Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет радиоэлектроники

Кафедра информационных радиотехнологий

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

ПРОГРАММНОЕ СРЕДСТВО «АПТЕКА»

Студент гр. 444601 В. Г. Пока

Руководитель С. А. Кучук

МИНСК 2016

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc468814630)

[1 ОБЗОР ЛИТЕРАТУРЫ 5](#_Toc468814631)

[2 СТРУКТУРНОЕ ПРОЕКТИРОВАНИЕ 9](#_Toc468814632)

[3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ 15](#_Toc468814633)

[3.1 Модуль работы с данными приложения 15](#_Toc468814634)

[3.2 Модуль инфраструктуры приложения 16](#_Toc468814635)

[3.3 Модуль хранения данных 17](#_Toc468814636)

[3.4 Модуль пользовательского интерфейса 17](#_Toc468814637)

[4 РАЗРАБОТКА МОДУЛЕЙ 19](#_Toc468814638)

[5 ТЕСТИРОВАНИЕ 21](#_Toc468814639)

[6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 23](#_Toc468814640)

[ЗАКЛЮЧЕНИЕ 29](#_Toc468814641)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 30](#_Toc468814642)

[ПРИЛОЖЕНИЕ А – ФРАГМЕНТЫ ИСХОДНОГО КОДА 31](#_Toc468814643)

**ВВЕДЕНИЕ**

XXI век невозможно представить без информационных технологий, компьютеров, современных программных продуктов, которые так тесно вошли в нашу жизнь. Компьютеры, мобильные телефоны, плееры прочно обосновались в нашем ежедневном использовании. Современные программные продукты характеризуются сложностью описания: достаточно большим количеством различных функций и процессов, тесным взаимодействием компонентов. Программные средства стали очень активно применяться в каждой сфере деятельности (финансы, производство, торговля, бухгалтерия). Программы создаются, обновляются, сменяются. Но иногда возникает проблема, когда в определенной сфере явно не хватает хороших, очень удобных и практичных программных средств.

Разработка программного средства предполагает наличие в программе следующих функций:

- обеспечение хранения информации по исследуемой предметной области;

- возможность редактирования записей базы данных (корректно добавлять, изменять, удалять информацию);

- возможность осуществления поиска по заданным критериям;

- вывод на печать необходимой информации.

Цель курсовой работы – разработка программного средства «Аптека», которое имеет авторизацию. ПС администратора: регистрация, изменение, удаление аптеки, регистрация, изменение, удаление номенклатуры и количества лекарств в аптеке, цены. ПС пользователя: отчет «Поиск заданного лекарства в заданном количестве».

**1 ОБЗОР ЛИТЕРАТУРЫ**

Основным источником литературы, используемым в данной курсовой работе, является библиотека официальной технической документации для разработчиков программного обеспечения под операционные системы Microsoft Windows [1].

Библиотека содержит документацию на программные интерфейсы (API) продуктов Microsoft, а также код примеров, технические статьи и другую полезную информацию.

На данном интернет ресурсе было почерпнуто множество приёмов использования различных методов, классов и структур данных, определённых в программной платформе .NET Framework.

Почерпнутые на ресурсе MSDN знания касаются непосредственной разработки программного обеспечения, эти знания применимы для конкретных случаев разработки, таких как реализация определённых методов пользовательских классов, поддержка стандартных интерфейсов платформы NET Framework, работа с некоторыми интерфейсами приложений.

При проектировании структуры программы, а так же при выборе паттернов, реализующих основную функциональность, были использованы следующие источники.

В источнике описываются способы решения типичных задач, возникающих в объектно-ориентированном проектировании. Рассматриваются основные паттерны проектирования, и приводится их каталог[2]. Книга хорошо демонстрирует роль паттернов в создании архитектуры сложных систем. Даёт практические советы по разработке архитектуры приложений в рамках объектно-ориентированного подхода.

В данном источнике была почерпнута основная идея разрабатываемой архитектуры – чёткое разделение кода на модель данных и набор визуальных элементов, представляющих эти данные. Такое разделение является паттерном объектно-ориентированного проектирования под названием «модель-вид-контролер» (MVC).

Источником, используемым на этапе рефакторинга кода, является [3]. Под рефакторингом [3] понимается процесс изменения внутренней структуры программы, не затрагивающий её внешнего поведения и имеющий целью облегчить понимание её работы. В основе рефакторинга лежит последовательность небольших эквивалентных (то есть сохраняющих поведение) преобразований. Поскольку каждое преобразование маленькое, программисту легче проследить за его правильностью, и в то же время вся последовательность может привести к существенной перестройке программы и улучшению её согласованности и четкости.

