МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Информационных систем и технологий

Специальность 1-98 01 03 «Программное обеспечение информационной безопасности мобильных систем»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированное программирование»

Тема «Аптека»

Исполнитель

студентка 2 курса 7 группы Лисункова А.Д.

Руководитель

Ассистент Панченко О. Л.

Курсовой проект защищен с оценкой

Руководитель Пацей Н.В.

Минск 2020

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПЕУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий   
Кафедра программной инженерии

Утверждаю

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_ \_\_ Н.В. Пацей

подпись инициалы и фамилия

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020г.

**ЗАДАНИЕ**

**к курсовому проектированию**

**по дисциплине** "Объектно-ориентированное программирование"

|  |  |
| --- | --- |
| Специальность: 1-98 01 03 Программное обеспечение информационной безопасности мобильных систем.  Студент: Лисункова А.Д. | Группа:\_\_7\_\_ |
| **Тема: Аптека** | |

**1. Срок сдачи студентом законченной работы**: "20 мая 2020 г."

**2. Исходные данные к проекту:**

**2.1**. Функционально ПС поддерживает:

* Выполнять функции администратора аптеки:
  + Возможность удаления, добавления и редактирования товаров в базе данных;
  + Оповещать клиента о выполнении заказа.
* Выполнять функции клиента для пользователя:
  + Выполнять регистрацию и авторизацию;
  + Выполнять поиск товаров;
  + Осуществлять заказ.

**2.2.** При выполнении курсового проекта необходимо использовать принципы и приемы ООП. Приложение разрабатывается под ОС Windows и представляет собой настольное приложение (desktop). Отображение, бизнес логика должны быть максимально независимы друг от друга для возможности расширения. Диаграммы вариантов использования, классов реализации задачи, взаимодействия разработать на основе UML. Язык разработки проекта – C#. Управление программой должно быть интуитивно понятным и удобным. При разработке использовать несколько наиболее подходящих шаблонов проектирования ПО.

**3. Содержание расчетно-пояснительной записки**

(перечень вопросов подлежащих разработке)

* Введение
* Постановка задачи и обзор литературы (алгоритмы решения, обзор прототипов, актуальность задачи)
* Проектирование архитектуры проекта (структура модулей, классов).
* Разработка функциональной модели и модели данных ПС (выполняемые функции)
* Тестирование
* Заключение
* Список используемых источников
* Приложения

**4. Форма представления выполненной курсовой работы:**

* + Теоретическая часть курсового проекта должны быть представлены в формате docx. Оформление записки должно быть согласно выданным правилам.
  + Листинги программы представляются частично в приложении.
  + Пояснительную записку, листинги, проект (инсталляцию проекта) необходимо загрузить диск, указанный преподавателем.

***Календарный план***

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Введение | 19.02.2020 |  |
| 2 | Аналитический обзор литературы по теме проекта. Изучение требований, определение вариантов использования | 12.03.2020 |  |
| 3 | Анализ и проектирование архитектуры приложения (построение диаграмм, проектирование бизнес-слоя, представления и данных) | 26.03.2020 |  |
| 4 | Проектирование структуры базы данных. Разработка дизайна пользовательского интерфейса | 2.04.2020 |  |
| 5 | Кодирование программного средства | 23.04.2020 |  |
| 6 | Тестирования и отладка программного средства | 30.04.2020 |  |
| 7 | Оформление пояснительной записки | 7.05.2020 |  |
| 9 | Сдача проекта | 20.05.2020 |  |

**5. Дата выдачи задания \_\_\_\_**12.02.2020**\_\_\_\_**

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *О. Л. Панченко*

(подпись)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата и подпись студента)

# 

Оглавление

[ВВЕДЕНИЕ 4](#_Toc41536866)

[1. Анализ прототипов и формирование требований к проектируемому программному средству 5](#_Toc41536867)

[2.Проектирование программного средства 8](#_Toc41536868)

[3. Разработка программного средства 11](#_Toc41536869)

[4. Тестирование, проверка работоспособности и анализ полученных результатов 15](#_Toc41536870)

[5. Руководство по установке и использованию 20](#_Toc41536871)

[ЗАКЛЮЧЕНИЕ 21](#_Toc41536872)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 22](#_Toc41536873)

[ПРИЛОЖЕНИЕ А 23](#_Toc41536874)

[ПРИЛОЖЕНИЕ Б 24](#_Toc41536875)

### ВВЕДЕНИЕ

Данная курсовая работа посвящена разработке приложения «Аптека».

Услуги аптек играют одну из важных ролей в сфере здоровья каждого человека. Люди болеют время от времени, а значит, имеют дело с лекарственными средствами, которые помогают выйти из состояния болезни. И в этом деле современная фармакология и промышленность медицинских материалов играют свою важную роль, чтобы медицинская помощь была возможной, своевременной и эффективной.

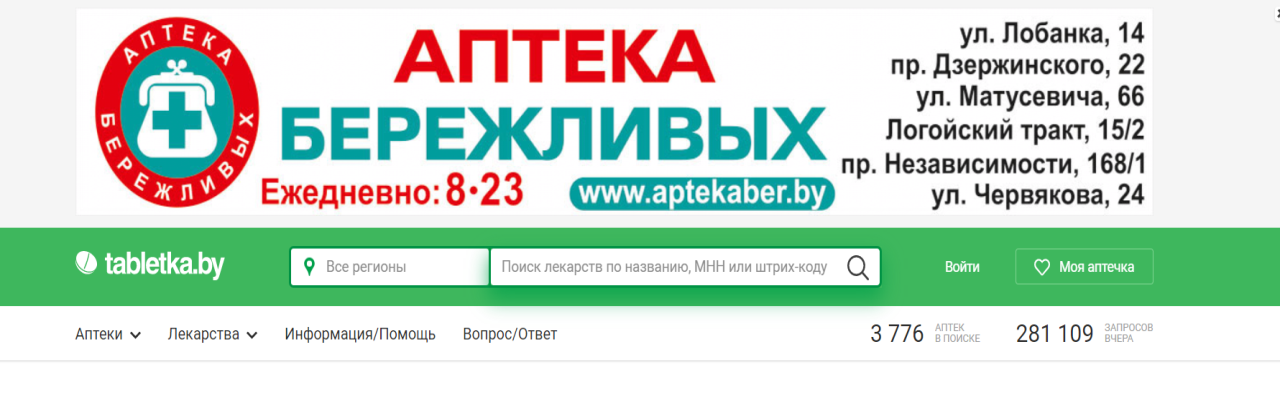
Готовое программное средство должно предоставлять возможность людям осуществлять поиск лекарств по определенным параметрам фильтра, а также осуществлять заказы выбранных товаров. Приложение также предоставляет рациональный подход к организации кабинета администратора, его взаимодействие с базой данных.

### Анализ прототипов и формирование требований к проектируемому программному средству

* 1. **Обзор и анализ прототипов конкурентов**

В данном разделе я буду сравнивать наиболее популярные приложения с аналогичной тематикой, чтобы просмотреть достоинства и функционал всех приложений.

* + 1. Первый объект обзора - программа «tabletka.by» (рис.1). [1]

