**Оглавление**

[**Введение** 2](#_Toc501282033)

[**Глава** **1. Разработка модели базы данных** 3](#_Toc501282034)

[**Глава 2. Разработка необходимых объектов** 5](#_Toc501282035)

[**2.1. Таблицы** 5](#_Toc501282036)

[**2.2. Процедуры** 5](#_Toc501282037)

[**2.3. Триггеры** 5](#_Toc501282038)

[**Глава 3. Описание процедуры импорта данных** 6](#_Toc501282039)

[**Глава 4. Технология SSIS (SQL Server Integration Services)** 7](#_Toc501282040)

[**Глава 5.** **Резервное копирование и восстановление** 10](#_Toc501282041)

[**Глава 6. Тестирование** 12](#_Toc501282042)

[**6.1. Тестирование производительности базы данных** 12](#_Toc501282043)

[**6.2. Тестирование приложения.** 13](#_Toc501282044)

[**Глава 7. Руководство пользователя** 16](#_Toc501282045)

[**Заключение** 22](#_Toc501282046)

[**Список использованной литературы** 23](#_Toc501282047)

[**Приложение А** 24](#_Toc501282048)

[**Приложение Б** 29](#_Toc501282049)

[**Приложение В** 34](#_Toc501282050)

[**Приложение Г** 35](#_Toc501282051)

[**Приложение Д** 36](#_Toc501282052)

[**Приложение E** 42](#_Toc501282053)

# **Введение**

Система управления базами данных (СУБД) – совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных. В нашем проекте мы будем использовать реляционную модель. Реляционная модель представляет собой совокупность данных, состоящую из набора двумерных таблиц. В теории множеств таблице соответствует термин отношение(relation), физическим преставлением которого является таблица, отсюда и название модели – реляционная. Реляционная модель является удобной и наиболее привычной формой представления данных. Любая таблица в реляционной базе состоит из строк, которые называют записями, и столбцов, которые называют полями.

Основные функции СУБД:

* определение структуры создаваемой базы данных, ее инициализация и проведение начальной загрузки;
* предоставление пользователям возможности манипулирования данными (выборка необходимых данных, выполнение вычислений, разработка интерфейса ввода/вывода, визуализация);
* обеспечение логической и физической независимости данных;
* защита логической целостности базы данных;
* защита физической целостности;
* управление полномочиями пользователей на доступ к базе данных;
* синхронизация работы нескольких пользователей;
* управление ресурсами среды хранения;
* поддержка деятельности системного персонала.

Обычно современная СУБД содержит следующие компоненты:

* ядро, которое отвечает за управление данными во внешней и оперативной памяти и журнализацию;
* процессор языка базы данных, обеспечивающий оптимизацию запросов на извлечение и изменение данных и создание, как правило, машинно-независимого исполняемого внутреннего кода;
* подсистему поддержки времени исполнения, которая интерпретирует программы манипуляции данными, создающие пользовательский интерфейс с СУБД.

СУБД существует огромное множество: PostgreSql, Oracle, MS SQL Server, Microsoft Access, MySQL, MongoDB, Firebird и так далее. В данной работе будет использовано решение от компании MS SQL Server.

Так же для разработки приложения использовалась технология WPF и язык программирования C#.

**Глава** **1. Разработка модели базы данных**

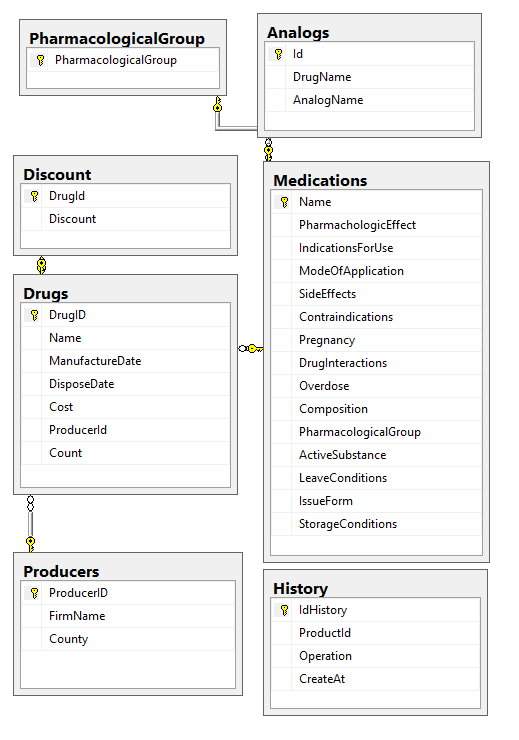


Рис. 1.1 – Диаграмма базы данных

Для базы данных аптека были разработаны 7 таблиц. Диаграмма связей таблиц для необходимой базы данных представлена н рисунке 1.1.

Таблица PharmacologicalGroup, содержащая фармакологическую группу препаратов:

* PharmacologicalGroup – уникальный идентификатор, содержит имя фармакологической группы;

Таблица Medications, предназначенная для хранения аннотации к препарату:

* Name – уникальный идентификатор, содержит название препарата;
* PharmachologicEffect – фармакологический эффект;
* IndicationsForUse – показания к применению;
* ModeOfApplication – способ применения;
* SideEffect – побочные эффекты;
* Contraindications – противопоказания;
* Pregnancy – применение во время беременности;
* DrugInteractions – лекарственное взаимодействие;
* OverDose – побочные действия от передозировки;
* Composition – состав;
* PharmacologicalGroup – имя фармакологической группы;
* ActiveSubstance – активное вещество;
* LeaveConditions – условия отпуска;
* IssueForm – форма выпуска препарата;
* StorageConditions – условия хранения.

Таблица Producers, предназначенная для хранения производителей препаратов:

* ProducerID – уникальный идентификатор производителя;
* FirmName – название компании;
* Country – страна производства.

Таблица Drugs, предназначенная для хранения препаратов:

* DrugID – уникальный идентификатор препарата;
* Name – название препарата;
* ManufactureDate – дата производства;
* DisposeDate – дата утилизации;
* Cost – цена;
* ProducerId – id производителя;
* Count – количество.

Таблица Analogs предназначенная для хранения аналогов:

* Id – уникальный идентификатор;
* DrugName– имя препарата;
* AnalogName – имя аналога.

Таблица Discount, предназначенная для хранения скидки к препарату:

* DrugId – уникальный идентификатор препарата;
* Discount – размер скидки.

Таблица History, предназначенная для просмотра изменений препарата:

* IdHistory – уникальный идентификатор;
* ProductId – идентификатор препарата;
* Operation – операция;
* CreateAt – время операции.

Скрипты для создания всех таблиц базы данных представлены в Приложении А.

# **Глава 2. Разработка необходимых объектов**

## **2.1. Таблицы**

Таблицы являются основой любой базы данных, именно в них хранится вся информация. При проектировании базы данных было создано 7 таблиц, которые подробно описаны ранее в разделе 1, а SQL-скрипты для их создания находятся в Приложении А.

## **2.2. Процедуры**

Хранимая процедура – это объект базы данных, предоставляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере. Использование хранимых процедур необходимо для доступа к данным и произведение манипуляции с ними. Листинги хранимых процедур представлены в приложении Б.

Были разработаны процедуры:

* AddAnalogs – для добавления аналогов;
* AddDrug – для добавления препаратов;
* AddMedications – для добавления аннотаций;
* AddProducers – для добавления производителей;
* DeleteAllDrugs – для удаления всех препаратов;
* DeleteAllMedic – для удаления всех аннотаций;
* DeleteDrugs – для удаления препарата по Id;
* DeleteProcuder – для удаления производителя по Id;
* insertProducerFromXML – для импорта производителей из XML;
* UpdateDrugs – для обновления препарата;
* SelectName – для поиска препарата по имени;
* SelectProducers – для поиска препарата по производителю

Помимо них было разработаны и CRUD-процедуры.

## **2.3. Триггеры**

Триггер – хранимая процедура особого типа, которую пользователь не вызывает непосредственно, а использование которой обусловлено действием по модификации данных. Листинги предоставлены в приложении В.

# **Глава 3.** **Описание процедуры импорта данных**

**Процедура импорта данных из XML-файла**

XML **–** расширяемый язык разметки. Рекомендован Консорциумом всемирной паутины(W3C). Спецификация XML описывает XML-документы и частично описывает поведение XML-процессоров.

Для импорта используется стандартная функция, входящая в набор OLE DB – OPENROWSET, в которую передаются параметры о типе импортируемых данных и пути файла, где они находятся. Также для генерации XML файла используется программа которая написана на языке программирования C#. Примеры скрипта и исходного кода программы представлены в Приложении Г.

# **Глава 4. Технология SSIS (SQL Server Integration Services)**

SSIS – это инструмент, который позволяет в удобном виде реализовать интеграцию, т.е. реализовать процесс переноса данных из одного источника в другой. Этот процесс иногда называют ETL (от англ. Extract, Transform, Load – дословно «извлечение, преобразование, загрузка»).

