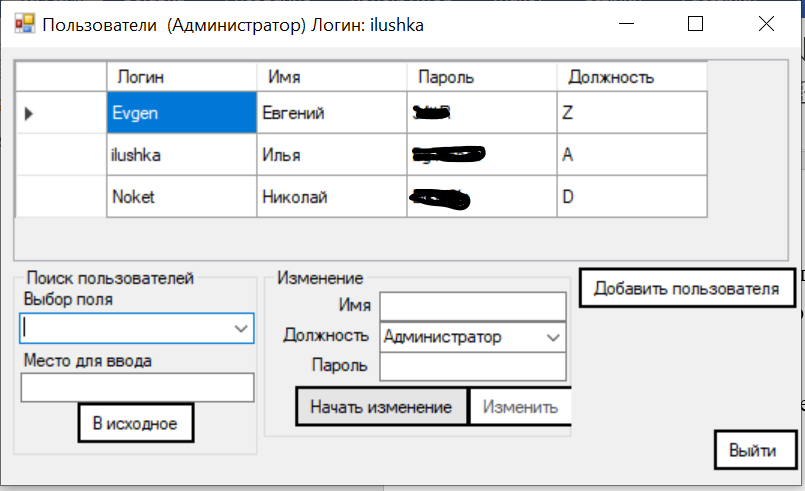
* По товару
* По дате
* По логину
* Меньше цены
* Больше цены

По товару - производится отчет содержащий хотя бы один введенный товар.

По дате – производится отчет транзакций, которые находятся между введенных дат, чтобы совершить выборку по сегодняшнему дню, нужно выбрать на обоих полях сегодняшний день.

По логину – производится отчет транзакций, которых совершил определенный пользователь программы.

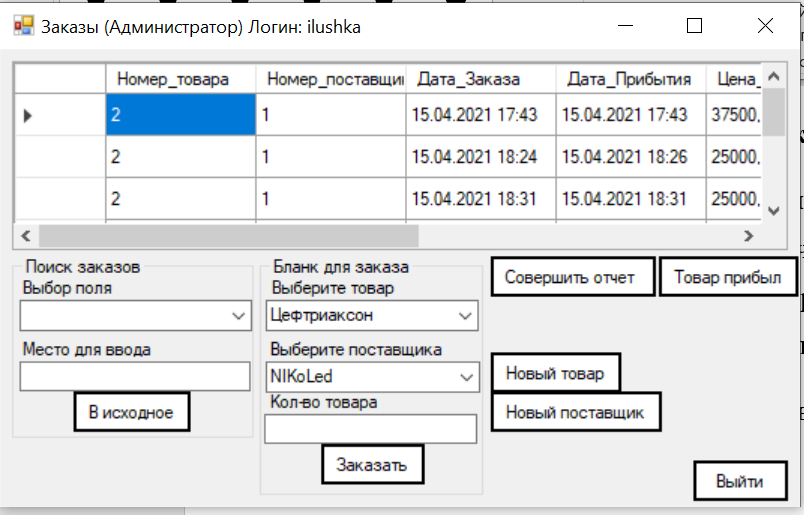
Больше цены и меньше цены – производится отчет транзакций, которые больше или меньше введенной цены. Для этого в поле ввода цены был выключен ввод символов.



**Рисунок 12 «Окно пользователей»**

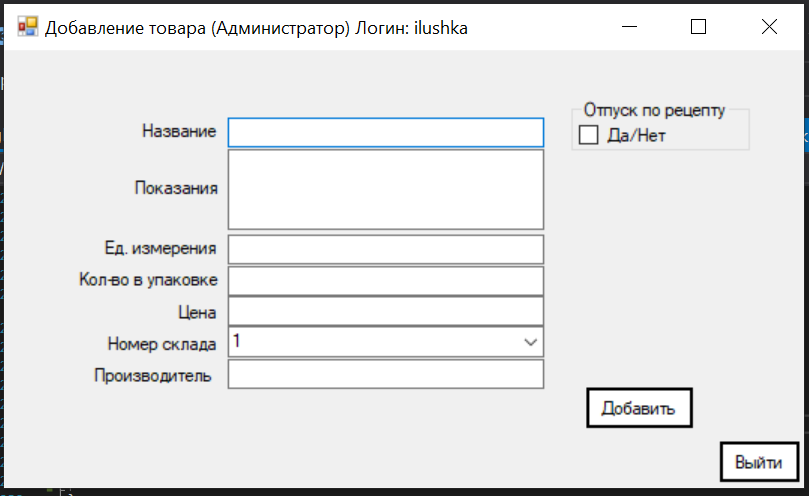
К данной таблице имеет доступ только администратор, но в избежание рассекречивания данных, при ошибочном входе постороннего пользователя не под идентификации администратора пароли будут скрыты, а при нажатии кнопки начать изменение не будет выводиться пароль в поле ввода.