Рис.1 – Главная

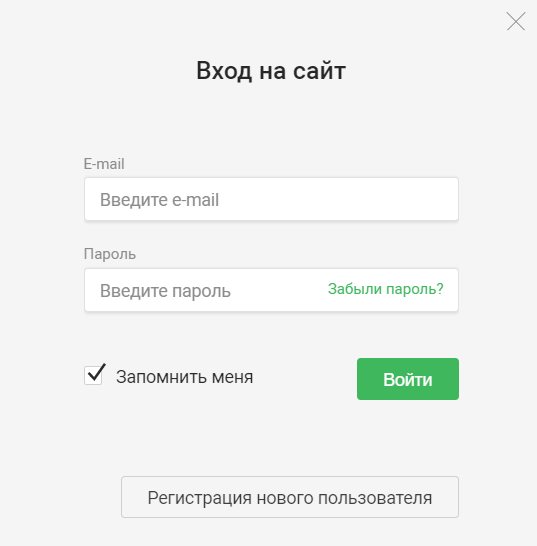


Рис.2 – Форма авторизации и регистрации пользователя

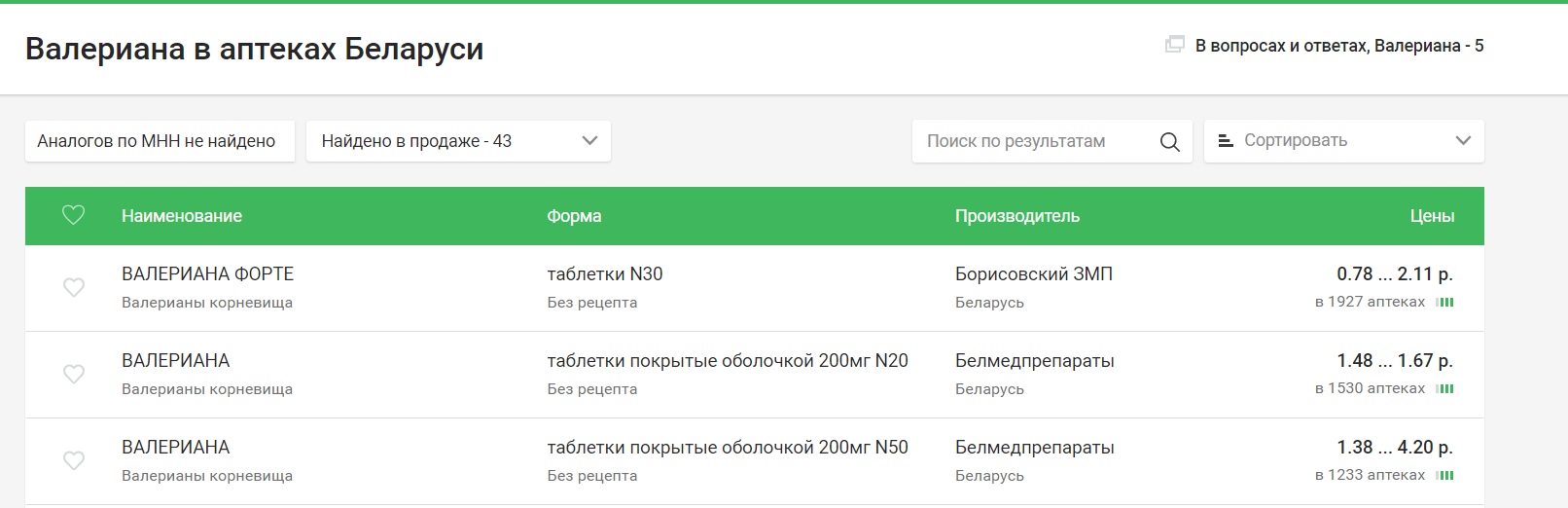


Рис. 3 – Результаты поиска

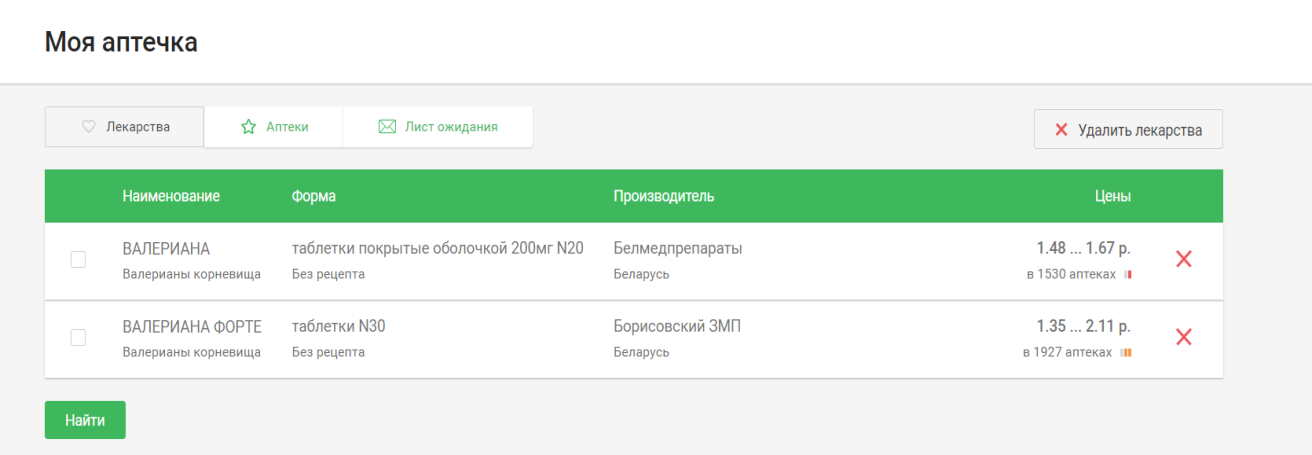


Рис. 4 – Корзина пользователя

Сразу можно выделить:

* Имеется возможность создать аккаунт (рис.2);
* Просматривать результаты поиска (рис. 3);
* Добавление товара в корзину (рис. 4);

Можно утверждать, что все сделано максимально удобно и приемлемо для пользователя в целом, является достойным примером.

* + 1. Второй объект обзора - «Аптека Adel» (рис. 5). [2]

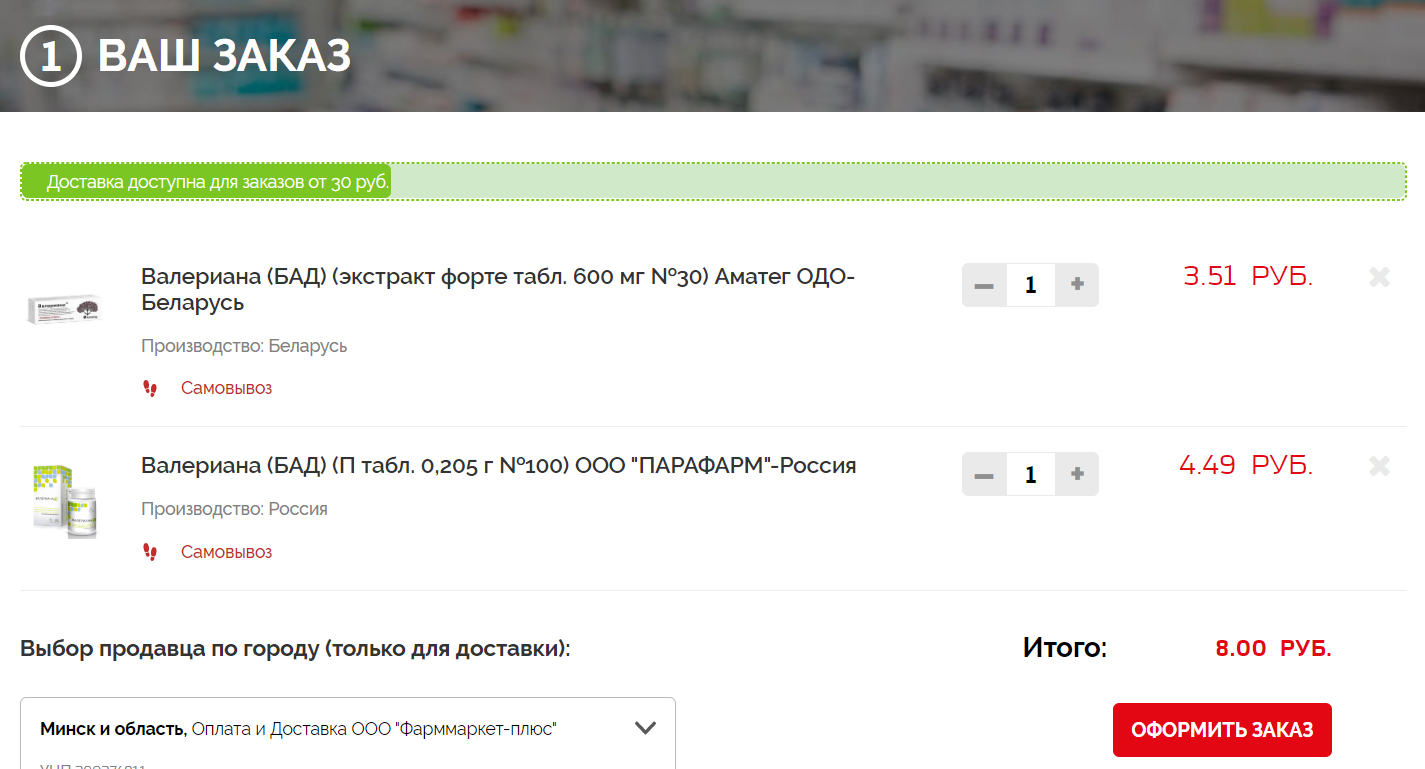


Рис.5 – Страницы заказов

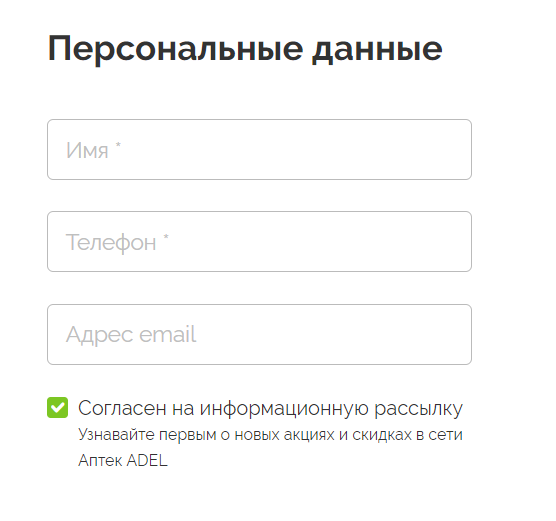


Рис.6 – Заполнение персональных данных пользователя при заказе

Сразу можно выделить:

-Добавлять товар в корзину (рис. 5);

-Заполнение формы для формирования заказа (рис. 6).

* + 1. Третий объект обзора - «Pilula.by» (рис. 7). [3]

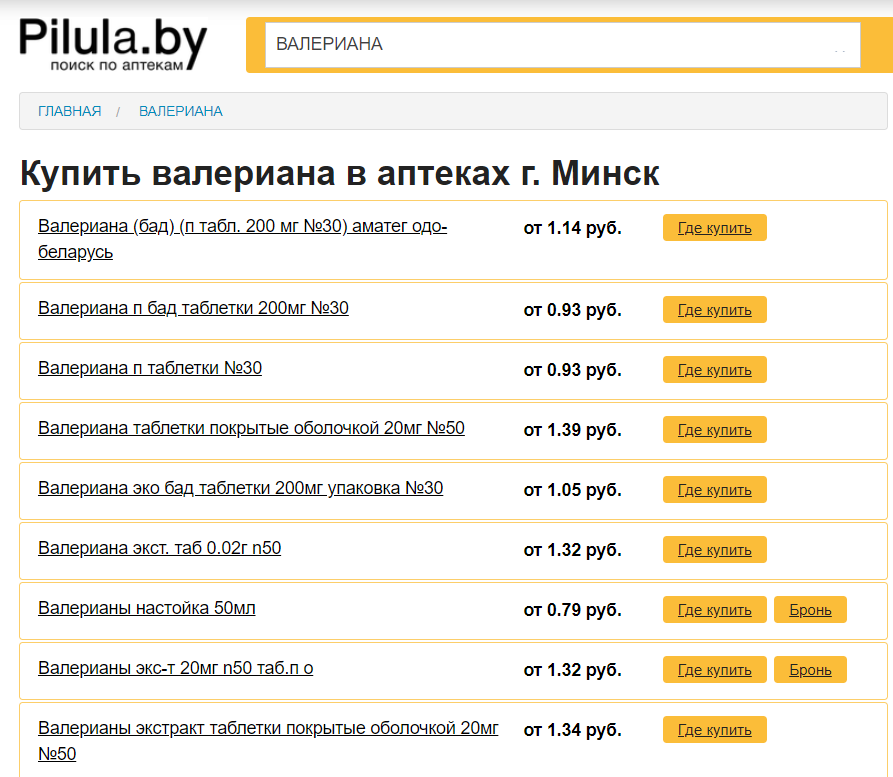


Рис.7 – Результаты поиска по именованию товара

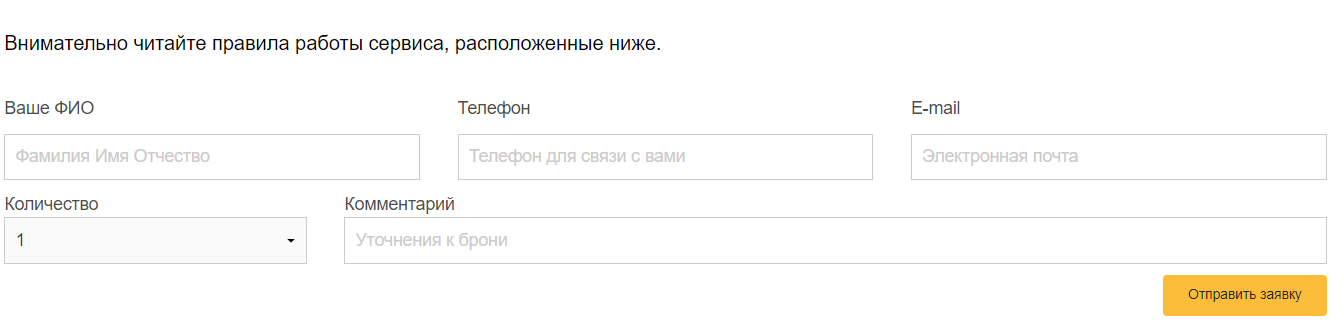


Рис.8 – Форма заполнения персональных данных для оформления заказа

Сразу можно выделить:

-Поиск по названию товара(рис. 7);

-Заполнение формы для формирования заказа (рис. 8).