На рисунках 4.1- 4.5 показано пошаговое выполнение.

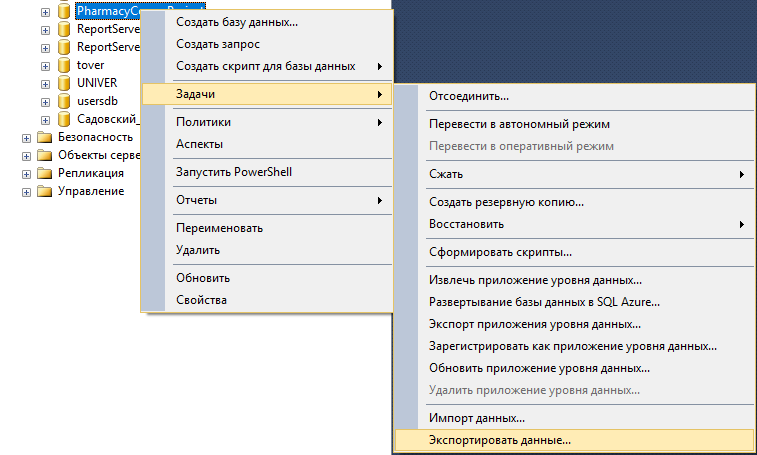


Рис.4.1 – Экспорт данных

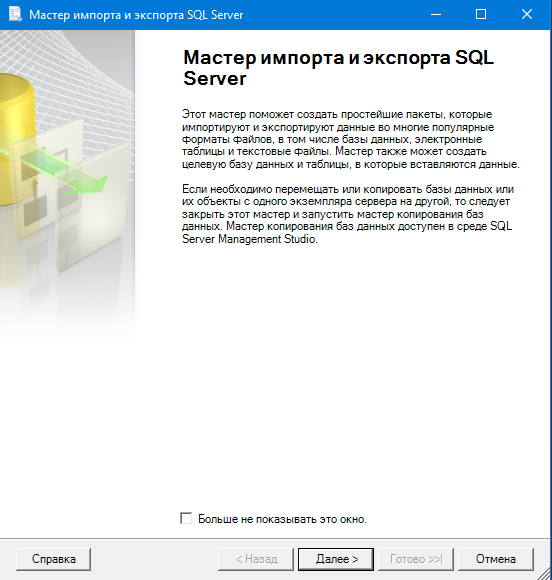


Рис. 4.2 – Мастер экспорта SQL Server

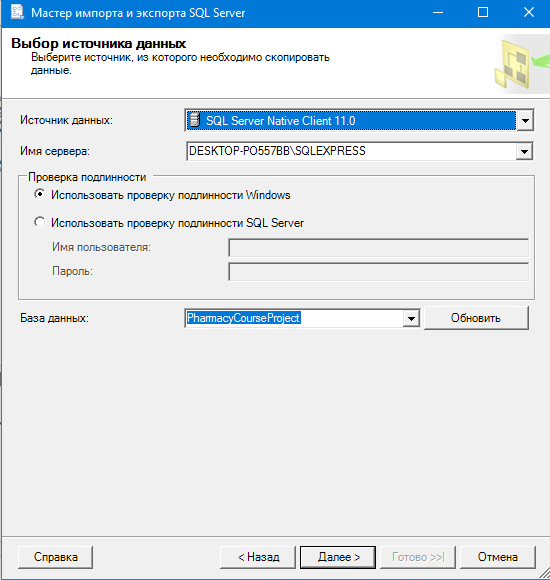


Рис. 4.3 – Выбор источника данных

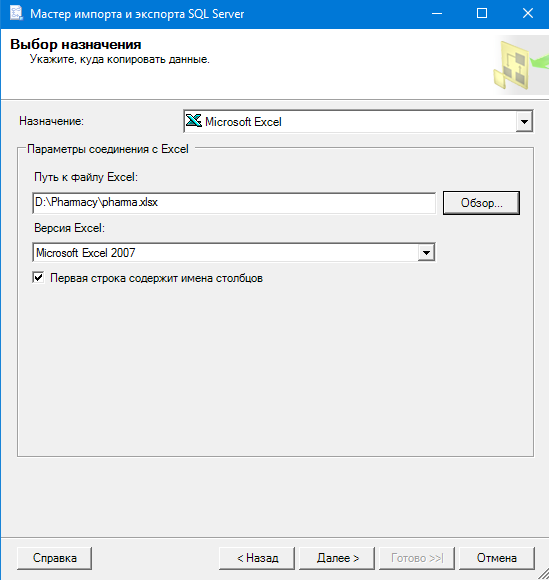


Рис. 4.4 – Выбор источника экспорта

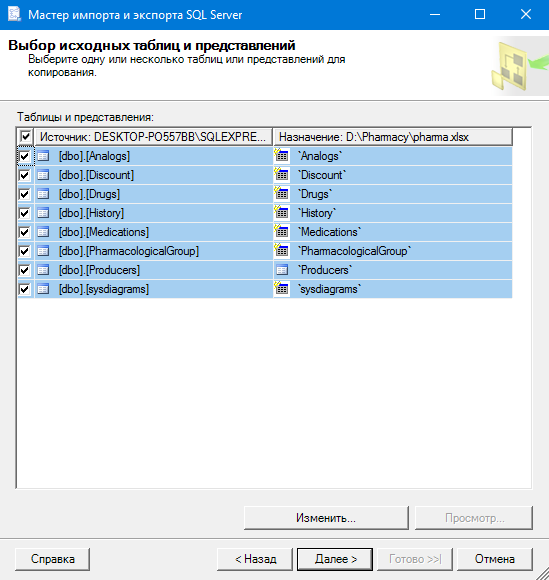


Рис. 4.5 – Выбор информации для экспорта

# **Глава 5.** **Резервное копирование и восстановление**

Резервное копирование базы данных будет осуществляться в папку BackUpDB на диске D:\ каждый день и имя файла будет состоять из даты создания бэкапа.

Листинг исходного кода резервного копирования находится в Приложении E.

Далее создаем файл расширения bat, который и будет запускать скрипт резервного копирования. Листинг исходного кода исполняемого файла находится в Приложении E.

Создание задачи будем осуществлять с помощью планировщика заданий (taskschd.msc). На рисунках 5.1 и 5.2 демонстрируется создание задачи.

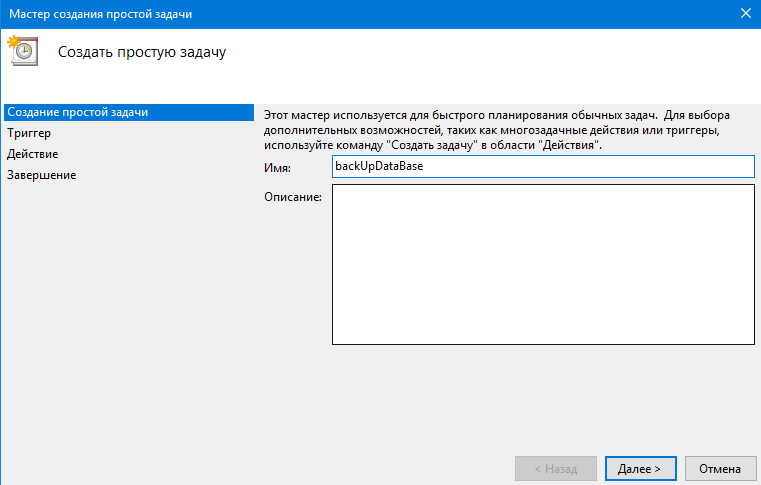


Рис. 5.1 – Создание действия

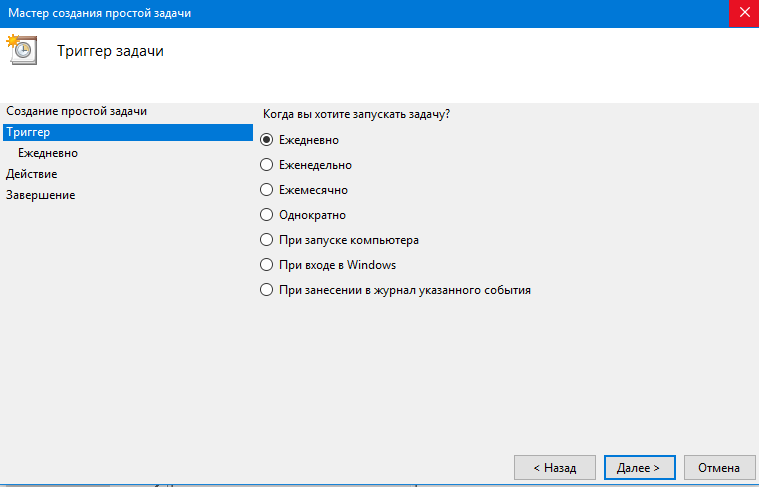


Рис. 5.2 – Создание расписания

Восстановление базы данных можно осуществить с помощью выполнения задачи восстановления.

