**Содержание**

[**Введение** 2](#_Toc185397361)

[1. Аналитический обзор прототипов и литературных источников 3](#_Toc185397362)

[Обзор аналогов 3](#_Toc185397363)

[1.1.1. Аналог – YouDo 3](#_Toc185397364)

[1.1.2. Аналог - Yandex Услуги 5](#_Toc185397365)

[1.1.3. Аналог - TaskRabbit 6](#_Toc185397366)

[2. Спецификация бизнес-требований к программному средству 8](#_Toc185397367)

[2.2 Спецификация пользовательских требований к программному средству 8](#_Toc185397368)

[2.3 Спецификация функциональных требований к программному средству 9](#_Toc185397369)

[**3. Проектирование программного средства** 10](#_Toc185397370)

[3.1 Проектирование архитектуры приложения 10](#_Toc185397371)

[**4. Проектирование программного средства** 14](#_Toc185397372)

[4.1 Реализация архитектуры MVVM 14](#_Toc185397373)

[4.2 Реализация классов Models и ViewModels 15](#_Toc185397374)

[4.3 Реализация паттерна Command 16](#_Toc185397375)

[4.4 Реализация подключения к БД и паттерны Repository и UnitOfWork 16](#_Toc185397376)

[4.5 Реализация вспомогательных классов и методов 16](#_Toc185397377)

[4.6 Реализация Views 18](#_Toc185397378)

[**5. Тестирование, проверка работоспособности и анализ полученных результатов** 19](#_Toc185397379)

[5.1 Тестирование авторизации и регистрации 19](#_Toc185397380)

[5.2 Тестирование функций администратора 21](#_Toc185397381)

[5.2 Тестирование функций пользователя 22](#_Toc185397382)

[**6. Руководство по использованию** 25](#_Toc185397383)

[**Заключение** 31](#_Toc185397384)

[**Список литературы** 32](#_Toc185397385)

[**Приложение А** 33](#_Toc185397386)

[**Приложение Б** 33](#_Toc185397387)

[**Приложение В** 34](#_Toc185397388)

[**Приложение Г** 35](#_Toc185397389)

[**Приложение Д** 37](#_Toc185397390)

[**Приложение Е** 39](#_Toc185397391)

# **Введение**

Основной целью курсового проекта является разработка программного средства для электронной площадки по предоставлению услуг. Данное программное средство позволит клиентам легко и быстро находить необходимые услуги, а также выполнять заказ прямо из дома или офиса, экономя время и средства на личные визиты. Площадка будет обеспечивать доступ к более широкому ассортименту услуг, чем традиционные способы поиска, а также предоставлять возможность онлайн-оплаты.

В основу проектирования легли следующие принципы:

1. Легкость использования. Приложение должно быть интуитивно понятным и простым в использовании, даже для тех пользователей, которые не имеют опыта покупок в интернете.
2. Безопасность. Приложение должно обеспечивать защиту персональной информации.
3. Оперативность. Приложение должно быстро открываться и быстро реагировать на действия пользователя.
4. Многофункциональность. Приложение должно предоставлять множество возможностей для пользователя, включая поиск, заказ услуг, быстрое выполнение и удобный способ оплаты, а также возможность отслеживания статуса и возможность связаться со службой поддержки.
5. Надежность. Приложение должно быть надежным и стабильным, чтобы пользователи могли заказывать услуги без каких-то проблем или сбоев.

Программного средство для управления площадкой по предоставлению услуг должно реализовывать все вышеперечисленные принципы и решать поставленные задачи. При выполнении курсового проекта будут использованы язык программирования С#, принципы и приемы ООП, база данных MS SQL Server и технология Windows Presentation Foundation (WPF).

Для успешной реализации курсового проекта необходимо:

* провести анализ соответствующей литературы;
* ознакомиться с прототипами программных средств выбранной мной темы;
* определить функциональные требования;
* продумать структуру базы данных;
* продумать структуру проекта;
* реализовать программное средство;
* протестировать программное средство;
* написать руководство пользователя.

Содержание данной пояснительной записки отражает все этапы выполнения моего курсового проекта.

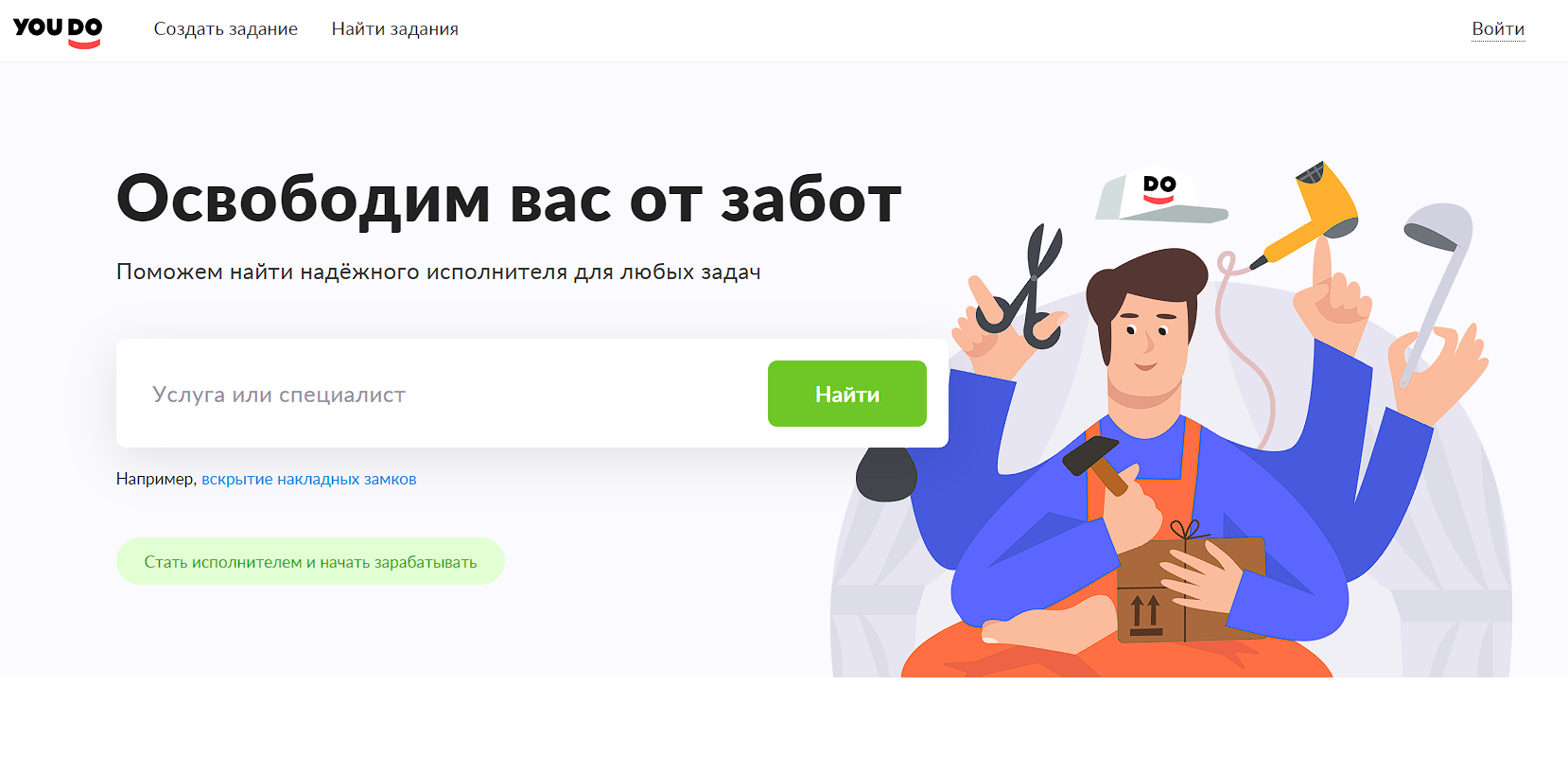
1. Аналитический обзор прототипов и литературных источников