В первой части излагаются принципы, паттерны и приёмы написания чистого кода, вторая часть состоит из практических сценариев, предоставляющих собой упражнения по рефакторингу кода, третья часть представляет собой базу знаний, дающую представление о процессе чтения, написания и рефакторинга кода.

Из данного источника были почерпнуты знания, касающиеся написания лаконичного, хорошо читаемого и, следовательно, легко поддерживаемого кода. Были проанализированы и переписаны некоторые части программы: определённые методы и классы были разбиты на составные части, имена стали отражать суть своего содержания, функции сокращены. Источник помог в ходе рефакторинга сделать исходный код прозрачным и легко читаемым.

Подробную информацию о технологии Microsoft под названием Windows Management Instrumentation можно почерпнуть в [4].

Windows Management Instrumentation (WMI) в дословном переводе – это инструментарий управления Windows. Если говорить более развернуто, то WMI – это одна из базовых технологий для централизованного управления и слежения за работой различных частей компьютерной инфраструктуры под управлением операционной системы Windows.

Данная технология представляет собой расширенную и адаптированную под Windows реализацию стандарта WBEM, принятого многими компаниями, в основе которого лежит идея создания универсального интерфейса мониторинга и управления различными системами и компонентами распределённой информационной среды предприятия с использованием объектно-ориентированных идеологий и протоколов HTML и XML.

В основе структуры данных в WBEM лежит Common Information Model (CIM), реализующая объектно-ориентированный подход к представлению компонентов системы. CIM является расширяемой моделью, что позволяет программам, системам и драйверам добавлять в неё свои классы, объекты, методы и свойства.

WMI, основанный на CIM, также является открытой унифицированной системой интерфейсов доступа к любым параметрам операционной системы, устройствам и приложениям, которые функционируют в ней.

Важной особенностью WMI является то, что хранящиеся в нём объекты соответствуют динамическим ресурсам, то есть параметры этих ресурсов постоянно меняются, поэтому параметры таких объектов не хранятся постоянно, а создаются по запросу потребителя данных. Хранилище свойств объектов WMI называется репозиторием и расположено в системной папке операционной системы Windows.

Также в рассматриваемом источнике было почерпнуто множество знаний касающихся практического применения тех или иных методов, описания свойств и структур данных.

**2 СТРУКТУРНОЕ ПРОЕКТИРОВАНИЕ**

Целями создания автоматизированной системы является повышение эффективности работы сотрудников аптеки, сокращение потерь рабочего времени, ведение учета лекарств.

Задачами создаваемой автоматизированной системы являются:

– экономия времени за счет сокращения времени на выполнение работ;

– повышение качества выходной информации путем автоматизации процессов учета лекарств;

– защита информации от несанкционированного доступа.

Назначением разрабатываемой автоматизированной системы являются:

– хранить данные в удобном для использования виде;

– получать доступ к необходимой информации в любой момент времени;

– редактировать данные по мере их изменения.

В автоматизированной системе должны быть автоматизированы следующие функции:

– ведение данных о лекарствах;

– ведение данных об аптеках;

– учет наличия лекарств в аптеках.

Входная информация вводится администратором. Введение информации должно быть максимально удобно.

Разрабатываемая автоматизированная система должна предоставлять следующие возможности:

– добавление или изменение существующих данных;

– обеспечение защиты данных от несанкционированного доступа;

– обеспечение защиты от ввода неверной информации.

Автоматизированная обработка информации может прекратиться в следующих случаях:

– при нарушении целостности структуры базы данных;

– при сбое работы сервера приложений.

Исходя из целей и задач, можно построить структурную схему автоматизированной системы (рисунок 2.1).

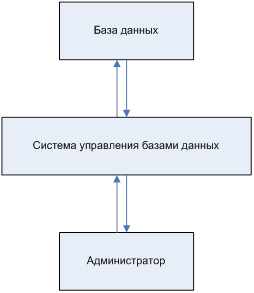


Рисунок 2.1 – Структурная схема автоматизированной системы

Язык моделирования может быть представлен нотациями UML, Буча, OMT и др. В данном случае была выбрана нотация UML, а в качестве инструментального средства — Rational Rose.

CASE-средство IBM Rational Rose представляет собой современный интегрированный инструментарий для проектирования архитектуры, анализа, моделирования и разработки программных систем. Язык UML в IBM Rational Rose стал базовой технологией визуализации и разработки программных систем.

