МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Программной инженерии

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

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТА**

по дисциплине «Современные технологии программирования мобильных систем»

Тема «Программное средство для заказа сантехники на дом»

Исполнитель

студент 2 курса 8 группы Беринчик Владислав Андреевич

(Ф.И.О.)

Руководитель работы ассистент Панченко О.Л.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой ­

Председатель Пацей Н.В

(подпись)

Минск 2021

# **Содержание**

Содержание 2

ВВЕДЕНИЕ 3

1. Аналитический обзор литературы 4

2. Анализ требований к программному средству и разработка функциональных требований 6

3. Проектирование программного средства 7

4. Создание программного средства 11

5. Тестирование, проверка работоспособности и анализ полученных результатов 25

Заключение 33

Список использованных источников 34

Приложение А. UML-диаграммы проекта 35

Приложение B. Логическая структура базы данных 36

Приложение С. Листинги программы 37

# **ВВЕДЕНИЕ**

Важной частью жизни людей всегда было удобство и экономия своего времени. В современном мире одним из основных способов сэкономить время является заказ услуг и товаров на дом.

Покупка каких-либо товаров постепенно все больше и больше перемещается из оффлайн в онлайн-пространство, и это открывает альтернативность как для людей, так и для компаний.

Многим людям гораздо удобнее и практичнее выбирать товар онлайн приложениях или на сайтах.

Чаще всего это удобнее всего делать через специальные приложения, которые помогают компаниям расширить свою сеть, а покупателям – экономить время. С помощью разрабатываемого программного средства можно будет выбирать и оформлять заказ в любое удобное вам время не появляясь лично в магазине.

Приложение имеет 2 типа пользователей: администратор и пользователь. Они имеют разные полномочия. Пользователи могут следить за каталогом и просматривать информацию о товарах, и, непосредственно формировать свой заказ. Администратор имеет полные права. Он может удалять любых пользователей, просматривать информацию о пользователях, а также следит за каталогом.

Также реализована авторизация всех пользователей. Это значит, что перед тем, как сформировать заказ нужно пройти регистрацию. Хранение всех пользователей будет осуществлено в базе данных, и при авторизации или регистрации будет непосредственное обращение к ней. Также вся информация о заказах, каталоге будет хранится в базе данных.

Реализована возможность редактирования аккаунта.

# **Аналитический обзор литературы**

В данном разделе рассматриваются программные средства с аналогичной тематикой с целью выделения преимуществ и недостатков. Разработка своего программного продукта будет основываться на их анализе.

**santego.by**

В качестве примера я выбрал вот этот сайт. На нем отсутствует регистрация и авторизация, что я считаю недостатком.

На сайте доступен обширный каталог со всей возможной сантехникой, что несомненно является достоинством (рис.1).

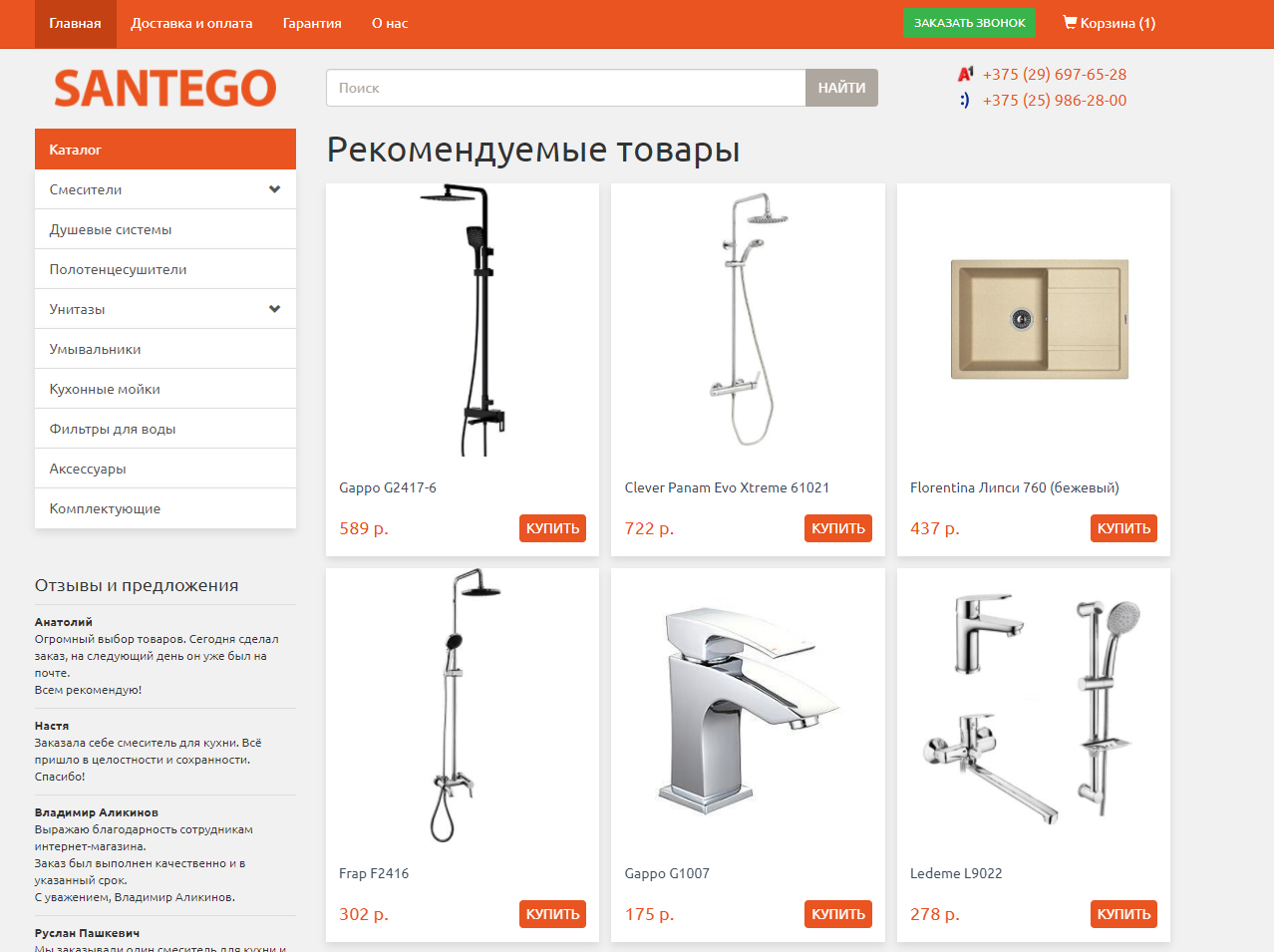


Рис.1.1 – Каталог на santego.by

Как только пользователь выбрал и намерен купить товар, он нажимает кнопку “купить” и товар переходит в корзину (рис.2).

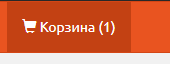


Рис.1.2 – Добавление товара в корзину

Далее нажатием на “корзину” открывается форма где нужно заполнить данные для заказа (рис.3).

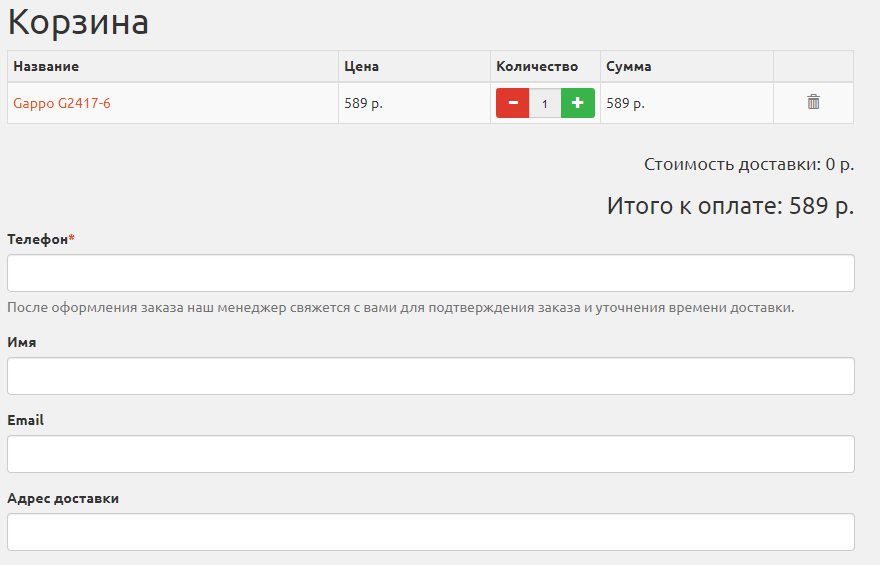


Рис.1.3 – Корзина пользователя

Далее по окончанию заполнения данных формы, пользователю следует нажать на кнопку “Оформить заказ”, после чего вылезает сообщение: “Мы с вами свяжемся!”.

# **Анализ требований к программному средству и разработка функциональных требований**

В данном разделе формулируются и описываются функциональные требования для проектирования программного средства.

При разработке приложения используются:

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

В качестве разрабатываемого функционала выделяем следующие основные функции приложения для обычного пользователя:

* Просмотр каталога.
* Просмотр подробной информации о товаре из каталога.
* Просмотр своего заказа.
* Удаление товаров из заказа.
* Поиск по критериям.
* Фильтрация в алфавитном порядке.
* Выбор категории поиска.
* Регистрация в приложении.
* Авторизация в приложении.
* Просматривать и редактировать информацию своего аккаунта

К разрабатываемому функционалу также относятся следующие функции суперпользователя (admin):

* Добавления товара в БД.
* Удаление товара из БД.
* Просмотр информации о товарах из каталога.
* Удаление пользователей из БД.
* Изменение привелегий пользователей.
* Просматривать и редактировать информацию своего аккаунта
* Просмотр заказов пользователей
* Удаление заказов пользоватлей
* Регистрация в приложении.
* Авторизация в приложении.

# **Проектирование программного средства**

Программное средство разделяется на две части по виду пользователя: часть для обычных пользователей и часть для суперпользователей.

Взаимодействие приложения с обычным пользователем представлено следующей блок-схемой (Рис. 4):

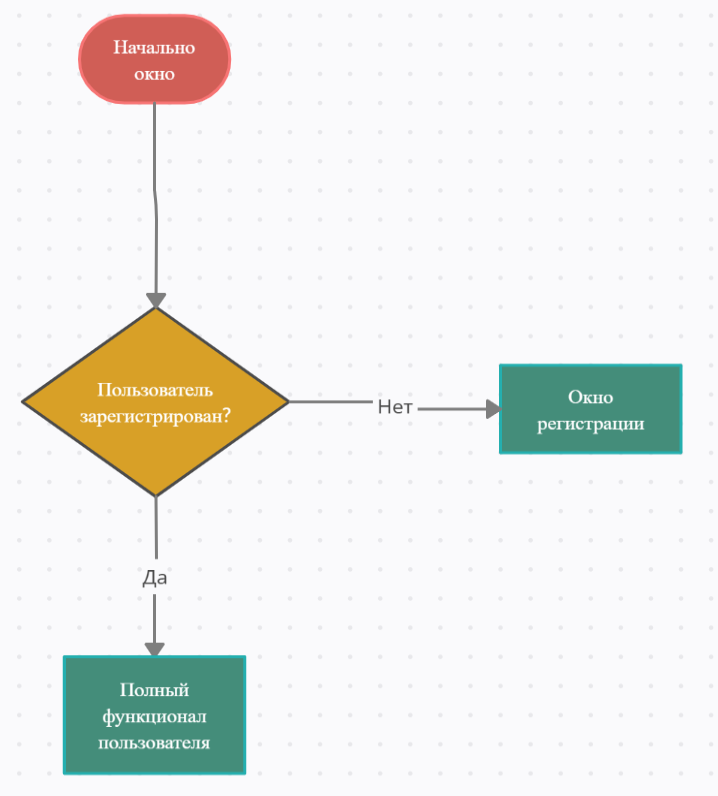


Рис. 3.1 – Схема взаимодействия пользователя с приложением.