Обзор аналогов

В рамках курсового проекта по разработке базы данных для электронной площадки по предоставлению услуг планируется провести аналитический обзор аналогичных платформ. Этот обзор поможет лучше понять, какие функции и возможности могут быть реализованы на платформе, а также какие инструменты и подходы могут быть применены для разработки базы данных, соответствующей требованиям и особенностям данной электронной площадки.

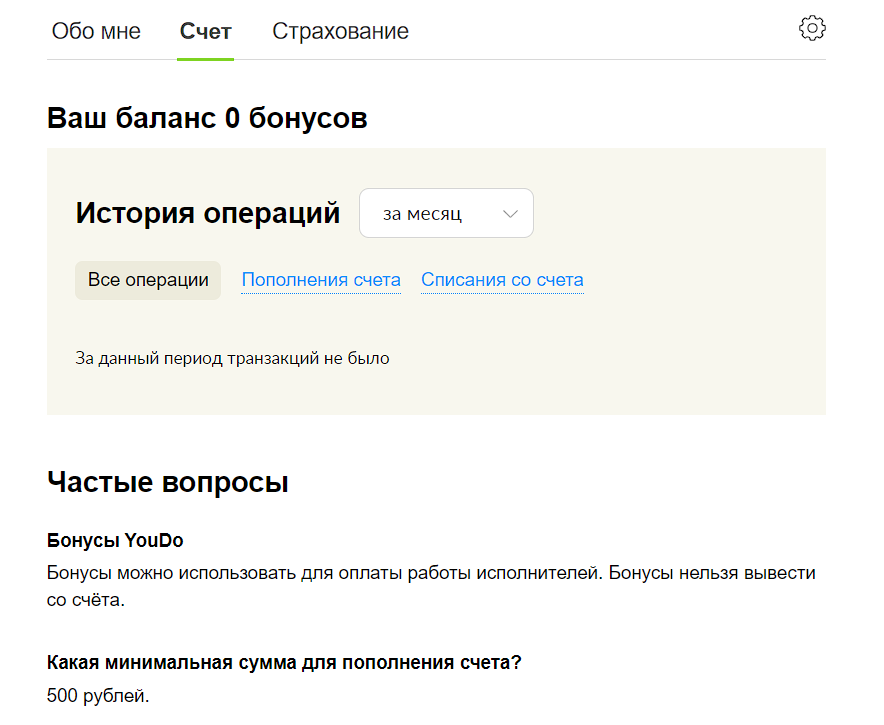
**1.1.1. Аналог – YouDo**

Внешний вид: YouDo представляет собой платформу с современным и привлекательным дизайном, который отражает тематику предоставляемых услуг. Интерфейс сайта обеспечивает удобство использования и хорошую читаемость. Цветовая палитра подобрана гармонично, что способствует приятному визуальному восприятию пользователей.

Оформление главной страницы: Главная страница YouDo привлекает внимание своим удобным и интуитивно понятным поисковым окном. Это окно позволяет пользователям быстро и легко найти интересующие их услуги, просто вводя ключевые слова или фразы. Такой подход обеспечивает эффективный поиск и удобство пользования платформой.

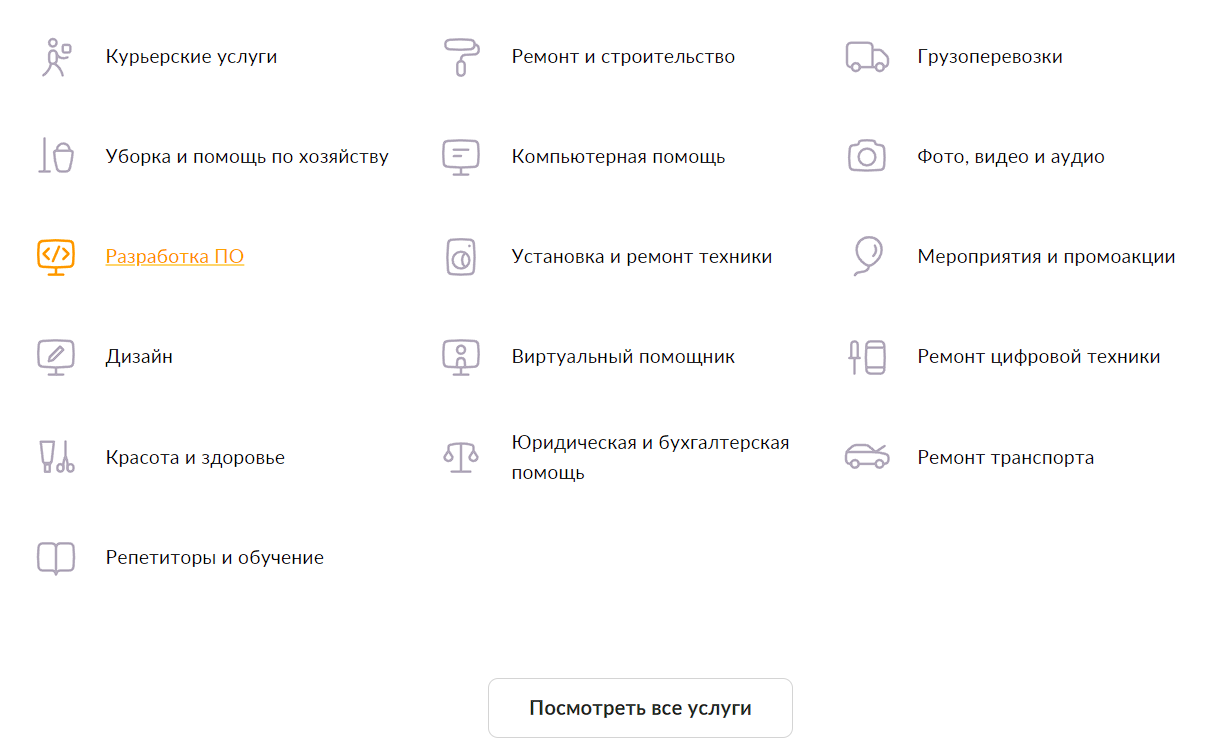
Изображение 1.1 — Главная страница YouDo

Корзина: На платформе YouDo корзина является важной частью процесса заказа услуг. Она интегрирована в удобный интерфейс сайта, что позволяет пользователям легко добавлять выбранные услуги и управлять ими. В корзине предоставляются возможности для просмотра выбранных услуг, изменения их количества или удаления из списка. Этот функционал обеспечивает удобство пользовательского опыта и позволяет легко управлять заказами перед их окончательным оформлением.



