МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий   
Кафедра программной инженерии

Утверждаю

Заведующий кафедрой ПИ

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

подпись инициалы и фамилия

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021г.

**ЗАДАНИЕ**

**к курсовому проектированию**

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

|  |  |
| --- | --- |
| Специальность: 1-40 05 01-03 «Информационные системы и технологии»  Студент: Конопляник Е. Е. | Группа:\_\_1\_\_ |
| **Тема: Программное средство «Система записи на услуги парикмахерской»** | |

**1. Срок сдачи студентом законченной работы**: "20 мая 2021 г."

**2. Исходные данные к проекту:**

**2.1**. Функционально ПС поддерживает:

* Функции администратора сервиса:
* Добавление/удаление/редактирование информации об услугах;
* Добавление/удаление/редактирование информации о продуктах;
* Редактирование информации о записях клиентов;
* Функции клиента:
* Резервирование услуг;
* Заказ товаров;
* Регистрация/вход в личный профиль;

**2.2.** При выполнении курсового проекта необходимо использовать принципы проектирования 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 на диск, указанный преподавателем.

#### Календарный план

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Введение | 19.02.2021 |  |
| 2 | Аналитический обзор литературы по теме проекта. Изучение требований, определение вариантов использования | 12.03.2021 |  |
| 3 | Анализ и проектирование архитектуры приложения (построение диаграмм, проектирование бизнес-слоя, представления и данных) | 26.03.2021 |  |
| 4 | Проектирование структуры базы данных. Разработка дизайна пользовательского интерфейса | 2.04.2021 |  |
| 5 | Кодирование программного средства | 23.04.2021 |  |
| 6 | Тестирования и отладка программного средства | 30.04.2021 |  |
| 7 | Оформление пояснительной записки | 7.05.2021 |  |
| 9 | Сдача проекта | 20.05.2021 |  |

**5. Дата выдачи задания \_\_\_\_**12.02.2021**\_\_\_\_**

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *И. Г. Сухорукова*

(подпись)

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

(дата и подпись студента)

**Оглавление**

[Введение 5](#_Toc73088314)

[1 Аналитический обзор аналогов 6](#_Toc73088315)

[1.1 Анализ прототипов 6](#_Toc73088316)

[1.2 Постановка задачи 14](#_Toc73088317)

[2. Анализ требований к программному средству и разработка функциональных требований 15](#_Toc73088318)

[2.1 Определение требований к программному средству 15](#_Toc73088319)

[2.2 Описание средств разработки 15](#_Toc73088320)

[2.3 Описание функциональности программного средства 17](#_Toc73088321)

[2.4 Спецификация функциональных требований 17](#_Toc73088322)

[3 Проектирование и создание программного средства 19](#_Toc73088323)

[3.1 Архитектура системы 19](#_Toc73088324)

[3.2 Модель базы данных 20](#_Toc73088325)

[3.3 UML диаграммы 23](#_Toc73088326)

[3.3 Структура проекта и выполняемые функции 24](#_Toc73088327)

[3.4 Пример использования 26](#_Toc73088328)

[4 Реализация программного средства 27](#_Toc73088329)

[4.1 Реализация паттерна MVVM 27](#_Toc73088330)

[4.2 Реализация авторизации, регистрации и восстановления пароля 28](#_Toc73088331)

[4.3 Покупка книги 30](#_Toc73088332)

[4.4 Удаление пользователей редакторами 32](#_Toc73088333)

[4.5 Уведомления на почту о покупках в приложении 32](#_Toc73088334)

[4.6 Выставление оценок книгам 33](#_Toc73088335)

[5 Тестирование, проверка работоспособности и анализ полученных результатов 35](#_Toc73088336)

[6 Руководство по использованию 42](#_Toc73088337)

[Заключение 46](#_Toc73088338)

[Приложение А 48](#_Toc73088339)

[Приложение Б 50](#_Toc73088340)

[Приложение В 51](#_Toc73088341)

[Приложение Г 52](#_Toc73088342)

[Приложение Д 53](#_Toc73088343)

[Приложение Е 55](#_Toc73088344)

[Приложение Ж 57](#_Toc73088345)

# Введение

Основная цель моего курсового проекта – разработка программного средства «Система записи на услуги парикмахерской». Разработанное ПС должно предоставить пользователям возможность записаться на услуги и зарезервировать необходимые им продукты без непосредственного посещения рабочего помещения парикмахерской.

В ходе выполнения курсового проекта мне необходимо произвести анализ конкурентов, составить диаграммы использования, диаграммы последовательности, диаграммы классов, выполнить реализацию программного средства, провести тестирование и проверку работоспособности конечного приложения.

Многие компании из сферы услуг до сих пор ведут запись клиентов по телефону и заполняют бумажный журнал. Это устаревшая система работы, в которой легко перепутать даты, проблематично отследить постоянных заказчиков и анализировать эффективность работы.

Сервис записи – это также инструмент для оптимизации работы администраторов. Он упрощает процесс взаимодействия между компанией и её клиентами. Потребители могут записаться на услугу в любое время суток даже с телефона без очередей и нервов.[1]

Стоит подчеркнуть активную актуализацию идеи, вложенной в данное программное средство в наши дни в связи с пандемией коронавирусной инфекцией и желанием всё более широких групп населения производить определенного рода рутинные действия, не выходя из дома.

# 1 Аналитический обзор аналогов

Для корректного проектирования приложения, определения его функциональности, а также требований пользователей необходимо провести обзор аналогов. Данная процедура поможет нам подчеркнуть лучшие практики, используемые в современных приложениях данного типа.

В целом приложений данной категории для Windows довольно мало, так как все-таки книги редко читаются на большом экране компьютера. Намного комфортнее их читать на смартфоне, планшете и, естественно, на электронной книге.

## 1.1 Анализ прототипов

Аналог №1 – «Локон».

На рисунке 1.1 можно видеть главную страницу онлайн-сервиса для записи на услуги парикмахерской «Локон.by».

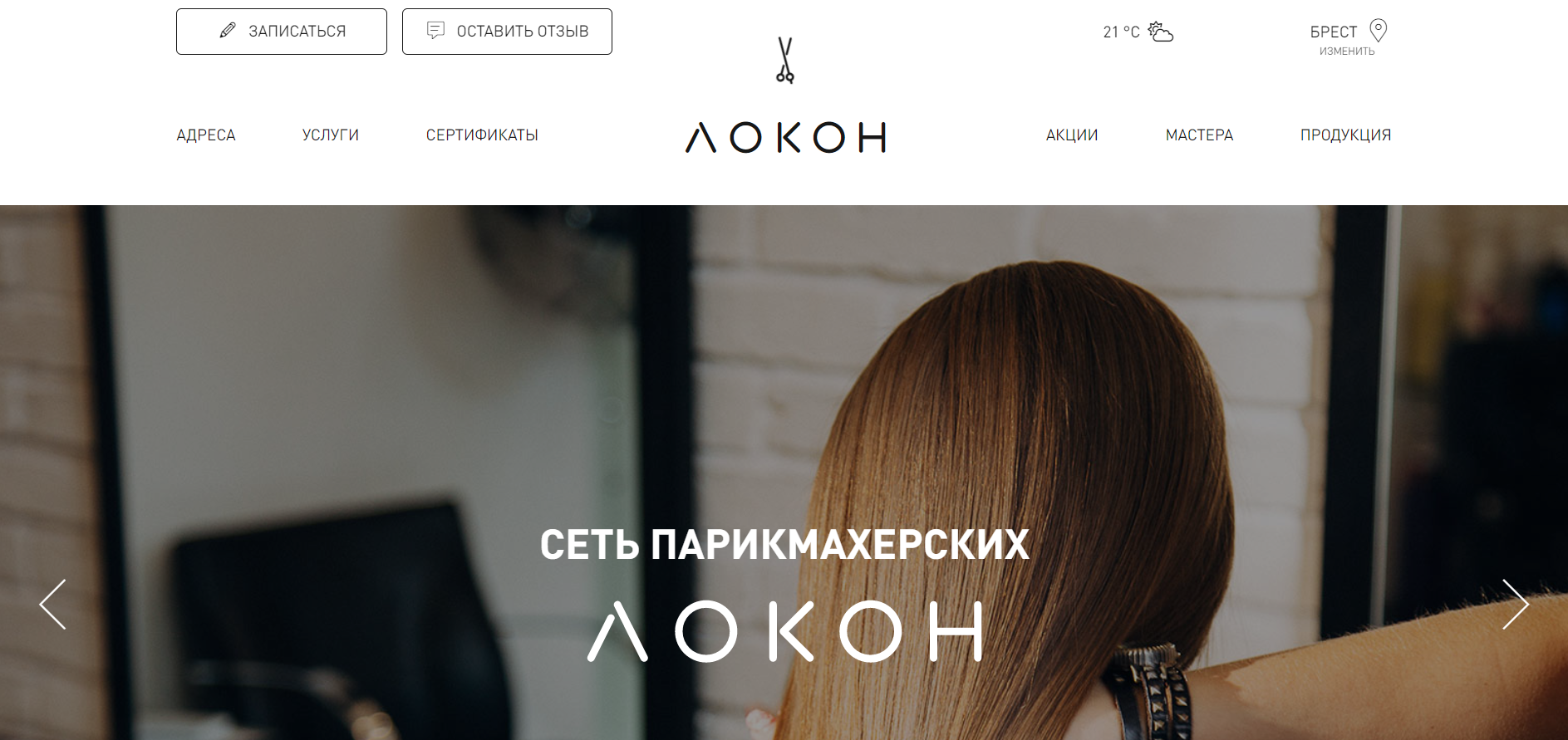


Рисунок 1.1 – Главная страница «Локон.by»

Данный сервис обладает всем необходимым функционалом для приложений данной категории. Гибкость поиска заключается в возможности искать не только по названию услуг, но и по имени мастера. Результаты поиска выводятся в удобном виде (Рисунок 1.2).