Взаимодействие приложения с суперпользователем представлено следующей блок-схемой (Рис.5):

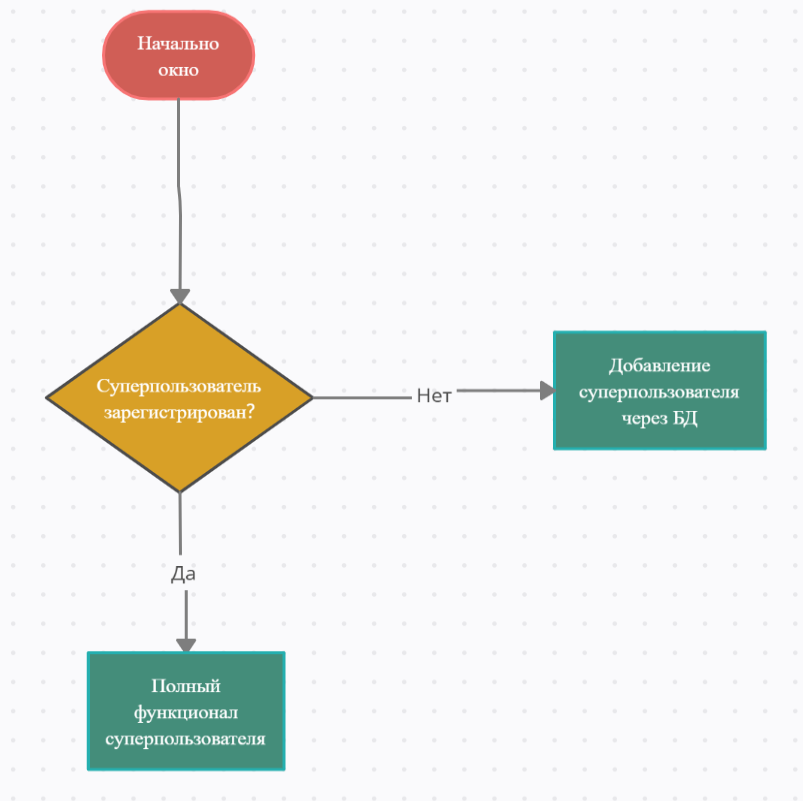


Рис. 3.2 – Схема взаимодействия суперпользователя с приложением.

Начало работы с приложением для пользователя заключается в возможности выбора либо зайти под своими логином и паролем, если он зарегистрирован в системе, либо зарегистрировать аккаунт в соответствующем окне. Доступа без регистрации не предоставляется. Суперпользователи имеют доступ к большему функционалу по сравнению с другими. Администратор либо заранее зарегистрирован в БД, либо добавляется путем регистрации обычного пользователя и сменой его привилегии.

Для реализации поставленной задачи проектирования программного средства была создана база данных MyShop посредством реализации подхода Code First технологии Entity Framework. Для работы с БД использовалась система управления реляционными базами данных Microsoft SQL Server.

Необходимая база данных состоит из следующих таблиц (Рис. 6):

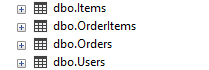


Рис. 3.3 – Таблицы базы данных для работы с приложением.

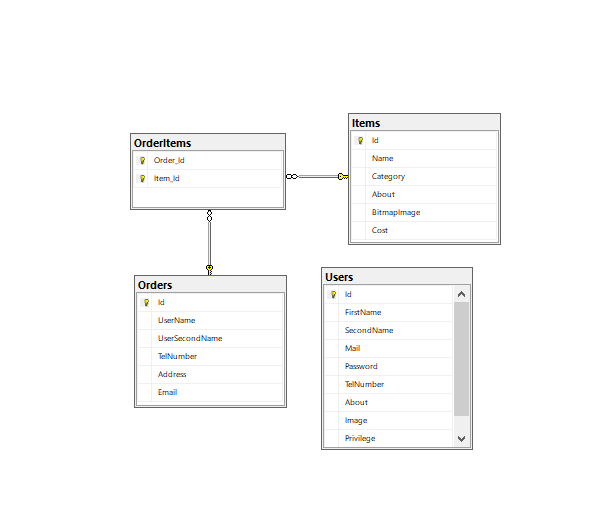


Рис. 3.4 – Диаграмма базы данных MyShop.

Таблица Users. Содержит данные о пользователях.

Таблица Orders. Хранит данные о заказах.

Таблица OrderItems. Содержит данные о заказе и заказанном товаре.

Таблица Items. Содержит данные о товарах.

В приложении к пояснительной записке представлены диаграммы классов для приложения, описывающие отношения между классами и их взаимодействие между собой. На диаграмме показано, что основным классом в программе выступает класс Window. В программе реализован паттерн MVVM для разделения визуальной и логической частей приложения, поэтому классы моделей представления наследуются от класса ViewModelBase, который реализует интерфейс INotifyPropertyChanged.

На рисунке 8 представлена диаграмма возможностей для приложения:

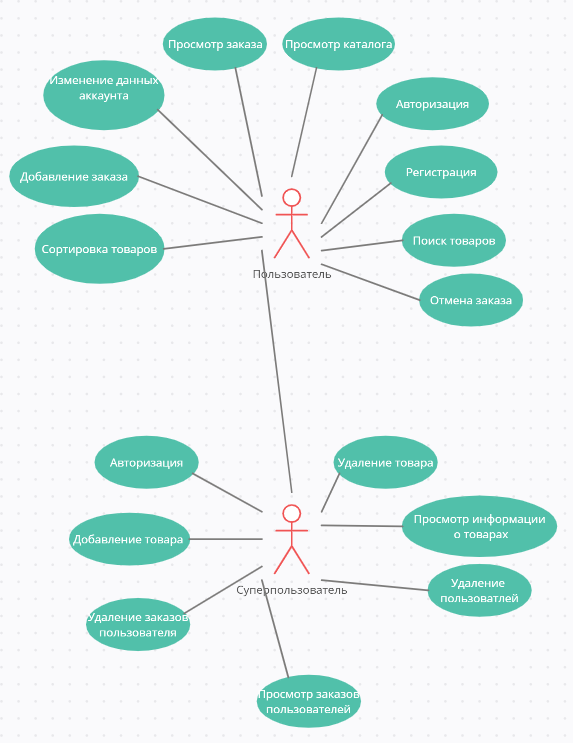


Рис. 3.4 – Диаграмма возможностей.

# **Создание программного средства**

В процессе создания приложения был реализован архитектурный паттерн MVVM, который позволяет отделить логику от части представления, задаёт общую архитектуру приложения. MVVM состоит из трех компонентов: модели (Model), модели представления (ViewModel) и представления (View).

Модель описывает используемые в приложении данные. Модели могут содержать логику, непосредственно связанную этими данными, например, логику валидации свойств модели. В то же время модель не должна содержать никакой логики, связанной с отображением данных и взаимодействием с визуальными элементами управления.

View или представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Применительно к WPF представление - это код в xaml, который определяет интерфейс в виде кнопок, текстовых полей и прочих визуальных элементов.

ViewModel или модель представления связывает модель и представление через механизм привязки данных. Если в модели изменяются значения свойств, при реализации моделью интерфейса INotifyPropertyChanged автоматически идет изменение отображаемых данных в представлении, хотя напрямую модель и представление не связаны.

Решение CourseProject(WPF) представлено проектом, имеющим следующую структуру:

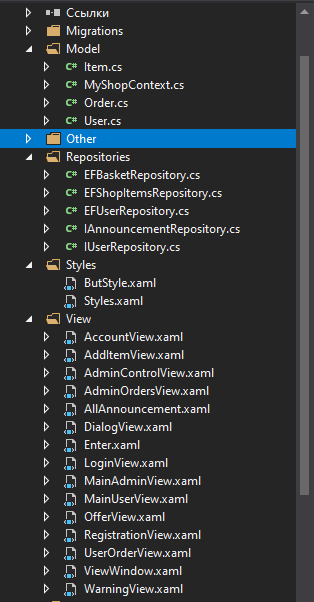
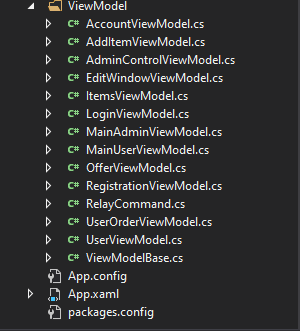
 

Рис. 4.1 – Структура проекта

Models – содержит классы моделей базы данных.

ViewModels – классы моделей представления, промежуточных сущностей между логической и визуальной частью.

Views – окна, страницы и юзер-контролы приложения.

Внешний вид представлен XAML-документами, находящимися в папке Views проекта. Переключение между ними обеспечивается с помощью кнопок, к которым привязаны команды, реализованные с помощью классов из папки Commands. Дизайн всех окон и элементов является адаптивным, так как для организации расположения элементов управления использовался Grid.

Для работы с базой данных используется технология Entity Framework с подходом Code First. Таким образом, все модели базы данных были написаны на C#, после этого был создан класс MyShopContext, представляющий базу данных. Далее в программе работали с этим классом и его свойствами для обращения к информации в базе данных.

Вход в приложение осуществляется с помощью почты и пароля, которые были введены во время регистрации:

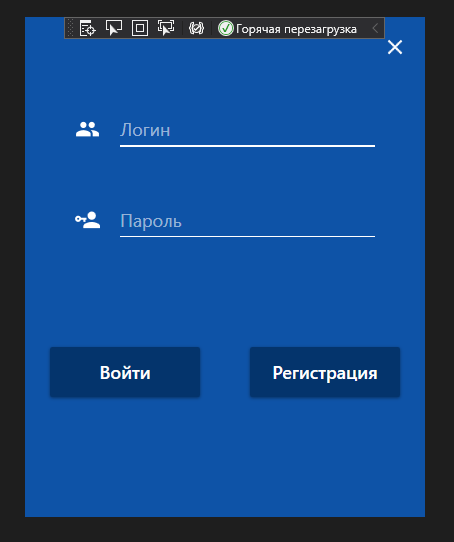


Рис. 4.2 – Окно авторизации пользователей.

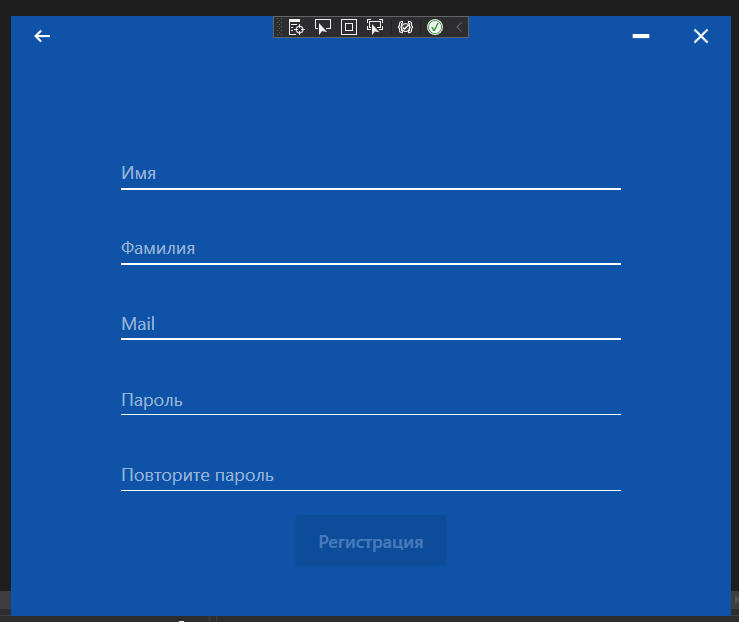


Рис. 4.3 – Окно регистрации пользователей.

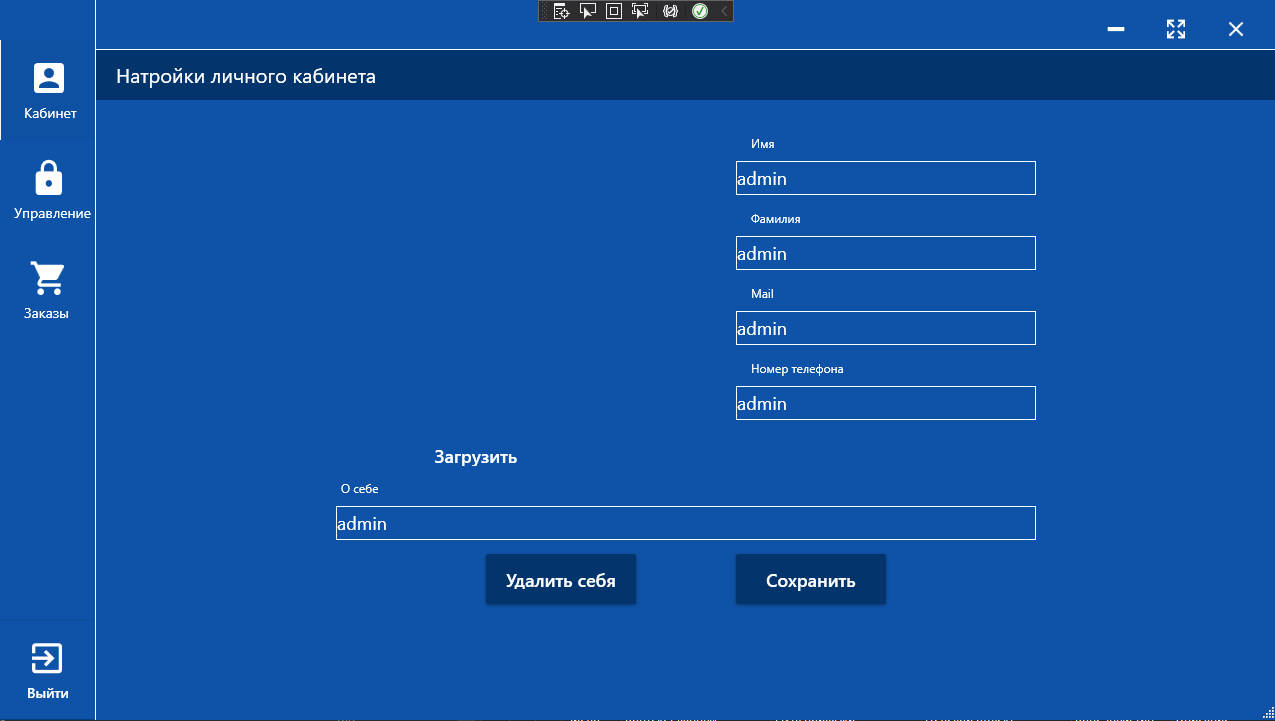


Рис. 4.4 – Окно суперпользователя.