Изображение 1.2 — Корзина YouDo

После анализа стало очевидно, что платформа YouDo обладает современным и привлекательным дизайном, который значительно улучшает пользовательский опыт. Функционал корзины и раздела «Избранное» обеспечивает пользователям простоту использования и интуитивную понятность, что содействует позитивному восприятию и способствует успешному завершению заказов.



Изображение 1.3 — Коллекции услуг YouDo

Ключевые аспекты, которые можно использовать и адаптировать для своей базы данных:

1. Категории услуг и подкатегории

В YouDo есть разные категории услуг (например, ремонт, уборка, программирование и т.д.), которые делятся на подкатегории. Это может быть реализовано через таблицу категорий(Categories), где каждая категория может ссылаться на родительскую категорию через parent. Это позволит создавать иерархическую структуру категорий услуг.

2. Система рейтингов и отзывов

В YouDo каждый пользователь имеет систему рейтингов на основе выполненных заданий. Можно реализовать таблицу Reviews, которые будут связываться с таблицей пользователей (Users) и заказов (Orders). Это позволит хранить оценки после завершения заказа, что улучшит механизм оценки на платформе.

3. Админ-панель с функционалом Data-Mining

Визуализированная панель, которая доступна только администраторам, позволяет извлекать скрытые паттерны, тенденции и важную информацию из данных, что может помочь улучшить работу сервиса и удовлетворенность пользователей.

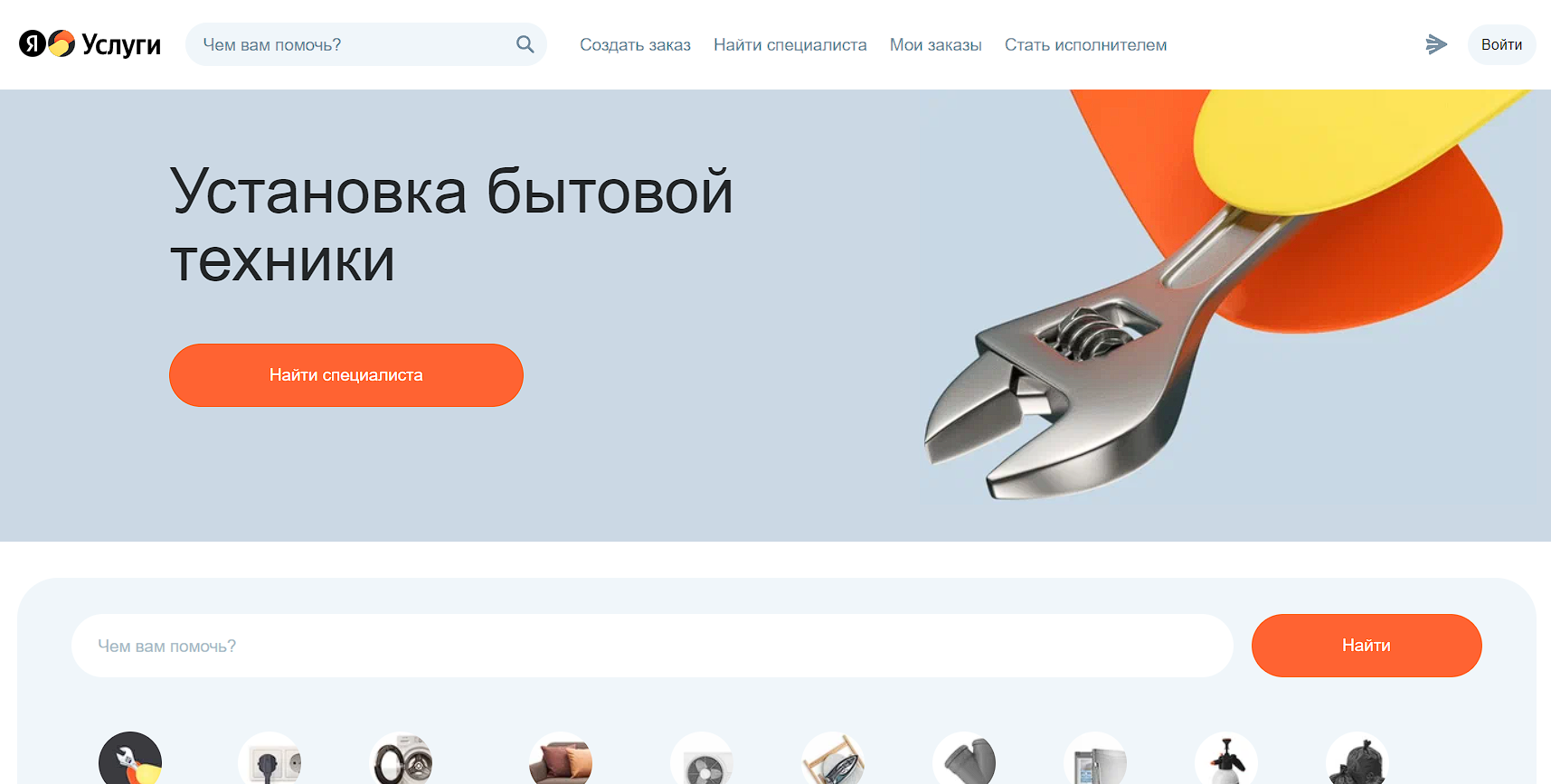
4. Возможность изменения статуса заказа

Функционал **изменения статуса заказа** с условием, что статусы можно изменять только для **неотмененных заказов**, является важной частью управления процессом выполнения заказов на онлайн-платформе. Это предотвращает ошибки, когда, например, заказ уже был отменён, но кто-то пытается обновить его статус на "в процессе" или "выполнен".