# **Глава 6. Тестирование**

## **6.1. Тестирование производительности базы данных**

Для тестирования производительности была взята за основу таблица Producers, так как она заполнена гораздо больше всех остальных таблиц.

Изначально таблица была заполнена на 100000 строк. После этого был применён SELECT-запрос к данной таблице и при помощи стандартных средств IDE MS SQL Server Management Studio оценена цена выборки к таблице. Результат данной оценки запроса приведён на рисунке 6.1.

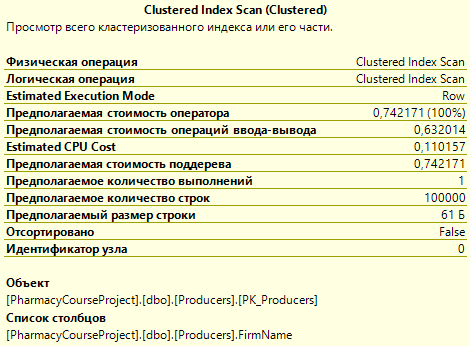


Рис. 6.1 – Оценка запроса к таблице Producers без некластеризованного индекса

После проведения первоначальной оценки был построен некластеризованный индекс к таблице Producers по столбцу FirmName и проведена оценка такого же SELECT-запроса к таблице Producers. Результаты, полученные во время оценки, представлены на рисунке 6.2.

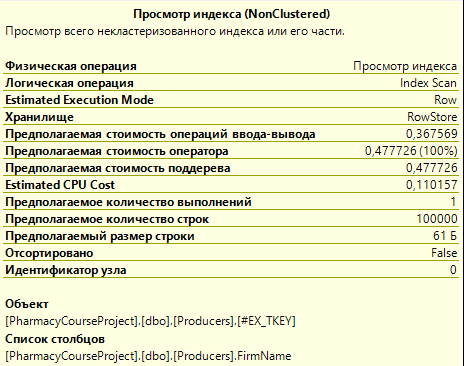


Рисунок 6.2 – Оценка запроса к таблице с построенным некластеризованным индексом

По результатам проведённых оценок до и после построения некластеризованного индекса, можно сделать вывод, что после создания индекса мы ускорили многие параметры.

Таким образом, постройка индекса к таблице была более чем оправдана, так как мы получили прирост производительности, в зависимости от параметра, характеризующего запрос.

## **6.2. Тестирование приложения.**

При разработке приложения были учтены возможные исключительные ситуации.

Предусмотрена следующая обработка исключительных ситуаций:

* В случае отсутствия выбранных аналогов выводится соответствующее сообщение. Рисунок 6.2.1 демонстрирует такую ситуацию.

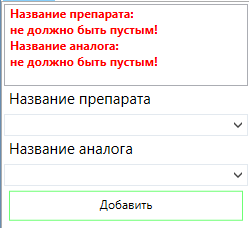


Рис. 6.2.1 – Окно при отсутствии информации о аналоге.

* В случае несоответствия фармакологических групп выводится соответствующее сообщение. Рисунок 6.2.2 демонстрирует такую ситуацию.

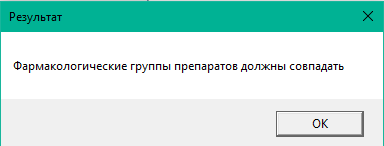


Рис. 6.2.2 – Окно при несоответствии фармакологических групп аналогов

* В случае ввода некорректной даты выводится соответствующее сообщение. Рисунок 6.2.3 демонстрирует такую ситуацию.

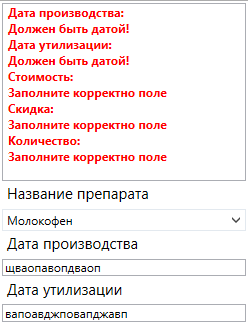


Рис. 6.2.3 – Окно при несоответствии формата дат

* В случае пустых полей для ввода выводится соответствующее сообщение. Рисунок 6.2.4 демонстрирует такую ситуацию.

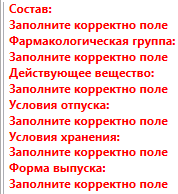


Рис. 6.2.4 – Окно при отсутствии данных для заполнения

* В случае ввода некорректной цены при поиске препарата выводится соответствующее сообщение. Рисунок 6.2.5 демонстрирует такую ситуацию.

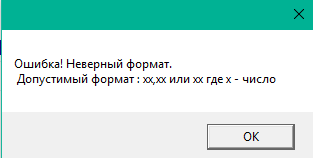


Рис. 6.2.5 – Окно при вводе некорректной даты пользователем

* В случае если пользователь не ввел данные для поиска выводится соответствующее сообщение. Рисунок 6.2.6 демонстрирует такую ситуацию.

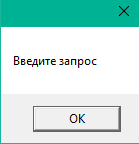


Рис. 6.2.6 – Окно при отсутствии введенных данных

Следует отметить, что данное приложение предусматривает проверку на корректность введенных данных пользователем.

# **Глава 7. Руководство пользователя**

При запуске приложения мы видим стартовое окно. Мы можем сразу задать критерий поиска и найти интересующий препарат. Рисунки 7.1 и 7.2 демонстрируют это. Листинги основных функций приложения находятся в Приложении Д.

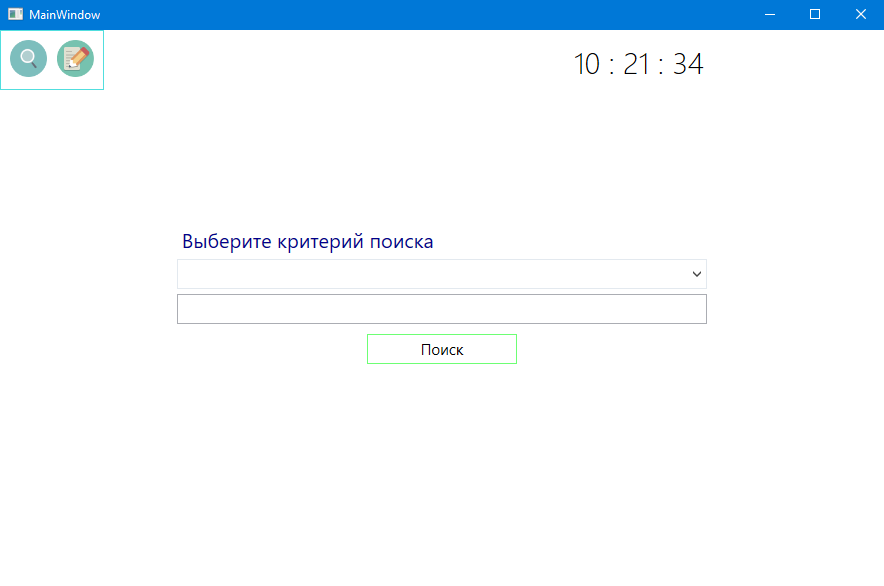


Рис. 7.1 - Стартовое окно приложения

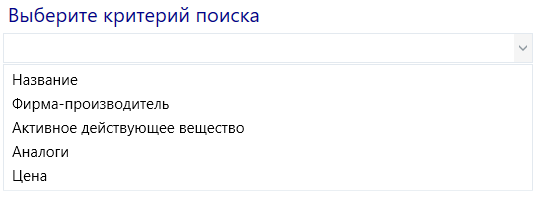


Рис. 7.2 – Окно выбора критерия для поиска

При вводе корректных данных мы получаем результат. Рисунок 7.3 демонстрирует это.

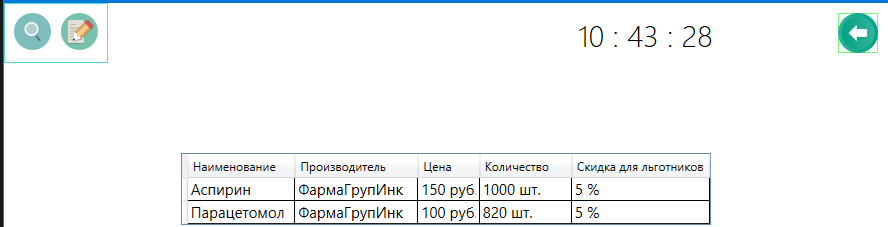


Рис. 7.3 – Окно результата поиска

Далее мы можем просмотреть аннотацию интересующего нас препарата.

Рисунок 7.4 демонстрирует это.