Модель Rose — это картина системы. Она содержит:

- все диаграммы UML;

- действующих лиц;

- варианты использования;

- объекты;

- классы;

- компоненты;

- узлы системы.

Она детально описывает, что система содержит и как функционирует. Поэтому разработчики могут использовать её в качестве эскиза или чертежа создаваемой системы.

Действующее лицо (actor) — это роль, которую «субъект» играет по отношению к системе, а не конкретный человек или наименование работы.

В данной программе имеется несколько действующих лиц — администратор и пользователи.

Вариант использования представляет собой последовательность действий (транзакций), выполняемых системой в ответ на событие, которое инициируется некоторым действующим лицом.

Администратор может изменять, удалять, добавлять, просматривать данные.

Пользователи могут осуществить поиск лекарства, создать отчет о результатах поиска.

На диаграмме вариантов использования показывают взаимодействия (связи) между всеми действующими лицами и вариантами использования. Диаграмма должна показывать, какие действующие лица инициируют варианты использования, а также должна отображать, когда действующие лица получают информацию от вариантов использования.

На рисунке 2.2 представлена диаграмма вариантов использования программы:



Рисунок 2.2 — Диаграмма вариантов использования программы

Обобщенный алгоритм работы приложения и модуля сервера показан на рисунке 2.3.



Рисунок 2.3 - Алгоритм работы приложения в целом

На рисунке 2.4 представлена схема базы данных.



Рисунок 2.4 – Схема базы данных

База данных состоит из четырех таблиц:

Таблица «Собственность» - содержит информацию о формах собственности аптек.

Таблица «Лекарства» содержит код и название лекарств.

Таблица «Аптеки» содержит информацию о названиях, форме собственности, адресах, телефонах аптек.

Таблица «Наличие» предназначена для учета конкретного лекарства в конкретной аптеке.

**3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ**

Разрабатываемая информационная система представляет собой программу, установленную на компьютер, имеющий подключение к локальной вычислительной сети. Посредством пользовательского интерфейса данной программы, пользователю предоставляются возможности перечисленные ниже:

- обеспечение хранения информации по исследуемой предметной области;

- возможность редактирования записей базы данных (корректно добавлять, изменять, удалять информацию);

- возможность осуществления поиска по заданным критериям;

- вывод на печать необходимой информации.

Таким образом, при проектировании структуры информационной системы было определено четыре основных модулей:

- работы с данными приложения (модель);

- инфраструктуры приложения (инфраструктура);

- хранения данных;

- пользовательского интерфейса.

Рассмотрим подробнее каждый из перечисленных модулей.

**3.1 Модуль работы с данными приложения**

Задача модели данных состоит в том, что бы предоставить в распоряжение приложения всю необходимую для работы информацию и обеспечить выполнение команд пользователя.

После получения структур данных модель устанавливает необходимые зависимости, которые позволяют пользователю управлять полученной информацией. Происходит это следующим образом. В рассматриваемом модуле описаны все пользовательские команды. Команда представляет собой функцию с определённой сигнатурой и общей логикой выполнения, оперирующую некоторыми виртуальными методами абстрактных классов. Когда модель получает структуры данных от модуля работы с программным обеспечением, она формирует объекты конкретных классов, которые являются наследниками, используемых в системе команд, абстрактных структур. После этих действий, посредствам механизма виртуальных функций, выполняемая команда вместо абстракций манипулирует конкретными данными. Таким образом, модель данных приложения предоставляет доступ пользователя к управлению полученными данными, и исполняет роль обратной связи.

**3.2 Модуль инфраструктуры приложения**

Основная задача модуля инфраструктуры приложения – это предоставление некоторых глобальных сервисов уровня приложения и обеспечение связей между модулями. Этот модуль не влияет явным образом ни на информацию, полученную от удалённых компьютеров, ни на выполнение пользовательских команд. Вместо этого он определяет внутреннюю структуру приложения. Первой задачей, которую выполняет инфраструктура, является хранение внутренних интерфейсов приложения. Под интерфейсом здесь понимается конструкция в коде программы, используемая для специфицирования услуг, предоставляемых классом или компонентом. Интерфейс определяет границу взаимодействия между классами или компонентами, специфицируя определенную абстракцию, которую осуществляет реализующая сторона. Таким образом, модуль инфраструктуры, предоставляя информацию обо всех используемых внутри приложения интерфейсах, обеспечивает коммуникацию между остальными модулями, входящими в его состав.