1.1.2. Аналог - Yandex Услуги

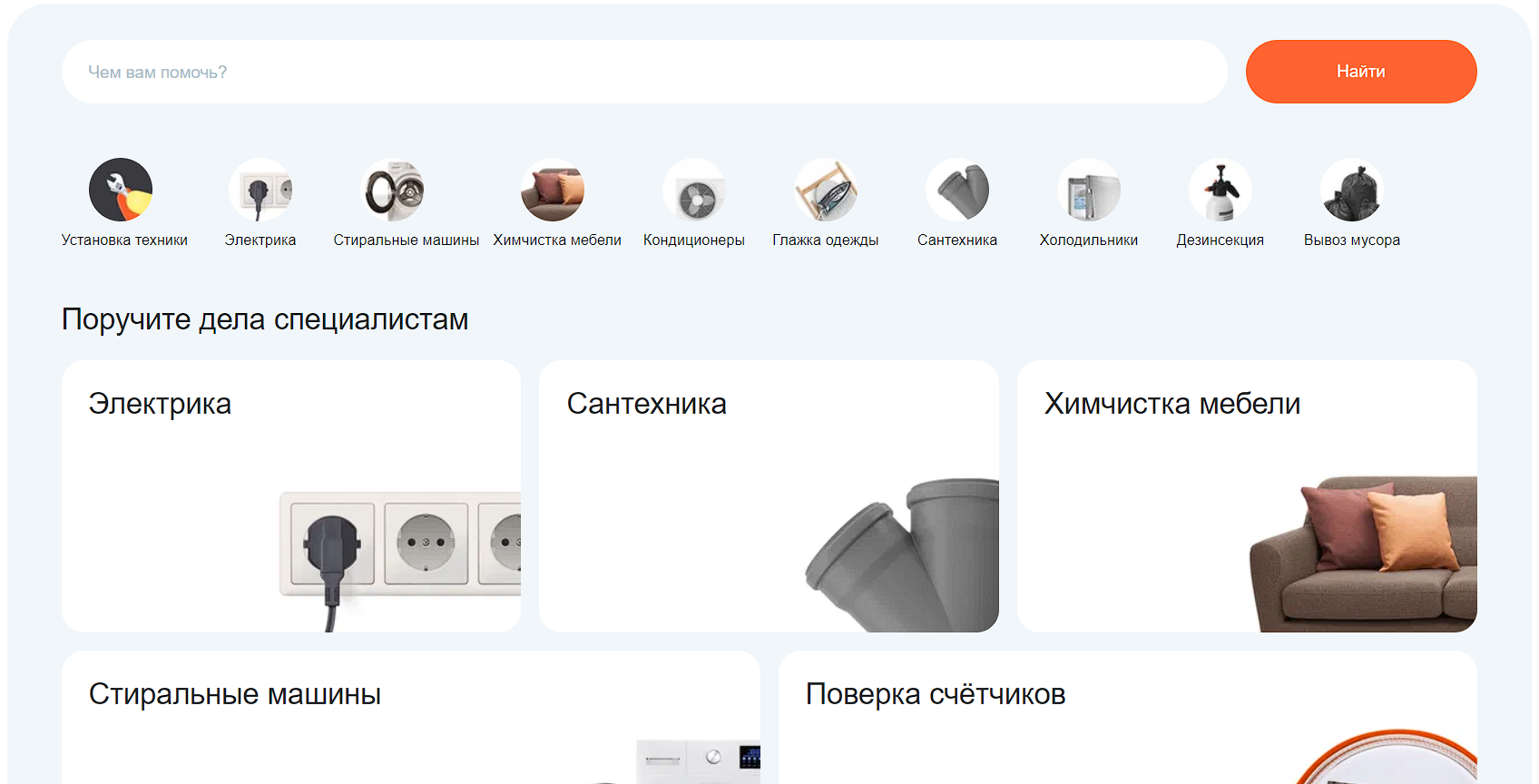
Уникальные особенности: Yandex Услуги предоставляют пользователям возможность взаимодействовать непосредственно с исполнителями и клиентами, создавая уникальное сообщество людей, нуждающихся в услугах. Кроме того, платформа активно поддерживает новых специалистов, предоставляя им возможность продвигаться и развиваться в своей профессиональной сфере.

Инновационный подход: Yandex Услуги используют инновационные технологии для обеспечения безопасности и удобства пользователей. Использование передовых методов автоматизации и проверки данных гарантирует надежность и прозрачность всех транзакций на платформе.



Изображение 1.4 — Главная страница Yandex Услуг

Широкий выбор контента: На платформе Yandex Услуги представлен обширный ассортимент услуг различных категорий и направлений. Это позволяет пользователям выбирать сервисы, соответствующие их потребностям и предпочтениям, а также получать новый опыт и знания в различных сферах услуг.



Изображение 1.5 — Актуальное Yandex Услуг

Поддержка специалистов: Yandex Услуги активно поддерживает исполнителей, обеспечивая им честное вознаграждение за их услуги и создавая условия для раскрытия их профессионального потенциала. Это способствует укреплению сообщества специалистов и улучшению взаимодействия с клиентами. Таким образом, Yandex Услуги выделяются своим уникальным сообществом и поддержкой специалистов, а также инновационным подходом к предоставлению услуг.

Ключевые аспекты, которые можно использовать и адаптировать для своей базы данных:

1. **Заказ и управление статусами**

Yandex Услуги поддерживает управление статусами заказов: в ожидании, в процессе, выполнен, отменён.

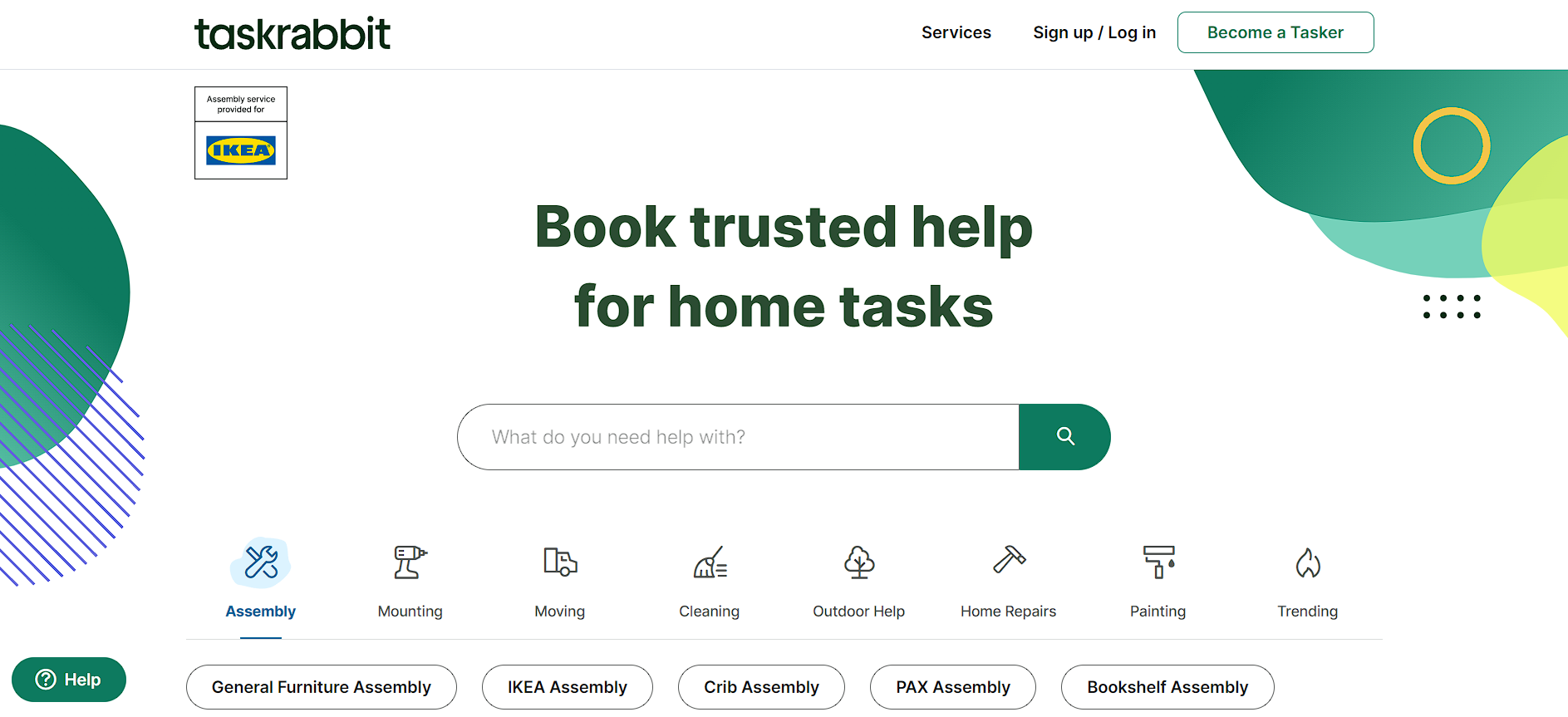
1. **Детализированный профиль исполнителя**