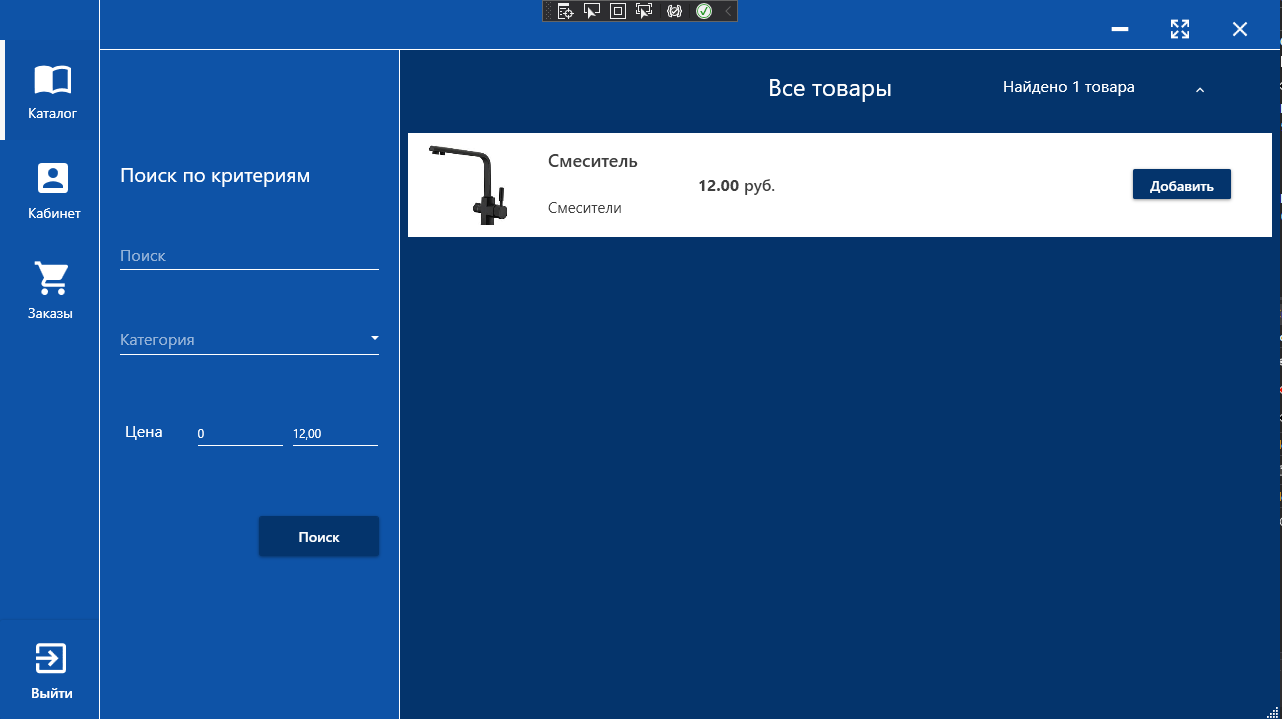


Рис. 4.5 – Окно пользователя.

В окне регистрации прописаны методы на добавление пользователей в таблицы, которые администратор может просмотреть и в дальнейшем изменить привилегию либо удалить. (Рис. 17).

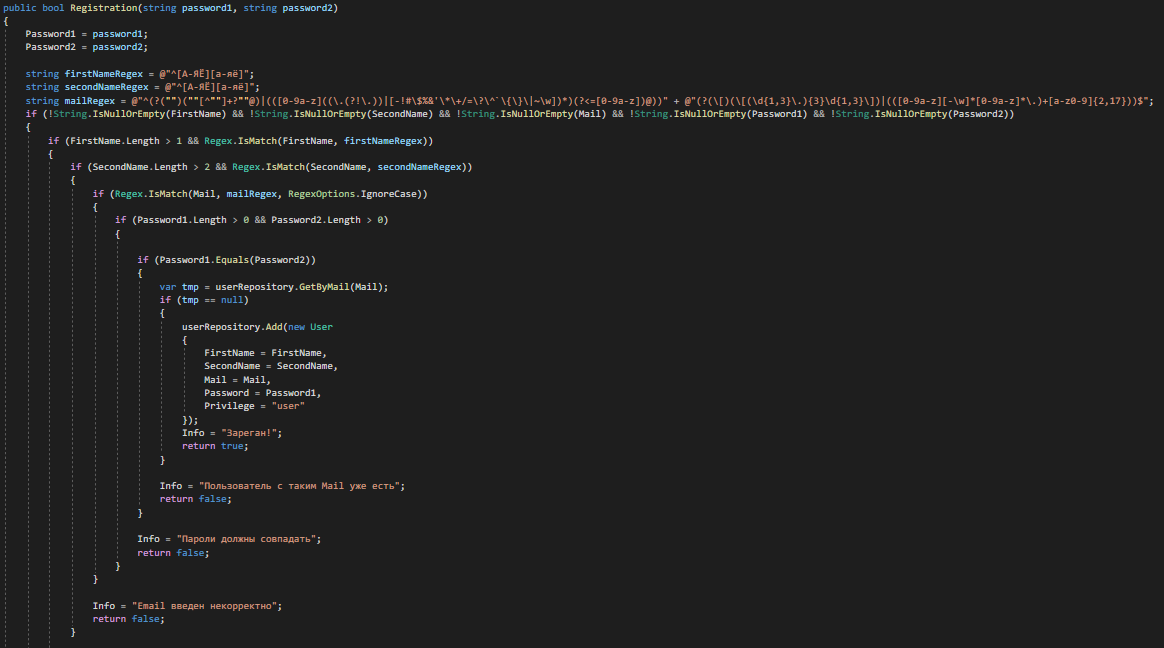


Рис. 4.6 – Реализация метода регистрации пользователя.



Рис. 4.7 – Реализация метода авторизации пользователя.

Методы перехода по страницам имеют практически одинаковую структуру:

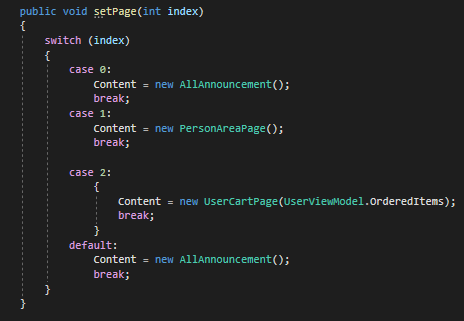


Рис. 4.7 – Реализация методов перехода у пользователя.

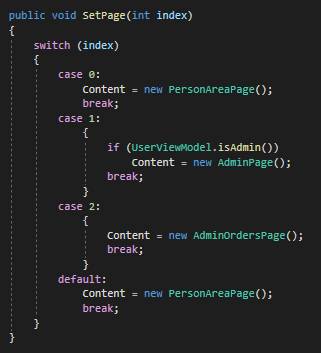


Рис. 4.8 – Реализация методов перехода у суперпользователя.

Когда суперпользователь захочет добавить товар в каталог ему нужно зайти в панель “управление” и сверху выбрать “товары”, далее нажать кнопку “добавить”, далее заполнить данные о товаре и нажать “добавить”:

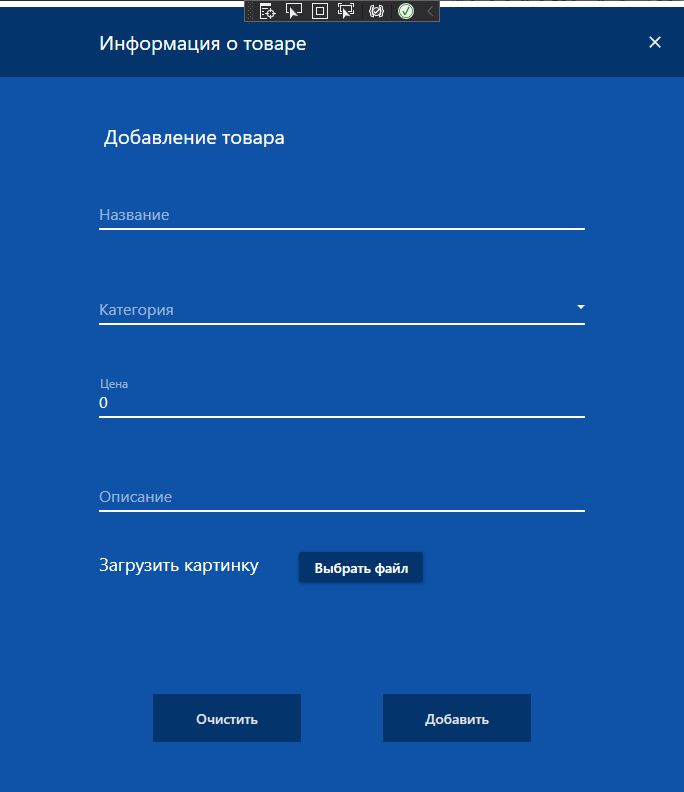


Рис. 4.9 – Реализация метода добавления товара у суперпользователя.

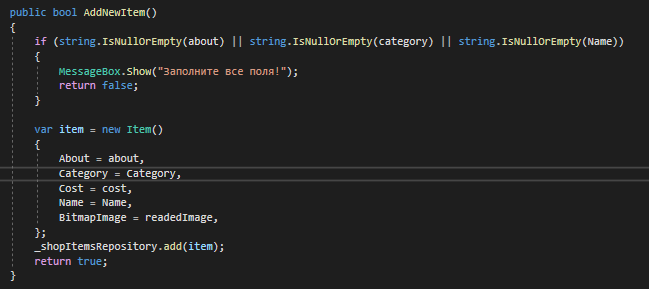


Рис. 4.10 – Реализация метода добавления товара у суперпользователя.