* 1. **Анализ требований к программному средству и разработка функциональных требований**

При разработке приложения используются:

* Объектно-ориентированный язык программирования C#;
* Система для построения клиентских приложений WPF;
* Расширяемый язык разметки XAML;
* Технология для работы с базами данных Entity Framework;
* Среда разработки Visual Studio 2019;
* Система управления реляционными базами данных Microsoft SQL Server;

В качестве разрабатываемого функционала выделяем следующие основные функции приложения:

* Возможность удаления, добавления и редактирования товаров в базе данных.
* Возможность добавления, удаления и редактирования информации о поставщиках аптеки в базе данных.
* Возможность удаления, добавления и редактирования категории товаров в базе данных
* Возможность оповещения пользователей о выполнении заказов.
* Поиск товара по заданному фильтру.
* Добавление товара в корзину.
* Удаление товара из корзины.
* Со стороны пользователя допускается осуществления заказов.

### 2.Проектирование программного средства

**2.1 Архитектура системы**

Для программного средства в курсовом проекте будет использоваться система пользователей со строго ограниченным функционалом: пользователь имеет право просматривать, искать товар, добавлять товар в корзину, осуществлять заказ.

В архитектуру моего проекта входит взаимодействие интерфейса и программного кода с помощью паттерна проектирования MVVM, что позволяет разделить проект на три составляющие для удобства и эффективности работы.

Для работы с базой данных выбрана технология Entity Framework, так как использование этой технологии облегчает взаимодействие с базой данных.

Ниже приведена подробная схема взаимодействия приложения с пользователем.





Рис.9 Структура взаимодействия пользователя с приложением

Суть приложения заключается в том, что изначально предлагается пользователю зайти под своей учетной записью, совершать поиск интересующих лекарств, производителей препаратов или определенной категории лекарство, помещать в корзину выбранные товары, осуществлять заказы, получение уведомлений о выполненном заказе.

Со стороны администратора подразумевается работа с данными, хранящимися в базе данных, осуществляемая с помощью удобного и понятного интерфейса взаимодействия. К такому рода действий можно отнести: взаимодействие с информацией о товарах, их поставщиков и категорий; отправка уведомлений пользователям о выполнении заказов.

**2.2 Проектирование логической структуры базы данных**

Для реализации поставленной задачи проекта была создана база данных. Для её создания использовалось система управления реляционными базами данных Microsoft SQL Server.

На рисунке 10 приведена диаграмма базы данных.

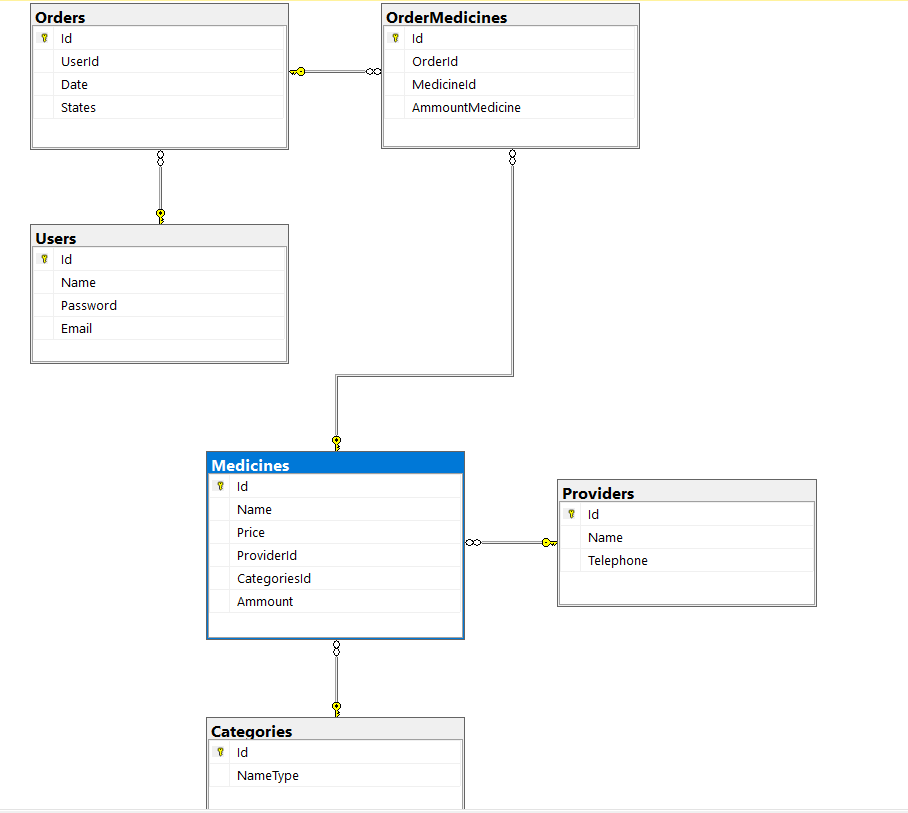


Рис. 10 – Диаграмма базы данных

Всего в базе данных 6 таблиц. В таблице Users хранятся данные о хранятся данные обо всех пользователях, зарегистрированных в программном средстве, в том числе, и об администраторе. В таблице Orders – данные об оформленных заказах.

Таблица OrderMedicines выполняет связующую функцию между таблицами заказов и препаратов, предоставляя связь многие-ко-многим. Таблица Medicines содержит препараты, таблица Providers хранит данные обо всех поставщиках аптеки, а таблица Categories – о категориях лекарств.

Проектирование таблиц было выполнено с помощью технологии Entity Framework с подходом Code First. Entity Framework представляет специальную объектно-ориентированную технологию на базе фреймворка .NET для работы с данными. Таким образом, все связи между таблицами и сами таблицы были спроектированы при создании модели данных.

Таблица Users состоит из 4 столбцов:

* Id – хранит идентификатор пользователя
* Name – хранит логин пользователя
* Password – хранит пароль пользователя
* Email – хранит Email пользователя

Таблица Orders состоит из 4 столбцов:

* Id – хранит идентификатор заказа
* UserId – хранит идентификатор пользователя
* Date – хранит дату и время заказа
* States – хранит состояние заказа

Таблица OrderMedicines состоит из 4 столбцов:

* Id – хранит свой идентификатор
* OrderId – хранит идентификатор заказа
* MedicineId – хранит идентификатор препарата
* AmmountMedicine – хранит количество препарата

Таблица Medicines состоит из 5 столбцов:

* Id – хранит идентификатор препарата
* Name – название препарата
* Price– хранит цену на один препарат
* ProviderId – хранит идентификатор поставщика
* CategoriesId – хранит идентификатор категории
* Ammount – общее количество поставляемого товара

Таблица Categories состоит из 2 столбцов:

* Id – хранит идентификатор категории
* NameType – хранит название категории

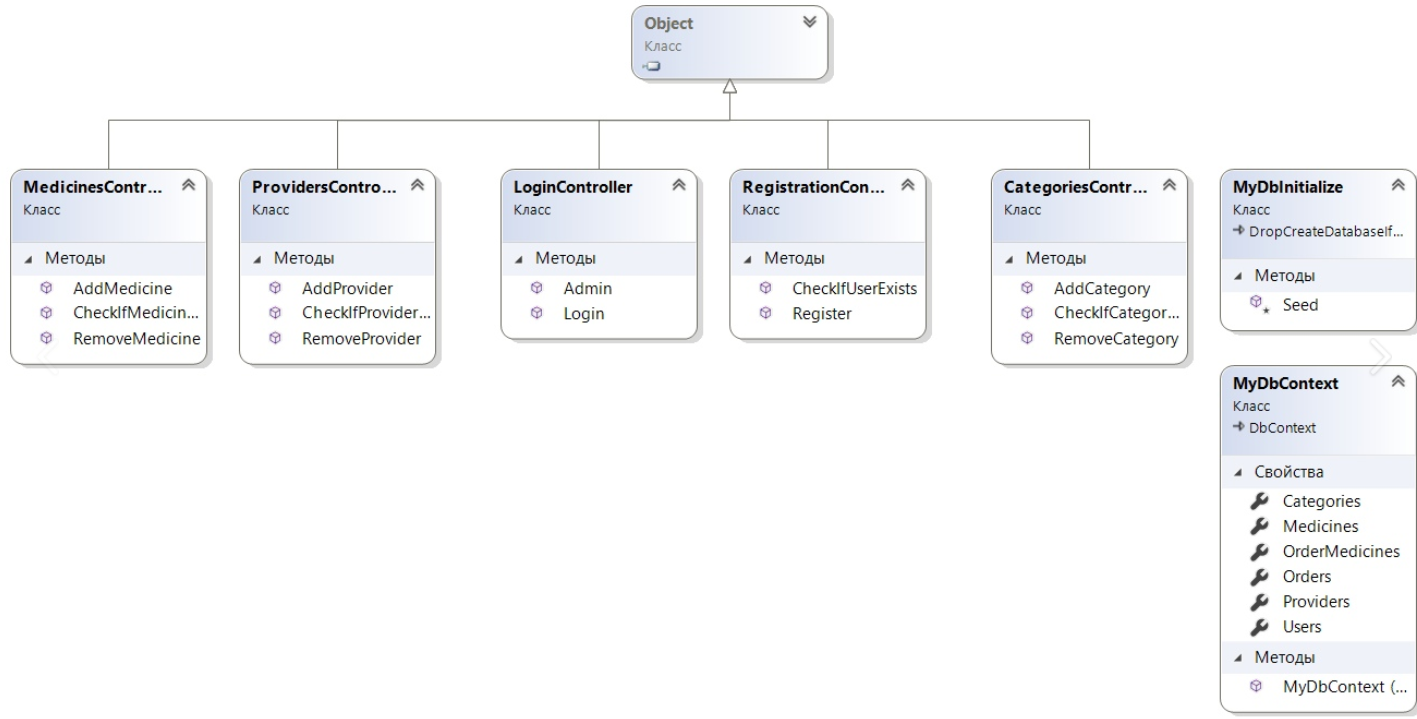
Таблица Providers состоит из 3 столбцов:

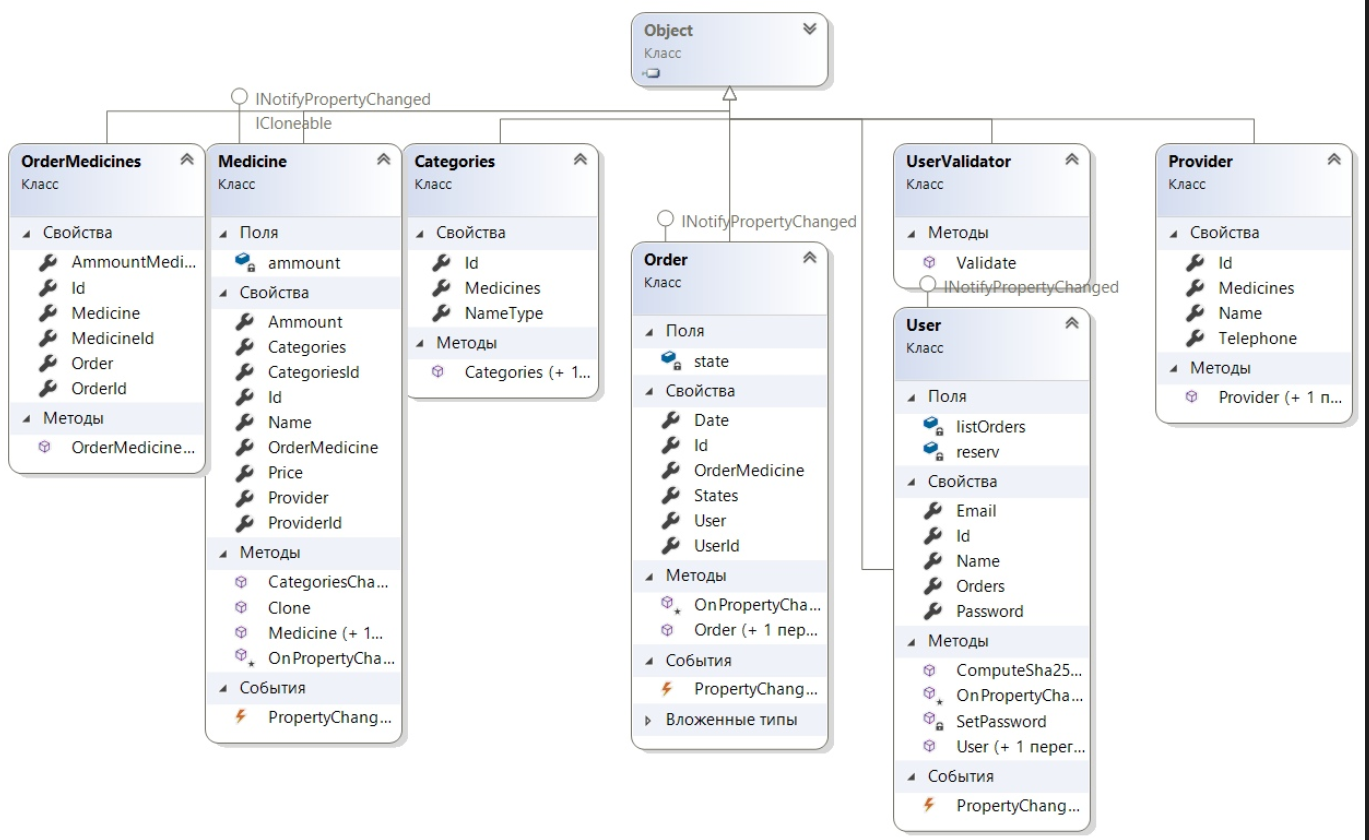
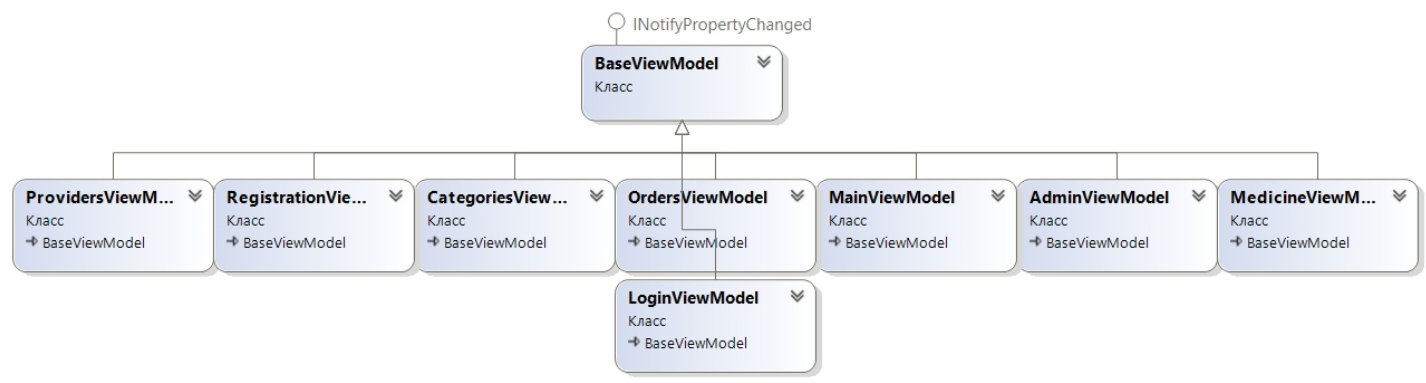
* Id – хранит идентификатор поставщика
* Name – хранит имя поставщика
* Telephone – хранит номер телефона поставщика

### 3. Разработка программного средства

Следующим этапом в проектировании является составление диаграмм, демонстрирующих возможности, классы, последовательности.

В следующих диаграммах рассмотрим общую концепцию классов в приложении и их взаимодействие между собой.





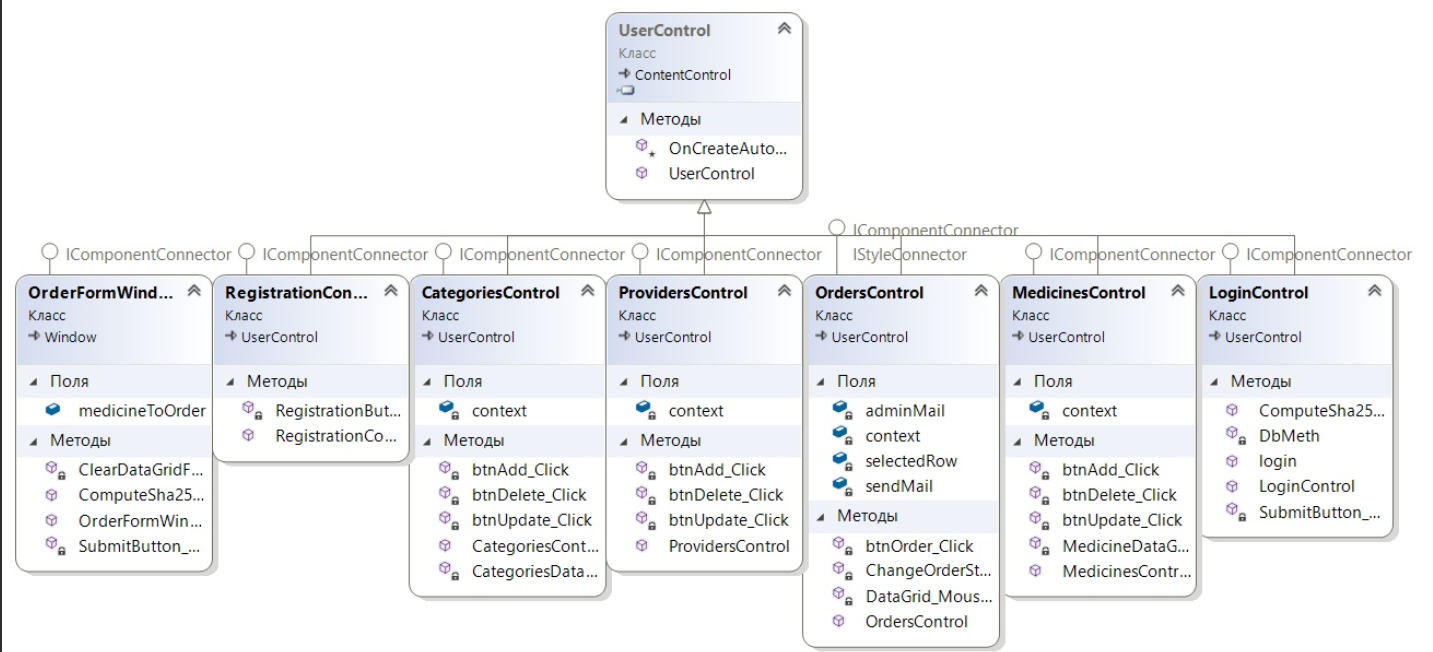


Рис.11 – Диаграммы классов

В качестве реализации технологии был применен подход Code First. Были созданы классы User.cs, Order.cs, Medicine.cs, OrderMedicines.cs, Categories.cs, Provider.cs, описывающие сущности базы данных.

Для реализации регистрации и авторизации были написаны классы RegistrationController.cs и LoginController.cs.

Класс регистрации содержит в себе 2 метода: CheckIfUserExists – метод для проверки наличия введенной учетной записи в базе данных, Register – метод для добавления учетной записи.

Класс авторизации содержит в себе методы для входа в приложения пользователя и администратора – Login и Admin.

Класс CategoriesController содержит в себе 3 метода: CheckIfCategoriesExists – метод проверки существования заданной категории в базе данных, AddCategories – для добавления новой категории в базу данных, RemoveCategories – для удаления существующей категории.

Класс MedicinesController содержит в себе 3 метода: CheckIfMedicinesExists – метод проверки существования заданного препарата в базе данных, AddMedicines – для добавления нового препарата в базу данных, RemoveMedicines – для удаления существующего препарата.

Класс ProvidersController содержит в себе 3 метода: CheckIfProvidersExists – метод проверки существования заданного поставщика в базе данных, AddProviders – для добавления нового поставщика в базу данных, RemoveProviders – для удаления существующего поставщика.

В проекте есть 10 классов окон, 7 из которых являются UserControl. Благодаря использованию возможностей паттерна MVVM, в проекте имеется функция переключения между окнами. AdminViewCommand.cs – управление переключениями между UserControl администратора. UpdateViewCommand.cs – управление переключениями между UserControl регистрации и авторизации

В окне LoginWindow пользователь может выбирать окна регистрации и авторизации (RegistrationControl и LoginControl). В окне AdminWindow, предназначенном для работы администратора, имеется переключение окон CategoriesControl, ProviderControl, OrderControl, MedicinesControl которые позволяют администратору осуществлять добавление, удаление, редактирование в базе данных, а также просматривать статус заказа и оповещать пользователя о выполнении заказа путем отправки уведомления на почту пользователя.

MainWindow – главное окно для работы пользователя в приложении. В окне пользователь может осуществлять поиск по заданным параметрам поиска, просматривать информацию о найденных товарах, добавлять необходимые препараты в корзину. В окне имеется кнопка для перехода к оформлению заказа из корзины.

OrderFormWindow – окно для ввода дополнительных данных для оформления заказа.

Реализация алгоритма отправления сообщения на почту показана в виде блок-схемы в Приложении А.

Ниже приведена диаграмма последовательности при авторизации пользователя.