**На данной платформе** каждый исполнитель имеет подробный профиль с информацией о навыках, опыте, рейтинге, отзывах и завершенных заказах.Таблицы Users могут хранить соответствующую информацию о пользователях и их умениях. Для завершенных заказов можно создать связь между таблицами Orders и Users, чтобы показать успешные заказы, выполненные пользователем.

1.1.3. Аналог - TaskRabbit

Уникальные особенности: TaskRabbit выделяется своим уникальным подходом к организации услуг и акцентом на качестве предоставляемых сервисов. Здесь пользователи могут найти высококвалифицированных специалистов в различных областях, от ремонта до переезда, прошедших проверку и одобренных платформой. Этот подход способствует созданию надежной и качественной сети профессионалов для заказчиков.

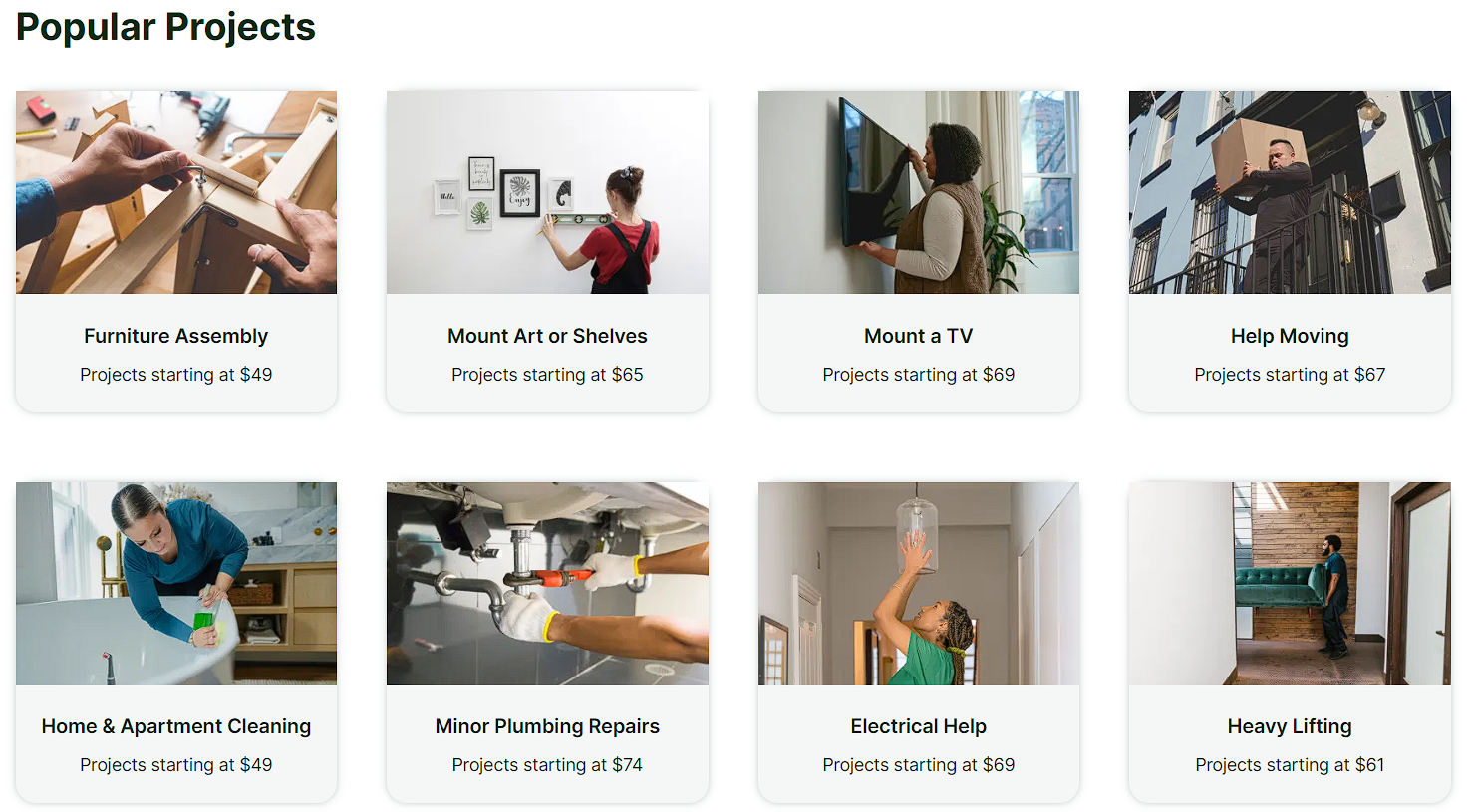
Специализация на услугах: TaskRabbit ориентирована на поддержку и развитие специалистов, обладающих высокими навыками и профессионализмом в своей сфере. Платформа предоставляет возможность специалистам представить свои услуги широкой аудитории, обеспечивая им новые возможности для развития и роста в своей профессиональной карьере.



Изображение 1.6 — Главная страница TaskRabbit

Эксклюзивные услуги: На платформе TaskRabbit пользователи могут найти эксклюзивные услуги, которые могут быть недоступны на других платформах.

Прямое общение с исполнителями: TaskRabbit обеспечивает прямое взаимодействие пользователей с исполнителями услуг, что способствует созданию сообщества и поддерживает тесные связи между исполнителями и заказчиками. Это позволяет пользователям получить уникальную перспективу на процесс выполнения услуги и взаимодействовать с профессионалами напрямую.