При нажатии кнопки «Добавить пользователя», появится окно регистрации.

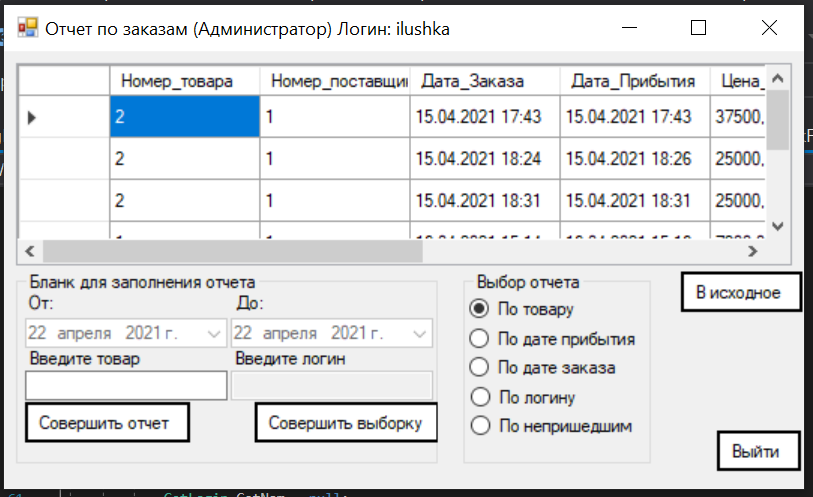


**Рисунок 13 «Окно заказов»**

Для того чтобы заказать новый товар или у нового поставщика требуется нажать соответствующие кнопки «Новый товар» и «Новый поставщик», при нажатии на «Новый товар», появится окно с бланком заполнения нового лекарства, при нажатии на кнопку «Новый поставщик», появится выше показанное окно поставщиков (рисунок 5.5). Для того чтобы заказать уже имеющийся товар в базе, достаточно выбрать его в списке, с поставщиком аналогично и выбрать количество заказываемого товара. Чтобы зафиксировать заказ нужно нажать кнопку «Заказать», после этого создастся новая неполная запись о заказе. Если товар прибыл, то требуется выбрать заказ нажатием на поле соответствующего заказа в таблице, и нажать кнопку «Товар прибыл», если товар уже прибыл программа сообщит об этом. Чтобы совершить отчет по заказам нужно нажать соответствующую кнопку «Совершить отчет», после нажатия появится форма аналогичная окна транзакций (рисунок 5.8).



**Рисунок 14 «Окно добавления товара»**



**Рисунок 15 «Окно заказов»**

Выборка по товару, дате прибытия, дате заказа и логину аналогичная окну транзакций, выборка по не пришедшим осуществляется простым нажатием на ее и последующем выводом в таблицу заказов, которые еще не пришли.

**Заключение**

Курсовой проект был выполнен в Microsoft SQL Server и Visual Studio. В ходе выполнения курсовой работы была разработана база данных и программное приложение с возможностью в будущем доработки программы.

В результате разработки приложения базы данных «AptekaDB» были выполнены следующие поставленные задачи:

* База данных должна отражает всю информацию о предметной области
* Программное обеспечение «AptekaApp» имеет интуитивно понятный построенный интерфейсом
* Программное обеспечение работает стабильно и без ошибок
* Данное ПО сильно упрощает менеджмент данных

Также в результате выполнения курсовой работы были улучшены навыки работы с разработкой и построение баз данных в СУБД SQL и разработкой приложений на С# в среде разработки Visual Studio.

**Список используемой литературы**

1. Крис Фиайли «SQL» - М.: Диалектика, 2018. https://www.labirint.ru/books/656532/.
2. Троелсен, Э., Джепкинс, Ф. Язык программирования С# и платформы .NET / Э. Троелсен, Ф. Джепкинс.- М.:Вильямс, 2018.https://www.labirint.ru/books/660321/.
3. Алабахари, Д. C# 7.0. Справочник. Полное описание языка / Д. Алабахари. – М.: Диалектика, 2018. <https://www.labirint.ru/books/643928/>.
4. Фленов, М. Библия C# / М. Фленов. – М.: BHV, 2019. <https://www.labirint.ru/books/724696/>.
5. Тюкачев, Н., Хлебостроев, В. С# Алгоритмы и структуры данных / Н. Тюкачев., В. Хлебостроев. – М.: Лань, 2018.: <https://www.labirint.ru/books/598757/>.
6. Алабахари, Д. C# Карманный справочник / Д. Алабахари. – М.: Диалектика, 2018. <https://www.labirint.ru/books/612651/>