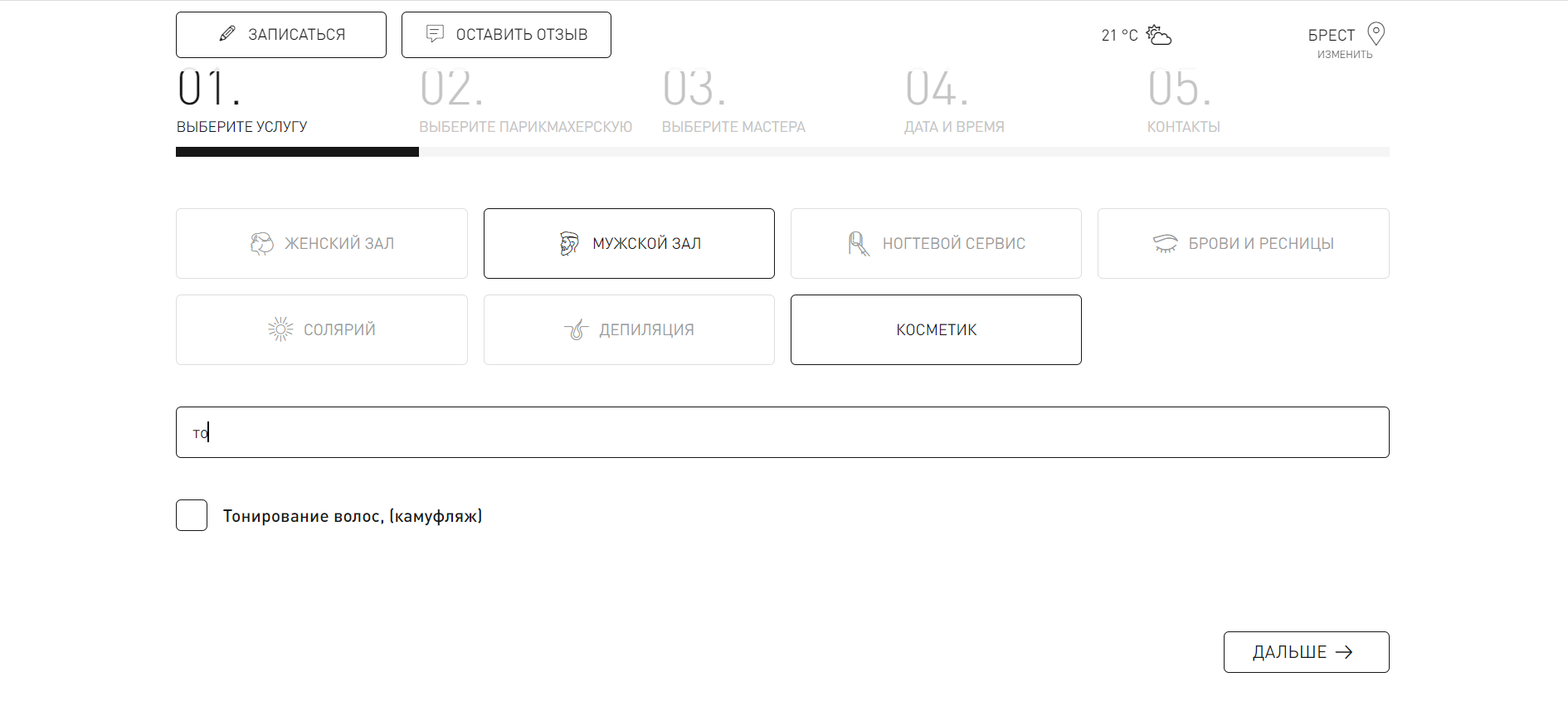


Рисунок 1.2 – Результат поиска

Переход по вкладкам в процессе записи на услугу происходит максимально ненавязчиво (Рисунок 1.3). На мой взгляд, все сделано невероятно продумано и приятно. Дизайн крайне лаконичен и понятен мне, как рядовому пользователю.

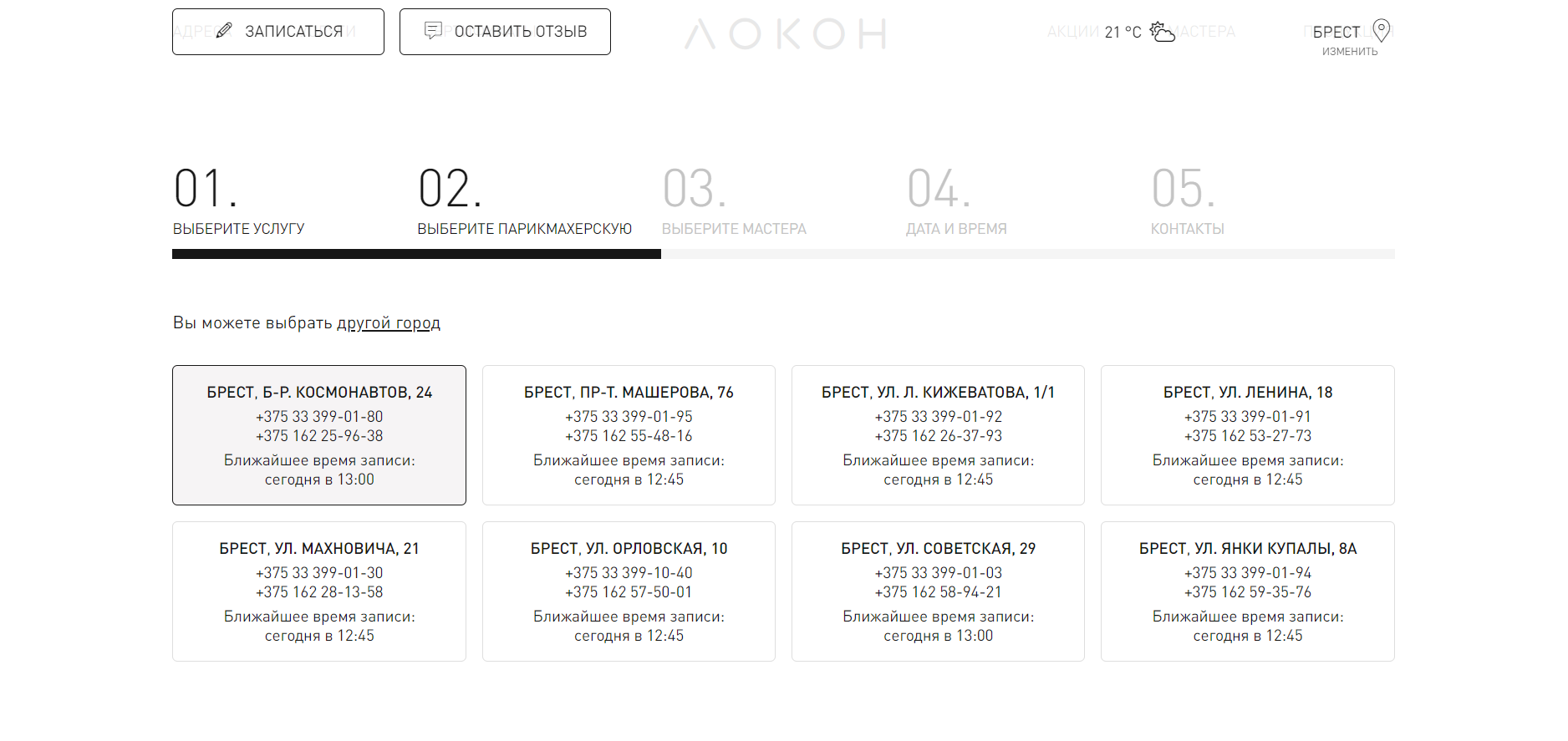


Рисунок 1.3 – Страница выбора адреса

Удобная навигация, приятный визуальный стиль (используется белый и оттенки серого для фона), отличный функционал делают этот сайт очень удобным и желанным для целевой аудитории.

Резюмируя, хочется еще раз выделить возможности данного сервиса:

* поиск услуг по ключевым словам;
* отзыв;
* запись на услугу;
* выбор локации распространения услуг;
* использование подарочных сертификатов;
* система рекомендаций.

Достоинства приложения:

* простой, но при этом очень богатый функционал;
* приятный неперегруженный интерфейс;
* быстрая скорость работы;
* отсутствие сторонней рекламы;
* надежность;
* наличие возможности оставить отзыв.

Недостатки приложения:

* отсутствует личный профиль.

Аналог №2 – «Парикмахерская №1».

На Рисунке 1.4 представлена главная страница сайта «Парикмахерская №1». Она не изобилует потрясающими стилистическими решениями. И слегка перегружена визуальными элементами в виде фонового видео.

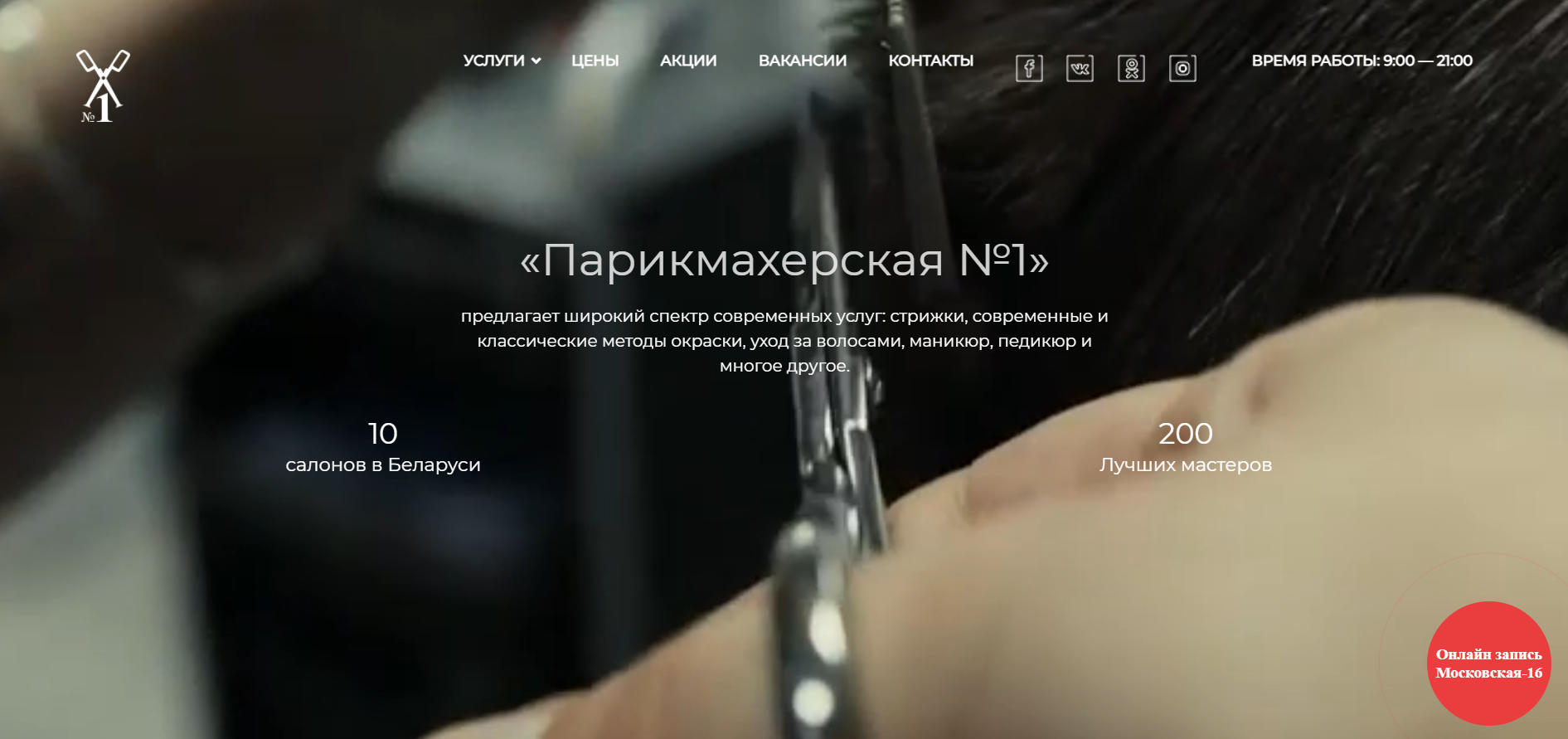


Рисунок 1.4 – Главная страница сайта «Парикмахерская №1»

Нажатие на красную кнопку в правом нижнем углу экрана вызывает меню записи на услугу (Рисунок 1.5). Стоит отметить нагромождение информации в этой зоне.