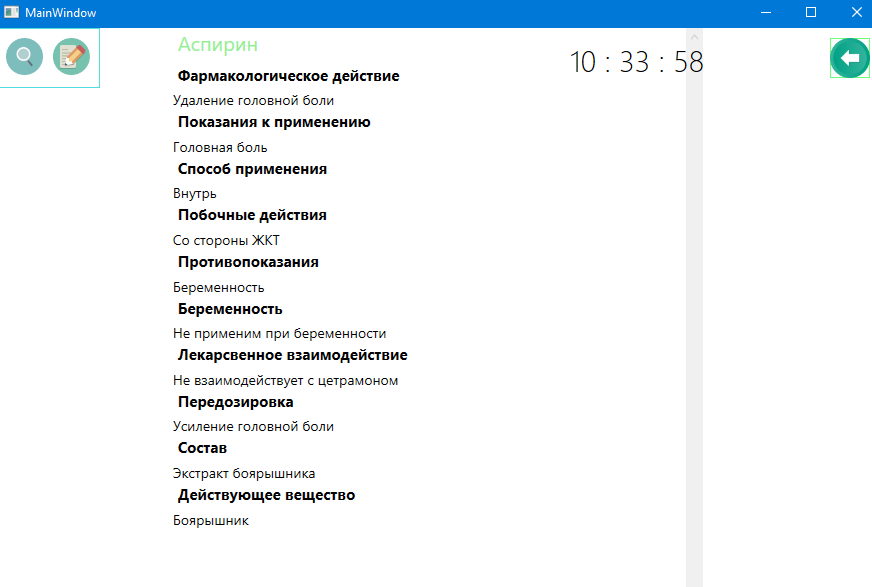


Рис. 7.4 – Окно просмотра аннотации

Если препарат не найден то программа выдаст нам результат показанный на рисунке 7.5



Рис. 7.5 – Окно результата поиска

Так как пользователь является одновременно и администратором приложения то он может добавлять препараты и аннотации. Перейдем в меню добавления аннотации и добавим ее. Демонстрация на рисунке 7.6.

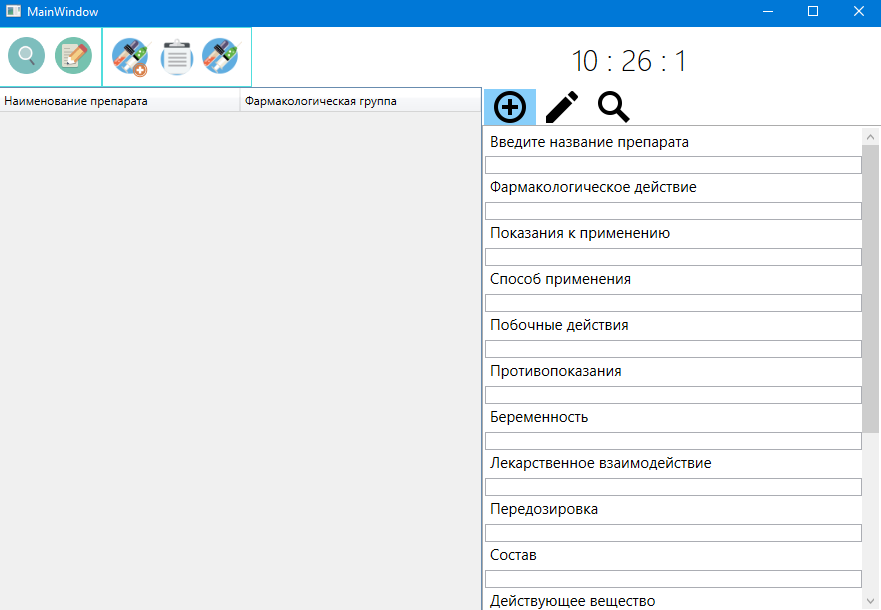


Рис. 7.6 Окно добавления аннотации

Результат мы увидим на рисунках 7.7 и 7.8 соответственно.

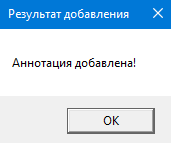


Рис. 7.7 Результат добавления аннотации



Рис. 7.8 Результат добавления аннотации

После добавления аннотации можно добавить препарат. Перейдем в окно добавления препарата и добавим его. Окно и результат добавления изображены на рисунках 7.9, 7.10, 7.11 соответственно.

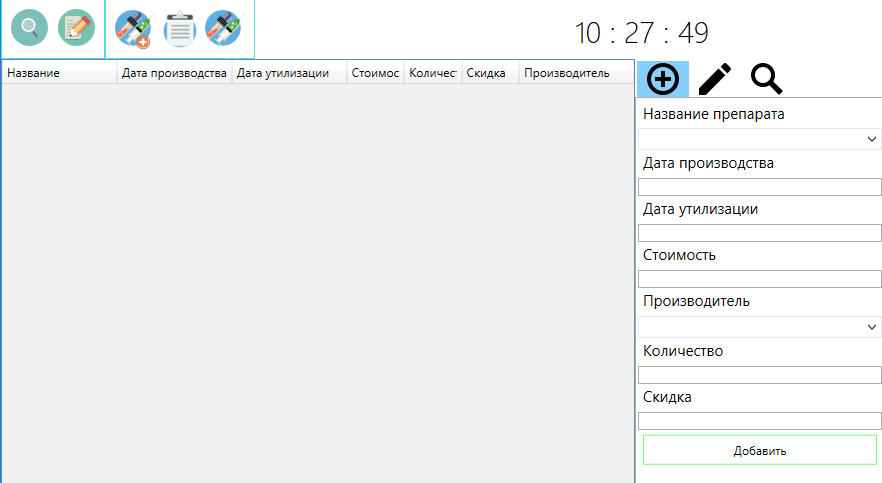


Рис. 7.9 – Окно добавления препарата

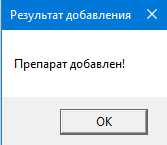


Рис. 7.10 – Результат добавления препарата



Рис. 7.11 – Результат добавления препарата

Если нам необходимо добавить аналог к уже существующему препарату то это можно сделать перейдя в меню добавления аналога. На рисунке 7.12, 7.13, 7.14 изображено меню добавления и соответственно результат.

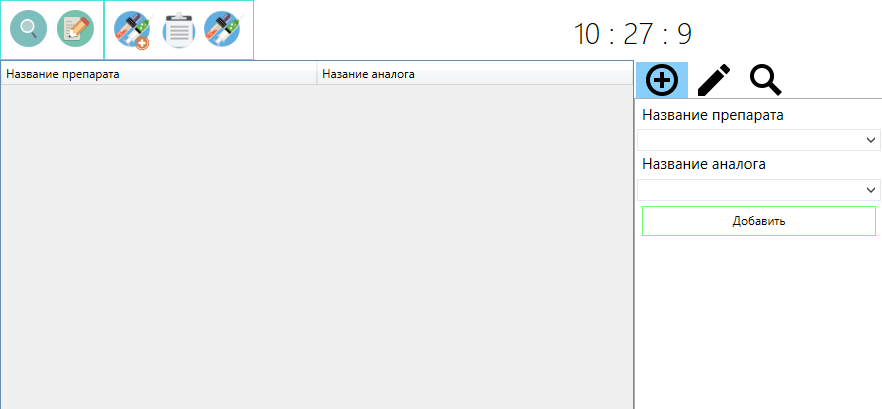


Рис. 7.12 – Окно добавления аналога

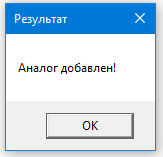


Рис. 7.13 – Результат добавления аналога



Рис. 7.14 – Результат добавления аналога

# **Заключение**

В данном курсовом проекте была разработана база данных для аптеки.

Она может служить хорошим помощником для хранения лекарственных средств. Также было разработано приложение которое значительно упрощает взаимодействия с базой данных. Помимо этого, было настроено резервное копирование базы данных на случай сбоя работы, а также разобрана технология интеграции.

В соответствии с полученным результатом, можно сказать, что разработанная программа работает верно, а требования технического задания выполнены в полном объеме.