**Листинг программы**

class GetLogin

{

public static string GetNam { get; set; }

public static void GetViewTran(DataGridView dg1)

{

try

{

var excelApp = new Microsoft.Office.Interop.Excel.Application();

excelApp.Visible = true;

excelApp.Workbooks.Add();

Microsoft.Office.Interop.Excel.Worksheet workSheet = (Microsoft.Office.Interop.Excel.Worksheet)excelApp.ActiveSheet;

workSheet.Rows[1].Font.Bold = true;

workSheet.Rows[4].Font.Bold = true;

workSheet.Cells[1, 2] = "Отчет по транзакциям";

workSheet.Cells[3, 1] = "Номер транзакции";

workSheet.Cells[3, 2] = "Товары";

workSheet.Cells[3, 3] = "Логин пользователя";

workSheet.Cells[3, 4] = "Время транзакции";

workSheet.Cells[3, 5] = "Цена таранзакции";

workSheet.Cells[3, 1].Borders.LineStyle = 1;

workSheet.Cells[3, 2].Borders.LineStyle = 1;

workSheet.Cells[3, 3].Borders.LineStyle = 1;

workSheet.Cells[3, 4].Borders.LineStyle = 1;

workSheet.Cells[3, 5].Borders.LineStyle = 1;

for (int i = 0; i < dg1.RowCount; i++)

{

for (int j = 0; j < dg1.ColumnCount; j++)

{

workSheet.Cells[i + 4, j + 1] = dg1[j, i].Value;

workSheet.Cells[i + 4, j + 1].Borders.LineStyle = 1;

workSheet.Columns[j + 1].AutoFit();

}

}

workSheet.Cells[dg1.RowCount + 5, 1] = "Дата отчета: " + DateTime.Now.ToString("dd.MM.yyyy");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

public static void GetViewZak(DataGridView dg1)

{

try

{

var excelApp = new Microsoft.Office.Interop.Excel.Application();

excelApp.Visible = true;

excelApp.Workbooks.Add();

Microsoft.Office.Interop.Excel.Worksheet workSheet = (Microsoft.Office.Interop.Excel.Worksheet)excelApp.ActiveSheet;

workSheet.Rows[1].Font.Bold = true;

workSheet.Rows[4].Font.Bold = true;

workSheet.Cells[1, 2] = "Отчет по заказам";

workSheet.Cells[3, 1] = "Номер товара";

workSheet.Cells[3, 2] = "Номер поставщика";

workSheet.Cells[3, 3] = "Дата заказа";

workSheet.Cells[3, 4] = "Дата прибытия";

workSheet.Cells[3, 5] = "Цена заказа";

workSheet.Cells[3, 6] = "Количество товара";

workSheet.Cells[3, 7] = "Номер заказа";

workSheet.Cells[3, 8] = "Логин заказчика";

workSheet.Cells[3, 1].Borders.LineStyle = 1;

workSheet.Cells[3, 2].Borders.LineStyle = 1;

workSheet.Cells[3, 3].Borders.LineStyle = 1;

workSheet.Cells[3, 4].Borders.LineStyle = 1;

workSheet.Cells[3, 5].Borders.LineStyle = 1;

workSheet.Cells[3, 6].Borders.LineStyle = 1;

workSheet.Cells[3, 7].Borders.LineStyle = 1;

workSheet.Cells[3, 8].Borders.LineStyle = 1;

for (int i = 0; i < dg1.RowCount; i++)

{

for (int j = 0; j < dg1.ColumnCount; j++)

{

workSheet.Cells[i + 4, j + 1] = dg1[j, i].Value;

workSheet.Cells[i + 4, j + 1].Borders.LineStyle = 1;

workSheet.Columns[j + 1].AutoFit();

}

}

workSheet.Cells[dg1.RowCount + 5, 1] = "Дата отчета: " + DateTime.Now.ToString("dd.MM.yyyy");

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

public static string DaiLogin(string t)

{

return t.Substring(t.LastIndexOf(" "));

}

public static string DaiInfo(string t)

{

return t.Substring(t.IndexOf("("));

}

public static string DaiRole(string t)

{

string[] sep = DaiInfo(t).Split('(',')');

return sep[1];

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp26

{

public partial class PostForm : Form

{

Postavshik postavshik;

ApteckaDBEntities9 db = new ApteckaDBEntities9();

public PostForm()

{

InitializeComponent();

this.FormClosed += PostForm\_FormClosed;

TelBox.MaxLength = 11;

OutPut(dg1,db);

NamBox.Text = GetLogin.GetNam;

}

public static void OutPut(DataGridView dg1, ApteckaDBEntities9 db)