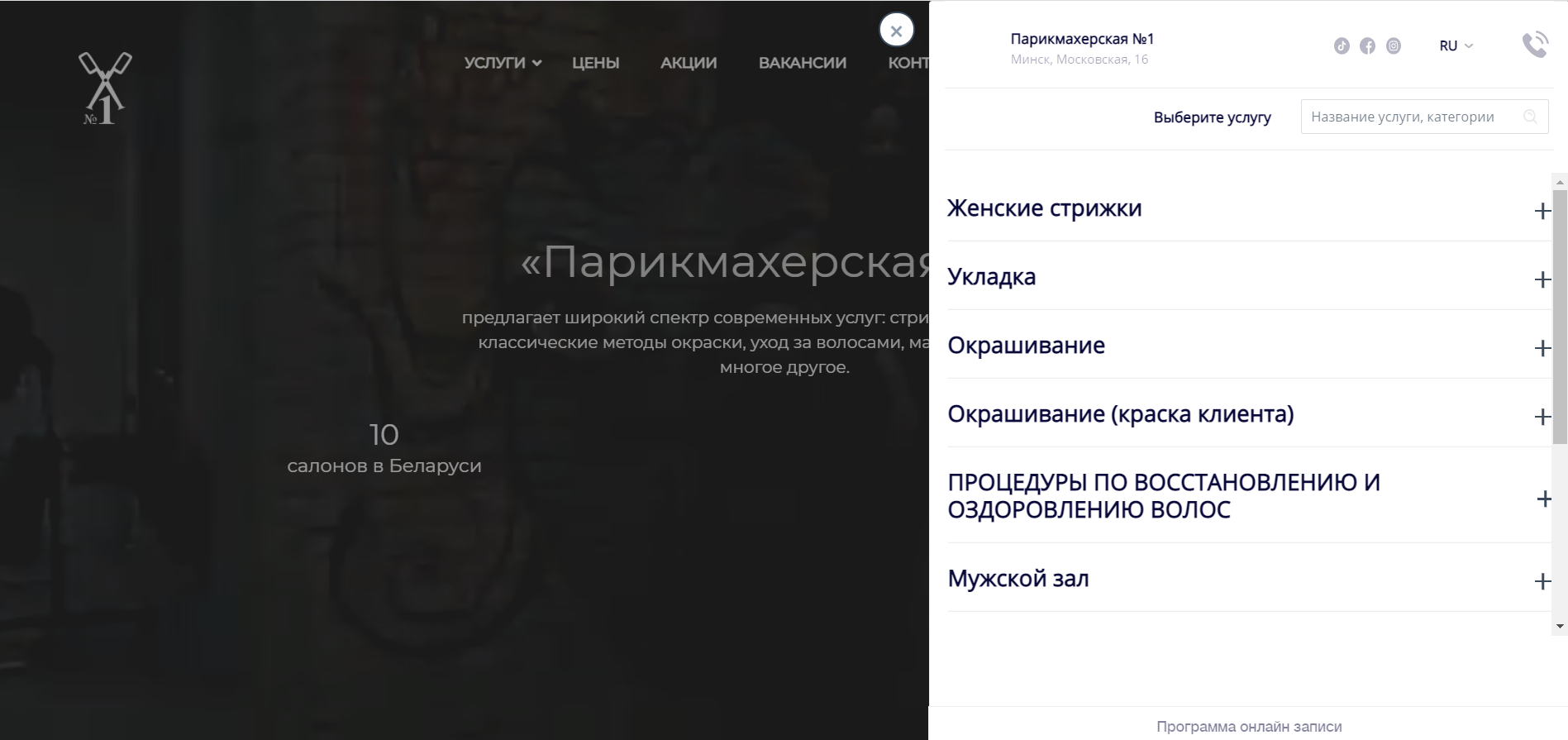


Рисунок 1.5 – Меню записи

Подводя итоги, хочется выделить возможности данного сервиса:

* отзыв;
* запись на услугу;
* возможность подать заявку на вакантное место в компании;
* система рекомендаций.

Достоинства приложения:

* отсутствие сторонней рекламы;
* полноценный функционал;
* наличие возможности оставить отзыв.

Недостатки приложения:

* отсутствует личный профиль;
* медленная загрузка главной страницы из-за тяжелой видеовставки;
* слегка перегруженный дизайн;
* навязчивая кнопка в правом нижнем углу экрана.

## 1.2 Постановка задачи

Таким образом, в результате исследования аналогов я пришел к выводу, что приложение данной тематики не должно быть очень сложным. Приложение должно содержать только ту информацию об услугах, которая исключительно важна для пользователя.

Приложение типа «система записи на услуги парикмахерской» в первую очередь должно предоставлять пользователям список из услуг с возможностью записи. Таким образом, данное приложение выступает в роли связующего звена между пользователем и салоном.

В ходе выполнения курсового проекта необходимо разработать приложение, в котором будет возможность резервирования услуг, ознакомления со списком записей.

В результате исследования аналогов были выявлены следующие цели:

* предоставить функционал, который будет соответствовать функционалу конкурентов;
* отличная скорость приложения;
* удержания пользователей;
* создать удобный интерфейс для пользователей.

Данное исследование позволяет нам перейти к анализу требований к программному средству и разработке функциональных требований.

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

Прежде, чем начать разработку проекта необходимо определить основные функциональные требования к программному средству.

## 2.1 Определение требований к программному средству

Основной задачей моего курсового проекта является разработка десктопного приложения, обладающего функционалом сервиса записи на услуги парикмахерской, которое обеспечит упрощение процесса взаимодействия клиента с компанией – парикмахерской, а также станет помощником в работе с БД персонала компании.

Мое программное средство должно поддерживать следующий функционал:

* авторизация со стороны администратора;
* добавление/удаление/редактирование информации об услугах со стороны администратора;
* добавление/удаление/редактирование информации о продуктах со стороны администратора;
* редактирование информации о записях клиентов со стороны администратора;
* резервирование услуг со стороны пользователя;
* заказ товаров со стороны пользователя;
* регистрация и авторизация со стороны пользователя.

## 2.2 Описание средств разработки

Во время процесса разработки приложения были использованы следующие программные средства:

* интегрированная среда разработки Microsoft Visual Studio 2019 Community;
* язык программирования C#;
* программная платформа .NET Framework 4.7.2;
* технология WPF;
* расширяемый язык разметки XAML;
* технология Entity Framework 6.4;
* язык визуальных образов Material Design In XAML;
* NuGet-пакет Extended WPF Toolkit.

Интегрированная среда разработки Visual Studio Community —полнофункциональная, расширяемая и бесплатная интегрированная среда разработки для создания современных приложений Android, iOS и Windows, а также веб-приложений и облачных служб. Visual Studio включает в себя [редактор исходного кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80_%D0%B8%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BA%D0%BE%D0%B4%D0%B0) с поддержкой технологии [IntelliSense](https://ru.wikipedia.org/wiki/IntelliSense) и возможностью простейшего [рефакторинга кода](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3). Встроенный [отладчик](https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio_Debugger) может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер [классов](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%81_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) и дизайнер [схемы базы данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Visual Studio позволяет создавать и подключать сторонние дополнения ([плагины](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD)) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем [контроля версий исходного кода](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%D0%BC%D0%B8). [2]

Для написания программного средства использовался язык программирования C#. C# – это современный типобезопасный объектно-ориентированный язык программирования, который позволяет разработчикам создавать множество типов безопасных и надежных приложений, работающих в экосистеме .NET. C# берет свое начало в семействе языков C.

Windows Presentation Foundation (WPF) – это платформа пользовательского интерфейса, которая создает настольные клиентские приложения. Платформа разработки WPF поддерживает широкий набор функций разработки приложений, включая модель приложения, ресурсы, элементы управления, графику, макет, привязку данных, документы и безопасность. Платформа является частью .NET. WPF использует расширяемый язык разметки приложений (XAML), чтобы предоставить декларативную модель для программирования приложений.

XAML (eXtensible Application Markup Language) - язык разметки, используемый для инициализации объектов в технологиях на платформе .NET. Применительно к WPF (а также к Silverlight) данный язык используется прежде всего для создания пользовательского интерфейса декларативным путем. Хотя функциональность XAML только графическими интерфейсами не ограничивается: данный язык также используется в технологиях WCF и WF, где он никак не связан с графическим интерфейсом. То есть его область шире. Применительно к WPF мы будем говорить о нем чаще всего именно как о языке разметки, который позволяет создавать декларативным путем интерфейс, наподобие HTML в веб-программировании. Однако опять же повторюсь, сводить XAML к одному интерфейсу было бы неправильно, и далее на примерах мы это увидим. [https://metanit.com/sharp/wpf/2.php]

ADO.NET Entity Framework (EF) — объектно-ориентированная технология доступа к данным, является [object-relational mapping](https://ru.wikipedia.org/wiki/ORM) (ORM) решением для [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework) от [Microsoft](https://ru.wikipedia.org/wiki/Microsoft). Предоставляет возможность взаимодействия с объектами как посредством [LINQ](https://ru.wikipedia.org/wiki/LINQ) в виде LINQ to Entities, так и с использованием Entity SQL. Для облегчения построения web-решений используется как [ADO.NET Data Services](https://ru.wikipedia.org/wiki/ADO.NET_Data_Services) ([*Astoria*](https://ru.wikipedia.org/wiki/ADO.NET_Data_Services)), так и связка из [Windows Communication Foundation](https://ru.wikipedia.org/wiki/Windows_Communication_Foundation) и [Windows Presentation Foundation](https://ru.wikipedia.org/wiki/Windows_Presentation_Foundation), позволяющая строить многоуровневые приложения, реализуя один из шаблонов проектирования [MVC](https://ru.wikipedia.org/wiki/Model-View-Controller), [MVP](https://ru.wikipedia.org/wiki/Model-View-Presenter) или [MVVM](https://ru.wikipedia.org/wiki/Model-View-ViewModel).[ <https://ru.wikipedia.org/wiki/ADO.NET_Entity_Framework>]

Extended WPF Toolkit - это коллекция номер один элементов управления, компонентов и утилит WPF для создания приложений Windows следующего поколения. Предоставляет 48 элементов управления, все из которых предлагаются в рамках лицензии сообщества Xceed Software Inc. v4.1.0 содержит 20 исправлений и улучшений. [https://www.nuget.org/packages/Extended.Wpf.Toolkit/]

## 2.3 Описание функциональности программного средства

Описание функциональности программного средства представлено с помощью UML-диаграммы вариантов использования.

Ниже представлена спецификация функциональности программного средства.

## 2.4 Спецификация функциональных требований

В своем приложении необходимым элементом является авторизация и регистрация пользователей, что требует наличия базы данных и связи программного средства с ней. При входе пользователи должны вводить свой email и пароль. При регистрации требуется ввести свой email и дважды ввести желаемый пароль. Если введенные данные проходят валидацию, то они записываются в базу данных и в следующий раз пользователь сможет войти в систему, используя email и пароль, которые он ввел при регистрации.

Данное программное средство поддерживает 2 типа учетных записей:

* администратор;
* пользователь.

При входе в систему администратор может видеть информацию обо всех зарезервированных услугах и изменять дату их записи в случае, если пользователь не смог явиться в указанное им время или же удалять информацию о записи в случае, если пользователь отказался от услуги. Также администратор имеет доступ к информации об услугах, предоставляемых в парикмахерской и продуктах, доступных для резервирования.

В набор возможностей пользователя входит просмотр списка всех доступных услуг и товаров, возможность их добавления в базу данных, носящую функционал похожий на функционал корзины. На странице *«Аккаунт»* пользователь может осуществлять работу со своими резервациями, отменяя их. На всех страницах ПС можно увидеть визуальное отображение роли. Пользователь может просмотреть количество зарезервированных услуг, их общую стоимость, количество зарезервированных товаров и их суммарную стоимость.

Все изменения, вносимые администратором и пользователем, вносятся в базу данных.

# 3 Проектирование и создание программного средства

Проектирование программного средства — процесс создания проекта программного обеспечения. Целью проектировaния является определение внутренних свойств системы и детализации её внешних свойств на основе исходных условий задачи. Этап проектирования также подразумевает анализ исходных условий задачи.