Если супер пользователь захочет удалить товар из каталога он должен зайти в панель “управление”, сверху выбрать “товары”, далее нажать кнопку “удалить” и кнопку “да”

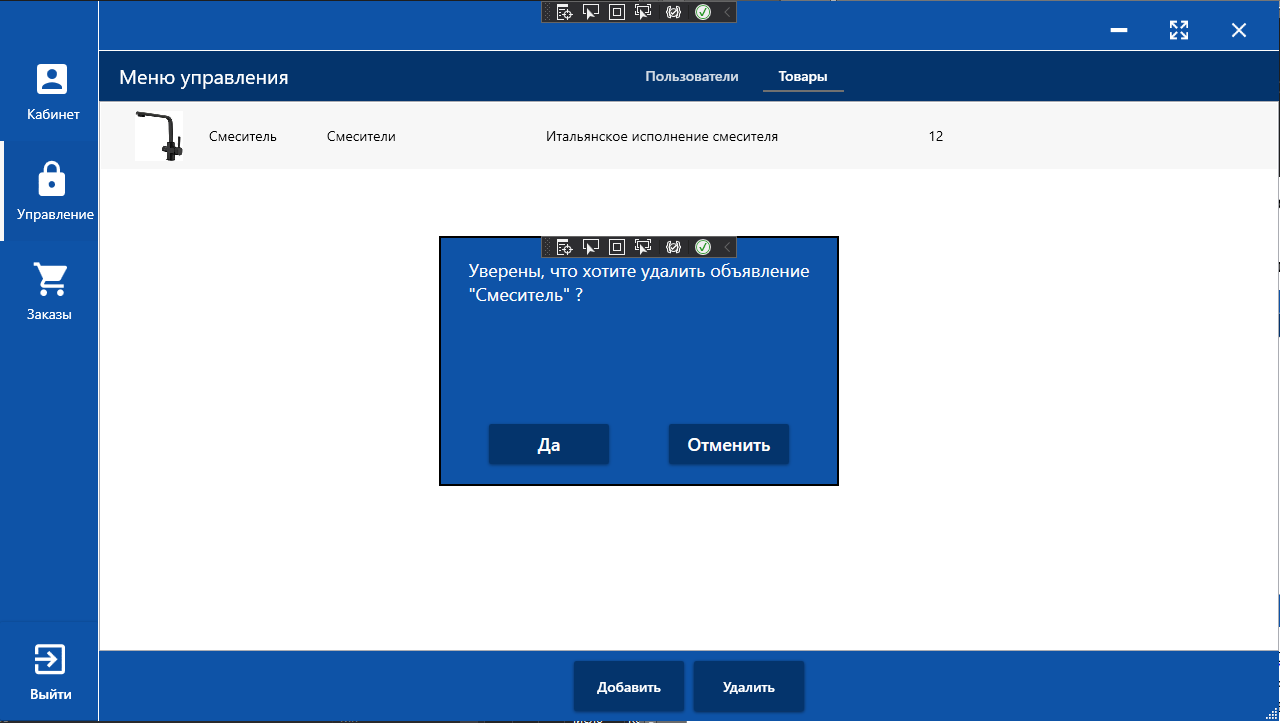


Рис. 4.11 – Реализация метода удаления товара у суперпользователя.

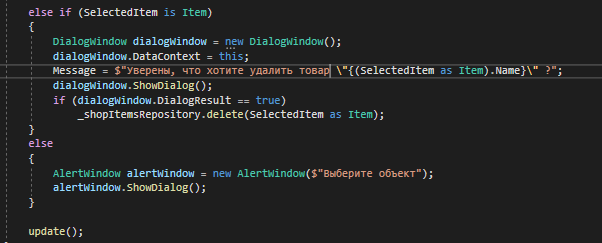


Рис. 4.12 – Реализация метода удаления товара у суперпользователя.

Если нужно изменить привилегию, суперпользователю нужно зайти в панель “управление”, сверху выбрать “пользователи”, далее нажать кнопку “изменить привилегию”:

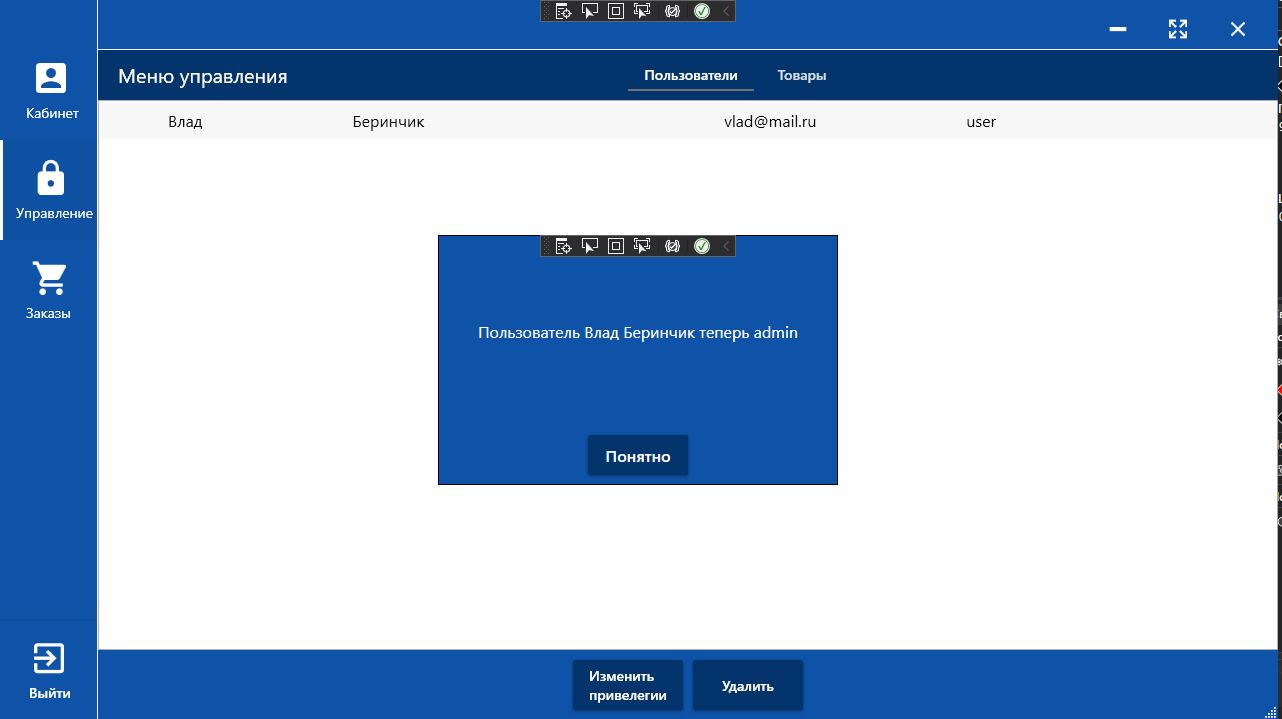


Рис. 4.13 – Окно изменения привилегии пользователя.

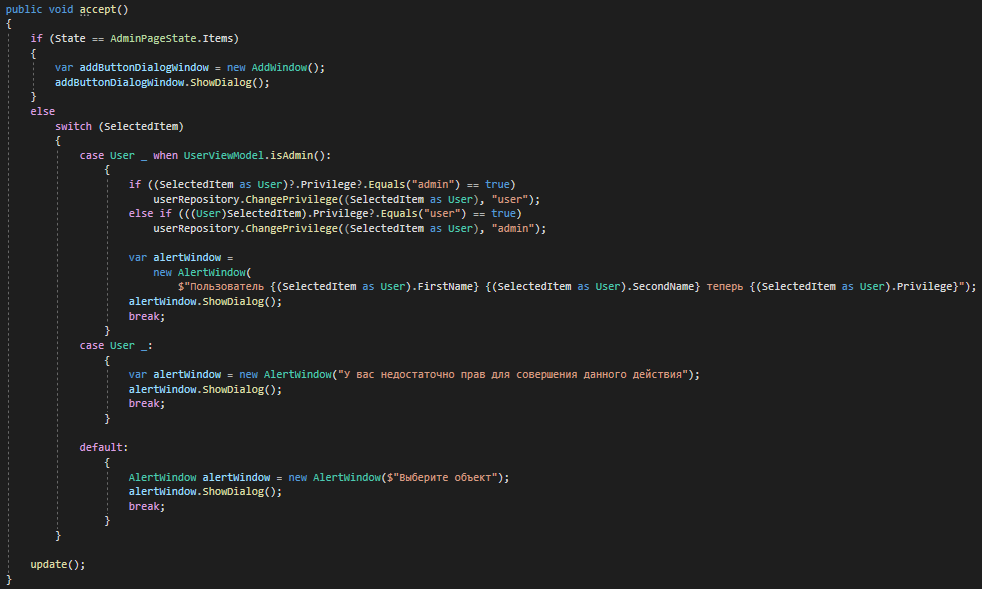


Рис. 4.14 – Реализация изменения привилегии у пользователя.

Если нужно удалить пользователя, суперпользователю нужно зайти в панель “управление”, сверху выбрать “пользователи”, далее нажать кнопку “удалить”:

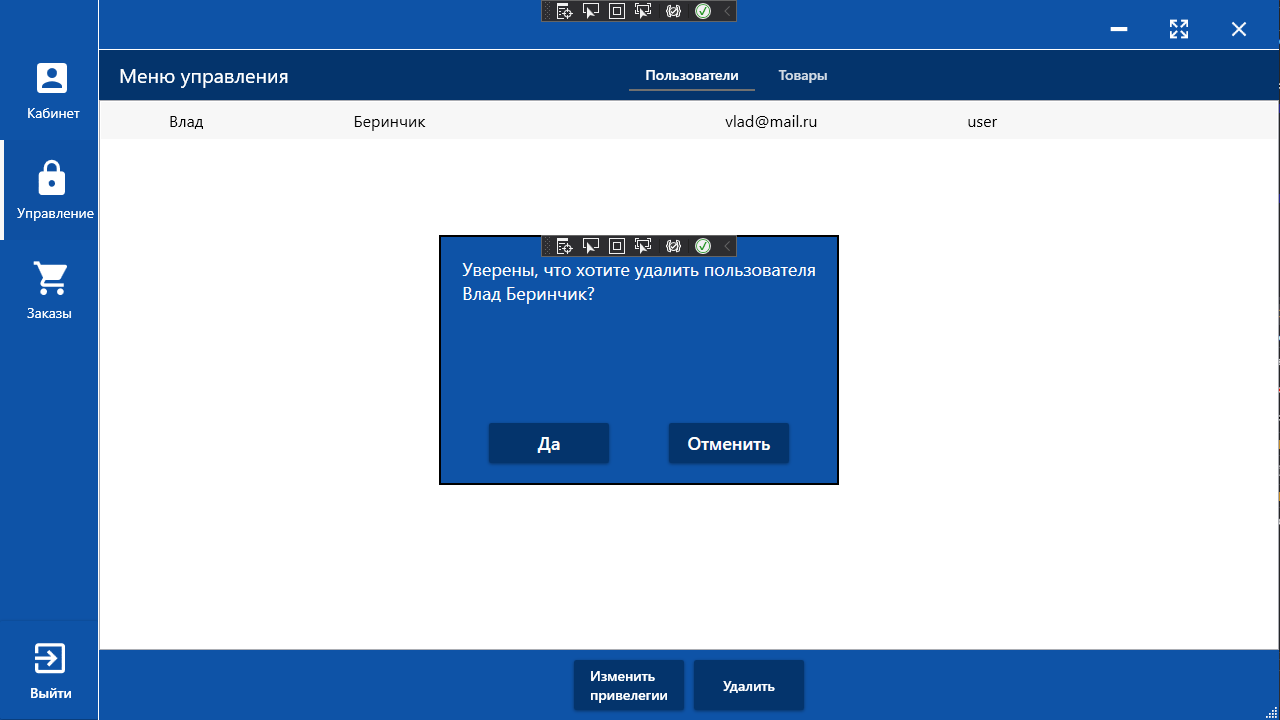


Рис. 4.15 – Окно удаления пользователя.

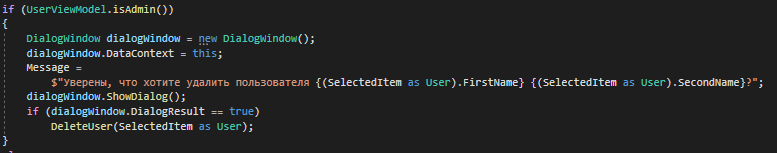


Рис. 4.16 – Реализация удаления пользователя.