Вторая задача инфраструктуры состоит в предоставлении сервиса глобальных событий. Описание этого сервиса имеет технический характер, и будет представлено в следующем разделе.

**3.3 Модуль хранения данных**

Как следует из названия данного модуля, задачей его является хранение результатов работы программы. Хранение осуществляется при помощи базы данных, а сам модуль представляет собой примитивную систему управления базой данных (СУБД).

Результатами работы программы являются два типа информации. К первому типу относятся данные, полученные программой в ходе сканирования и взаимодействия с сервисами удалённых компьютеров. Ко второму типу данных относиться введённая пользователем информация, состоящая из данных обеспечивающих учёт и некоторых специальных пометок, которые создаёт пользователь. Эта информация сохраняется в таблицы базы данных, структура которых, в свою очередь, имеет отражение в модели приложения. Доступ к этой информации обеспечивает инфраструктура, в которой находятся соответствующие интерфейсы для взаимодействия с сохраненными данными. Таким образом, модуль хранения, используя средства используемой базы данных, устанавливает зависимости между двумя рассмотренными видами информации и облегчает доступ к ней.

**3.4 Модуль пользовательского интерфейса**

Модуль пользовательского интерфейса предоставляет пользователю полученную информацию и даёт возможность изменять её. Пользовательский интерфейс представлен совокупностью визуальных компонентов, каждый из которых занимается визуализацией строго определённого блока данных программы. Данный модуль организован таким образом, что бы позволить без серьёзных трудозатрат добавлять новые компоненты или модифицировать старые. Все компоненты, входящие в состав пользовательского интерфейса имеют возможность настройки внешнего вида во время выполнения программы. Пользователь может динамически отключать либо подключать необходимые компоненты, а так же изменять их размер и местоположение.

Таким образом, рассмотренная структура приложения позволяет решить поставленные задачи по обеспечению основных действий пользователя, которые были рассмотрены в начале пункта.

На основании приведённой выше структуры информационной системы, анализа используемых в проектировании подходов и субъективных предпочтений был произведён выбор программных средств разработки.

Так как источниками данных приложения являются службы и сервисы разработанные корпорацией Microsoft, работающие на вычислительных машинах под управлением операционных систем семейства Windows, наиболее подходящей платформой для разработки является .NET Framework. Эта платформа поддерживает все вышеперечисленные технологии, приложения, написанные с её помощью, легко переносимы между различными типами аппаратного обеспечения, а так же между различными операционными системами. .NET Framework поддерживает множество языков программирования, для курсовой работы выбран Microsoft Visual Studio 2010 C#.

**4 РАЗРАБОТКА МОДУЛЕЙ**

В настоящем разделе будут рассмотрены алгоритмы работы ключевых процедур и функций.

Для поиска необходимого лекарства используется фильтрация данных:

private void textBox1\_TextChanged(object sender, EventArgs e)

{

view1BindingSource.Filter = "[lekarstvo] LIKE'" + textBox1.Text + "%'";

}

Для формирования отчета в Excel используется следующий код:

private void button1\_Click\_1(object sender, EventArgs e)

{

Excel.Application exApp = new Excel.Application();

exApp.Visible = true;

exApp.Workbooks.Add();

Worksheet workSheet = (Worksheet)exApp.ActiveSheet;

workSheet.Cells[1, 1] = "Лекарство";

workSheet.Cells[1, 2] = "Цена";

workSheet.Cells[1, 3] = "Количество";

workSheet.Cells[1, 4] = "Аптека";

workSheet.Cells[1, 5] = "Телефон";

workSheet.Cells[1, 6] = "Адрес";

int rowExcel = 2;

for (int i = 0; i < view1DataGridView.Rows.Count; i++)

{

workSheet.Cells[rowExcel, "A"] = view1DataGridView.Rows[i].Cells[0].Value;

workSheet.Cells[rowExcel, "B"] = view1DataGridView.Rows[i].Cells[1].Value;

workSheet.Cells[rowExcel, "C"] = view1DataGridView.Rows[i].Cells[2].Value;

workSheet.Cells[rowExcel, "D"] = view1DataGridView.Rows[i].Cells[3].Value;

workSheet.Cells[rowExcel, "E"] = view1DataGridView.Rows[i].Cells[4].Value;

workSheet.Cells[rowExcel, "F"] = view1DataGridView.Rows[i].Cells[5].Value;

++rowExcel;

}

string pathToXmlFile;

pathToXmlFile = Environment.CurrentDirectory + "\\" + "MyFile.xlsx";

workSheet.SaveAs(pathToXmlFile);

exApp.Quit();

}

**5 ТЕСТИРОВАНИЕ**

Важной составляющей при разработке программы является ее тестирование. Под тестированием понимается процесс выполнения программы с намерением найти ошибки. Если рассматривать процесс создания ПС в общем, то процесс тестирования является одним из самых продолжительных по времени.