## 3.1 Архитектура системы

В данном программном средстве реализован паттерн MVVM (Model-View-ViewModel). Данный паттерн позволяет отделить логику приложения от визуальной части. Данный паттерн является архитектурным, то есть он задает общую архитектуру приложения.

MVVM состоит из трех компонентов: модели (Model), модели представления (ViewModel) и представления (View).

Model 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.

В моем случае модель реализует интерфейс INotifyPropertyChange, который позволяет сообщать системе об изменениях свойств модели. Благодаря этому облегчается привязка к представлению, причем прямое взаимодействие между моделью и представлением отсутствует.

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.

Хотя окно (класс Window) в WPF может содержать как интерфейс в xaml, так и привязанный к нему код C#, однако в идеале код C# не должен содержать какой-то логики, кроме разве что конструктора, который вызывает метод InitializeComponent и выполняет начальную инициализацию окна. Вся же основная логика приложения выносится в компонент ViewModel.

Однако иногда в файле связанного кода все может находиться некоторая логика, которую трудно реализовать в рамках паттерна MVVM во ViewModel.

Представление не обрабатывает события за редким исключением, а выполняет действия в основном посредством команд.

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.

## 3.2 Модель базы данных

Основой для хранения и взаимодействия данных в проекте служит база данных «CleanCutSalonDB», созданная с помощью встроенных средств взаимодействия с объектами SQL Server. База данных содержит 5 таблиц, выделенных под каждую модель объекта программного средства.

Во-первых, для хранения наиболее функционально важной информации в базе данных, а именно информации о пользователях, используется таблица под названием «Clients». Кроме зарегистрированных Email и пароля, таблица содержит id пользователя, являющийся уникальным для каждого клиента. Ключевым в таблице является столбец Email, который выступает в качестве связующего звена для некоторых запросов в БД.

Для добавления услуг и продуктов нам необходимы поля, включающие в себя данные, с помощью которых они могут быть описаны. Для наглядности к каждой услуге и продукту привязаны иллюстрации, для чего был выделен отдельный столбец в таблицах «Services» и «Products» типа данных nvarchar, хранящего текстовые данные соответственно. Также в случае с услугой важную роль играет цена и продолжительность выполнения – они получили по отдельному столбцу для хранения значений, передаваемых в поля соответствующих моделей. Также каждый товар имеет описание, для которого выделен столбец под названием «Description». Также в таблицах присутствуют идентифицирующие значения в столбце «Id».

В базе данных также присутствуют таблицы «OrderedServices» и «OrderedProducts», предназначенные для хранения идентификаторов основных моделей и связывания данных с помощью них.

В Приложении В представлена взаимосвязь таблиц базы данных.

Таблица «Clients» имеет зависимость в виде первичного ключа «Emails», который сопоставляется с внешним ключом «Login» таблиц «OrderedServices» и «OrderedProducts».

Внешние ключи «ServiceId» и «ProductId» таблиц «OrderedServices» и «OrderedProducts» соответственно связаны с первичными ключами таблицы «Service» и «Product». Все таблицы соединены между собой связями «один ко многим», так как несколько строк из дочерней таблицы зависят от одной строки в родительской таблице.

## 3.3 UML диаграммы

Унифицированный язык моделирования (Unified Modeling Language, UML) – это графический язык для визуализации, специфицирования, конструирования и документирования систем, в которых главная роль принадлежит программному обеспечению.

Диаграмма вариантов использования (Use Case Diagram) демонстрирует набор вариантов использования и действующих лиц (которые являются специальным видом классов), а также их связи. Диаграммы этого типа описывают статическое представление вариантов использования системы. Особенно важны для организации и моделирования поведения системы.

Диаграмма классов (Class Diagram) показывает набор классов, интерфейсов и коопераций, а также их связи. Диаграммы этого вида чаще всего используются для моделирования объектно-ориентированных систем. Предназначены для статического представления системы. Диаграммы классов, включающие активные классы, представляют статическое представление процессов системы.

Диаграмма классов представлена в Приложении A.

Диаграмма последовательности (Sequence Diagram) – это разновидность диаграммы взаимодействия, показывающая временную последовательность сообщений.

Диаграмма последовательности для данного программного средства представлен в Приложении Б.

База данных для этого приложения состоит из 5 таблиц, которые связаны между собой с помощью первичных и внешних ключей. Наглядно схема базы данных и ее связи представлены в приложении В.

## 3.3 Структура проекта и выполняемые функции

Данное программное средство предоставляет минимальный набор требований для полноценной работы подписочного книжного сервиса и выполняет следующие функции:

* авторизация со стороны администратора;
* добавление/удаление/редактирование информации об услугах со стороны администратора;
* добавление/удаление/редактирование информации о продуктах со стороны администратора;
* редактирование информации о записях клиентов со стороны администратора;
* резервирование услуг со стороны пользователя;
* заказ товаров со стороны пользователя;
* регистрация и авторизация со стороны пользователя.

Внутренняя структура проекта отображена на Рисунке 3.1.

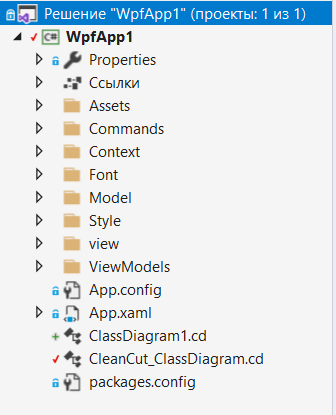


Рисунок 3.1 – Структура проекта

Папка «Command» содержит класс, реализующий паттерн Command для работы с событиями в модели MVVM.

Папка «Assets» содержит используемые при верстке иконки и изображения.

Папка «Context» содержит контексты БД (реализация Entity CRUD).

Папка «Font» содержит встраиваемые шрифты.

Папка «Model» содержит модели объектов приложения.

Папка «Style» содержит словари стилей для некоторых представлений.

Папка «view» содержит представления приложения.

Папка «ViewModels» содержит все «вьюмодели» приложения.

## 3.4 Пример использования

Диаграмма использования «Авторизованный пользователь» и «Администратор» в моем случае выглядит следующим образом (Рисунок 3.2).

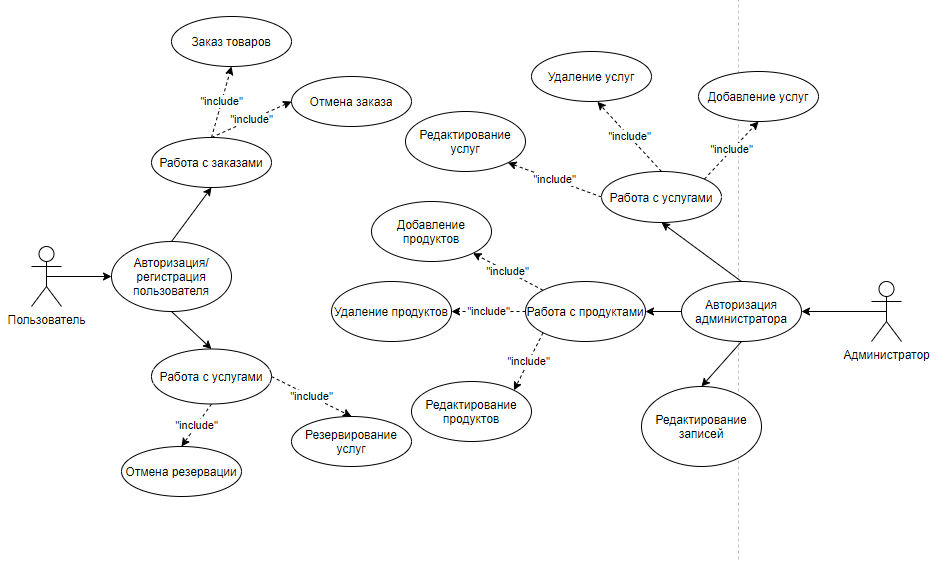


Рисунок 3.2 – Диаграмма использования ПС

# 4 Реализация программного средства

## 4.1 Реализация паттерна MVVM

Паттерн MVVM реализуется в приложение посредством создания базовой «вьюмодели» BaseViewModel (Рисунок 4.1), которая наследует интерфейс INotifyPropertyChanged.

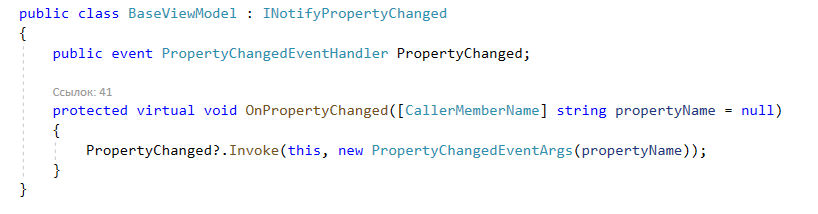


Рисунок 4.1 – Базовая модель представления BaseViewModel

Все остальные модели представления приложения наследуют этот класс (Рисунок 4.2). Реализация данного интерфейса позволяет сделать привязку данных для получения данных из представления в модель представления и наоборот. Делается это реализацией метода интерфейса INotifyPropertyChanged – OnPropertyChanged().

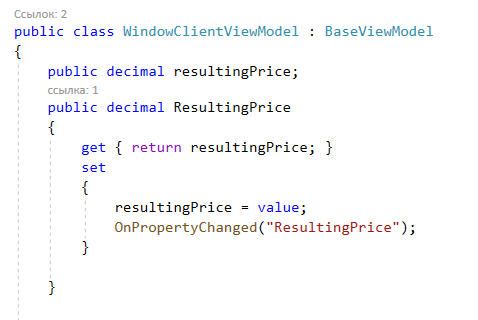


Рисунок 4.2 – Модель представления, которая наследует BaseViewModel и использует метод OnPropertyChanged()

Для взаимодействия пользователя и приложения в MVVM используются команды. Команды реализованы с помощью интерфейса ICommand (Рисунок 4.3) и является по сути реализацией поведенческого паттерна Команда.

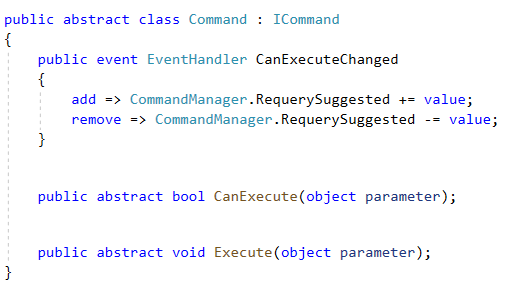


Рисунок 4.3 – Реализация интерфейса ICommand классом Command

## 4.2 Реализация авторизации и регистрации

Данное программное средство требует обязательной авторизации пользователей для возможности взаимодействия со своим функционалом. Если у пользователя нет учетной записи приложения, то он может зарегистрироваться, а в случае утраты пароля, воспользоваться восстановлением пароля.

В Приложении Г приведен код, реализующий вход пользователя в приложение.

При входе происходит валидация введенных данных, поиск пользователя в базе данных по Email и, в случае успешного поиска, заходим в систему.

В Приложении Д приведен листинг метода регистрации в приложении.

При регистрации все поля проходят валидацию и в случае успеха данные нового пользователя заносятся в таблицу БД.

## 4.3 Резервирование услуги

Процесс резервирования услуги происходит посредством перенесения данных из экземпляра ObservableCollection, используемой для отображения услуг на странице в соответствующую таблицу базы данных с помощью CRUD. При этом происходит комплексная валидация выбранной даты записи. Листинг кода исполняемого в процессе резервирования представлен в Таблице 4.1.