Изображение 1.7 — Коллекции услуг TaskRabbit

Таким образом, TaskRabbit выделяется своим упором на качество и кураторство услуг, специализацией в области разнообразных сервисов, предоставлением уникальных возможностей и прямым взаимодействием между исполнителями и клиентами.

2. Спецификация бизнес-требований к программному средству

Бизнес-требования содержат высокоуровневые цели организации или заказчиков системы. Как правило, их высказывают те, кто финансируют проект, покупатели системы, менеджер реальных пользователей, отдел маркетинга. Важно учесть, что бизнес-требования должны быть конкретными, измеримыми и достижимыми. Они должны быть согласованы с интересами и целями организации или заказчика системы, а также учитывать потребности и ожидания пользователей. Анализ и учет бизнес-требований позволяет определить основные задачи, которые должны быть решены программным средством, и направляет весь процесс разработки в соответствии с ожиданиями и потребностями пользователей и заказчика. Для курсового проекта были выдвинуты высокоуровневые требования, которые можно рассматривать как общие требования к разрабатываемому средству. К их числу относятся:

* простота и понятность интерфейса;
* использование системы управления базами данных (СУБД);
* поддержка нескольких способов оплаты.

Весь дальнейший процесс проектирования и разработки программного средства должен находиться в очерченных бизнес-требованиями границах.

2.2 Спецификация пользовательских требований к программному средству

Пользовательские требования играют важную роль в определении функциональности и возможностей разрабатываемого программного средства. Они описывают, каким образом система будет использоваться пользователями и какие конкретные задачи она поможет им решить. Учет пользовательских требований позволяет сосредоточиться на создании удобного и интуитивно понятного интерфейса, который соответствует потребностям и ожиданиям конечных пользователей. Это способствует повышению удовлетворенности пользователей и успешной адаптации системы в рабочей среде.

Программное средство должно предоставлять следующие функциональные возможности для пользователя:

* регистрация;
* авторизация;
* выполнение поисковых запросов, фильтрации/сортировки;
* добавление услуг в корзину;
* заполнение формы услуги (выбор способа оплаты и места выполнения);
* возможность просматривать и оставлять отзывы об услуге;
* возможность просмотра личной страницы пользователя с отображением всех заказанных услуг;
* просмотр всего списка услуг;
* возможность обратиться за поддержкой к администрации.

Администратор должен выполнять следующие функции:

* управление базой данных;
* изменение статуса услуг;
* просмотр информации о пользователях и услугах;
* техническая поддержка пользователей.

Таким образом, был проведен тщательный анализ требований к программному средству, который позволил разработать список пользовательских требований. Разработка данной программной системы должна включать в себя реализацию функций, соответствующих сформированным спискам.

2.3 Спецификация функциональных требований к программному средству

После проведения анализа были выявлены следующие функциональные требования:

* архитектура приложения должна соответствовать шаблонам проектирования, таким как MVVM, Command;
* вся информация должна храниться в базе данных;
* приложение должно обеспечивать поддержку добавления информации в базу данных;
* приложение должно производить валидацию вводимых пользователем данных;
* приложение должно корректным образом обрабатывать возникающие исключительные ситуации: отображать понятное для пользователя сообщение о возникшей ошибке;
* приложение должно предоставлять пользователям возможность создания нового аккаунта в виде регистрационной формы;
* приложение должно предоставлять возможность пользователям проходить аутентификацию и входить в систему под соответствующим введенным данным пользовательским именем;
* приложение должно предоставлять возможность поиска продуктов по следующим критериям: название, категория, цена;
* приложение должно предоставлять возможность оформления заказа;
* приложение должно предоставлять возможность просмотра информации о всех заказах;
* должна присутствовать средства для связи пользователя с администратором.

Таким образом, был проведен тщательный анализ требований к программному средству, который позволил разработать список функциональных требований. Разработка данной программной системы должна проводиться в соответствии с сформированными списком.

# **3. Проектирование программного средства**

3.1 Проектирование архитектуры приложения

Архитектура программного обеспечения — совокупность вaжнейших решений об организации программной системы. Архитектура включает:

* выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов;
* соединение выбранных элементов структуры и поведения во всё более крупные системы;
* архитектурный стиль, который направляет всю организацию — все элементы, их интерфейсы, их сотрудничество и их соединение.

Для удовлетворения проектируемой системы различным атрибутам качества применяются различные архитектурные шаблоны (паттерны). В разрабатываемом приложении используется архитектурный шаблон Model-View-ViewModel (MVVM).

Шаблон MVVM имеет три основных слоя: модель, которая представляет бизнес-логику приложения, представление пользовательского интерфейса, и представление-модель, в котором содержится вся логика построения графического интерфейса и ссылка на модель, поэтому он выступает в качестве модели для представления.

На рисунке 3.1 представлена диаграмма, которая показывает общую структуру приложения в рамках шаблона MVVM.

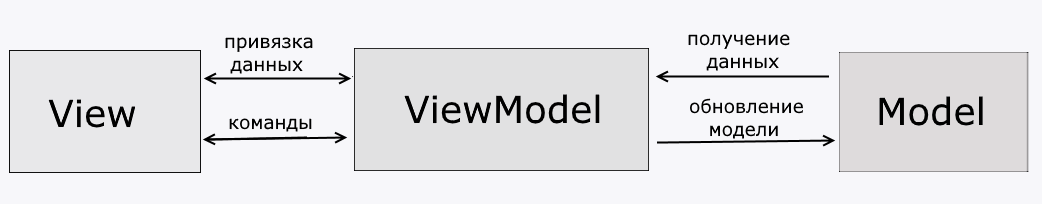


Рисунок 3.1 – Структура шаблона MVVM

View определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Так как пользовательский интерфейс и качество его реализации играет далеко не последнее место в конечном результате, разработка эффективного интерфейса, приятного и удобного для конечного пользователя, является важной задачей. Поэтому для хорошего проектирования View необходимо понять, как пользователь будет взаимодействовать с приложением. Для этого была составлена диаграмма использования представленная в приложении А, на которой представлен принцип работы приложения с точки зрения пользователя.

На рисунке 3.2 отображена диаграмма последовательности для авторизации. На диаграмме последовательности отображаются только те объекты, которые непосредственно принимают участие во взаимодействии.

****

Рисунок 3.2 – Диаграмма последовательности для авторизации

Диаграмма компонентов приложения, представленная на рисунке 3.3, визуально отображает различные компоненты, из которых состоит наше программное средство. Компоненты включают в себя модули, классы и другие структурные элементы, которые взаимодействуют друг с другом для обеспечения функциональности приложения. Эта диаграмма помогает нам понять архитектурную организацию приложения и его основные составляющие. Она также служит важным инструментом для разработчиков, позволяя лучше структурировать и понять внутреннюю логику и взаимодействие компонентов в системе.