Тесты бывают различных типов в зависимости от целей и приемов тестирования. Рассмотрим, для примера, следующую классификацию видов тестирования:

* тестирование модуля, или автономное тестирование (module testing, unit testing) – контроль отдельного программного модуля обычно в изолированной среде (т.е. изолированно от всех остальных модулей). Тестирование иногда может включать также математическое доказательство;
* тестирование сопряжений (integration testing) – контроль сопряжений между частями системы (модулями, компонентами, подсистемами);
* тестирование внешних модулей (external function testing) – контроль внешнего поведения системы, определенного внешними спецификациями;
* комплексное тестирование (system testing) – контроль и испытание системы по отношению к исходным целям;
* тестирование приемлемости (acceptance testing) – проверка соответствия программы требованиям пользователя;
* тестирование настройки (installation testing) – проверка соответствия каждого конкретного варианта установки системы с целью выявить любые ошибки, возникшие в процессе настройки системы;
* тестирование по методу «черного ящика» – программа рассматривается как одно целое и тесты генерируются на основе функциональной спецификации программы (то есть разделение на эквивалентные классы, анализ граничных значений). Тестировщик имеет доступ к ПО только через те же интерфейсы, что и заказчик или пользователь, либо через внешние интерфейсы, позволяющие другому компьютеру либо другому процессу подключиться к системе для тестирования;
* тестирование по методу «белого ящика» объединяет методы тестирования, основанные на структуре программы (т.е. прохождение операторов, прохождение ветвей, минимальное покрытие графа программы и т.д.). разработчик теста имеет доступ к исходному коду программ и может писать код, который связан с библиотеками тестируемого ПО. Это типично для юнит-тестирования (англ. unit testing), при котором тестируются только отдельные части системы. Оно обеспечивает то, что компоненты конструкции — работоспособны и устойчивы, до определённой степени. При тестировании белого ящика используются метрики покрытия кода;
* тестирование по методу «серого ящика». При тестировании серого ящика разработчик теста имеет доступ к исходному коду, но при непосредственном выполнении тестов доступ к коду, как правило, не требуется.

Для проведения испытаний разрабатываемой программы был выбран метод «черного ящика».

Таким образом, после проведения тестирования программы методами, описанными в данной главе, можно сделать вывод о том, что все требования, установленные заказчиком к разрабатываемой информационной системе, выполняются корректно.

**6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

Для корректной работы информационной системы «Аптека» необходимо следующее аппаратное обеспечение:

* платформа класса не ниже Intel Pentium IV;
* не менее 512Мб оперативной памяти (RAM);
* не менее 1 Гб свободного дискового пространства (HDD);
* сетевая карта;
* CD-ROM;
* клавиатура.

Требования к программному обеспечению следующие:

– операционная система Windows 7 и выше;

– платформа .Net Framework не ниже версии 4.0.

При запуске программы появляется окно авторизации. При выборе режима «Администратор» необходимо ввести пароль admin, при выборе режима «Пользователь» пароль вводить не надо (рисунок 6.1).

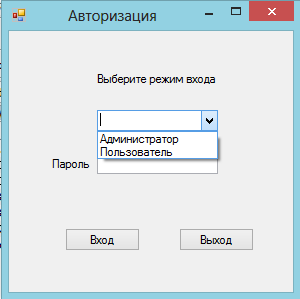


Рисунок 6.1 - Авторизация

На рисунке 6.2 представлена форма поиска лекарств.

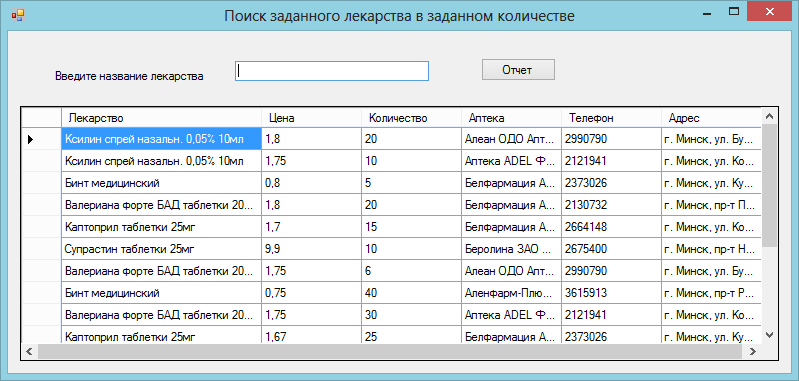


Рисунок 6.2 – Поиск заданного лекарства

Название лекарства вводится в текстовое поле и в таблице остаются только необходимые сведения (рисунок 6.3).