Чтобы просмотреть заказы пользователей, суперпользователь должен войти в панель заказы и если заказ был выполнен то закрыть его:

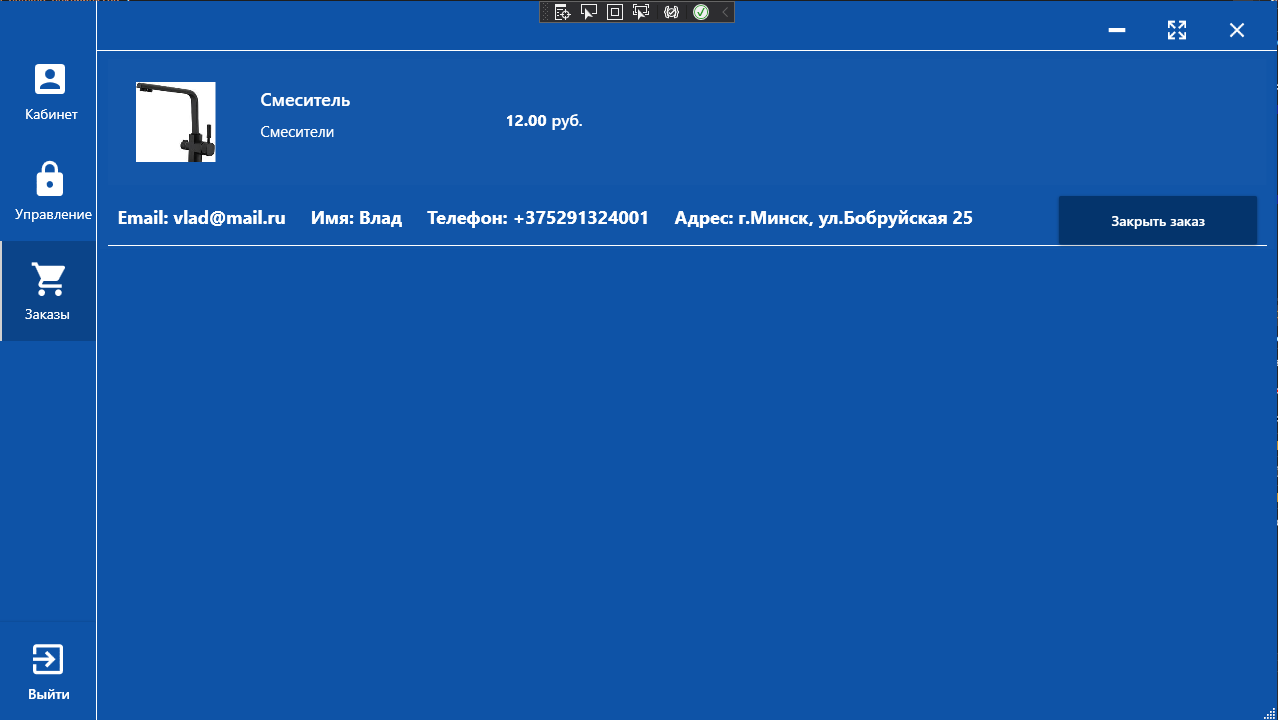


Рис. 4.17 – Окно оформленных заказов пользователями.

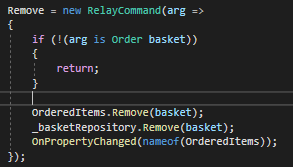


Рис. 4.18 – Реализация закрытия заказа.

После того как пользователь выбрал из каталога товары, которые хочет добавить, он должен на соответствующем товаре нажать кнопку “добавить”, после чего товар заблокируется и переместится в панель заказы:

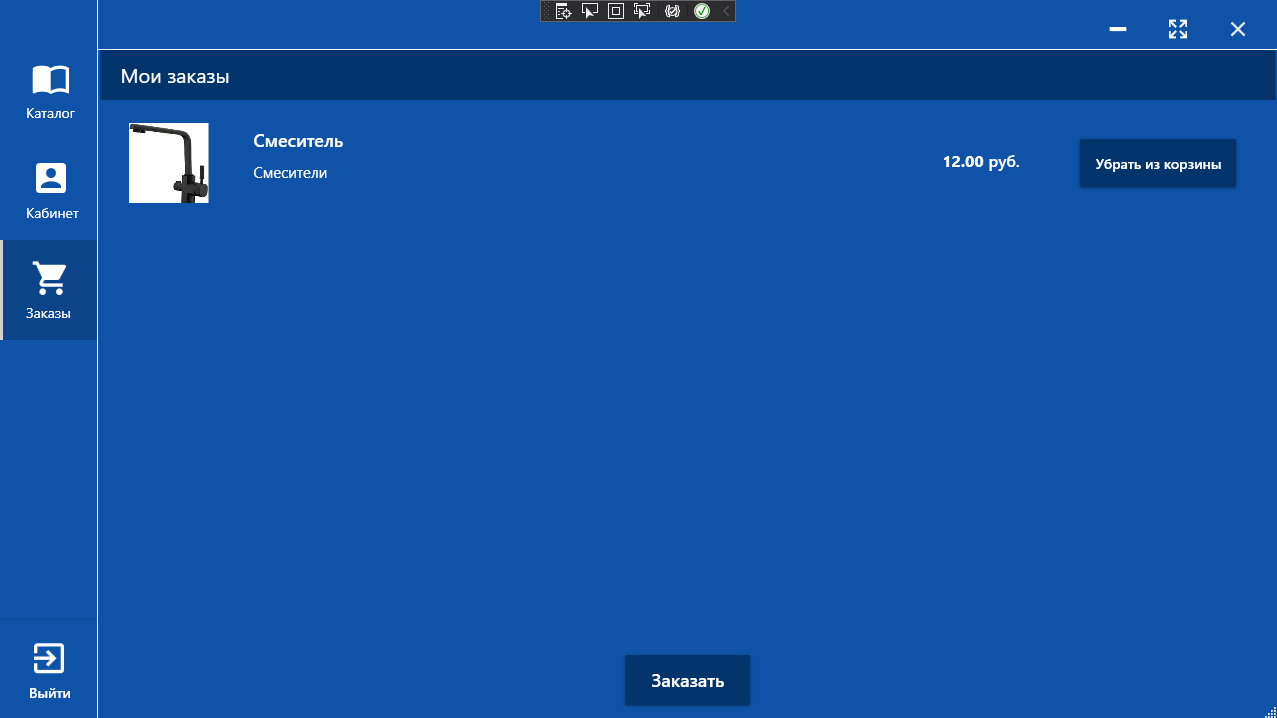


Рис. 4.19 – Окно заказа у пользователя.

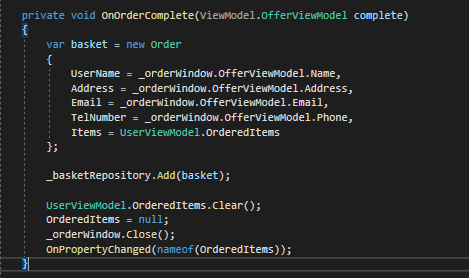


Рис. 4.20 – Реализация метода добавления товара в заказ у пользователя.

Для того, чтобы оформить заказ пользователю нужно в панели заказы нажать кнопку “заказать”, заполнить все данные во всплывающей форме и нажать кнопку “заказать”.

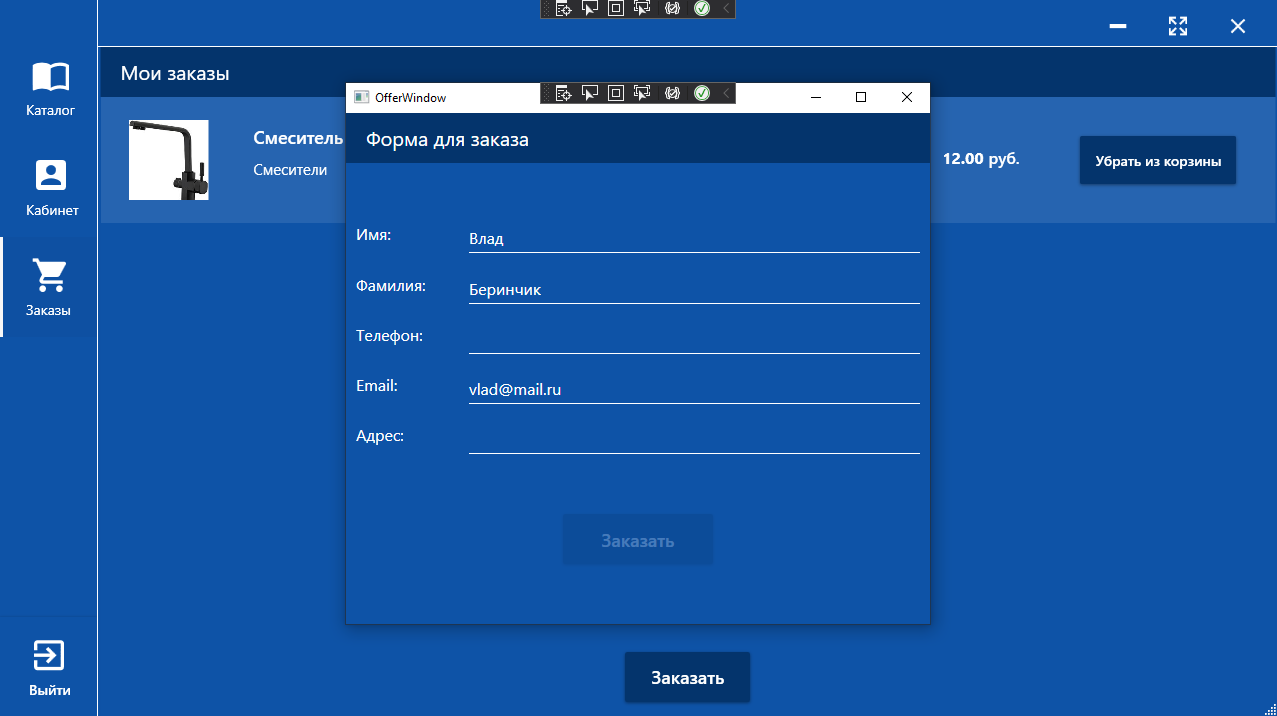


Рис. 4.21 – Окно оформления заказа.

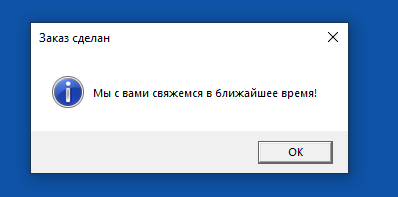


Рис. 4.22 – Окно оповещения о оформленном заказе.

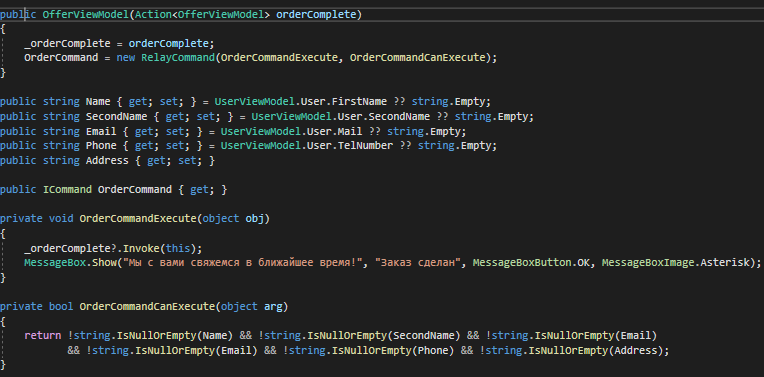


Рис. 4.23 – Реализация метода оформления заказа.

Если пользователь или суперпользователь захочет поменять информацию в аккаунте, то нужно перейти в панель “кабинет”, ввести данные в поля и нажать кнопку “сохранить”:

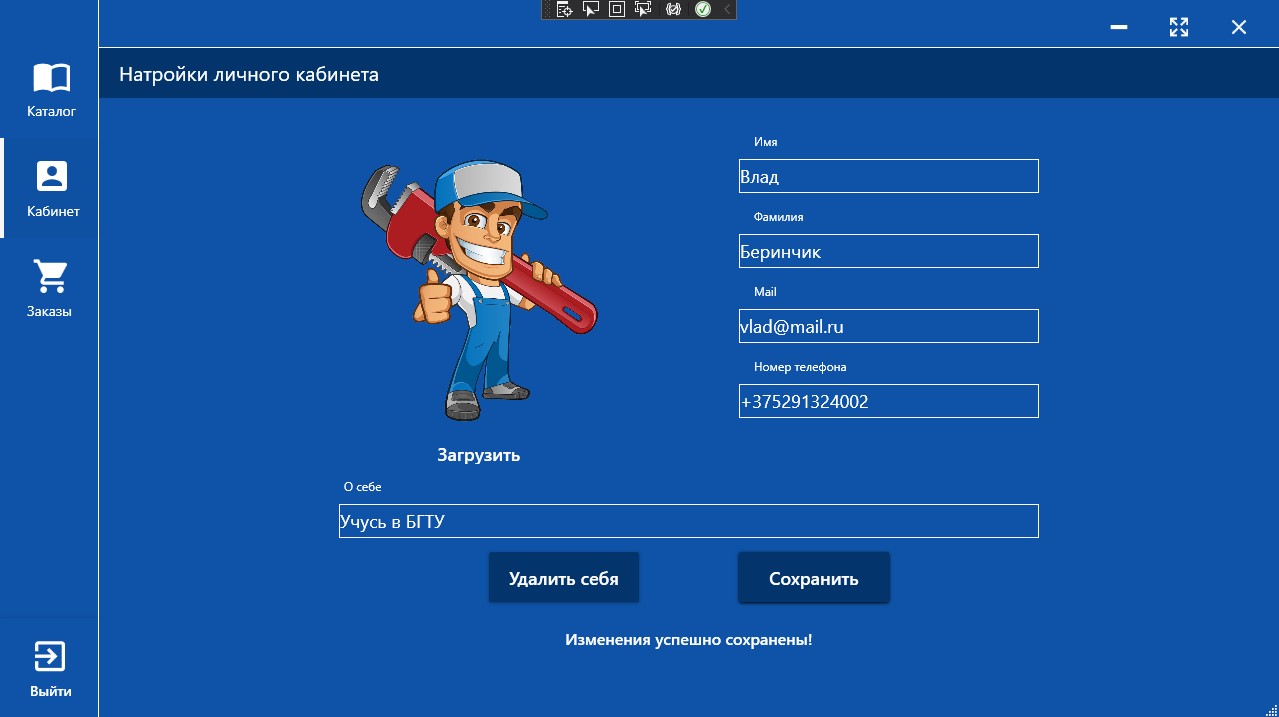


Рис. 4.34 – Окно аккаунта пользователя.