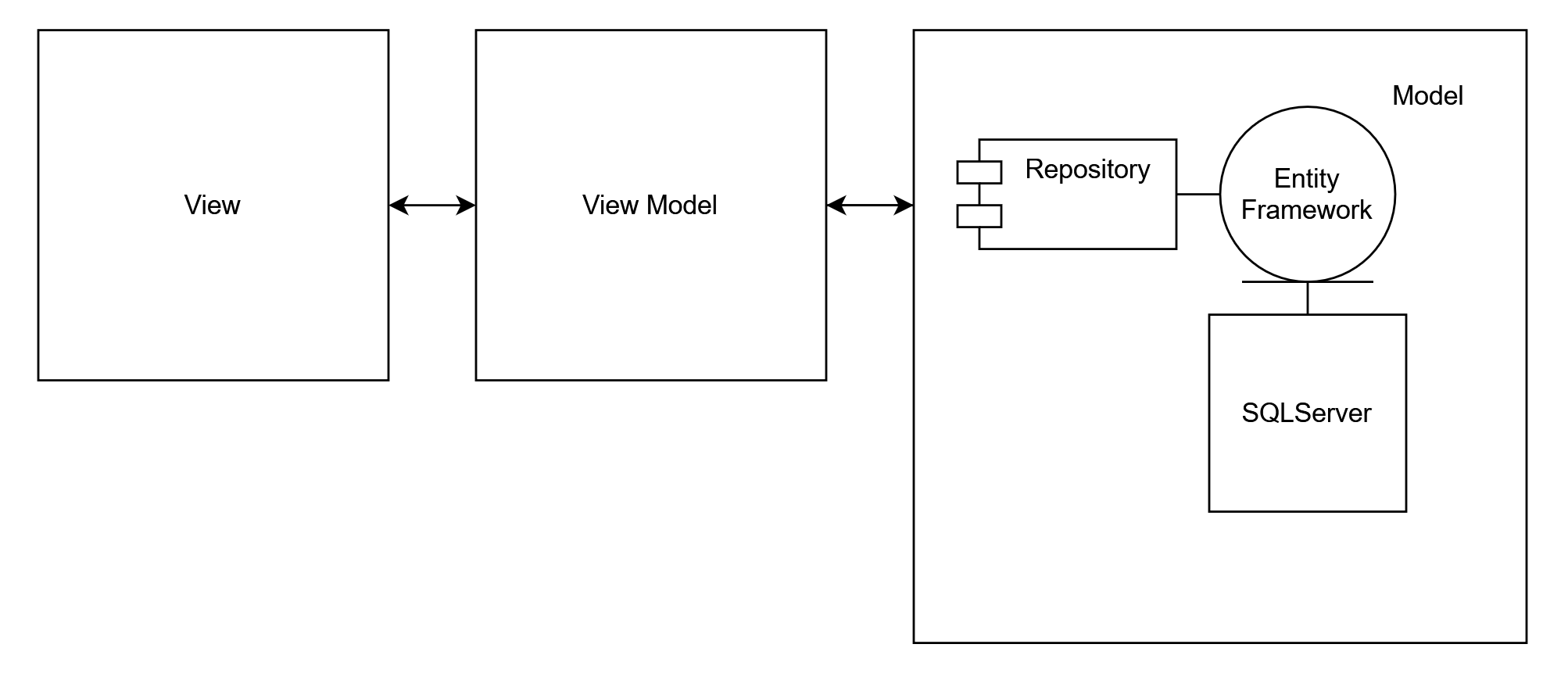


Рисунок 3.3 – Диаграмма последовательности для авторизации

На рисунке 3.4 представлена блок-схема, которая наглядно описывает последовательность шагов и логику, применяемую при регистрации пользователя в системе. Блок-схема помогает визуализировать основные этапы процесса регистрации, включая ввод данных, их валидацию, сохранение в базе данных и уведомление пользователя об успешной регистрации. Такая графическая представление алгоритма облегчает понимание его структуры и взаимосвязей между различными этапами.

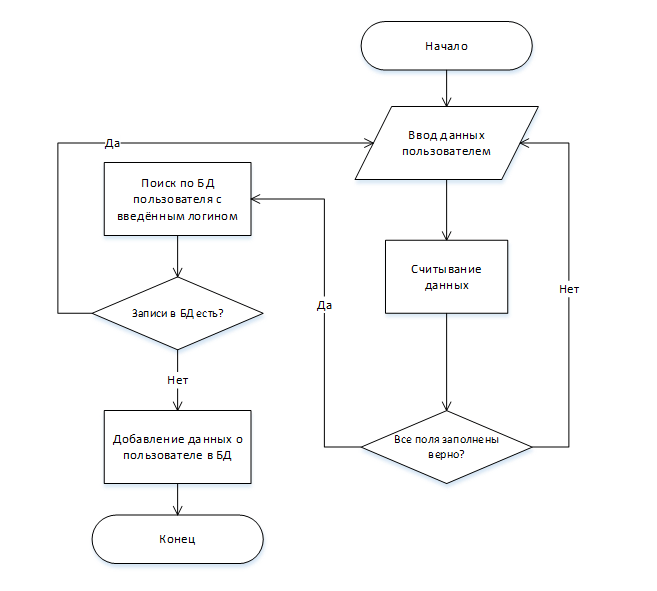


Рисунок 3.5 – Схема алгоритма регистрации

Диаграмма классов представлена в приложении Б.

Логика перехода между страницами пользовательской части реализована в классе MainWindowViewModel, а для администраторской части – в классе AbminViewModel.

Для получения и записи данных в БД используются классы AppConection, ManagerDBRepository и UnitOfWork. Эти классы предоставляют необходимый уровень абстракции, который облегчает и ускоряет доступ к БД. Они выполняют роль посредников между приложением и базой данных, предоставляя удобные методы для работы с данными.

Для взаимодействия с БД в приложении используется технология Entity Framework Core. Она обеспечивает удобный и эффективный способ работы с данными, абстрагируя от особенностей конкретной БД и позволяя использовать объектно-ориентированный подход к работе с данными.

Для администрирования приложения и управления доступом к БД был создан пользователь с логином admin, обладающий правами администратора. Это позволяет осуществлять контроль над базой данных, выполнение административных задач и обеспечивает безопасность и конфиденциальность данных приложения

**3.2. Модель базы данных**

Для реализации поставленной задачи была создана база данных «plat». Для ее создания использовалась система управления реляционными базами данных MSQL Server. База данных состоит из таблиц, представленных в приложении В.

Таблица «Users» содержит информацию о пользователях. В данной таблице поле ID является первичным ключом. Поле Login хранит логин пользователя, Password – пароль, Email – адрес электронной почты, Phone – номер телефона, Photo – фото, Name и Surname – соответственно имя и фамилию.

Таблица «Categories» содержит информацию о категориях услуг. Первичный ключ CategoryID хранит уникальный идентификатор услуги, Name – название, ImageLink – путь к изображению соответствующей услуги данной категории.