Таблица 4.1. Резервирование услуги

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Collections.ObjectModel;  using System.Linq;  using System.Text;  using System.Text.RegularExpressions;  using System.Threading.Tasks;  using System.Windows;  using System.Windows.Input;  using WpfApp1.Commands;  using WpfApp1.Model;  using WpfApp1.view;  using WpfApp1.view.Client.Buttons;  using WpfApp1.view.Client.Buy;  using WpfApp1.ViewModels.Base;  namespace WpfApp1.ViewModels.Client  {  public class OrderViewModel : BaseViewModel  {  public ICommand OrderService { get; private set; }  public ICommand Back { get; private set; }  private string selectedTime;  public string SelectedTime  {  get { return selectedTime; }  set  {  selectedTime = value;  OnPropertyChanged("SelectedTime");  }  }  public string logins;  public OrderViewModel(/\*Service init,\*/ string login)  {  logins = login;  Services = new ObservableCollection<Service>(App.db.Services);  OrderService = new RelayCommand(go\_order);  Back = new RelayCommand(go\_back);  }  private ObservableCollection<Service> services;  public ObservableCollection<Service> Services  {  get { return services; }  set  {  services = value;  OnPropertyChanged("Products");  }  }    private Service selectedService;  public Service SelectedService  {  get { return selectedService; }  set  {  selectedService = value;  OnPropertyChanged("SelectedProduct");  }  }    private string title\_box;  public string Title\_Box  {  get { return title\_box; }  set  {  title\_box = value;  OnPropertyChanged("Title\_Box");  }  }  private DateTime date\_box;  public DateTime Date\_Box  {  get { return date\_box; }  set { date\_box = value; }  }  private void go\_order(object sender)  {  DateTime oldDate = Date\_Box;  DateTime endTime = new DateTime(1, 1, 1, 20, 0, 0);  DateTime startTime = new DateTime(1, 1, 1, 8, 0, 0);  Regex timeValidation = new Regex("^(0[0-9]|1[0-9]|2[0-3]|[0-9]):[0-5][0-9]$");  if (!timeValidation.IsMatch(SelectedTime))  {  System.Windows.Forms.MessageBox.Show("Дурак, цифры научись вводить");  Date\_Box = oldDate;  return;  }  DateTime curDate = DateTime.Now;  bool flag = true;  string[] date = SelectedTime.Split(':');  Date\_Box = Date\_Box.AddHours(int.Parse(date[0]));  Date\_Box = Date\_Box.AddMinutes(int.Parse(date[1]));  DateTime dateTime = Date\_Box;  DateTime dateTimeFromService;  int duration = 0;  if (SelectedService != null)  {  duration = SelectedService.DurationInMinutes;  }  else flag = false;    dateTime = dateTime.AddMinutes(duration);  endTime = endTime.AddYears(Date\_Box.Year - 1);  endTime = endTime.AddMonths(Date\_Box.Month -1);  endTime = endTime.AddDays(Date\_Box.Day - 1);  startTime = startTime.AddYears(Date\_Box.Year - 1);  startTime = startTime.AddMonths(Date\_Box.Month - 1);  startTime = startTime.AddDays(Date\_Box.Day - 1);  if ((Date\_Box - startTime).TotalMinutes >= 0 && (dateTime - endTime).TotalMinutes <= 0)  {  foreach (var service in App.db.OrderedServices)  {  dateTimeFromService = service.DayReserv;  dateTimeFromService = dateTimeFromService.AddMinutes(SelectedService.DurationInMinutes);  if ((Date\_Box > service.DayReserv && Date\_Box >= dateTimeFromService) ||  (dateTime <= service.DayReserv && dateTime < dateTimeFromService))  {  }  else  {  flag = false;  Date\_Box = oldDate;  break;  }  }    }  else  {  MessageBox.Show("В выбранное вами время парикмахерская не работает!\nВыберите другое время!");  Date\_Box = oldDate;  return;  }  if (Date\_Box > DateTime.Now && flag)  {  try  {  OrderedService orderedService = new OrderedService  {  DayReserv = Date\_Box,  Login = logins,  ServiceId = SelectedService.Id    };  App.db.OrderedServices.Add(orderedService);  App.db.SaveChanges();  }  catch (Exception ex)  {  System.Windows.Forms.MessageBox.Show(ex.Message);  }  finally  {  Date\_Box = oldDate;  WindowClient winadm = new WindowClient(logins);  MessageBox.Show("Запись проведена успешно!!");  foreach (Window win in Application.Current.Windows)  {  if (win is Order)  {  win.Close();  }  }  winadm.Show();  }  }  else  {  MessageBox.Show("На это время уже забронировано место!\nВыберите другой вариант!");  }  }  private void go\_back(object sender)  {  WindowClient winadm = new WindowClient(logins);  foreach (Window win in Application.Current.Windows)  {  if (win is Order)  {  win.Close();  }  }  winadm.Show();  }  }  } |

## 4.4 Работа администратора с услугами

Для администратора доступно удаление, создание и изменение услуг. В случае с созданием мы обращаемся к базе данных, при этом записывая в строку нужной нам таблицы соответствующие данные. В случае же с удалением и редактированием мы передаем значения, полученные из запроса Entity в переменную типа var, затем обращаясь к полям получаемой модели данных, и присваиваем им новые значения, полученные из свойств, привязанных к элементам управления представления. Листинг с примером кода добавления приведен ниже в Таблице 4.2.

Таблица 4.2. Добавление услуги

|  |
| --- |
| using Microsoft.Win32;  using System;  using System.Windows;  using System.Windows.Input;  using WpfApp1.Commands;  using WpfApp1.Model;  using WpfApp1.view.Admin;  using WpfApp1.view.Admin.Buttons;  using WpfApp1.ViewModels.Base;  namespace WpfApp1.ViewModels  {  public class AddViewModel : BaseViewModel  {  public ICommand Add\_Service { get; private set; }  public ICommand GoBack\_AddView { get; private set; }  public ICommand Select\_Image { get; private set; }  #region Fields and Properties  private string title\_box;  public string Title\_Box  {  get { return title\_box; }  set { title\_box = value;  OnPropertyChanged("Title\_Box");  }  }  private decimal cost\_box;  public decimal Cost\_Box  {  get { return cost\_box; }  set  {  cost\_box = value;  OnPropertyChanged("Cost\_Box");  }  }  private int time\_box;  public int Time\_Box  {  get { return time\_box; }  set  {  time\_box = value;  OnPropertyChanged("Time\_Box");  }  }    private string imagepath;  public string ImagePath  {  get { return imagepath; }  set { imagepath = value; }  }  #endregion  public AddViewModel()  {  Add\_Service = new RelayCommand(go\_add);  GoBack\_AddView = new RelayCommand(go\_back);  Select\_Image = new RelayCommand(select\_image);  }  private void go\_add(object obj)  {  if (ImagePath == null || Title\_Box == null || Cost\_Box.ToString() ==null || Time\_Box.ToString() == null)  {  MessageBox.Show("Выберите изображение или заполните пустые поля!");    }    else  {  if (Cost\_Box == 0 || Time\_Box == 0)  {  MessageBox.Show("Проверьте правильность введенных данных о цене и времени!");  }  else  {  try  {  App.db.Services.Add(new Service { Title = Title\_Box, Cost = Cost\_Box, DurationInMinutes = Time\_Box, MainImagePath = ImagePath });  App.db.SaveChanges();  }  catch (Exception ex)  {  System.Windows.Forms.MessageBox.Show(ex.Message);  }  finally  {  MessageBox.Show("Услуга успешно добавлена!");  WindowAdminService winadm = new WindowAdminService();  foreach (Window win in Application.Current.Windows)  {  if (win is Add)  {  win.Close();  }  }  winadm.Show();  }  }  }    }    private void select\_image(object sender)  {  try  {  OpenFileDialog openFileDialog = new OpenFileDialog();  if (openFileDialog.ShowDialog() == true)  {  imagepath = openFileDialog.FileName;  }  }  catch (Exception ex)  {  System.Windows.Forms.MessageBox.Show(ex.Message);  }  finally  {  Console.WriteLine("path" + imagepath);  }    }  private void go\_back(object sender)  {  try  {  WindowAdminService winadm = new WindowAdminService();  foreach (Window win in Application.Current.Windows)  {  if (win is Add)  {  win.Close();  }  }  winadm.Show();  }  catch (Exception ex)  {  System.Windows.Forms.MessageBox.Show(ex.Message);  }  }  }  } |

## 4.5 Резервация услуги пользователем

Для администратора доступно удаление, создание и изменение услуг. В случае с созданием мы обращаемся к базе данных, при этом записывая в строку нужной нам таблицы соответствующие данные. В случае же с удалением и редактированием мы передаем значения, полученные из запроса Entity в переменную типа var, затем обращаясь к полям получаемой модели данных, и присваиваем им новые значения, полученные из свойств, привязанных к элементам управления представления. Листинг с примером кода добавления приведен ниже в Таблице 4.2.

# 5 Тестирование, проверка работоспособности и анализ полученных результатов

Для оценки удобства использования программного средства и проверки его работоспособности необходимо провести симуляцию критических ситуаций, в которых пользователь допускает какого-либо рода ошибки.

Один из самых важных параметров для тестирования – проверка всех текстовых полей. При введении некорректных данных в поля, приложение не должно ломаться и вылетать. Если данные в полях неверные, то нужно выводить соответствующие сообщения для уведомления пользователя.

Примеры ввода некорректных данных в окне входа представлены на Рисунках 5.1 – 5.3.