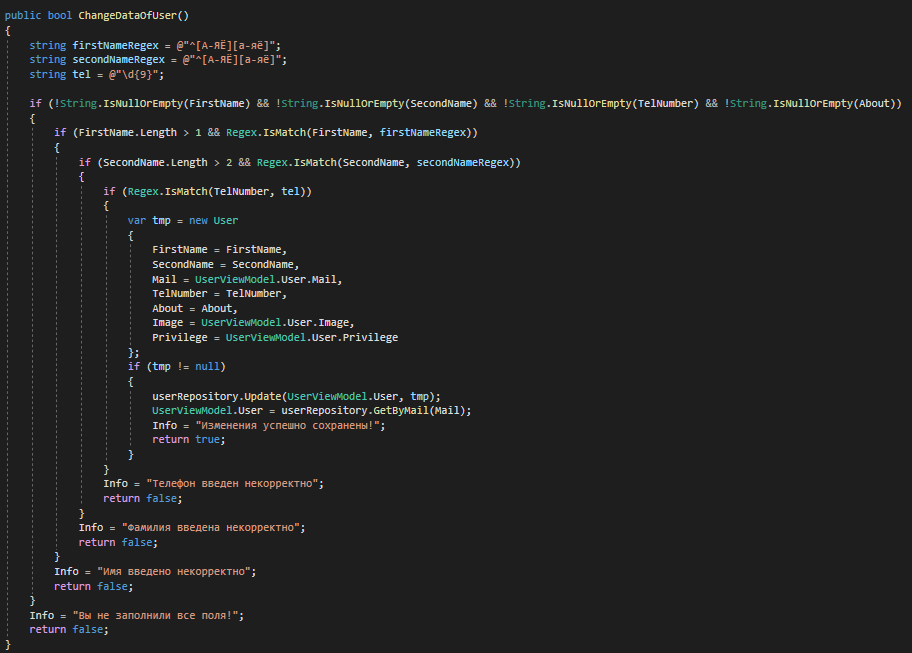


Рис. 4.35 – Реализация метода изменения данных в аккаунте.

Пользователю и суперпользователю запрещено менять почту, поэтому я добавил функцию удаления аккаунта:

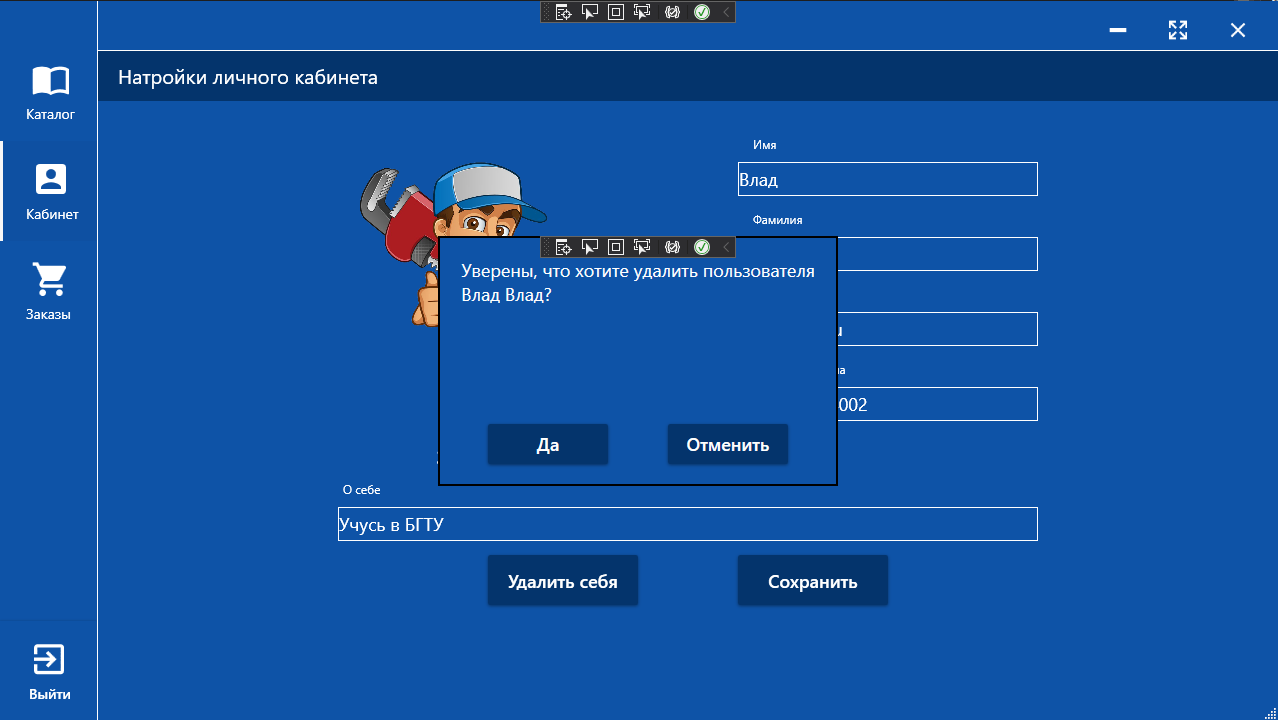


Рис. 4.36 – Окно подтверждения удаления данных аккаунта.

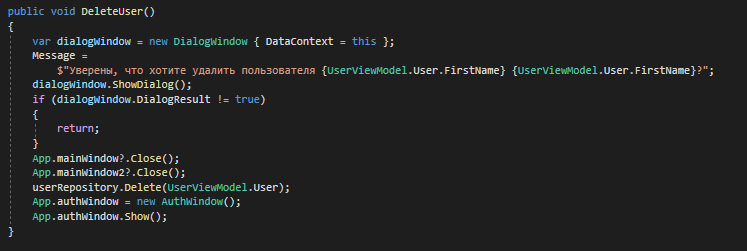


Рис. 4.37 – Реализация метода удаления аккаунта.

В приложении есть поиск и сортировка по каталогу:

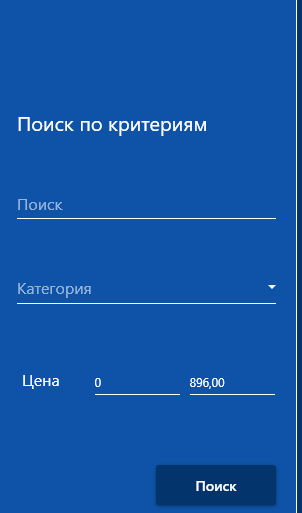


Рис. 4.38 – Поиск и сортировка по категориям.

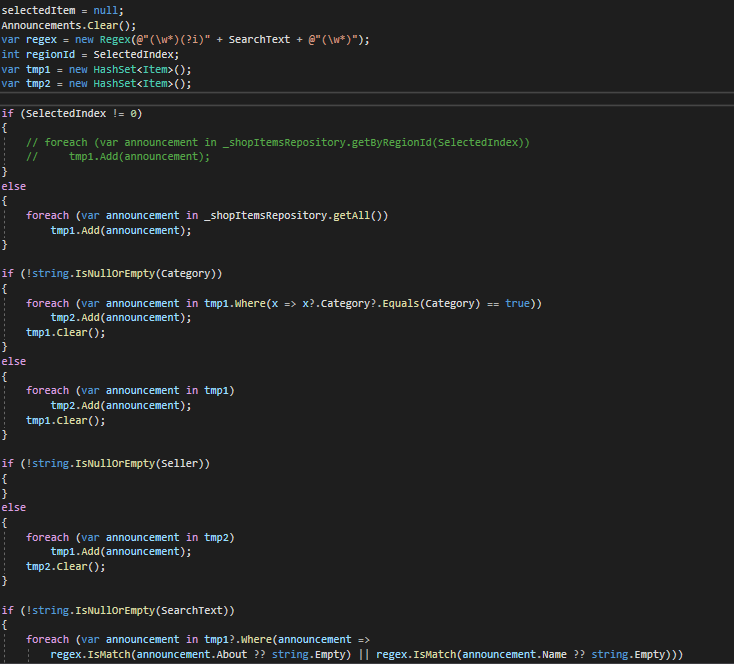


Рис. 4.39 – Реализация метода поиска.

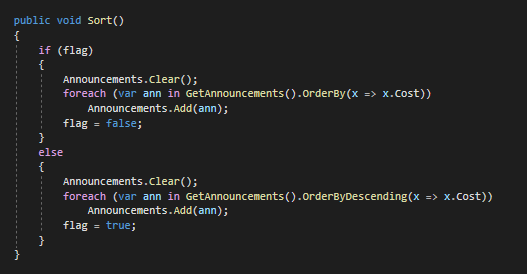


Рис. 4.40 – Реализация сортировки товаров по каталогу.

# **Тестирование, проверка работоспособности и анализ полученных результатов**

Приложение разрабатывалось таким образом, чтобы были предусмотрены все варианты, когда пользователь мог не заметить поле, ввести неправильные данные или попытаться повторно зарегистрироваться в системе.

Тестирование производилось в ходе разработки приложения. В случае обнаружения ошибок при отладке приложения, эти ошибки исправлялись, чтобы добиться более высокого уровня работоспособности программы.

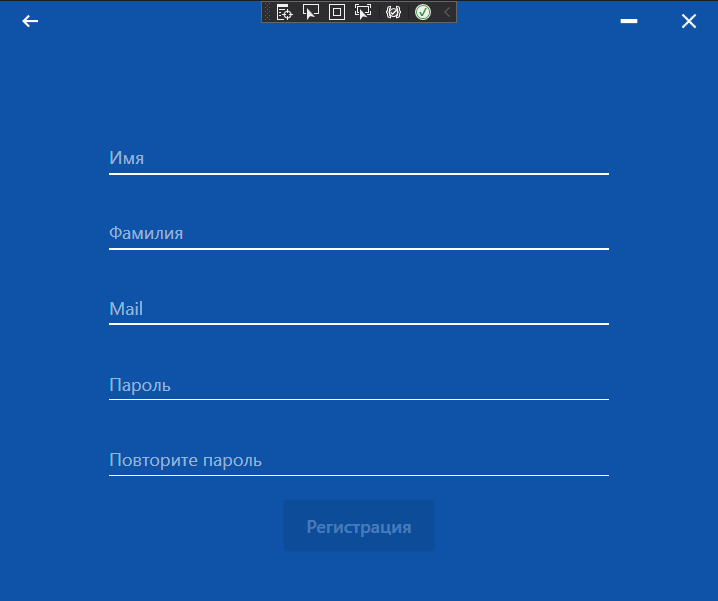


Рис. 5.1 – Блокировка кнопки, когда не заполнены поля формы.

Также присутствует проверка на наличие пользователей с введённым логином в базе данных приложения:

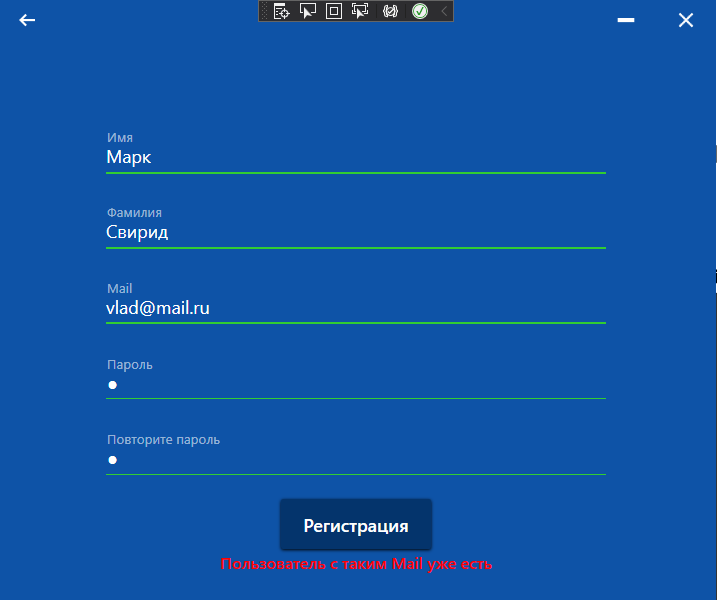


Рис. 5.2 – Обработка ошибки повторяющегося логина.