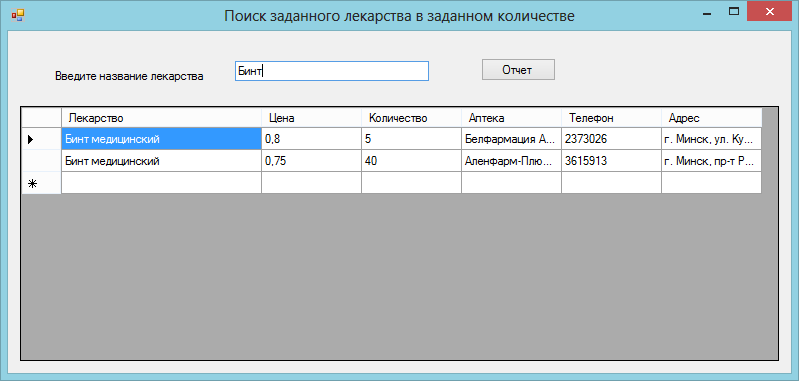


Рисунок 6.3 – Ввод названия лекарства

При нажатии на кнопку «Отчет» формируется отчет по найденным лекарствам, представленный на рисунке 6.4.

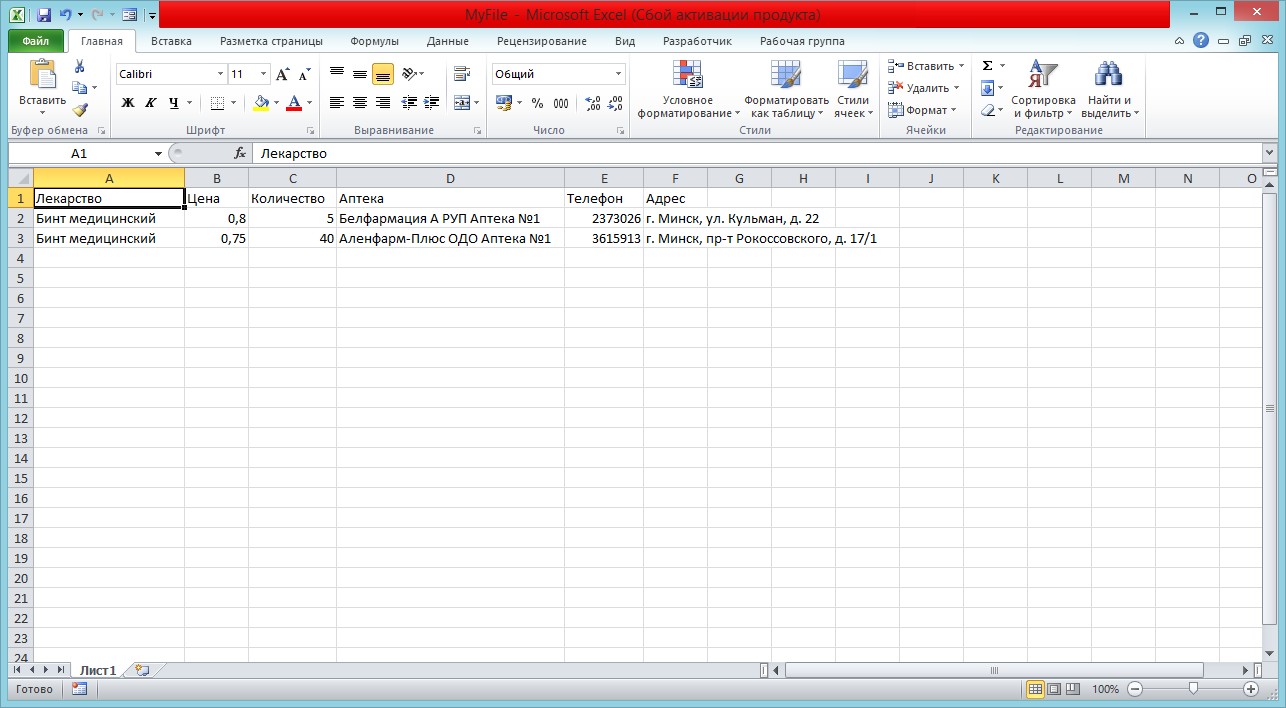


Рисунок 6.4 – Отчет по найденным лекарствам

На рисунке 6.5 представлена главная форма администратора.