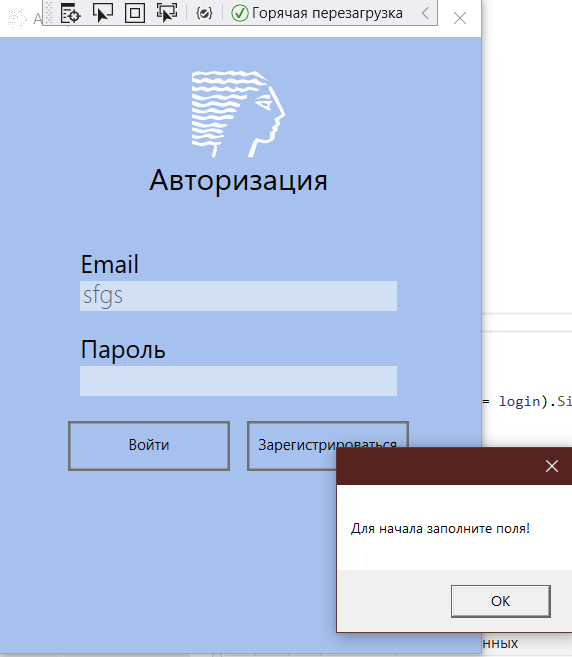


Рисунок 5.1 – Пустые строки в форме авторизации

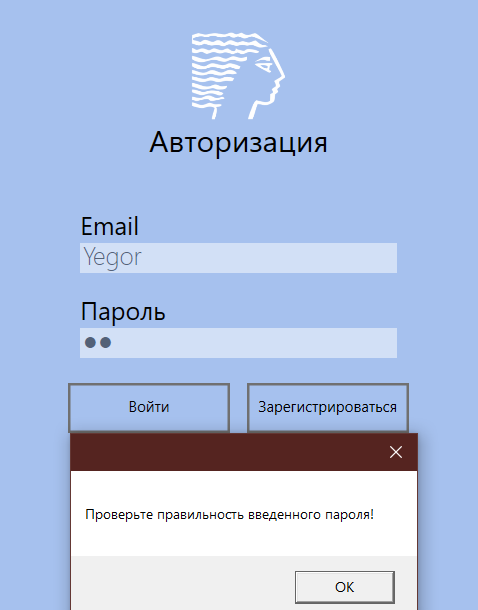


Рисунок 5.2 – Введен неверный пароль

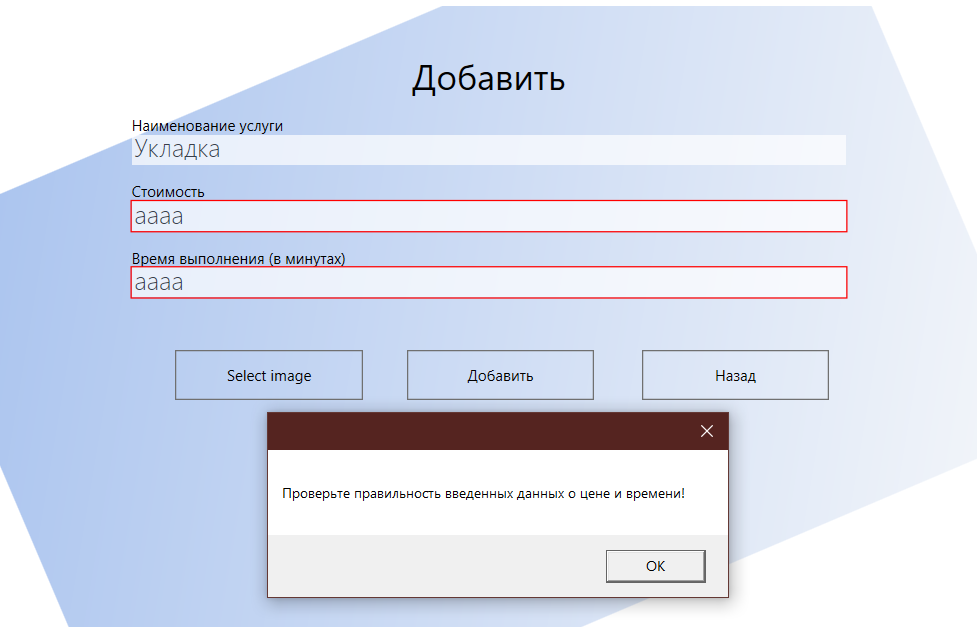


Рисунок 5.3 – Попытка ввести строковые значения в числовые поля

Проверка формы для входа показала, что валидация работает успешно, все возможные некорректные данные обрабатываются и не вызывают ошибок со стороны приложения. Идентичным образом работает валидация в форме регистрации.

Далее протестируем более сложный ЭУ, а именно календарь. Все возможные ошибочные операции показаны на Рисунках 5.4 – 5.6.

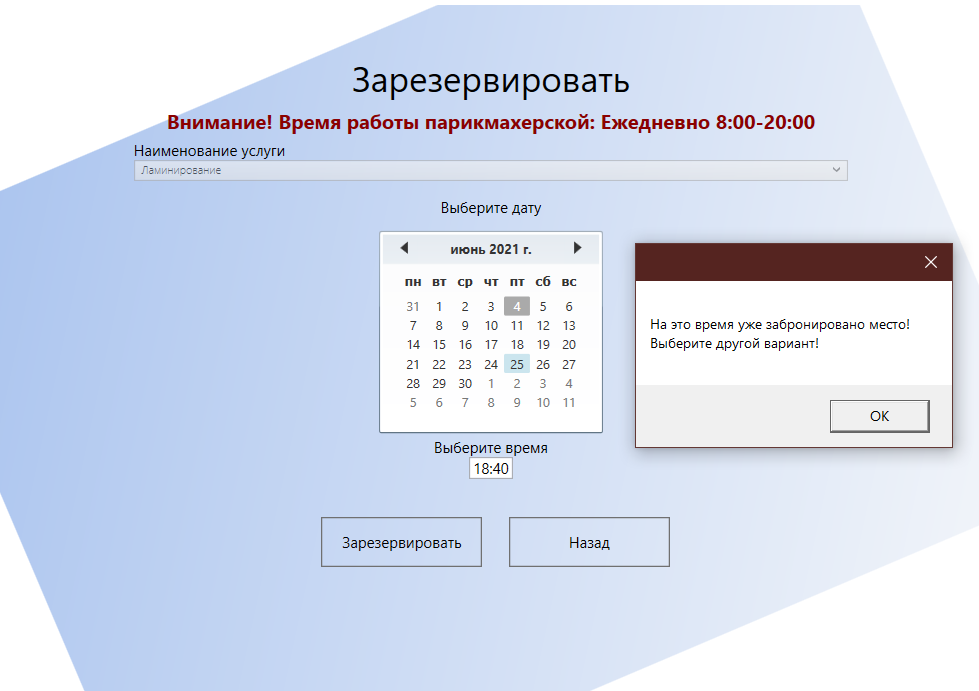


Рисунок 5.4 – Выбираемые дата и время уже заняты

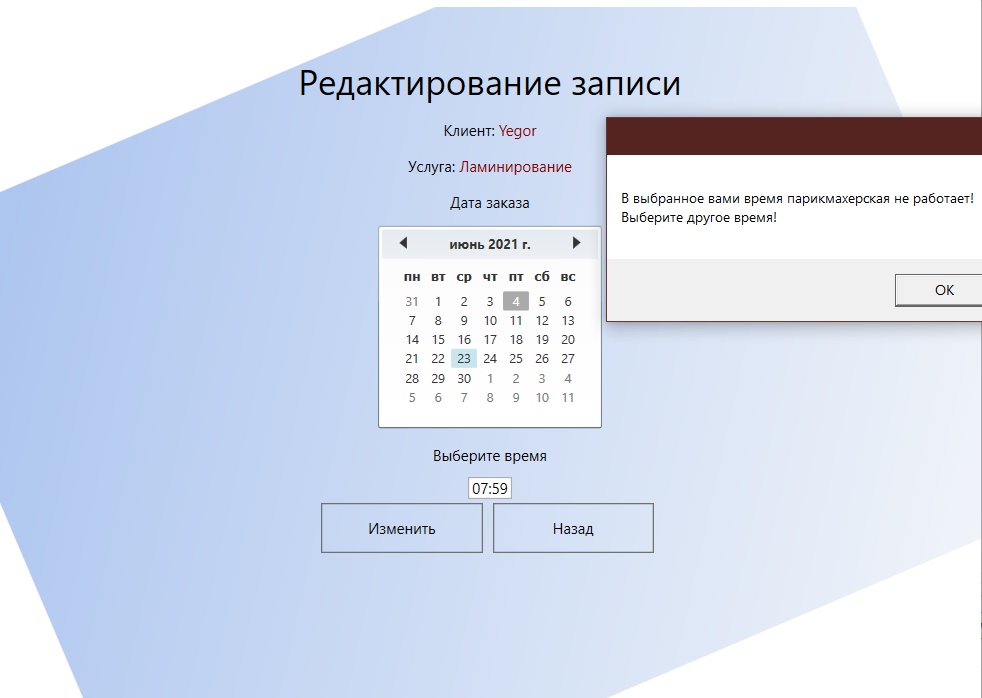


Рисунок 5.5 – В выбираемое время парикмахерская закрыта

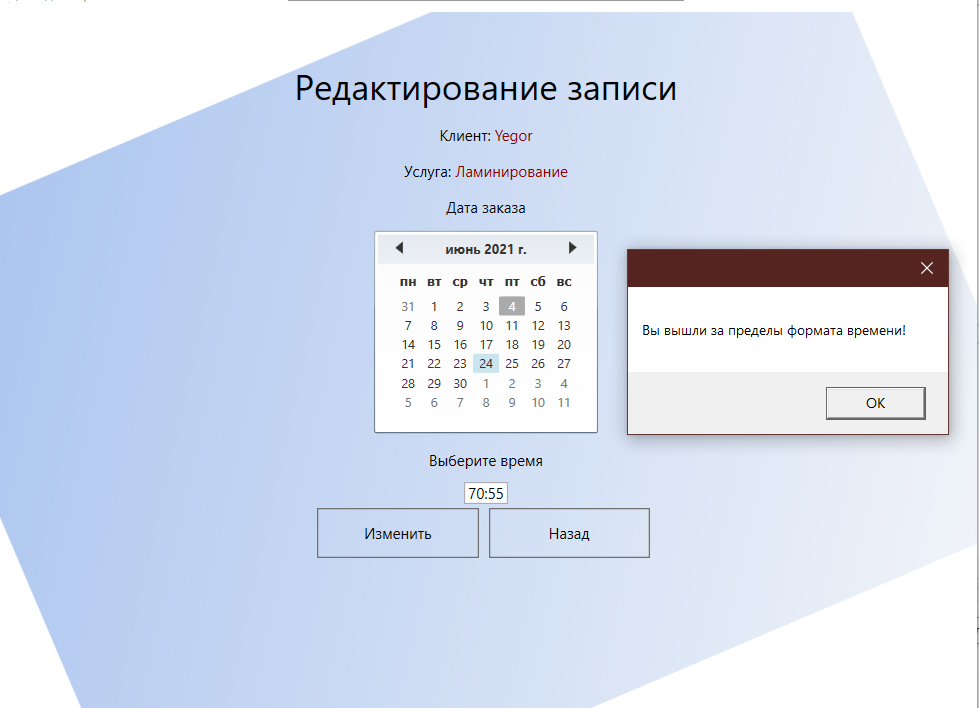


Рисунок 5.6 – Несовпадение введенных данных с форматом времени

Для проверки используются математические алгоритмы, а также регулярные выражения.

# 

# 6 Руководство по использованию

После запуска приложения необходимо авторизоваться. Если у вас нет учетной записи, то открыв окно с формой для регистрации, вы можете создать аккаунт пользователя (Рисунок 6.1).

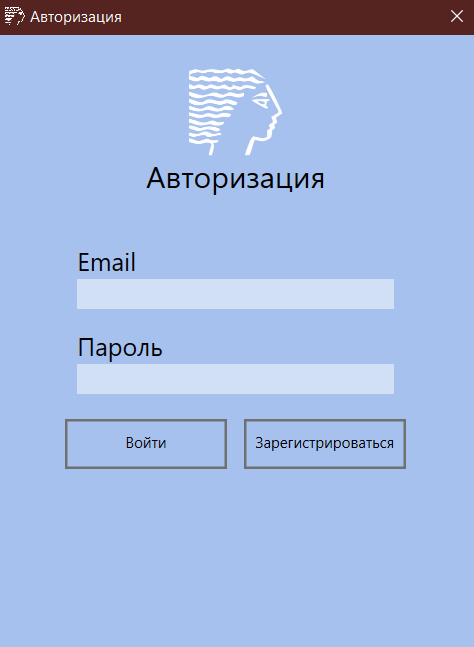


Рисунок 6.1 – Форма для регистрации

Когда мы вошли под одной из возможных учетных записей двух представленных в проекте ролей, перед нами появляется главная страница приложения. В случае с администратором – это окно c функциональными кнопками, надписи которых сообщают нам о том, что мы сможем сделать, если нажмем на соответствующую кнопку (Рисунок 6.2). В случае с авторизацией пользователя, он увидит список всех услуг, доступных для резервирования (Рисунок 6.3).