# **Список использованной литературы**

1. Смелов В.В. Курс лекций по базам данных / В.В. Смелов – Минск БГТУ
2. Пацей, Н.В. Технология разработки программного обеспечения / Н.В. Пацей. – Минск: БГТУ, 2016. – 129 с.
3. ProfessorWeb .NET & Web Programming [Электронный ресурс] – Режим доступа: [https://professorweb.ru](https://vk.com/away.php?to=https%3A%2F%2Fprofessorweb.ru&cc_key=)
4. Надёжное обслуживание баз MS SQL Server [Электронный ресурс] – Режим доступа: <https://habrahabr.ru/company/softlab/blog/266033/>
5. Руководство по Microsoft для разработчиков [Электронный ресурс] –

Режим доступа: <https://msdn.microsoft.com/ru-ru/dn308572.aspx>/

1. Руководство по MS SQL Server [Электронный ресурс] –

Режим доступа: https://metanit.com/sql/sqlserver/

1. Руководство по WPF [Электронный ресурс] –

Режим доступа: https://metanit.com/sharp/wpf/

1. SQL Server Repository [Электронный ресурс] –

Режим доступа: https://github.com/Microsoft/sql-server-samples

1. Паттерн MVVM [Электронный ресурс]

Режим доступа: <https://metanit.com/sharp/wpf/22.1.php>

1. Язык XML [Электронный ресурс]

Режим доступа: <https://ru.wikipedia.org/wiki/XML>

# **Приложение А**

USE [master]

GO

CREATE DATABASE [PharmacyCourseProject]

CONTAINMENT = NONE

ON PRIMARY

( NAME = N'Pharmacy', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL12.MSSQLSERVER\MSSQL\DATA\Pharmacy.mdf' , SIZE = 5120KB , MAXSIZE = UNLIMITED, FILEGROWTH = 1024KB )

LOG ON

( NAME = N'Pharmacy\_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL12.MSSQLSERVER\MSSQL\DATA\Pharmacy\_log.ldf' , SIZE = 2048KB , MAXSIZE = 2048GB , FILEGROWTH = 10%)

GO

ALTER DATABASE [PharmacyCourseProject] SET COMPATIBILITY\_LEVEL = 120

GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))

begin

EXEC [PharmacyCourseProject].[dbo].[sp\_fulltext\_database] @action = 'enable'

end

GO

ALTER DATABASE [PharmacyCourseProject] SET ANSI\_NULL\_DEFAULT OFF

GO

ALTER DATABASE [PharmacyCourseProject] SET ANSI\_NULLS OFF

GO

ALTER DATABASE [PharmacyCourseProject] SET ANSI\_PADDING OFF

GO

ALTER DATABASE [PharmacyCourseProject] SET ANSI\_WARNINGS OFF

GO

ALTER DATABASE [PharmacyCourseProject] SET ARITHABORT OFF

GO

ALTER DATABASE [PharmacyCourseProject] SET AUTO\_CLOSE OFF

GO

ALTER DATABASE [PharmacyCourseProject] SET AUTO\_SHRINK OFF

GO

ALTER DATABASE [PharmacyCourseProject] SET AUTO\_UPDATE\_STATISTICS ON

GO

ALTER DATABASE [PharmacyCourseProject] SET CURSOR\_CLOSE\_ON\_COMMIT OFF

GO

ALTER DATABASE [PharmacyCourseProject] SET CURSOR\_DEFAULT GLOBAL

GO

ALTER DATABASE [PharmacyCourseProject] SET CONCAT\_NULL\_YIELDS\_NULL OFF

GO

ALTER DATABASE [PharmacyCourseProject] SET NUMERIC\_ROUNDABORT OFF

GO

ALTER DATABASE [PharmacyCourseProject] SET QUOTED\_IDENTIFIER OFF

GO

ALTER DATABASE [PharmacyCourseProject] SET RECURSIVE\_TRIGGERS OFF

GO

ALTER DATABASE [PharmacyCourseProject] SET DISABLE\_BROKER

GO

ALTER DATABASE [PharmacyCourseProject] SET AUTO\_UPDATE\_STATISTICS\_ASYNC OFF

GO

ALTER DATABASE [PharmacyCourseProject] SET DATE\_CORRELATION\_OPTIMIZATION OFF

GO

ALTER DATABASE [PharmacyCourseProject] SET TRUSTWORTHY OFF

GO

ALTER DATABASE [PharmacyCourseProject] SET ALLOW\_SNAPSHOT\_ISOLATION OFF

GO

ALTER DATABASE [PharmacyCourseProject] SET PARAMETERIZATION SIMPLE

GO

ALTER DATABASE [PharmacyCourseProject] SET READ\_COMMITTED\_SNAPSHOT OFF

GO

ALTER DATABASE [PharmacyCourseProject] SET HONOR\_BROKER\_PRIORITY OFF

GO

ALTER DATABASE [PharmacyCourseProject] SET RECOVERY FULL

GO

ALTER DATABASE [PharmacyCourseProject] SET MULTI\_USER

GO

ALTER DATABASE [PharmacyCourseProject] SET PAGE\_VERIFY CHECKSUM

GO

ALTER DATABASE [PharmacyCourseProject] SET DB\_CHAINING OFF

GO

ALTER DATABASE [PharmacyCourseProject] SET FILESTREAM( NON\_TRANSACTED\_ACCESS = OFF )

GO

ALTER DATABASE [PharmacyCourseProject] SET TARGET\_RECOVERY\_TIME = 0 SECONDS

GO

ALTER DATABASE [PharmacyCourseProject] SET DELAYED\_DURABILITY = DISABLED

GO

EXEC sys.sp\_db\_vardecimal\_storage\_format N'PharmacyCourseProject', N'ON'

GO

USE [PharmacyCourseProject]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

use PharmacyCourseProject

CREATE TABLE [dbo].[Analogs](

[DrugName] [nvarchar](50) NOT NULL,

[AnalogName] [nvarchar](50) NOT NULL,

CONSTRAINT [PK\_Analogs] PRIMARY KEY CLUSTERED

(

[DrugName] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Discount](

[DrugId] [int] NOT NULL,

[Discount] [int] NOT NULL,

CONSTRAINT [PK\_Discount] PRIMARY KEY CLUSTERED

(

[DrugId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Drugs](

[DrugID] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](50) NOT NULL,

[ManufactureDate] [date] NOT NULL,

[DisposeDate] [date] NOT NULL,

[Cost] [float] NOT NULL,

[ProducerId] [int] NOT NULL,

[Count] [int] NOT NULL,

CONSTRAINT [PK\_Drugs] PRIMARY KEY CLUSTERED

(

[DrugID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Medications](

[Name] [nvarchar](50) NOT NULL,

[PharmachologicEffect] [nvarchar](3000) NOT NULL,

[IndicationsForUse] [nvarchar](1000) NOT NULL,

[ModeOfApplication] [nvarchar](1000) NOT NULL,

[SideEffects] [nvarchar](1000) NOT NULL,

[Contraindications] [nvarchar](1000) NOT NULL,

[Pregnancy] [nvarchar](1000) NOT NULL,

[DrugInteractions] [nvarchar](1000) NOT NULL,

[Overdose] [nvarchar](1000) NOT NULL,

[Composition] [nvarchar](1000) NOT NULL,

[PharmacologicalGroup] [nvarchar](100) NOT NULL,

[ActiveSubstance] [nvarchar](50) NOT NULL,

[LeaveConditions] [nvarchar](20) NOT NULL,

[IssueForm] [nvarchar](500) NOT NULL,

[StorageConditions] [nvarchar](500) NOT NULL,

CONSTRAINT [PK\_Medications\_1] PRIMARY KEY CLUSTERED

(

[Name] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[PharmacologicalGroup](

[PharmacologicalGroup] [nvarchar](100) NOT NULL,

CONSTRAINT [PK\_PharmacologicalGroup] PRIMARY KEY CLUSTERED

(

[PharmacologicalGroup] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Producers](

[ProducerID] [int] IDENTITY(1,1) NOT NULL,

[FirmName] [nvarchar](50) NOT NULL,

[County] [nvarchar](50) NOT NULL,

CONSTRAINT [PK\_Producers] PRIMARY KEY CLUSTERED

(

[ProducerID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Analogs] WITH CHECK ADD CONSTRAINT [FK\_Analogs\_Medications] FOREIGN KEY([DrugName])

REFERENCES [dbo].[Medications] ([Name])

ON UPDATE CASCADE

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[Analogs] CHECK CONSTRAINT [FK\_Analogs\_Medications]

GO

ALTER TABLE [dbo].[Discount] WITH CHECK ADD CONSTRAINT [FK\_Discount\_Drugs] FOREIGN KEY([DrugId])

REFERENCES [dbo].[Drugs] ([DrugID])

ON UPDATE CASCADE

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[Discount] CHECK CONSTRAINT [FK\_Discount\_Drugs]

GO

ALTER TABLE [dbo].[Drugs] WITH CHECK ADD CONSTRAINT [FK\_Drugs\_Medications] FOREIGN KEY([Name])

REFERENCES [dbo].[Medications] ([Name])

ON UPDATE CASCADE

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[Drugs] CHECK CONSTRAINT [FK\_Drugs\_Medications]

GO

ALTER TABLE [dbo].[Drugs] WITH CHECK ADD CONSTRAINT [FK\_Drugs\_Producers] FOREIGN KEY([ProducerId])

REFERENCES [dbo].[Producers] ([ProducerID])

GO

ALTER TABLE [dbo].[Drugs] CHECK CONSTRAINT [FK\_Drugs\_Producers]

GO

ALTER TABLE [dbo].[Medications] WITH CHECK ADD CONSTRAINT [FK\_Medications\_PharmacologicalGroup] FOREIGN KEY([PharmacologicalGroup])

REFERENCES [dbo].[PharmacologicalGroup] ([PharmacologicalGroup])

GO

ALTER TABLE [dbo].[Medications] CHECK CONSTRAINT [FK\_Medications\_PharmacologicalGroup]