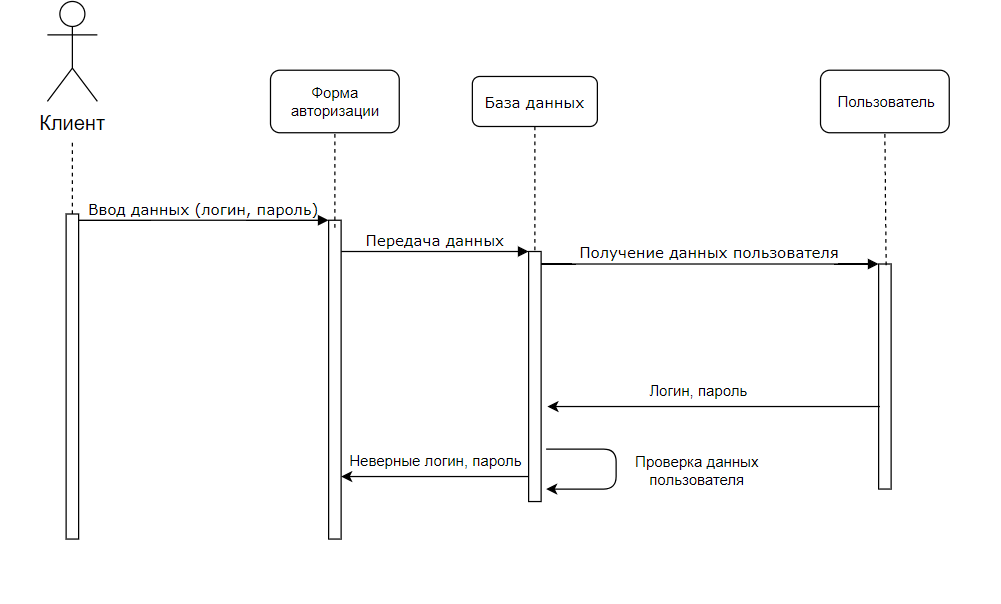


Рис. 12– Диаграмма последовательности при авторизации пользователя

На данной диаграмме клиент, чтобы войти в аккаунт, должен ввести свои данные: логин и пароль. После этого, программа начинает проверку с наличием соответствующих данных в своей базе данных, и, в случае совпадения, пользователь может получить доступ к своему аккаунту. В противном случае, если совпадений не найдено, программа оповестит пользователя о том, что необходимо проверить свои введенные данные.

Хотелось бы отметить действие паттерна проектирования MVVM при переходе по разным UserControl в одном окне.

Данный паттерн позволяет отделить работу с данными от представления. Модель данных связывает модель и представление через механизм привязки данных. Представлена она на следующем скриншоте:



Рис. 13 – Класс MainViewModel

Данный класс наследуется от BaseViewModel, который в свою очередь реализует интерфейс INotifyPropertyChanged, что позволяет оповещать клиента об изменении свойств:

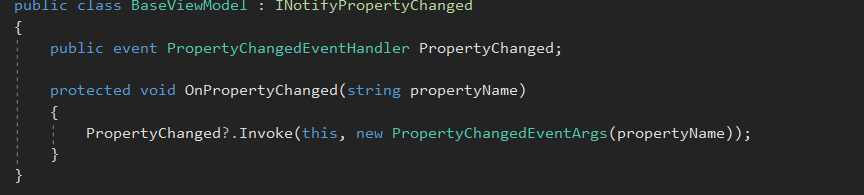


Рис. 14 – класс BaseViewModel

В XAML разметке окна, в котором будет происходить перемещение по двум окнам UserControl, прописаны шаблоны данных, что будут подставляться в StackPanel при нажатие на определенную кнопку.

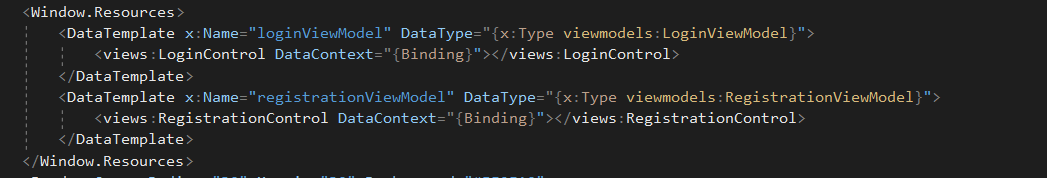


Рис.15 – LoginWindow

Нажатие на кнопку вызывает команду UpdateViewCommand, и в зависимости отпередеваемого параметра создается соответствующая ViewModel.

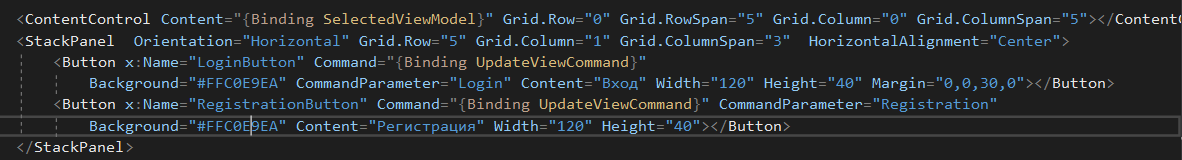


Рис.16 – Command Login

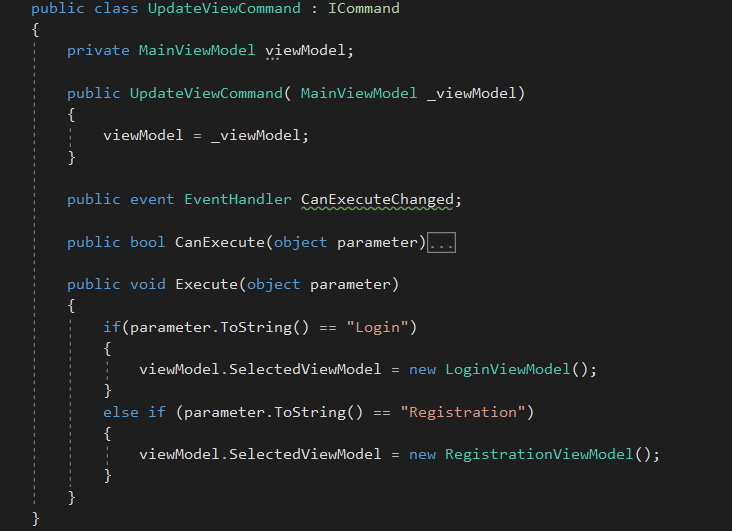


Рис.17 - UpdateViewCommand

### 4. Тестирование, проверка работоспособности и анализ полученных результатов

При разработке приложение было сконструировано так, чтобы все данные, с которыми работает пользователь, вводились корректно.

На рисунке 11 продемонстрировано поведение формы при введении неправильного пароля или логина.

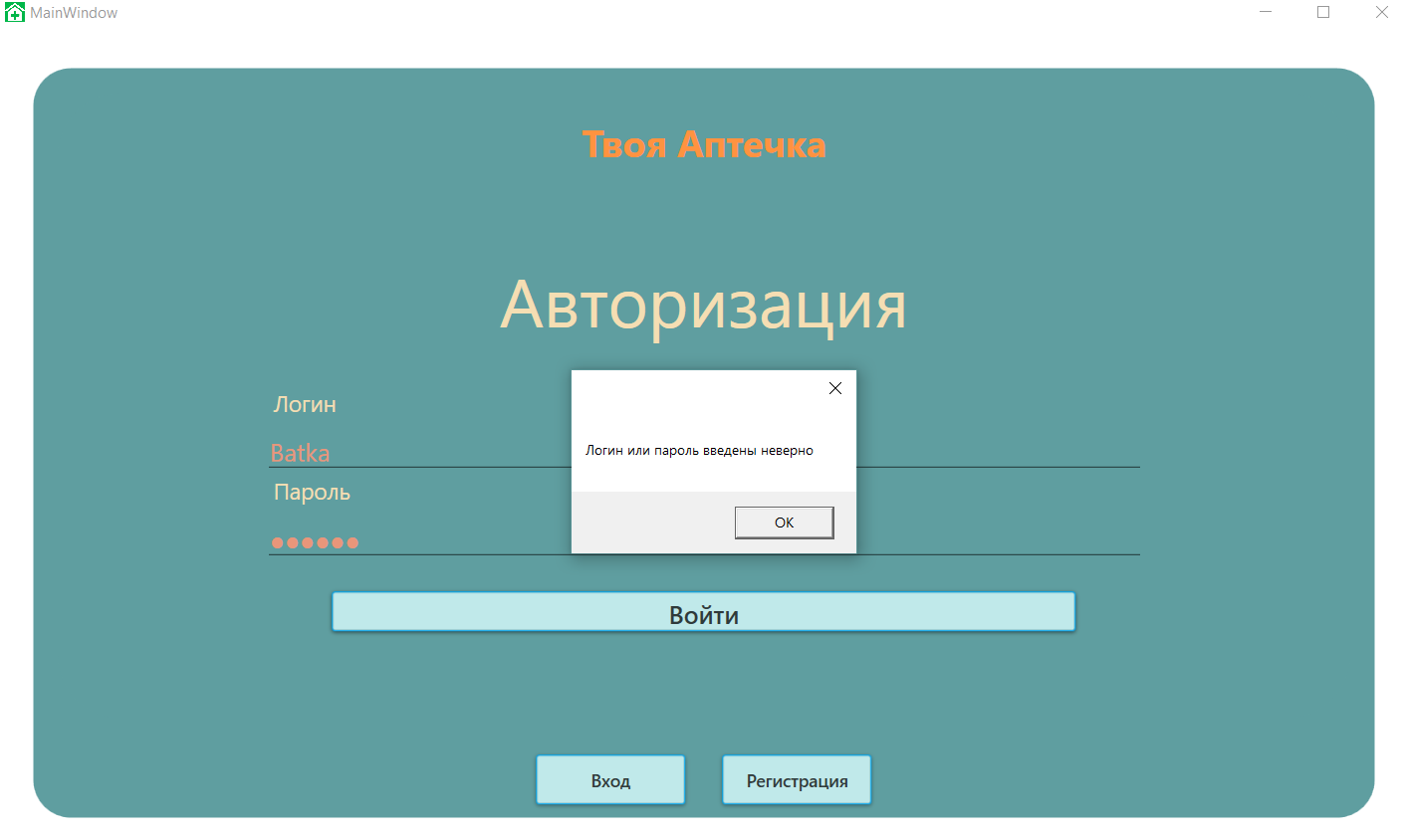


Рис.18 – Форма входа

Следующая ситуация возникает при несопадении введенных данных в полях пароля и повторного пароля либо, при слишком коротком пароле:

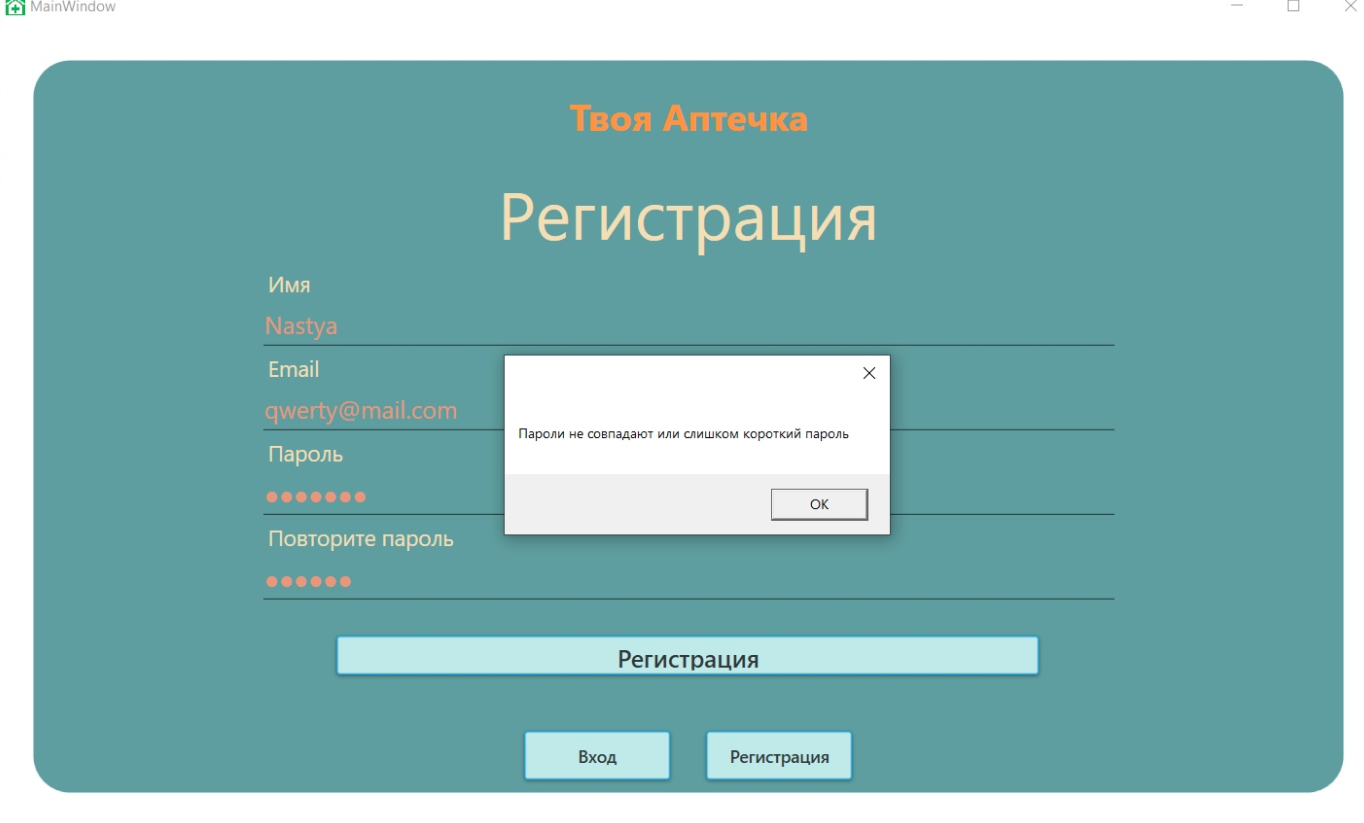


Рис.18 – Форма регистрации

При введении в поле пароля или логина сиволов, неудовлетворящих условию,заданному в регулярном выражении, выдается следующее предупреждение:

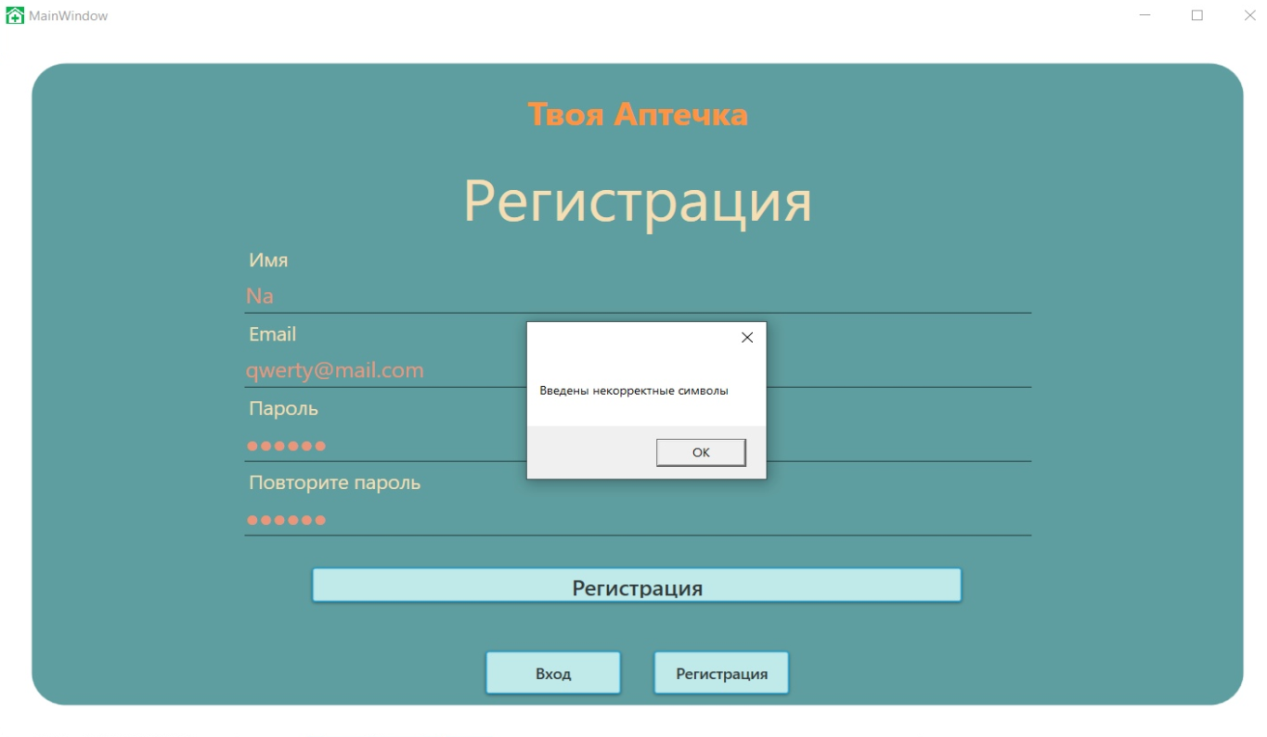


Рис.19 – Проверка на ввод определенных символов

При добавлении существующей категории в учетной записи администратора выходит следующее:

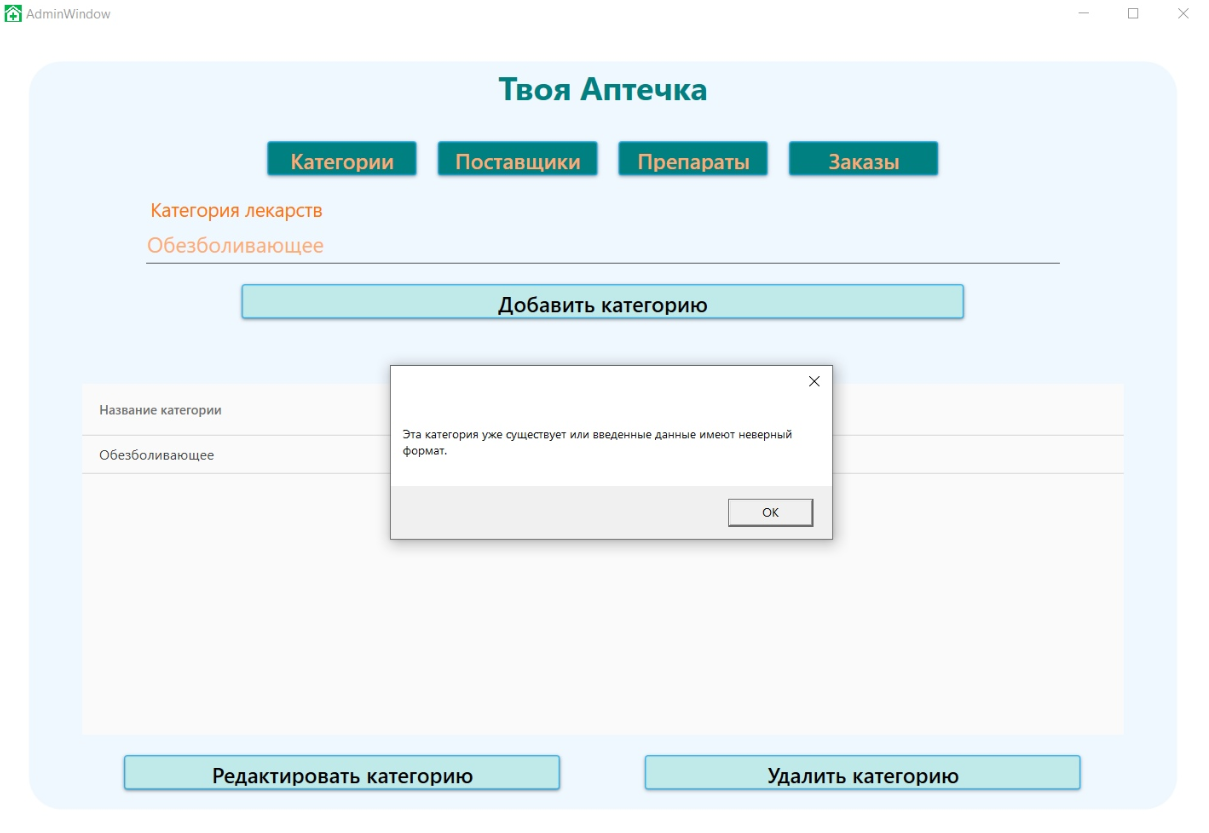


Рис.20 – Проверка на существование этого же элемента

Появление данного сообщения провоцирует несуществующий адрес электронной почты:

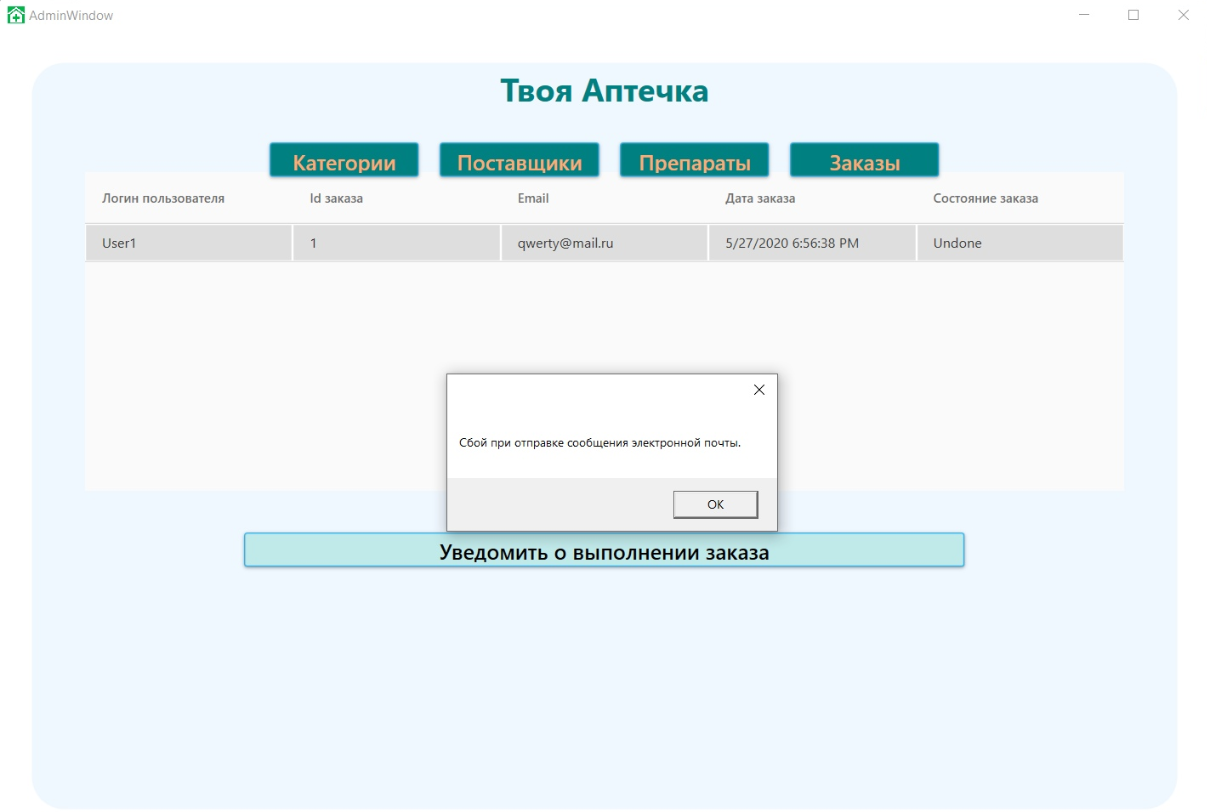


Рис.21 – Отправление уведомлений

При добавлении нового товара возможна ситуация, когда администратор пытается добавить его, не заполнив все поля или введя некорректные данные. Обработка данного исключения продемонстрирована на рисунке ниже:

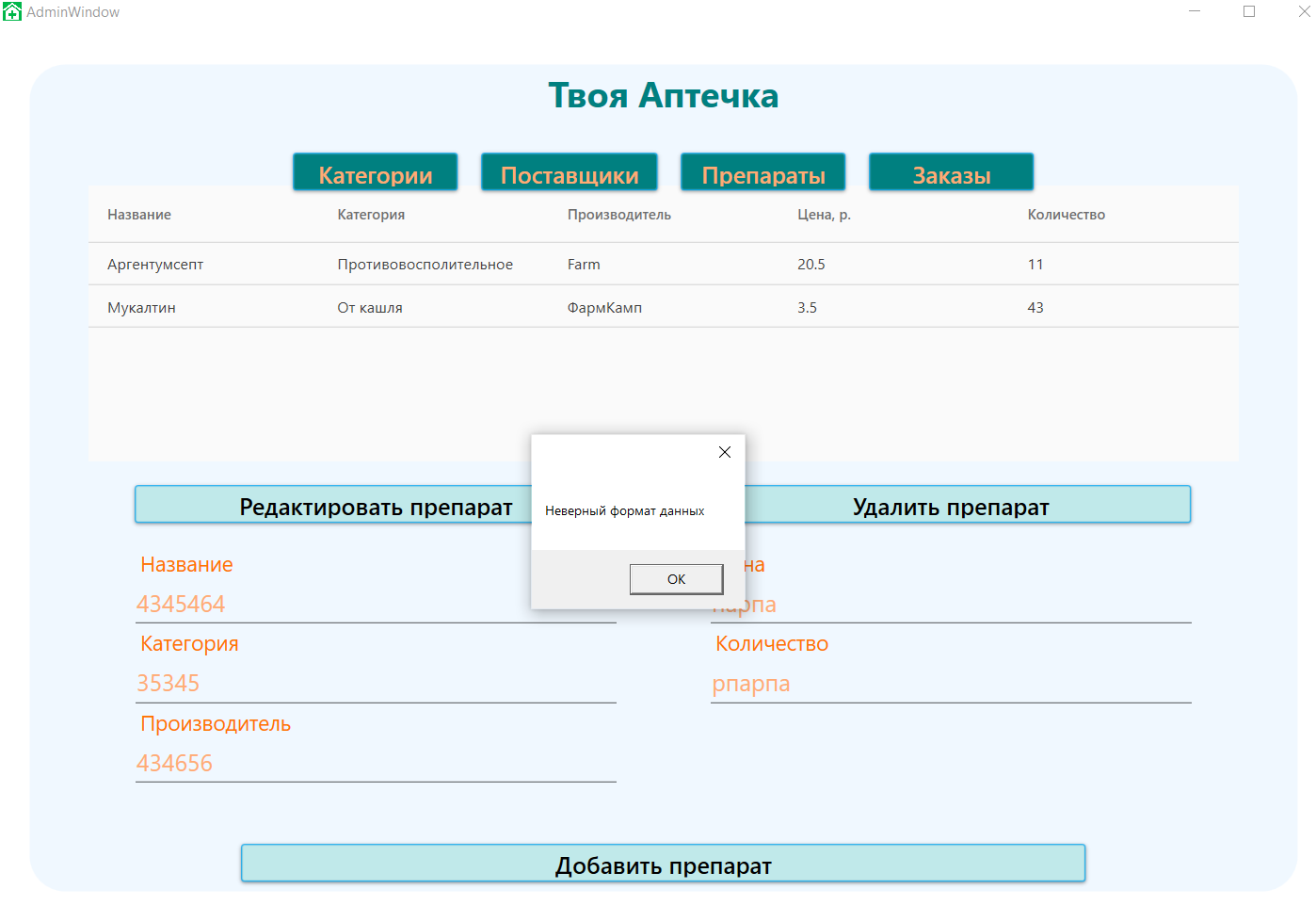


Рис.23– Администрирование препаратов

При поиске необходимого товара возможна ситуация, когда пользователь не заполняет поле для поиска или не задает фильтр поиска. Обработка данного исключения продемонстрирована на рисунке 17.

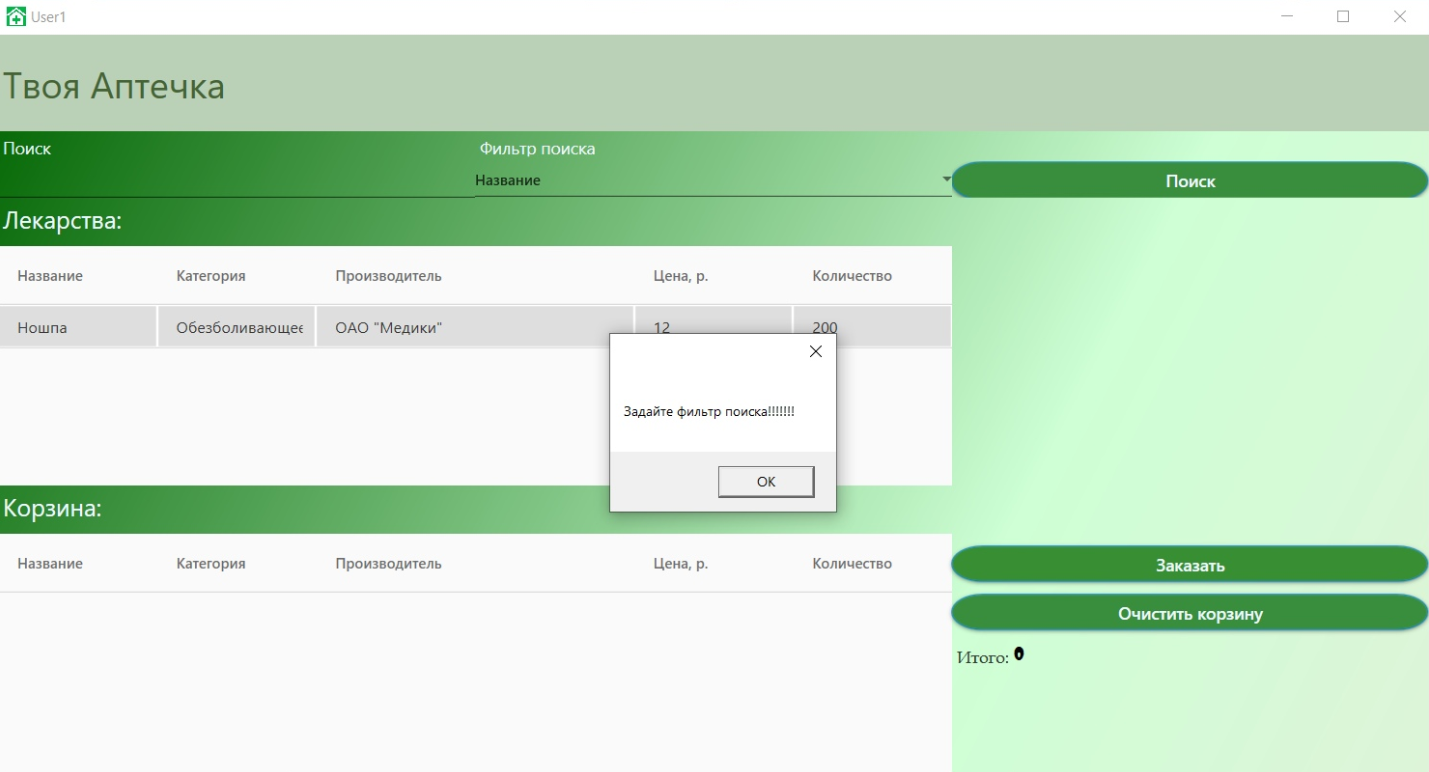


Рис.24 – Поиск препарата

При заказе необходимого препарата возможна ситуация, когда пользователь пытается заказать товар, не добавив ничего в корзину. Обработка данного исключения продемонстрирована на рисунке внизу:

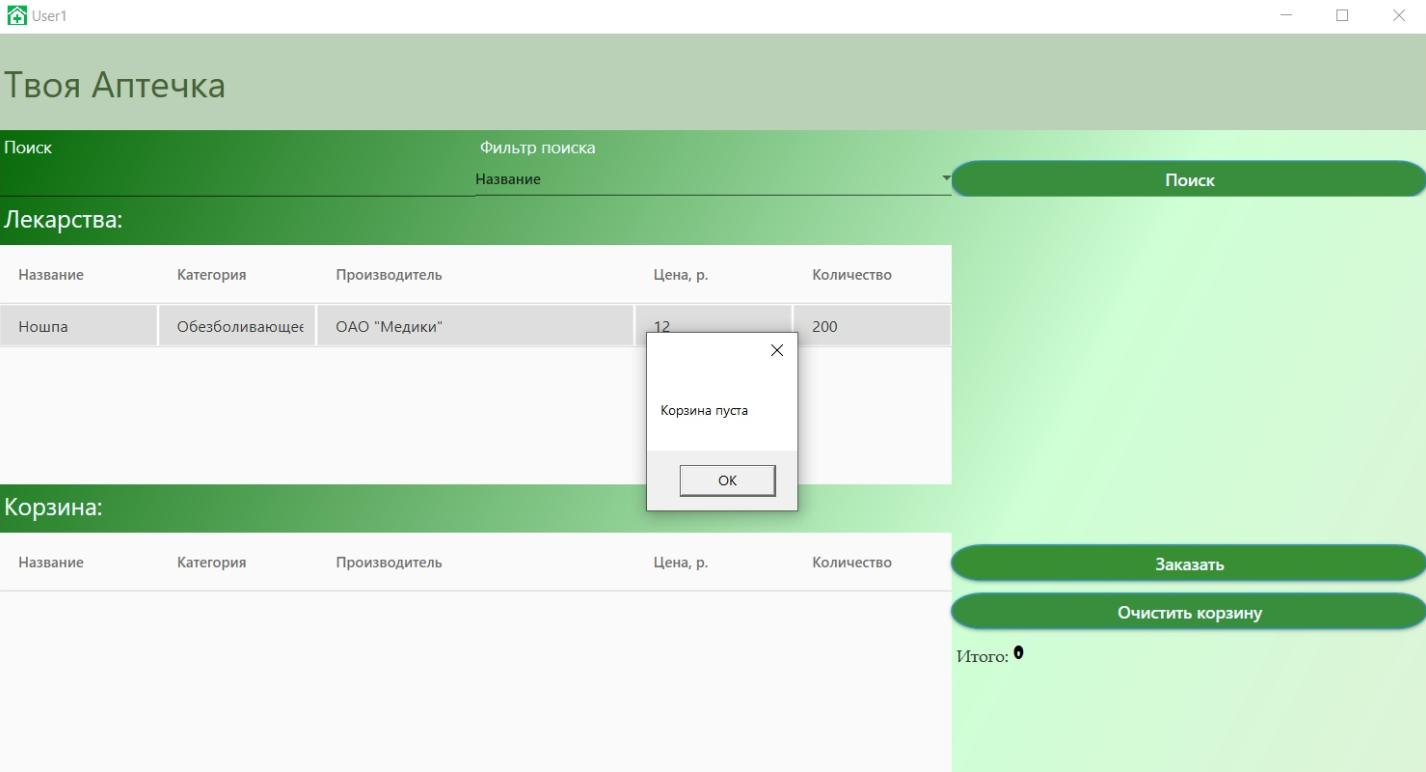


Рис.25 – Оформление заказа

При очистке пустой корзины пользователь получает следующую обработку исключения на следующем рисунке



Рис.26 – Очистка корзины

### 5. Руководство по установке и использованию

**5.1 Назначение программы**

Программа предназначена для предоставления возможности удаленного оформления заказа препаратов и их поиска

Программа предназначена для эксплуатации на ПК, установленном в офисе или в домашних условиях, ноутбуках и нетбуках.

Программа поставляется в виде папки с файлами содержащей файл исполняемого кода(.ехе-файлов) и связанный с ней необходимых файлов на компьютер. Единственным требованием для запуска будет являться наличие Microsoft SQL Server.

**5.2 Интерфейс программы**

Данное приложение максимально упрощено, не требует от пользователя каких-либо знаний в конкретной области. Дизайн приложения сделан максимально просто и комфортно для пользователя. В любых местах, где пользователь может допустить ошибку, приведены специальные инструкции, в виде всплывающих окон с предупреждениями о некорректных действиях со стороны пользователя.

Для вход в программу, изначально необходимо зарегистрироваться, пройдя валидацию.

### ЗАКЛЮЧЕНИЕ

В процессе выполнения курсового проекта была достигнута поставленная цель по созданию программного средства для поиска и заказа лекарственных препаратов. При разработке были выполнены все пункты из указанного списка предполагаемого основного функционала приложения.

В программном средстве были реализованы следующие функции:

* Возможность добавления, удаления и редактирования товаров в базе данных;
* Оповещать пользователей о выполнении заказа;
* Совершать поиск по товарам;
* Выполнять регистрацию и авторизацию;
* Осуществлять заказ;

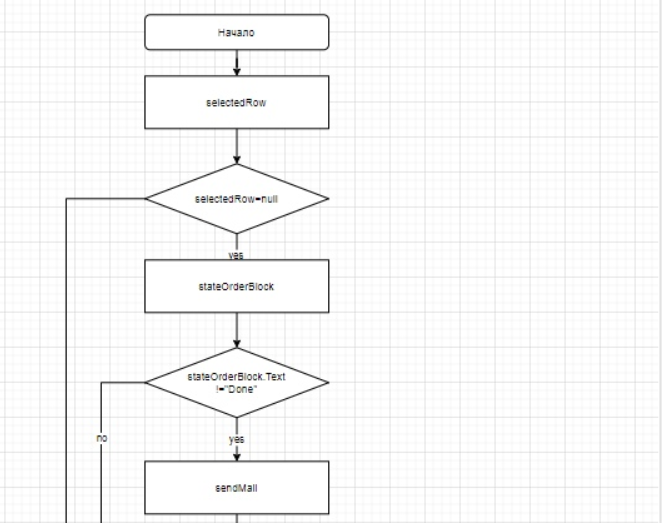
В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает, верно, а требования технического задания выполнены в полном объеме.

### СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Пацей, Н.В. Курс лекций по языку программирования С# / Н.В. Пацей. – Минск: БГТУ, 2016. – 175 с.
2. Пацей, Н.В. Технология разработки программного обеспечения / Н.В. Пацей. – Минск: БГТУ, 2016. – 129 с.
3. MSDN сеть разработчиков в Microsoft [Электронный ресурс]
4. METANIT.COM Сайт о программировании [Электронный ресурс]
5. ProfessorWeb .NET & Web Programming [Электронный ресурс]

### ПРИЛОЖЕНИЕ А

Алгоритм отправки уведомления на почту пользователя о выполненном заказе.





### ПРИЛОЖЕНИЕ Б

Листинги

**User.cs**

public class User: INotifyPropertyChanged

{

private List<Order> listOrders = new List<Order>();

List<Order> reserv = new List<Order>();

[Key]

public int Id { get; set; }

[Required]

[MinLength(4)]

[MaxLength(16)]

public string Name { get; set; }

[Required]

[MinLength(6)]

public string Password { get; set; }

private void SetPassword(string password)

{

Password = ComputeSha256Hash(password);

}

[Required]

[EmailAddress]

public string Email { get; set; }

public IList<Order> Orders{ get; set;}

public User()

{

Orders = new List<Order>();

}

public User(string name, string password, string email)

{

SetPassword(password);

Name = name;

Email = email;

Orders = new List<Order>();

}

static public string ComputeSha256Hash(string rawData)

{

using (SHA256 sha256Hash = SHA256.Create())

{

byte[] bytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(rawData));

StringBuilder builder = new StringBuilder();

for (int i = 0; i < bytes.Length; i++)

{

builder.Append(bytes[i].ToString("x2"));

}

return builder.ToString();

}

}

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void OnPropertyChanged(string propertyName = "")

{

PropertyChanged?.Invoke(this,new PropertyChangedEventArgs(propertyName));

}

}

**Medicine.cs**

public class Medicine:INotifyPropertyChanged, ICloneable

{

private int ammount;

[Key]

public int Id { get; set; }

[Required]

public string Name { get; set; }

public double Price { get; set; }

public int ProviderId { get; set; }

public Provider Provider { get; set; }

public int CategoriesId{ get; set; }

public Categories Categories{ get; set;}

public List<OrderMedicines> OrderMedicine { get; set; }

public int Ammount

{ get { return ammount; }

set

{

if (ammount == value)

return;

else

{

if (value >= 0 )

{

ammount = value;

OnPropertyChanged("Ammount");

}

}

}

}

public Medicine()

{

}

public Medicine(string name,string categoriesName,string providerName,double price,int ammountMed)

{

Name = name;

using(MyDbContext context = new MyDbContext())

{

try

{

Categories categoryFromDb = context.Categories.First(category => category.NameType == categoriesName);

this.CategoriesId = categoryFromDb.Id;

}

catch(Exception)

{

Categories categories = new Categories(categoriesName);

this.Categories = categories;

}

try

{

Provider providerFromDb = context.Providers.First(provider1 => provider1.Name == providerName);

this.ProviderId = providerFromDb.Id;

}

catch (Exception)

{

Provider providers = new Provider()

{

Name = providerName

};

this.Provider = providers;

}

}

Price = price;

Ammount = ammountMed;

}

public object Clone()

{

return new Medicine()

{

Provider = new Provider() { Id = this.Provider.Id, Medicines = this.Provider.Medicines,

Name = this.Provider.Name, Telephone = this.Provider.Telephone},

Categories = new Categories() { Id = this.Categories.Id, NameType = this.Categories.NameType },

Id = this.Id,

Price = this.Price,

Name = this.Name,

CategoriesId = this.CategoriesId,

Ammount = this.Ammount

};

}

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void OnPropertyChanged(string propertyName = "")

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

}

**OrderMedicines.cs**

public class OrderMedicines

{

[Key]

public int Id { get; set; }

public int OrderId { get; set; }

public Order Order { get; set; }

public int MedicineId { get; set; }

public Medicine Medicine { get; set; }

public int AmmountMedicine { get; set; }

public OrderMedicines()

{

}

public OrderMedicines(int orderId, int medicineId,int ammount)

{

OrderId = orderId;

MedicineId = medicineId;

AmmountMedicine = ammount;

}

}

**Order.cs**

public class Order:INotifyPropertyChanged

{

[Key]

public int Id { get; set; }

public int? UserId { get; set; }

[Required]

public DateTime Date { get; set; }

public User User { get; set; }

public List<OrderMedicines> OrderMedicine { get; set; }

public Order()

{

}

public Order(User user, DateTime dateTime)

{

User = user;

Date = dateTime;

}

public enum State

{

Done, Undone

}

State state = State.Undone;

public State States

{

get { return state; }

set

{

state = value;

OnPropertyChanged("States");

}

}

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void OnPropertyChanged(string propertyName = "")

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

}

**Categories.cs**

public class Categories

{

[Key]

public int Id { get; set; }

[Required]

public string NameType { get; set; }

public List<Medicine> Medicines

{

get; set;

}

public Categories()

{

}

public Categories(string nameType)

{

NameType = nameType;

}

}

**Provider.cs**

public class Provider

{

[Key]

public int Id { get; set; }

[Required]

public string Name { get; set; }

public string Telephone { get; set; }

public List<Medicine> Medicines{ get; set;}

public Provider()

{

}

public Provider(string name, string telephone)

{

Name = name;

Telephone = telephone;

}

}