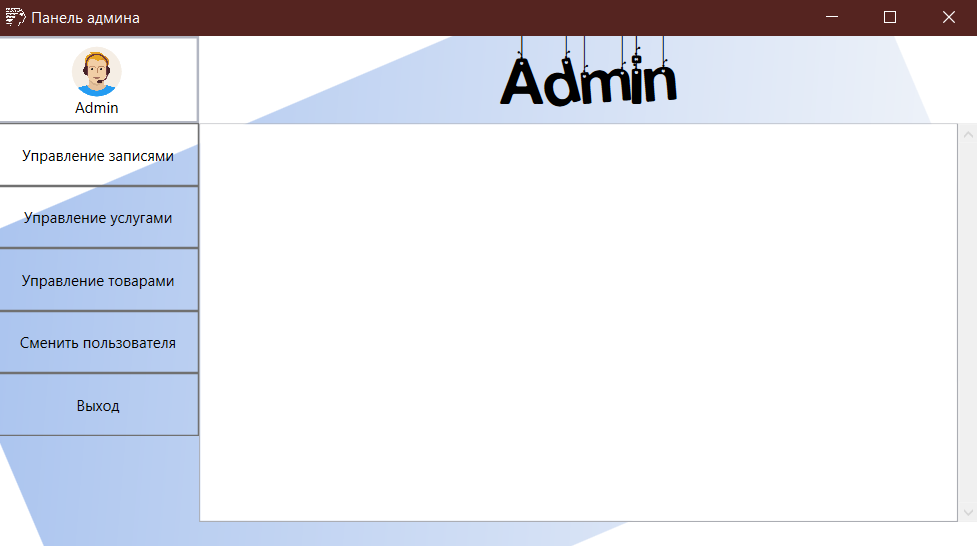


Рисунок 6.2– Главная страница для администратора

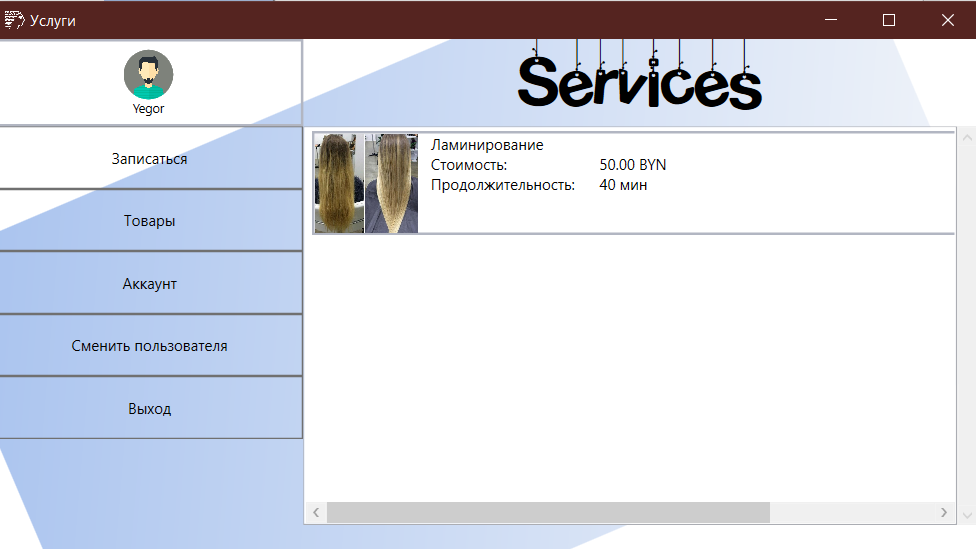


Рисунок 6.3 – Главная страница пользователя

Администратор, переходя на страницу «Управления записями», «Управления услугами» и «Управления товарами» встречает примерно одинаковый функционал из трех операций «Удаление», «Добавление» и «Редактирование». В ЭУ «ListBox» WPF, расположенном на данных страницах справа выводится наиболее важная информация о соответствующих рассматриваемых объектах. Нажимая на представление объекта, мы выбираем его для последующего редактирования или удаления соответственно. (Рисунок 6.4 – 6.5).

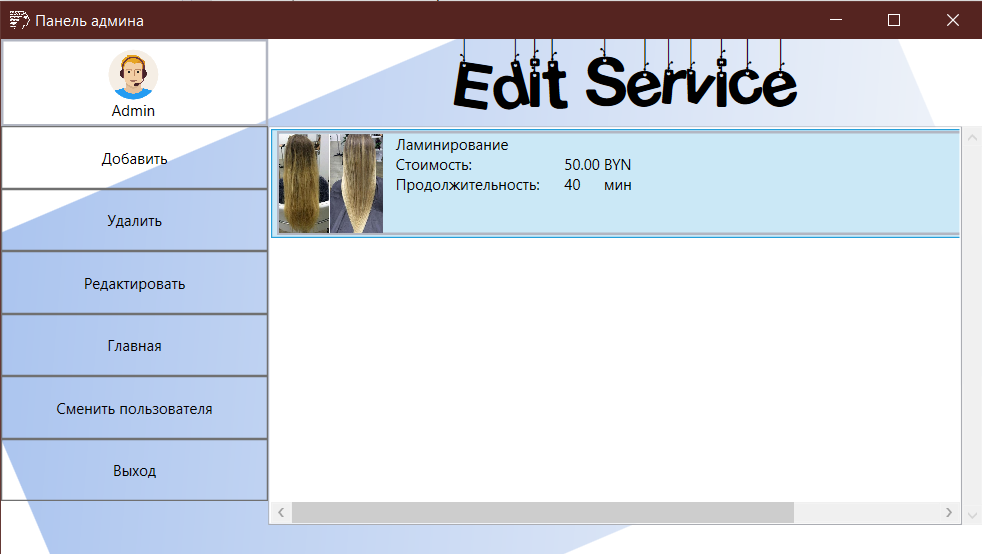


Рисунок 6.4 – Страница управления услугами администратора

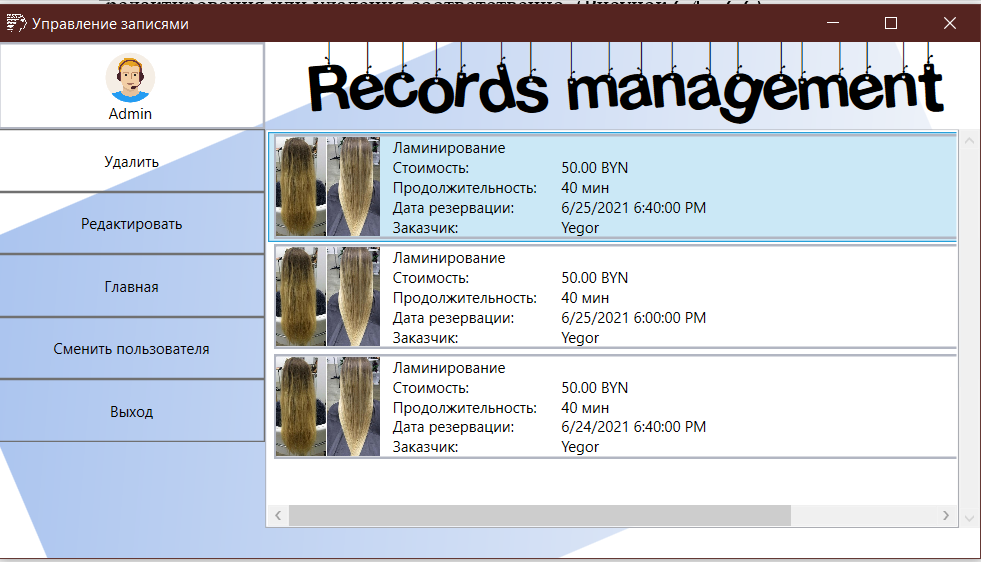


Рисунок 6.5 – Страница управления записями администратора

Пользователи могут отслеживать резервацию как товаров, так и услуг, перейдя на вкладку «Аккаунт». Здесь отображаются дата резервации, стоимость услуги и продолжительность, а также суммарная стоимость всех зарезервированных услуг. Также здесь можно найти информацию о зарезервированных товарах, об их суммарной стоимости. Кроме того, резервацию товара и услуги можно отменить. (Рисунок 6.6).

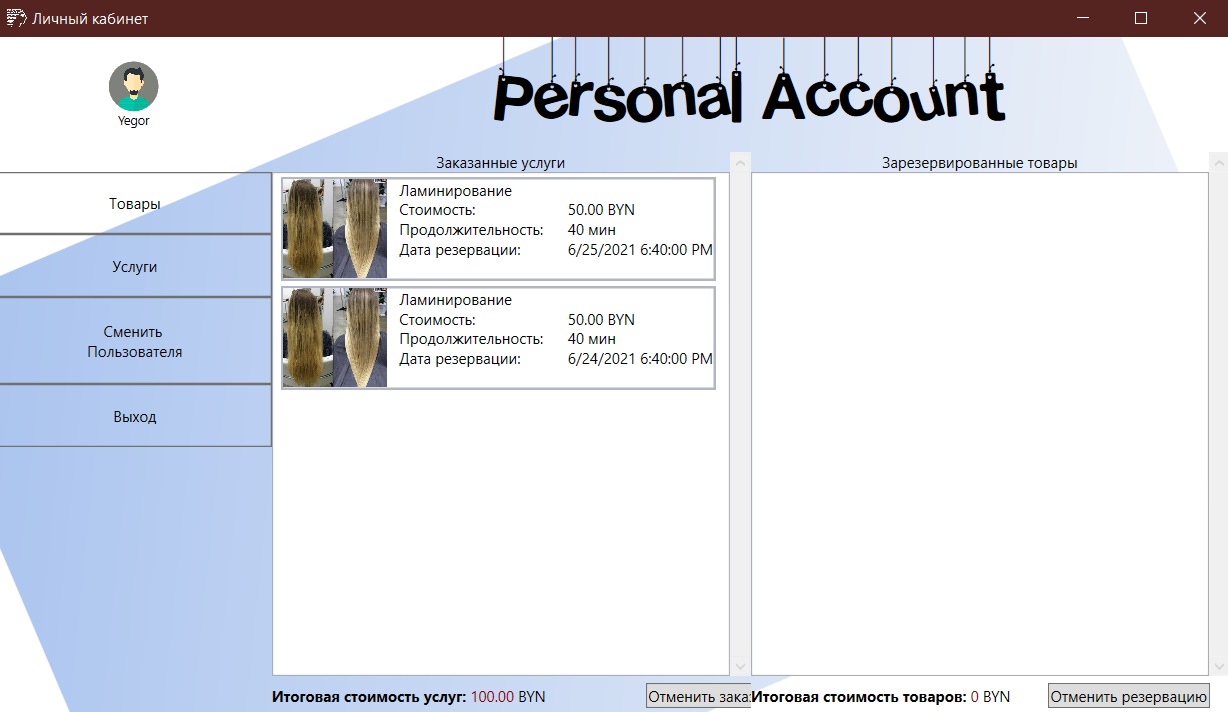


Рисунок 6.16 – Личный кабинет

Общей для двух ролей является возможность смены пользователя при нахождении на любой из функциональных страниц программного средства.

# Заключение

В ходе выполнения курсового проекта было разработано программное средство «CleanCut» на языке C# с использованием технологий Entity Framework, WPF.

При разработке данного приложения были выполнены все пункты из указанного списка предполагаемого основного функционала приложения, а именно:

* составлена база данных;
* сделано разграничение на 2 типа учетных записей;
* реализован функционал для администрирования базы данных администратором;
* реализован функционал для резервирования товаров и услуг пользователем;
* реализован функционал для процесса регистрации;

Результаты проведенного тестирования не выявили нарушений в работе программы, так что приложение можно назвать стабильным.

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

**Список использованных источников**

1. Сервисы онлайн-записи // [Электронный ресурс]. – Режим доступа: <https://postium.ru/programmy-i-servisy-onlajn-zapisi-klientov-10-luchshix/> – Дата доступа: 20.03.2021.