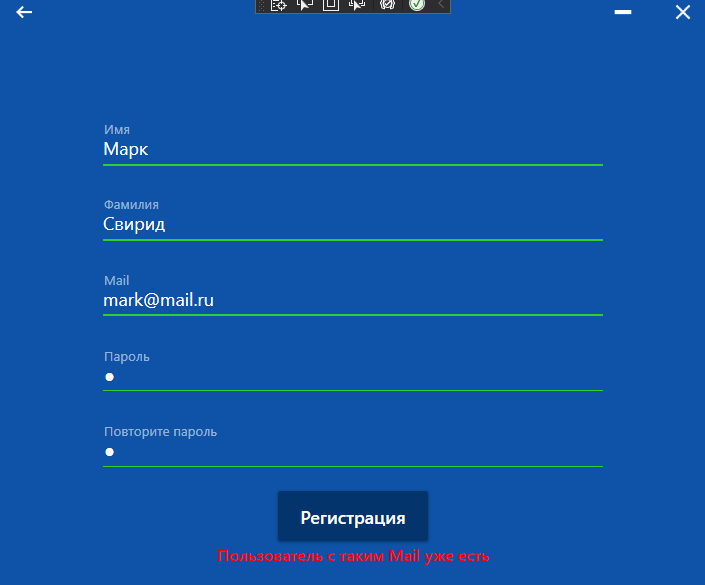


Рис. 5.3 – Обработка ошибки повторной регистрации в приложении.

У администратора приложения существует функция добавления товара. Чтобы предотвратить ошибки в этой форме была сделана валидация.

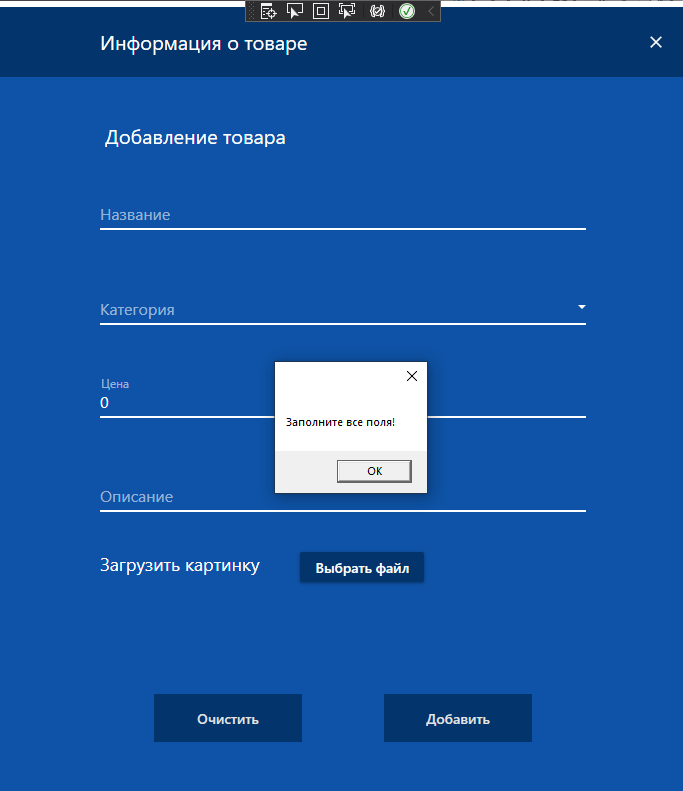
****

Рис. 5.4 – Обработка не заполненных товаров.

Также предусмотрена обработка различных ошибок при авторизации пользователей в системе.

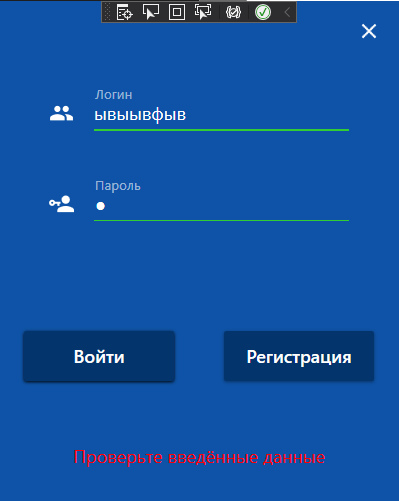


Рис. 5.5 – Попытка авторизации несуществующего пользователя.

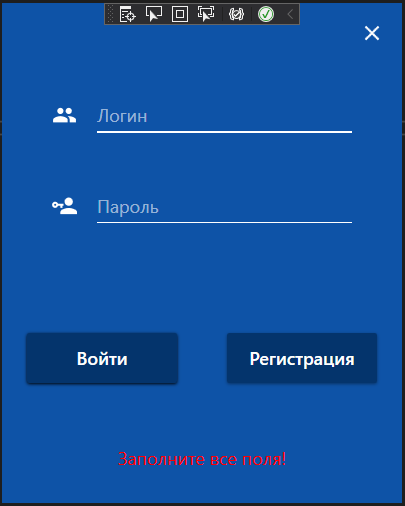


Рис. 5.5 – Пустые поля логина и пароля.

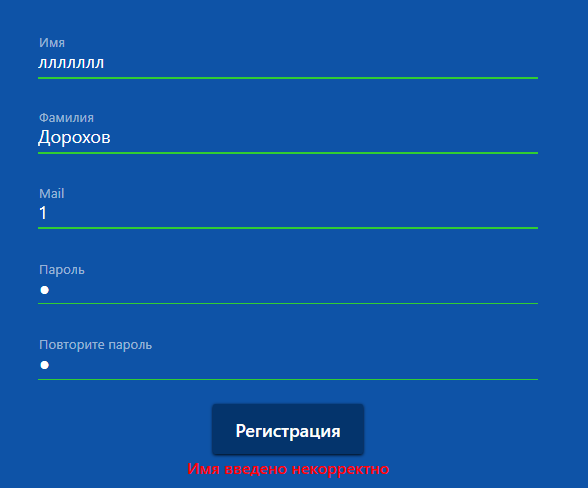
****

Рис. 5.6 – Валидация имени.

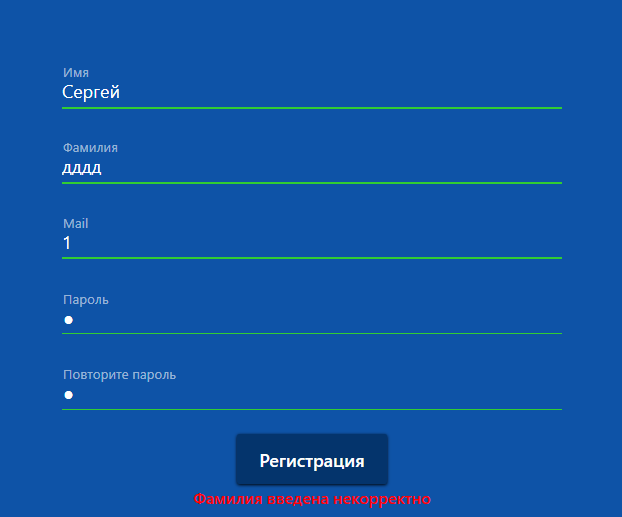
****

Рис. 5.7 – Валидация фамилии.

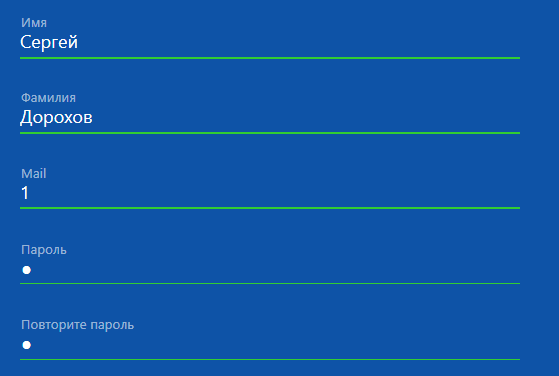
****

Рис. 5.8 – Валидация почты.

1. **Руководство по установке и использованию**

Перед установкой программы необходимо иметь на компьютере Microsoft SQL Server. Далее необходимо изменить в проекте приложения строку подключения, поменяв имя сервера на свой. Далее при запуске вся база данных будет создана автоматически.

Интерфейс программы прост и интуитивно понятен, поэтому у пользователя не должно возникнуть проблем с тем, чтобы разобраться, как работает приложения и что в нём нужно делать. В случаях, когда могут возникнуть вопросы, что сделано не так, предусмотрены сообщения об ошибках.

Работа с программой начинается с окна авторизации. Оно выглядит одинаково для всех пользователей, так как они ещё не вошли в систему под своими аккаунтами.

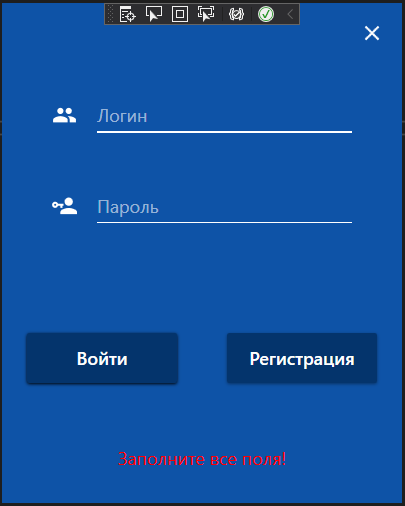


Рис. 6.1 – Окно авторизации.

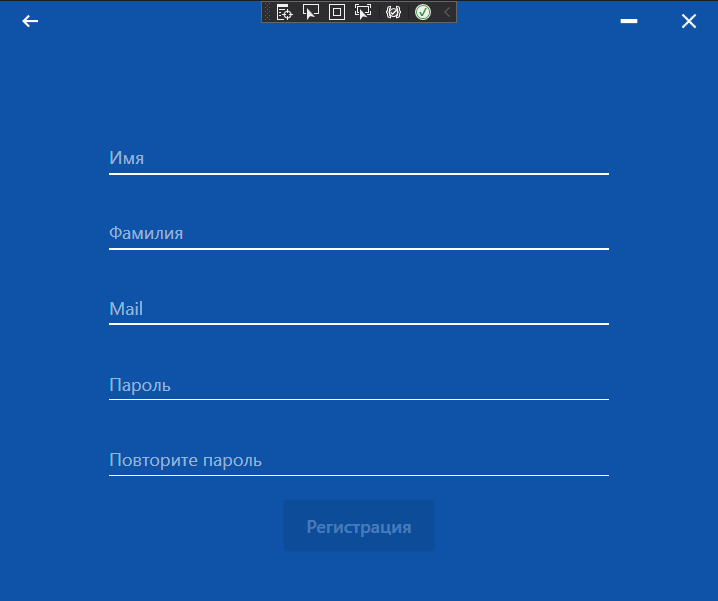


Рис. 6.2 – Окно регистрации пользователей

После регистрации пользователя необходимо через БД поменять привилегию пользователя. После того как действия выше сделаны, можно приступать к заполнению каталога товаров.

Окна пользователей отличаются внешне и по функционалу от суперпользователей. Суперпользователь взаимодействует с каталогом и другими пользователями, а обычный пользователь просматривает и делает заказы.