GO

USE [master]

GO

ALTER DATABASE [PharmacyCourseProject] SET READ\_WRITE

GO

CREATE TABLE History

(

Id INT IDENTITY PRIMARY KEY,

ProductId INT NOT NULL,

Operation NVARCHAR(200) NOT NULL,

CreateAt DATETIME NOT NULL DEFAULT GETDATE(),

);

GO

# **Приложение Б**

USE PHARMACYCOURSEPROJECT;

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

-- ДОБАВЛЕНИЕ АННОТАЦИИ -- ПОСЛЕ ФАРМГРУППЫ

CREATE PROCEDURE ADDMEDICATIONS(@NAME NVARCHAR(50), @PHARMACHOLOGICEFFECT NVARCHAR(3000), @INDICATIONSFORUSE NVARCHAR(1000), @MODEOFAPPLICATION NVARCHAR(1000),

@SIDEEFFECTS NVARCHAR(1000), @CONTRAINDICATIONS NVARCHAR(1000), @PREGNANCY NVARCHAR(1000), @DRUGINTERACTIONS NVARCHAR(1000),

@OVERDOSE NVARCHAR(1000), @COMPOSITION NVARCHAR(1000), @PHARMACOLOGICALGROUP NVARCHAR(100), @ACTIVESUBSTANCE NVARCHAR(50),

@LEAVECONDITIONS NVARCHAR(20), @ISSUEFORM NVARCHAR(500), @STORAGECONDITIONS NVARCHAR(500))

AS BEGIN

INSERT INTO MEDICATIONS VALUES (@NAME, @PHARMACHOLOGICEFFECT, @INDICATIONSFORUSE, @MODEOFAPPLICATION, @SIDEEFFECTS, @CONTRAINDICATIONS,

@PREGNANCY, @DRUGINTERACTIONS, @OVERDOSE, @COMPOSITION, @PHARMACOLOGICALGROUP, @ACTIVESUBSTANCE,

@LEAVECONDITIONS, @ISSUEFORM, @STORAGECONDITIONS)

END

GO

EXEC ADDMEDICATIONS @NAME = 'СЕРГЕЙ', @PHARMACHOLOGICEFFECT = 'КАЕФ', @INDICATIONSFORUSE = 'ГОЛОВНАЯ БОЛЬ', @MODEOFAPPLICATION = 'ВНУТРЬ',

@SIDEEFFECTS = 'ЗУД', @CONTRAINDICATIONS = 'БЕРЕМЕННОСТЬ', @PREGNANCY = 'НЕ РЕКОМЕНДУЕТСЯ ПРИ БЕРЕМЕННОСТИ',

@DRUGINTERACTIONS = 'НЕ ПРИМЕНИМО С МАЗЬЮ ВИШНЕВСКОГО', @OVERDOSE = 'БОЛЬ В ЖЕЛУДКЕ', @COMPOSITION = 'МЕЛ',

@PHARMACOLOGICALGROUP = 'МЕТАБОЛИКИ', @ACTIVESUBSTANCE = 'МАКАРОНЫ', @LEAVECONDITIONS= 'ПО РЕЦЕПТУ',

@ISSUEFORM = 'ТАБЛЕТКИ', @STORAGECONDITIONS = 'В СУХОМ МЕСТЕ'

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

-- ДОБАВЛЕНИЕ АНАЛОГА -- ПОСЛЕ ДОБАВЛЕНИЯ ПРЕПАРАТА

CREATE PROCEDURE ADDANALOGS(@DRUGNAME NVARCHAR(50), @ANALOGNAME NVARCHAR(50))

AS BEGIN

DECLARE @NAME NVARCHAR(50)

DECLARE @ANALOGNAMED NVARCHAR(50)

DECLARE @PHARMAGROUP NVARCHAR(50)

DECLARE @ID INT

SET @NAME = (

SELECT PHARMACOLOGICALGROUP

FROM MEDICATIONS

WHERE NAME = @DRUGNAME

)

SET @ANALOGNAMED = (

SELECT PHARMACOLOGICALGROUP

FROM MEDICATIONS

WHERE NAME = @ANALOGNAME

)

SET @ID = ( SELECT ANALOGS.ID FROM ANALOGS WHERE (ANALOGS.ANALOGNAME = @DRUGNAME AND ANALOGS.DRUGNAME = @ANALOGNAME) OR

(ANALOGS.ANALOGNAME = @ANALOGNAME AND ANALOGS.DRUGNAME = @DRUGNAME))

IF (@NAME = @ANALOGNAMED) AND (@DRUGNAME <> @ANALOGNAME) AND @ID IS NULL

BEGIN

INSERT INTO ANALOGS VALUES(@DRUGNAME, @ANALOGNAME)

END

END

GO

EXEC ADDANALOGS @DRUGNAME = 'СЕРГЕЙ', @ANALOGNAME = 'АНДРЕЙ'

DELETE ANALOGS;

DROP PROCEDURE ADDANALOGS

SELECT \* FROM MEDICATIONS

SELECT \* FROM ANALOGS

SELECT \* FROM PHARMACOLOGICALGROUP

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

-- ДОБАВЛЕНИЕ ПРОИЗВОДИТЕЛЯ

CREATE PROCEDURE ADDPRODUCERS(@FIRMNAME NVARCHAR(50), @COUNTRY NVARCHAR(50))

AS BEGIN

INSERT INTO PRODUCERS VALUES(@FIRMNAME, @COUNTRY)

END

GO

EXEC ADDPRODUCERS @FIRMNAME = 'ФАРМАГРУПИНК', @COUNTRY = 'США'

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

--ДОБАВЛЕНИЕ СКИДКИ --ПОСЛЕ ДОБАВЛЕНИЯ ПРЕПАРАТА

CREATE PROCEDURE ADDDISCOUNT(@DRUGID INT, @DISCOUNT INT)

AS BEGIN

DECLARE @DRUGIQ INT;

SET @DRUGIQ = (SELECT DRUGS.DRUGID FROM DRUGS WHERE DRUGS.DRUGID = @DRUGID)

IF @DISCOUNT >=0 AND @DRUGID IS NOT NULL

INSERT INTO DISCOUNT VALUES(@DRUGID, @DISCOUNT)

END

GO

DROP PROCEDURE ADDDISCOUNT

EXEC ADDDISCOUNT @DRUGID = '46', @DISCOUNT = '50'

SELECT \* FROM DISCOUNT

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

-- ДОБАВЛЕНИЕ ФАРМГРУППЫ

CREATE PROCEDURE ADDPHARMAGROUP(@PHARMACOLOGICALGROUP NVARCHAR(100))

AS BEGIN

INSERT INTO PHARMACOLOGICALGROUP VALUES(@PHARMACOLOGICALGROUP)

END

GO

--EXEC ADDPHARMAGROUP @PHARMACOLOGICALGROUP = ''

--DELETE PHARMACOLOGICALGROUP WHERE PHARMACOLOGICALGROUP = ''

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

-- ДОБАВЛЕНИЕ ПРЕПАРАТА -- ПОСЛЕ ДОБАВЛЕНИЯ АННОТАЦИИ

CREATE PROCEDURE ADDDRUG (@NAME NVARCHAR(50), @MANUFACTUREDATE DATE, @DISPOSEDATE DATE, @COST FLOAT, @COUNT INT, @FIRMNAME NVARCHAR(50))

AS BEGIN

DECLARE @PRODUCER INT

DECLARE @NICK NVARCHAR(50)

DECLARE @ID INT;

SET @NICK = ( SELECT MEDICATIONS.NAME FROM MEDICATIONS WHERE MEDICATIONS.NAME = @NAME );

SET @PRODUCER = ( SELECT PRODUCERS.PRODUCERID FROM PRODUCERS WHERE PRODUCERS.FIRMNAME = @FIRMNAME );

SET @ID = ( SELECT DRUGS.DRUGID FROM DRUGS WHERE DRUGS.NAME = @NAME AND DRUGS.MANUFACTUREDATE = @MANUFACTUREDATE AND DRUGS.DISPOSEDATE = @DISPOSEDATE AND DRUGS.COST = @COST AND DRUGS.PRODUCERID = @PRODUCER)

-- ДОБАВЛЕНИЕ ПРЕПАРАТОВ ОДНОЙ ПАРТИИ

IF @ID IS NOT NULL

BEGIN

IF @COST = ( SELECT DRUGS.COST FROM DRUGS WHERE DRUGS.DRUGID = @ID)

UPDATE DRUGS SET DRUGS.COUNT = DRUGS.COUNT + @COUNT WHERE DRUGS.DRUGID = @ID

END

ELSE

BEGIN

IF @PRODUCER <> 0 AND @NICK IS NOT NULL

BEGIN

PRINT 'HELLO'