{

dg1.DataSource = db.Postavshik.Select(p => new {

Номер\_поставщика = p.Id\_Postavshika,

Название = p.Nazvaniye,

Адрес = p.Adres,

Телефон = p.Telephon,

Руководитель = p.Rykovoditel

}).ToList();

}

private void PostForm\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

private void ExtBtn\_Click(object sender, EventArgs e)

{

GetLogin.GetNam = null;

this.Hide();

TablesForm tablesForm = new TablesForm();

tablesForm.Text = $"Меню таблиц {GetLogin.DaiInfo(this.Text)}";

tablesForm.Show();

}

private void AddBtn\_Click(object sender, EventArgs e)

{

try

{

string telephon = $"+{TelBox.Text}";

Postavshik postavshik = new Postavshik

{

Nazvaniye = NamBox.Text,

Adres = AdresBox.Text,

Telephon = telephon,

Rykovoditel = RukBox.Text

};

db.Postavshik.Add(postavshik);

db.SaveChanges();

MessageBox.Show("Поставщик добален");

OutPut(dg1, db);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void TelBox\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!(Char.IsDigit(e.KeyChar)))

{

if (e.KeyChar != (char)Keys.Back)

{

e.Handled = true;

}

}

}

private void ChangeEnd\_Click(object sender, EventArgs e)

{

postavshik.Nazvaniye = NamBox.Text;

postavshik.Telephon = TelBox.Text;

postavshik.Adres = AdresBox.Text;

postavshik.Rykovoditel = RukBox.Text;

db.SaveChanges();

MessageBox.Show("Изменение успешно");

ChangeEnd.Enabled = false;

OutPut(dg1,db);

}

private void ChangeStart\_Click(object sender, EventArgs e)

{

ChangeEnd.Enabled = true;

int idpos = (int)dg1.CurrentRow.Cells[0].Value;

postavshik = db.Postavshik.FirstOrDefault(p => p.Id\_Postavshika == idpos);

NamBox.Text = postavshik.Nazvaniye;

AdresBox.Text = postavshik.Adres;

TelBox.Text = postavshik.Telephon;

RukBox.Text = postavshik.Rykovoditel;

}

private void PostForm\_Load(object sender, EventArgs e)

{

for (int i = 0; i < dg1.ColumnCount; i++)

{

Searchbox.Items.Add(dg1.Columns[i].HeaderText);

}

}

private void SeachTb\_TextChanged(object sender, EventArgs e)

{

try

{

switch (Searchbox.SelectedIndex)

{

case 0:

dg1.DataSource = db.Postavshik.ToArray().Where(p => p.Id\_Postavshika == int.Parse(SeachTb.Text)).Select(p => new {

Номер\_поставщика = p.Id\_Postavshika,

Название = p.Nazvaniye,

Адрес = p.Adres,

Телефон = p.Telephon,

Руководитель = p.Rykovoditel

}).ToList();

break;

case 1:

dg1.DataSource = db.Postavshik.ToArray().Where(p => p.Nazvaniye.Contains(SeachTb.Text)).Select(p => new {

Номер\_поставщика = p.Id\_Postavshika,

Название = p.Nazvaniye,

Адрес = p.Adres,

Телефон = p.Telephon,

Руководитель = p.Rykovoditel

}).ToList();

break;

case 2:

dg1.DataSource = db.Postavshik.ToArray().Where(p => p.Adres.Contains(SeachTb.Text)).Select(p => new {

Номер\_поставщика = p.Id\_Postavshika,

Название = p.Nazvaniye,

Адрес = p.Adres,

Телефон = p.Telephon,

Руководитель = p.Rykovoditel

}).ToList();

break;

case 3:

dg1.DataSource = db.Postavshik.ToArray().Where(p => p.Telephon.Contains(SeachTb.Text)).Select(p => new {

Номер\_поставщика = p.Id\_Postavshika,

Название = p.Nazvaniye,

Адрес = p.Adres,

Телефон = p.Telephon,

Руководитель = p.Rykovoditel

}).ToList();

break;

case 4:

dg1.DataSource = db.Postavshik.ToArray().Where(p => p.Rykovoditel.Contains(SeachTb.Text)).Select(p => new {

Номер\_поставщика = p.Id\_Postavshika,

Название = p.Nazvaniye,

Адрес = p.Adres,

Телефон = p.Telephon,

Руководитель = p.Rykovoditel

}).ToList();

break;

default:

MessageBox.Show("Невозможен поиск");

break;

}

}

catch

{

MessageBox.Show("Не найдено");

}

}

private void Back2St\_Click(object sender, EventArgs e)

{

OutPut(dg1, db);

}

}

}