2. Microsoft Visual Studio // [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio> – Дата доступа: 15.05.2021.

3. Руководство по Entity Framework // [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/entityframework/ – Дата доступа: 27.04.2021.

4. Добро пожаловать в интегрированную среду разработки Visual Studio // [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2019 – Дата доступа: 17.03.2021.

5. A tour of the C# language // [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/ – Дата доступа: 27.04.2021.

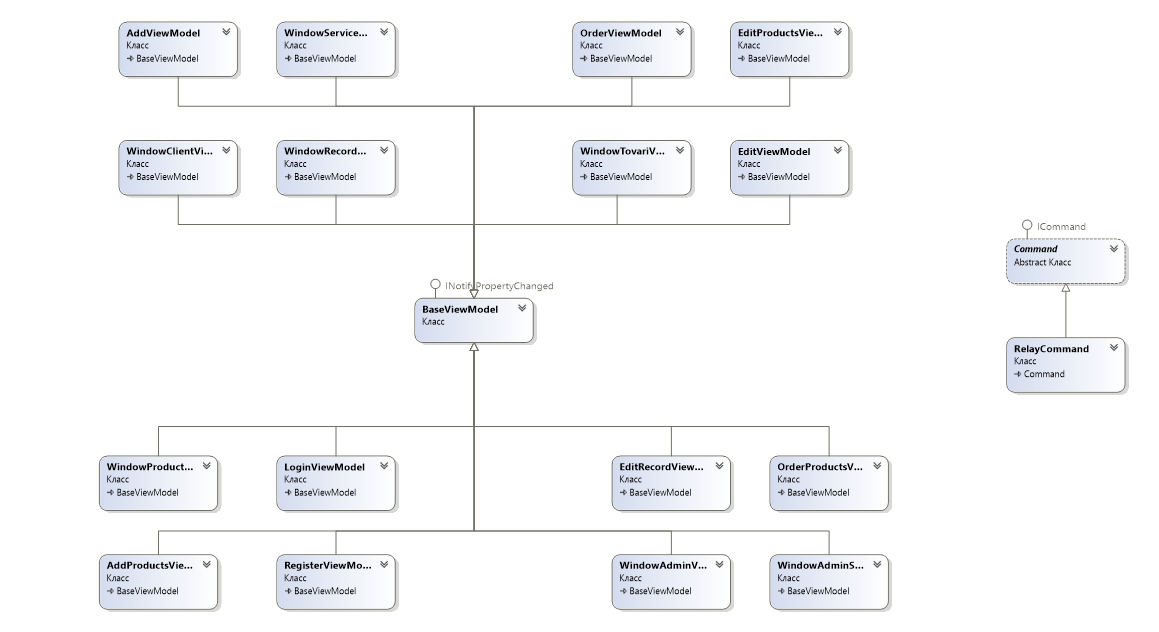
6. Get started with WPF // [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/en-us/visualstudio/designers/getting-started-with-wpf?view=vs-2019 – Дата доступа: 20.05.2021.

7. Обзор XAML (WPF .NET) // [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/desktop/wpf/xaml/?view=netdesktop-5.0&viewFallbackFrom=netdesktop-5.0 – Дата доступа: 20.05.2021.

8. Сравнение EF Core и EF6 // [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/ef/efcore-and-ef6/ – Дата доступа: 27.04.2021.

# Приложение А

**Диаграмма классов**



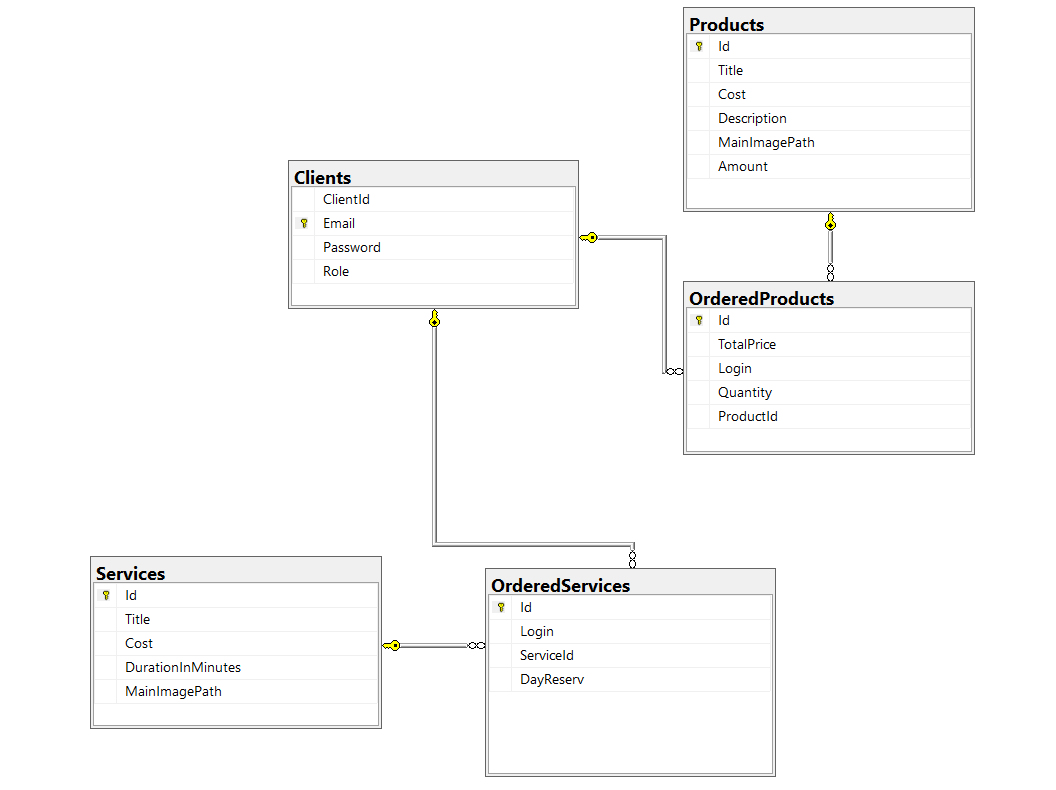
# Приложение Б

**Диаграмма последовательности**

# 

# Приложение В

**Диаграмма базы данных CleanCutDB**



# Приложение Г

**Реализация входа в приложение**

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Data.SqlClient;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows;  using System.Windows.Input;  using WpfApp1.Commands;  using WpfApp1.view;  using WpfApp1.ViewModels.Base;  namespace WpfApp1.ViewModels.Client  {  public class LoginViewModel : BaseViewModel  {  public ICommand Login { get; private set; }  public ICommand Register { get; private set; }  private string email\_box;  public string Email\_Box  {  get { return email\_box; }  set { email\_box = value; }  }  private string password\_box;  public string Password\_Box  {  get { return password\_box; }  set { password\_box = value; }  }  public LoginViewModel()  {  Login = new RelayCommand(Button\_Click);  Register = new RelayCommand(Button\_Click\_1);  }  private void Button\_Click(object sender)  {    if (Email\_Box != null && Password\_Box != null)  {  string login = Email\_Box.Trim();  string password = Password\_Box.Trim();  try  {  //Model.Client entity = null;  var entity = App.db.Clients.Where(x => x.Email == login && x.Password == password || x.Email == login).SingleOrDefault();  if (entity != null)  {  if (entity.Password == password)  {  if (entity.Role == 0)  {  WindowService client = new WindowService(login);  foreach (Window win in Application.Current.Windows)  {  if (win is Login)  {  win.Close();  }  }  client.Show();  }  else  {  //Model.Client adminentity = null;  WindowAdmin windowCompany = new WindowAdmin();  foreach (Window win in Application.Current.Windows)  {  if (win is Login)  {  win.Close();  }  }  windowCompany.Show();  }  }  else  {  MessageBox.Show("Проверьте правильность введенного пароля!");  }  }  else  {  MessageBox.Show($"Пользователя с логином \"{login}\" не существует!");  }  }  catch (SqlException)  {  MessageBox.Show("Отсутствует подключение с базой данных");  }  }  else  {  MessageBox.Show("Для начала заполните поля!");  }  }    private void Button\_Click\_1(object sender)  {  Register reg = new Register();  reg.Show();  foreach (Window win in Application.Current.Windows)  {  if (win is Login)  {  win.Close();  }  }  }  }    } |

# Приложение Д

**Реализация регистрации в приложении**

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Security;  using System.Text;  using System.Threading.Tasks;  using System.Windows;  using System.Windows.Input;  using WpfApp1.Commands;  using WpfApp1.ViewModels.Base;  using WpfApp1.Model;  using System.Text.RegularExpressions;  namespace WpfApp1.ViewModels.Client  {  public class RegisterViewModel : BaseViewModel  {  public ICommand RegisterButton { get; private set; }  public ICommand LoginButton { get; private set; }  public RegisterViewModel()  {  RegisterButton = new RelayCommand(on\_register);  LoginButton = new RelayCommand(on\_login);  }  private string email\_box;  public string Email\_Box  {  get { return email\_box; }  set { email\_box = value;  OnPropertyChanged("Email\_Box");  }  }  private string firstPasswordBox;  public string FirstPasswordBox  {  get { return firstPasswordBox; }  set { firstPasswordBox = value;  OnPropertyChanged("FirstPassword\_Box");  }  }  private string secondPasswordBox;  public string SecondPasswordBox  {  get { return secondPasswordBox; }  set { secondPasswordBox = value;  OnPropertyChanged("SecondPassword\_Box");  }  }    private void on\_login(object sender)  {  Login log = new Login();  log.Show();  foreach (Window win in Application.Current.Windows)  {  if (win is Register)  {  win.Close();  }  }  }  private void on\_register(object sender)  {  //bool flag = true;  Regex complexPassword = new Regex("(?=.\*[0-9])(?=.\*[!@#$%^&\*])(?=.\*[a-z])(?=.\*[A-Z])[0-9a-zA-Z!@#$%^&\*]{6,}");  Regex complexEmail = new Regex(@"^([\w\.\-]+)@([\w\-]+)((\.(\w){2,3})+)$");    /// <summary>  /// Свойства, в которые передаются значения для полей из окна  /// </summary>  Email\_Box = Email\_Box.Trim();  FirstPasswordBox = FirstPasswordBox.Trim();  SecondPasswordBox = SecondPasswordBox.Trim();  if (!complexPassword.IsMatch(FirstPasswordBox))  {  MessageBox.Show("Пароль должен быть в длину более 6 символов, содержать цифры, спец символы, латинские буквы в верхнем и нижнем регистре");  FirstPasswordBox = null;  return;  }  if (!complexEmail.IsMatch(Email\_Box))  {  MessageBox.Show("Введите настоящий E-mail адрес!");  Email\_Box = null;  return;  }  if (Email\_Box == null || FirstPasswordBox == null || SecondPasswordBox == null)  {  MessageBox.Show("Заполните все поля!");  }  else  {  if (FirstPasswordBox == SecondPasswordBox)  {  /// <summary>  /// Валидация всех ЭУ в окне  /// </summary>  try  {  App.db.Clients.Add(new Model.Client { Email = Email\_Box, Password = FirstPasswordBox, Role = 0 });  App.db.SaveChanges();  }  catch  {  }  finally  {  Login login = new Login();  foreach (Window win in Application.Current.Windows)  {  if (win is Register)  {  win.Close();  }  }  login.Show();  }  }  else  {  MessageBox.Show("Пароли должны совпадать!");  }  }  }  }  } |

# Приложение Е