IF @MANUFACTUREDATE < @DISPOSEDATE AND @COST > 0 AND @COUNT >= 0

INSERT INTO DRUGS VALUES(@NAME, @MANUFACTUREDATE, @DISPOSEDATE, @COST, @PRODUCER, @COUNT)

END

END

END

GO

EXEC ADDDRUG @NAME = 'СЕРГЕЙ', @MANUFACTUREDATE = '10-10-2017', @DISPOSEDATE = '20-10-2019', @COST = '400', @COUNT = '100', @FIRMNAME = 'БЕЛМЕДПРЕПАРАТЫ'

DROP PROCEDURE ADDDRUG

SELECT \* FROM DRUGS

DELETE DRUGS

SELECT \* FROM MEDICATIONS

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

SELECT \* FROM HISTORY

use PharmacyCourseProject;

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

CREATE PROCEDURE DELETEALLDRUGS

AS BEGIN

DELETE DRUGS

END

EXEC DELETEALLDRUGS

DROP PROCEDURE DELETEALLDRUGS

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

CREATE PROCEDURE DELETEALLMEDIC

AS BEGIN

DELETE MEDICATIONS

END

EXEC DELETEALLMEDIC

DROP PROCEDURE DELETEALLMEDIC

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

CREATE PROCEDURE DELETEDRUGS(@ID INT)

AS BEGIN

IF(SELECT DRUGS.DRUGID FROM DRUGS WHERE DRUGS.DRUGID = @ID) IS NOT NULL

DELETE DRUGS WHERE DRUGS.DRUGID = @ID

END

EXEC DELETEDRUGS @ID = '75'

DROP PROCEDURE DELETEDRUGS

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

CREATE PROCEDURE DELETEPRODUCER(@PRODNAME NVARCHAR(50))

AS BEGIN

IF(SELECT PRODUCERS.FIRMNAME FROM PRODUCERS WHERE PRODUCERS.FIRMNAME = @PRODNAME) IS NOT NULL

DELETE PRODUCERS WHERE PRODUCERS.FIRMNAME = @PRODNAME

END

EXEC DELETEPRODUCER @PRODNAME = 'TEVA'

SELECT \* FROM PRODUCERS

DROP PROCEDURE DELETEPRODUCER

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

# **Приложение В**

USE PHARMACYCOURSEPROJECT;

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

CREATE TRIGGER DRUGS\_INSERT

ON DRUGS

AFTER INSERT

AS

INSERT INTO HISTORY(PRODUCTID, OPERATION)

SELECT DRUGID, 'ДОБАВЛЕН ПРЕПАРАТ ' + NAME

FROM INSERTED

DROP TRIGGER DRUGS\_INSERT

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

CREATE TRIGGER DRUGS\_DELETE

ON DRUGS

AFTER DELETE

AS

INSERT INTO HISTORY(PRODUCTID, OPERATION)

SELECT DRUGID, 'УДАЛЕН ПРЕПАРАТ ' + NAME

FROM DELETED

DROP TRIGGER DRUGS\_DELETE

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

CREATE TRIGGER DRUGS\_UPDATE

ON DRUGS

AFTER UPDATE

AS

INSERT INTO HISTORY(PRODUCTID, OPERATION)

SELECT DRUGID, 'ИЗМЕНЕН ПРЕПАРАТ ' + NAME

FROM INSERTED

DROP TRIGGER DRUGS\_UPDATE

# **Приложение Г**

CREATE PROC INSERTPROCUDERFROMXML

@PATH NVARCHAR(256)

AS

BEGIN

SET NOCOUNT ON

SET XACT\_ABORT ON

BEGIN TRAN

DECLARE @RESULTS TABLE (X XML)

DECLARE @SQL NVARCHAR(300)=

'SELECT

CAST(REPLACE(CAST(X AS VARCHAR(MAX)), ''ENCODING="UTF-16"'', ''ENCODING="UTF-8"'') AS XML)

FROM OPENROWSET(BULK '''+@PATH+''', SINGLE\_BLOB) AS T(X)';

INSERT INTO @RESULTS EXEC (@SQL)

DECLARE @XML XML = (SELECT TOP 1 X FROM @RESULTS);

INSERT INTO PRODUCERS(FIRMNAME, COUNTY)

SELECT

C3.VALUE('FIRM\_NAMES[1]', 'NVARCHAR(50)') AS FIRMNAME,

C3.VALUE('COUNTRY\_NAMES[1]', 'NVARCHAR(50)') AS COUNTY

FROM @XML.NODES('ROOT/ORDER') AS T3(C3)

COMMIT;

END;

GO

EXEC INSERTPROCUDERFROMXML 'D:\PHARMACY\COMP.XML'

DROP PROCEDURE INSERTPROCUDERFROMXML

// генерация XML на C#

class Program

{ static void Main(string[] args)

{

var xmldoc = new XDocument(

new XElement("Root"));

Random ran = new Random();

for (int i = 0; i < 100000; i++)

{

string Name = "Country";

string FirmName = "FirmName";

Name += i;

FirmName +=i;

xmldoc.Root.Add(new XElement("Order", new XElement("Country\_Names",Name), new XElement("Firm\_Names", FirmName)));

}

xmldoc.Save("comp.xml");

}

}

# **Приложение Д**

private bool ValidateAdd()

{

ErrorsAdd = null;

ErrorsAdd += ReturnErrorsString(addName, "Название препарата", 50);

ErrorsAdd += ReturnErrorsDate(addStartDate, "Дата производства");

ErrorsAdd += ReturnErrorsDate(addEndDate, "Дата утилизации");

ErrorsAdd += ReturnErrorsInt(addCost, "Стоимость", 5);

ErrorsAdd += ReturnErrorsInt\_(addDiscount, "Скидка", 100);

ErrorsAdd += ReturnErrorsInt\_(addCount, "Количество", 10000);

if (!ErrorsAdd.Equals(""))

{

ErrorVisibility = Visibility.Visible;

return false;

}

else

ErrorVisibility = Visibility.Collapsed;

return true;

}

private bool ValidateEdit()

{

ErrorsEdit = null;

ErrorsEdit += ReturnErrorsString(editName, "Название препарата", 50);

ErrorsEdit += ReturnErrorsDate(editStartDate, "Дата производства");

ErrorsEdit += ReturnErrorsDate(editEndDate, "Дата утилизации");

ErrorsEdit += ReturnErrorsInt(editCost, "Стоимость", 5);

ErrorsEdit += ReturnErrorsInt\_(editDiscount, "Скидка", 100);

ErrorsEdit += ReturnErrorsInt\_(editCount, "Количество", 10000);

if (!ErrorsEdit.Equals(""))

{

ErrorEditVisibility = Visibility.Visible;

return false;

}

else

ErrorEditVisibility = Visibility.Collapsed;

return true;

}

private void EditClick(object obj)

{

if(ValidateEdit())

{

if(EditButton.Equals("Изменить"))

{

if (drugsDatabase.IsExist(drugsDatabase.GetDrugClassObject(editName, DateTime.Parse(editStartDate), DateTime.Parse(editEndDate), int.Parse(editCount), int.Parse(editCost), editProducer)))

{

changedDrug = drugsDatabase.GetDrugClassObject(editName, DateTime.Parse(editStartDate), DateTime.Parse(editEndDate), int.Parse(editCount), int.Parse(editCost), editProducer);

changedDiscount = drugsDatabase.GetDiscountClassObject(changedDrug, int.Parse(editDiscount));

EditButton = "Сохранить";

CancelButton = Visibility.Visible;

}

else

MessageBox.Show("Данного объекта не существует","Ошибка");

}

else

{

if (ValidateEdit())

{

MessageBox.Show(drugsDatabase.Change(changedDrug,

drugsDatabase.GetDrugClassObject(editName, DateTime.Parse(editStartDate), DateTime.Parse(editEndDate), int.Parse(editCount), int.Parse(editCost), editProducer),

changedDiscount, drugsDatabase.GetDiscountClassObject(changedDrug, int.Parse(editDiscount)))

, "Результат изменения");

CancelButton = Visibility.Collapsed;

EditButton = "Изменить";

FilterDrugs();

NotifyAdminIsEmpty();

}

else

MessageBox.Show("Введите корректные данные!");

}

}

}

private void CancelClick(object obj)

{

EditButton = "Изменить";

CancelButton = Visibility.Collapsed;

}

private void AddClick(object obj)

{

if(ValidateAdd())

{

Drugs drug;

MessageBox.Show(drugsDatabase.Add(

drug = drugsDatabase.GetDrugClassObject(addName, DateTime.Parse(addStartDate), DateTime.Parse(addEndDate), int.Parse(addCount), int.Parse(addCost), addProducer),

drugsDatabase.GetDiscountClassObject(drug, int.Parse(addDiscount))

),"Результат добавления");

FilterDrugs();

NotifyAdminIsEmpty();

}

}

private void SearchClick(object obj)

{

FilterDrugs();

}

public void NotifyAdminIsEmpty()