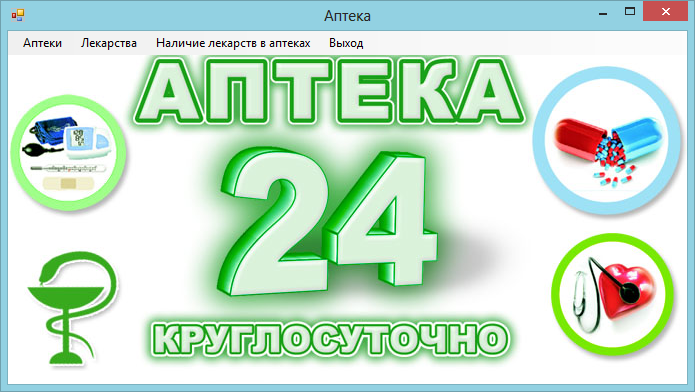


Рисунок 6.5 – Главная форма администратора

На рисунках 6.6 – 6.9 представлены формы для просмотра и редактирования информации, вызываемые из главного окна администратора.

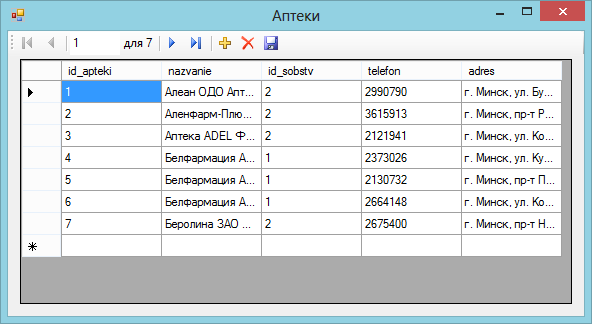


Рисунок 6.6 – Форма «Аптеки»

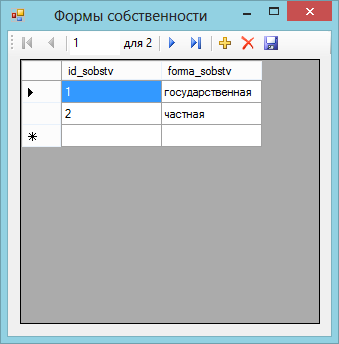


Рисунок 6.7 – Форма «Формы собственности»

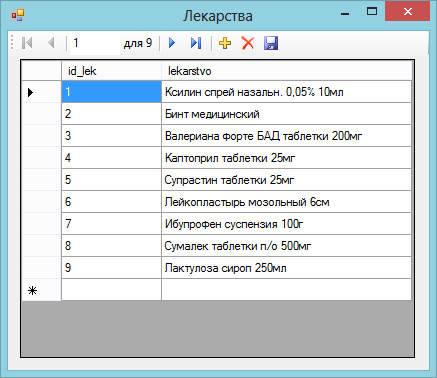


Рисунок 6.8 – Форма «Лекарства»

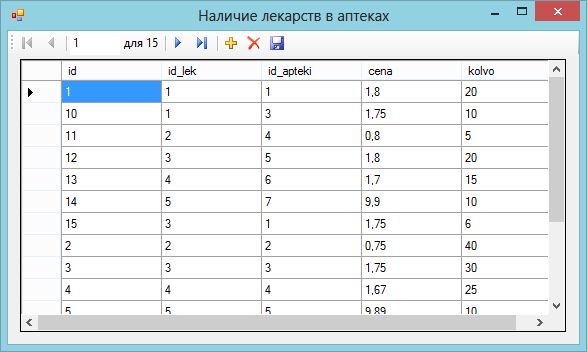


Рисунок 6.9 – Форма «Наличие лекарств в аптеках»

**ЗАКЛЮЧЕНИЕ**

В курсовой работе было разработано программное средство «Аптека», которое выполняет следующие функции:

- обеспечение хранения информации по исследуемой предметной области;

- возможность редактирования записей базы данных (корректно добавлять, изменять, удалять информацию);

- возможность осуществления поиска по заданным критериям;

- вывод на печать необходимой информации.

Таким образом, цель курсовой работы достигнута.

Результатом работы над курсовым проектом явилось приложение, которое может гармонично дополнить инструментарий системного администратора и перекрыть некоторые аспекты администрирования, упущенные в существующих программных продуктах.