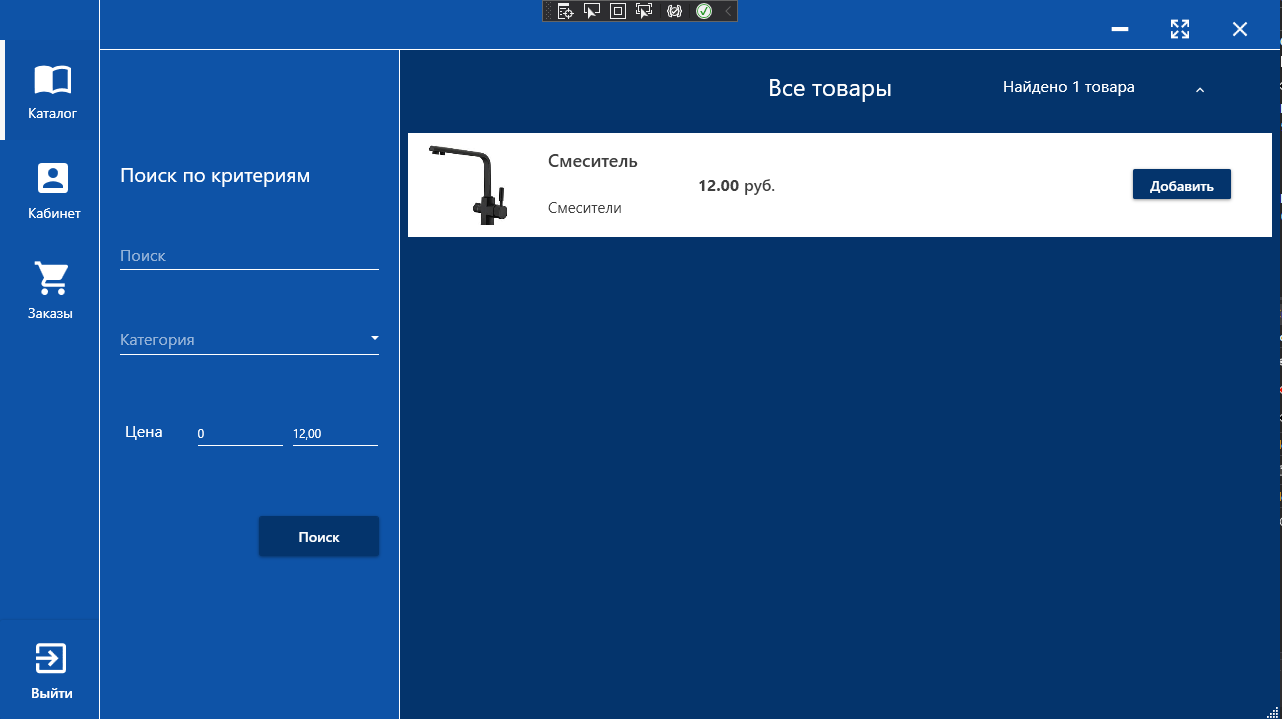


Рис. 6.3 – Окно пользователя для работы с приложением.

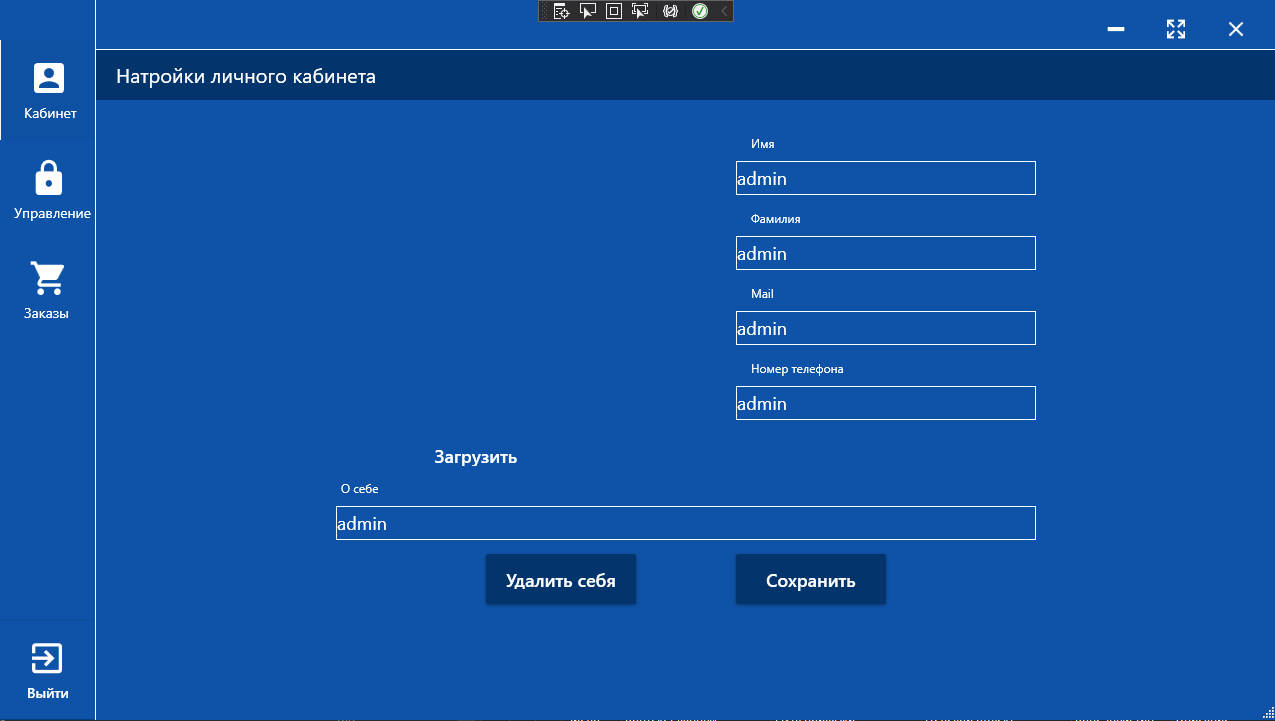


Рис. 6.3 – Окно суперпользователя для работы с приложением.

# **Заключение**

Результатом курсового проекта является разработанное на языке C# настольное приложение для заказа сантехники на дом.

В ходе выполнения курсового проекта были выполнены следующие действия:

– создана базы данных для хранения информации;

– разработана архитектура приложения;

– разработана функциональная часть приложения;

– разработан пользовательский интерфейс;

– написан исходный код приложения;

– реализованы все функции, которые были предусмотрены в листе задания на курсовой проект;

– протестировано приложение.

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

В ходе выполнения курсовой работы были изучены принципы создания приложений WPF, изучен язык XAML, приобретены знания и практические навыки по работе с технологией Entity Framework для работы с базами данных.

# **Список использованных источников**

1. schools.by URL: https://schools.by/ (дата обращения: 29.03.2021).
2. metanit.com URL: https://metanit.com/ (на протяжении всего времени затраченного на проект).
3. professorweb.ru URL: https://professorweb.ru/ (на протяжении всего времени затраченного на проект).
4. docs.microsoft.com URL: <https://docs.microsoft.com/> (на протяжении всего времени затраченного на проект).

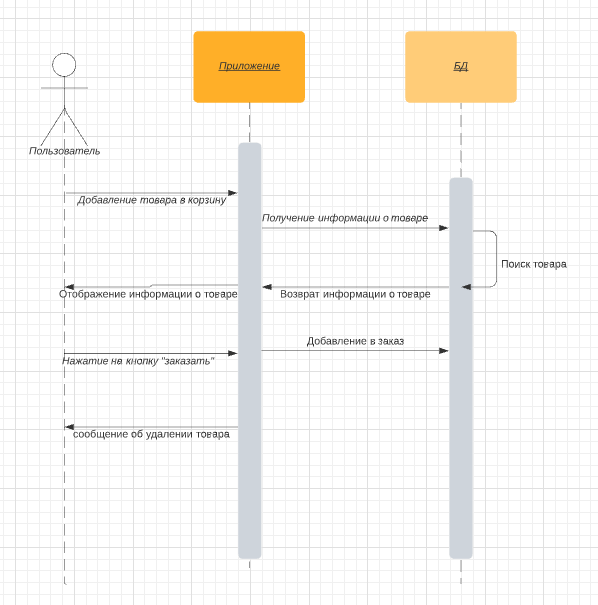
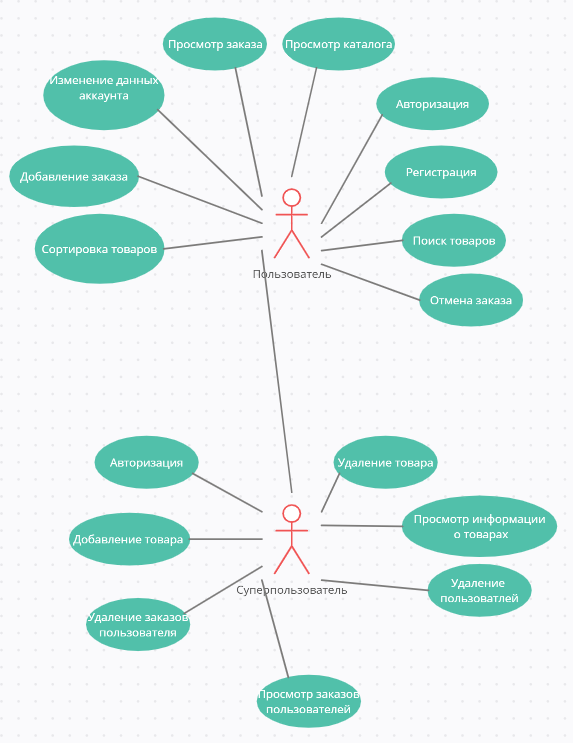


Диаграмма последовательности



**Диаграмма возможностей**

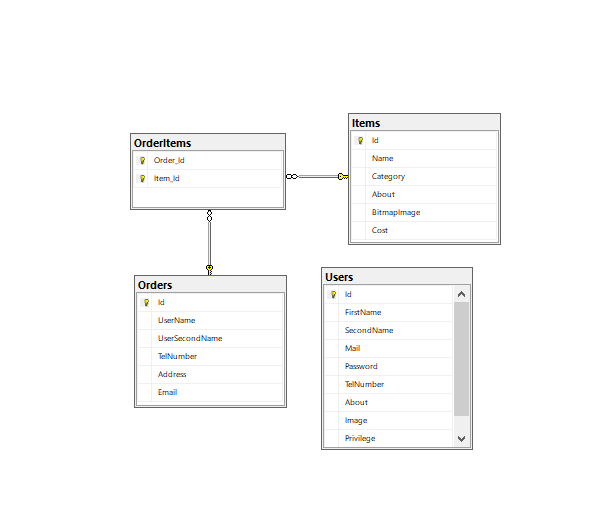


**UML Диаграмма классов**

# 

Диаграмма деятельности(добавление товара)

# **Приложение B. Логическая структура базы данных**



**Структура БД**

# **Приложение D. Листинги программы**

public bool Registration(string password1, string password2)

{

Password1 = password1;

Password2 = password2;

string firstNameRegex = @"^[A-ЯЁ][а-яё]";

string secondNameRegex = @"^[A-ЯЁ][а-яё]";

string mailRegex = @"^(?("")(""[^""]+?""@)|(([0-9a-z]((\.(?!\.))|[-!#\$%&'\\*\+/=\?\^`\{\}\|~\w])\*)(?<=[0-9a-z])@))" + @"(?(\[)(\[(\d{1,3}\.){3}\d{1,3}\])|(([0-9a-z][-\w]\*[0-9a-z]\*\.)+[a-z0-9]{2,17}))$";

if (!String.IsNullOrEmpty(FirstName) && !String.IsNullOrEmpty(SecondName) && !String.IsNullOrEmpty(Mail) && !String.IsNullOrEmpty(Password1) && !String.IsNullOrEmpty(Password2))

{

if (FirstName.Length > 1 && Regex.IsMatch(FirstName, firstNameRegex))

{

if (SecondName.Length > 2 && Regex.IsMatch(SecondName, secondNameRegex))

{

if (Regex.IsMatch(Mail, mailRegex, RegexOptions.IgnoreCase))

{

if (Password1.Length > 0 && Password2.Length > 0)

{

if (Password1.Equals(Password2))

{

var tmp = userRepository.GetByMail(Mail);

if (tmp == null)

{

userRepository.Add(new User

{

FirstName = FirstName,

SecondName = SecondName,

Mail = Mail,

Password = Password1,

Privilege = "user"

});

Info = "Зареган!";

return true;

}

Info = "Пользователь с таким Mail уже есть";

return false;

}

Info = "Пароли должны совпадать";

return false;

}

}

Info = "Email введен некорректно";

return false;

}

Info = "Фамилия введена некорректно";

return false;

}

Info = "Имя введено некорректно";

return false;

}

Info = "Вы не заполнили все поля!";

return false;

}

**Регистрация нового пользователя**

public bool CompareDataOfUser(string passwordParam)

{

if (!String.IsNullOrEmpty(Login) && !String.IsNullOrEmpty(passwordParam))

{

var tmp = eFUserRepository.GetByMail(Login);

if (tmp != null)

{

if (HashHelper.GetMd5Hash(passwordParam) == tmp.Password)

{

if (tmp.IsAdmin())

{

UserViewModel.User = tmp;

App.mainWindow = new MainWindow();

App.mainWindow.Show();

return true;

}

UserViewModel.User = tmp;

App.mainWindow2 = new MainWindow2();

App.mainWindow2.Show();

return true;

}

Info = "Проверьте введённые данные";

return false;

}

Info = "Проверьте введённые данные";

return false;

}

Info = "Заполните все поля!";

return false;

}

**Авторизация пользователя**