{

NotifyMessage = null;

NotifyMessage = "Добавьте пожалуйста препараты:\n";

foreach (var i in Drugs)

if (i.Count == 0)

NotifyMessage += '-' + i.Medications.Name + '\n';

if (!NotifyMessage.Equals("Добавьте пожалуйста препараты:\n"))

Notification = Visibility.Visible;

else

Notification = Visibility.Collapsed;

}

public void FilterDrugs()

{

Drugs\_.Clear();

if (Drugs != null)

foreach (var a in Drugs)

if(FilterDrugs(a))

Drugs\_.Add(a);

}

private bool FilterDrugs(Drugs a)

{

if (!string.IsNullOrWhiteSpace(searchCost) && !a.Cost.ToString().Contains(searchCost))

return false;

if (!string.IsNullOrWhiteSpace(searchEndDate) && !a.DisposeDate.ToString().Contains(searchEndDate))

return false;

if (!string.IsNullOrWhiteSpace(searchStartDate) && !a.ManufactureDate.ToString().Contains(searchStartDate))

return false;

if (!string.IsNullOrWhiteSpace(searchName) && !a.Medications.Name.Contains(searchName))

return false;

if (!string.IsNullOrWhiteSpace(searchProducer) && !a.Producers.FirmName.Contains(searchProducer))

return false;

return true;

}

private void DeleteClick(object obj)

{

if(ValidateEdit())

{

MessageBox.Show(drugsDatabase.Remove(drugsDatabase.GetDrugClassObject(editName, DateTime.Parse(editStartDate), DateTime.Parse(editEndDate), int.Parse(editCount), int.Parse(editCost), editProducer)), "Результаты удаления");

FilterDrugs();

NotifyAdminIsEmpty();

}

}

// добавление аннотации

private bool ValidateAdd()

{

ErrorsAdd = null;

ErrorsAdd += ReturnErrorsString(AddName, "Название препарата", 50);

ErrorsAdd += ReturnErrorsString\_(AddEffect, "Фармакологический эффект", 3000);

ErrorsAdd += ReturnErrorsString\_(AddIndicationsForUse, "Показания к применению", 1000);

ErrorsAdd += ReturnErrorsString\_(AddModeOfApplication, "Способ применения", 1000);

ErrorsAdd += ReturnErrorsString(AddSideEffects, "Побочные эффекты", 1000);

ErrorsAdd += ReturnErrorsString\_(AddPregnancy, "Беременность", 1000);

ErrorsAdd += ReturnErrorsString\_(AddContraindications, "Противопоказания", 1000);

ErrorsAdd += ReturnErrorsString\_(AddDrugInteractions, "Лекарственное взаимодействие", 1000);

ErrorsAdd += ReturnErrorsString(AddOverdose, "Передозировка", 1000);

ErrorsAdd += ReturnErrorsString\_(AddComposition, "Состав", 1000);

ErrorsAdd += ReturnErrorsString\_(AddPharmacologicalGroup, "Фармакологическая группа", 100);

ErrorsAdd += ReturnErrorsString\_(AddActiveSubstance, "Действующее вещество", 50);

ErrorsAdd += ReturnErrorsString\_(AddLeaveConditions, "Условия отпуска", 20);

ErrorsAdd += ReturnErrorsString\_(AddStorageConditions, "Условия хранения", 500);

ErrorsAdd += ReturnErrorsString\_(AddIssueForm, "Форма выпуска", 50);

if (ErrorsAdd.Equals(""))

{

ErrorVisibility = Visibility.Collapsed;

return true;

}

else

ErrorVisibility = Visibility.Visible;

return false;

}

private bool ValidateEdit()

{

ErrorsEdit = null;

ErrorsEdit += ReturnErrorsString(EditName, "Название препарата", 50);

ErrorsEdit += ReturnErrorsString\_(EditEffect, "Фармакологический эффект", 3000);

ErrorsEdit += ReturnErrorsString\_(EditIndicationsForUse, "Показания к применению", 1000);

ErrorsEdit += ReturnErrorsString\_(EditModeOfApplication, "Способ применения", 1000);

ErrorsEdit += ReturnErrorsString(EditSideEffects, "Побочные эффекты", 1000);

ErrorsEdit += ReturnErrorsString\_(EditPregnancy, "Беременность", 1000);

ErrorsEdit += ReturnErrorsString\_(EditContraindications, "Противопоказания", 1000);

ErrorsEdit += ReturnErrorsString\_(EditDrugInteractions, "Лекарственное взаимодействие", 1000);

ErrorsEdit += ReturnErrorsString(EditOverdose, "Передозировка", 1000);

ErrorsEdit += ReturnErrorsString\_(EditComposition, "Состав", 1000);

ErrorsEdit += ReturnErrorsString\_(EditPharmacologicalGroup, "Фармакологическая группа", 100);

ErrorsEdit += ReturnErrorsString\_(EditActiveSubstance, "Действующее вещество", 50);

ErrorsEdit += ReturnErrorsString\_(EditLeaveConditions, "Условия отпуска", 20);

ErrorsEdit += ReturnErrorsString\_(EditStorageConditions, "Условия хранения", 500);

ErrorsEdit += ReturnErrorsString\_(EditIssueForm, "Форма выпуска", 50);

if (ErrorsEdit.Equals(""))

{

ErrorEditVisibility = Visibility.Collapsed;

return true;

}

else

ErrorEditVisibility = Visibility.Visible;

return false;

}

private void AddClick(object obj)

{

if (ValidateAdd())

{

MessageBox.Show(aDb.Add(aDb.GetMedicationClassObject(addName,addActiveSubstance,addComposition,

addContraindications, addDrugInteractions, addIndicationsForUse, addIssueForm,

addLeaveConditions, addModeOfApplication, addOverdose, addEffect, addPharmacologicalGroup,

addPregnancy, addSideEffects, addStorageConditions)), "Результат добавления");

Med = new ObservableCollection<Medications>(MedicationDatabase);

ErrorsAdd = null;

}

}

private void SearchClick(object obj)

{

Med.Clear();

if (MedicationDatabase != null)

foreach (var m in MedicationDatabase)

if (IsContain(m))

Med.Add(m);

}

private bool IsContain(Medications m)

{

if (!string.IsNullOrWhiteSpace(searchName) && !m.Name.Contains(searchName))

return false;

if (!string.IsNullOrWhiteSpace(searchPharmacologicalGroup) && !m.PharmacologicalGroup.Contains(searchPharmacologicalGroup))

return false;

return true;

}

private void EditClick(object obj)

{

if(ValidateEdit())

{

if(EditButton.Equals("Изменить"))

{

EditButton = "Сохранить";

editMedication = aDb.GetMedicationClassObject(editName, editActiveSubstance, editComposition,

editContraindications, editDrugInteractions, editIndicationsForUse, editIssueForm,

editLeaveConditions, editModeOfApplication, editOverdose, editEffect, editPharmacologicalGroup,

editPregnancy, editSideEffects, editStorageConditions);

CancelButton = Visibility.Visible;

}

else

{

if(ValidateEdit())

{

CancelButton = Visibility.Collapsed;

EditButton = "Изменить";

MessageBox.Show(aDb.Change(editMedication,aDb.GetMedicationClassObject(editName, editActiveSubstance, editComposition,

editContraindications, editDrugInteractions, editIndicationsForUse, editIssueForm,

editLeaveConditions, editModeOfApplication, editOverdose, editEffect, editPharmacologicalGroup,

editPregnancy, editSideEffects, editStorageConditions)),"Рузультат изменения");

Med = new ObservableCollection<Medications>(MedicationDatabase);

}

}

}

}

private void DeleteClick(object obj)

{

if(ValidateEdit())

{

MessageBox.Show(aDb.Remove(aDb.GetMedicationClassObject(editName, editActiveSubstance, editComposition,

editContraindications, editDrugInteractions, editIndicationsForUse, editIssueForm,

editLeaveConditions, editModeOfApplication, editOverdose, editEffect, editPharmacologicalGroup,

editPregnancy, editSideEffects, editStorageConditions)),"Результат удаления");

Med = new ObservableCollection<Medications>(MedicationDatabase);

}

}

private void CancelClick(object obj)

{

EditButton = "Изменить";

CancelButton = Visibility.Collapsed;

}

# **Приложение E**

**//** резервное копирование

use PharmacyCourseProject;

DECLARE @pathName NVARCHAR(512)

SET @pathName = 'D:\BackUpDB\db\_backup\_' + Convert(varchar(8), GETDATE(), 112) + '.bak'

BACKUP DATABASE [PharmacyCourseProject] TO DISK = @pathName WITH NOFORMAT, NOINIT, NAME = N'db\_backup', SKIP, NOREWIND, NOUNLOAD, STATS = 10

// bat файл который запускает резервное копирование

sqlcmd -S DESKTOP-PO557BB -E -i backup.sql