В процессе выполнения данной курсовой работы были закреплены навыки в программировании на языке C#, проектировании баз данных и реализации их в СУБД MS SQL Server.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] MSDN [Электронный ресурс]. – Электронные данные. – Режим доступа: www.msdn.com.

[2] Э. Гамма. Примеры объектно-ориентированного проектирования. Паттерны проектирования, 2011 – 123 с.

[3] Р. Мартин. Чистый код. Создание, анализ рефакторинг, 2012 – 183 с.

[4] А. Попов. Администрирование Windowsс помощью WMI и WMIC, 2004 – 635 с.

[5] Г.Б.Розинский Разработка алгоритмов. Справочник. – М.: Мир и компьютер, 1991.

[6] Н.Кенрри Праг Библия пользователя Delphi 7. – Киев:   
ДИАЛЕКТИКА, 1996. – 576 с.

[7] П.Г. Дарахвелидзе, Е.П. Марков Программирования в Delphi 7. – СПб.: БХВ- Петербург, 2003. – 784 с.

[8] А.М.Лазаренков, В.А. Калиниченко. Охрана труда: учебник для студентов высших учебных заведений. –Мн.: ИВЦ Минфина, 2010.

[9] Технико-экономическое обоснование дипломных проектов. Методическое пособие для студентов всех специальностей БГУИР дневной и заочной форм обучения. Часть 2. Расчёт экономической эффективности инвестиционный проектов./Сост. А.А.Носенко, А.В. Грицай. – Мн.: БГУИР, 2002.

**ПРИЛОЖЕНИЕ А – ФРАГМЕНТЫ ИСХОДНОГО КОДА**

Form1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace Apteka

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

{

Application.Exit();

}

private void аптекиToolStripMenuItem1\_Click(object sender, EventArgs e)

{

Form2 FormDrives = new Form2();

FormDrives.Show();

}

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

{

Form3 FormDrives = new Form3();

FormDrives.Show();

}

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

{

Form4 FormDrives = new Form4();

FormDrives.Show();

}

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

{

Form5 FormDrives = new Form5();

FormDrives.Show();

}

}

}

Form7.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using Microsoft.Office.Interop.Excel;

using Excel = Microsoft.Office.Interop.Excel;

namespace Apteka

{

public partial class Form7 : Form

{

public Form7()

{

InitializeComponent();

}

private void Form7\_Load(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "aptekaDataSet.View1". При необходимости она может быть перемещена или удалена.

this.view1TableAdapter.Fill(this.aptekaDataSet.View1);

}

private void button1\_Click(object sender, EventArgs e)

{

Environment.Exit(0);

}

private void Form7\_FormClosed(object sender, FormClosedEventArgs e)

{

Environment.Exit(0);

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

view1BindingSource.Filter = "[lekarstvo] LIKE'" + textBox1.Text + "%'";

}

private void button1\_Click\_1(object sender, EventArgs e)

{

Excel.Application exApp = new Excel.Application();

exApp.Visible = true;

exApp.Workbooks.Add();

Worksheet workSheet = (Worksheet)exApp.ActiveSheet;

workSheet.Cells[1, 1] = "Лекарство";

workSheet.Cells[1, 2] = "Цена";

workSheet.Cells[1, 3] = "Количество";

workSheet.Cells[1, 4] = "Аптека";

workSheet.Cells[1, 5] = "Телефон";

workSheet.Cells[1, 6] = "Адрес";

int rowExcel = 2;

for (int i = 0; i < view1DataGridView.Rows.Count; i++)

{

workSheet.Cells[rowExcel, "A"] = view1DataGridView.Rows[i].Cells[0].Value;

workSheet.Cells[rowExcel, "B"] = view1DataGridView.Rows[i].Cells[1].Value;

workSheet.Cells[rowExcel, "C"] = view1DataGridView.Rows[i].Cells[2].Value;

workSheet.Cells[rowExcel, "D"] = view1DataGridView.Rows[i].Cells[3].Value;

workSheet.Cells[rowExcel, "E"] = view1DataGridView.Rows[i].Cells[4].Value;

workSheet.Cells[rowExcel, "F"] = view1DataGridView.Rows[i].Cells[5].Value;

++rowExcel;

}

string pathToXmlFile;

pathToXmlFile = Environment.CurrentDirectory + "\\" + "MyFile.xlsx";

workSheet.SaveAs(pathToXmlFile);

exApp.Quit();

}

}

}