Таблица «Products», содержащая информацию об услугах. Первичный ключ ID хранит уникальный идентификатор услуг, Titile – название, CategoryId – ID категории, Price – цена, Rating – текущий рейтинг в соответствии с оценками пользователей, ImageLink – путь к изображению соответствующей услуги.

Tаблица «Carts» содержит данные о корзине пользователя. В данной таблице первичный ключ ID является идентификатором корзины, поле UserID, ссылающееся на поле ID таблицы «Users», содержит ID пользователя, поле ProductID, ссылающееся на поле ID таблицы «Product», поле Count содержит информацию о количестве услуг в корзине, поле SumPrice определяет сумму услуг, а поле Cart – карту для оплаты.

Tаблица «Orders» содержит данные о заказах. В данной таблице первичный ключ OrderID является идентификатором заказа, поле UserID, ссылающееся на поле ID таблицы «Users», содержит ID пользователя, совершившего заказ, поле ProductID, ссылающееся на поле ID таблицы «Product», поле Count содержит информацию о количестве заказанной услуги, а поле OrderState – о текущем статусе заказа (В обработке, отправлен, доставлен). Поле SumPrice определяет сумму заказа, а поле Cart – карту для оплаты.

Tаблица «Reviews» содержит данные об отзывах пользователей на услуги. Таблица включает в себя: первичный ключ ReviewID, поле UserID, ссылающееся на поле ID таблицы «Users», содержит ID пользователя, поле ProductID, ссылающееся на поле ID таблицы «Product», поле UserRating содержащее оценку пользователя для данной услуги и поле ReviewContent определяющее содержание отзыва.

# **4. Проектирование программного средства**

4.1 Реализация архитектуры MVVM

Для реализации паттерна MVVM файлы программы были распределены по соответствующим директориям и реализовали соответствующие функции. Разделение проекта на логические модули представлено на рисунке 4.1.

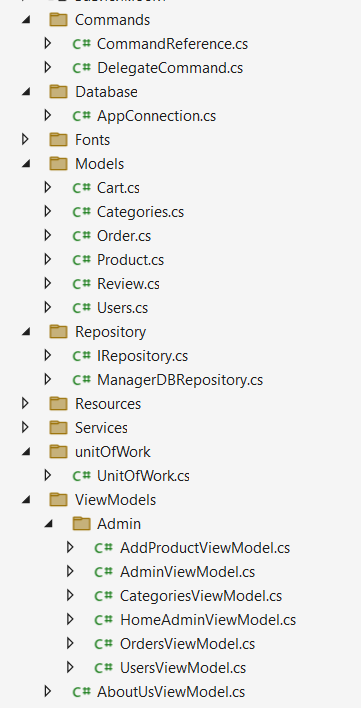
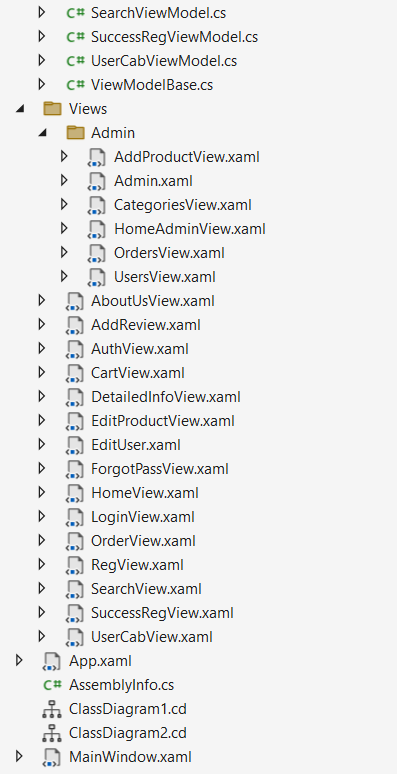
 

Рисунок 4.1 – Логические модули проекта

Архитектура MVVM (Model-View-ViewModel) является одним из популярных паттернов архитектуры для проектов, связанных с разработкой пользовательских интерфейсов. Модель представляет данные в приложении, Представление отображает данные пользователю, а ViewModel является связующим звеном между Моделью и Представлением, обрабатывая логику, связанную с UI и взаимодействуя с Моделью для запроса и обновления данных.

4.2 Реализация классов Models и ViewModels

В папке Models расположены сущностные классы, которые используются для создания БД. Этими классами являются Cart, Categories, Order, Product, User, Review. Пример реализации класса User представлен на рисунке 4.2.

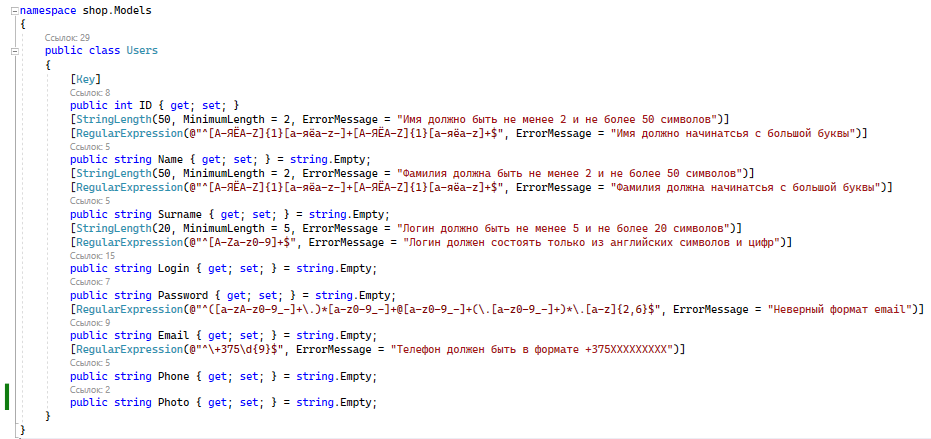


Рисунок 4.2 – Содержимое класса Users

Базовым классом для всей моделей представлений является ViewModelBase, который реализует интерфейс INotifyPropertyChanged. На рисунке 4.3 представлено содержимое класса ViewModelBase.

|  |
| --- |
| public abstract class ViewModelBase : INotifyPropertyChanged  {  public event PropertyChangedEventHandler PropertyChanged;  protected void OnPropertyChanged(string propertyName)  {  PropertyChangedEventHandler handler = PropertyChanged;  if (handler != null)  {  handler(this, new PropertyChangedEventArgs(propertyName));  }  }  } |
| Листинг 4.3 – Класс ViewModelBase |

Классы с суффиксом ViewModel содержат: закрытые поля, открытые свойства и команды, привязываемые к элементам управления на страницах View. ViewModel также содержит логику по получению данных из модели, которые потом передаются в представление. И также VewModel определяет логику по обновлению данных в модели. Диаграмма классов представлена в приложении Б.

4.3 Реализация паттерна Command

В приложении используется паттерн Command который позволяет инкапсулировать запрос на выполнение определенного действия в виде отдельного объекта. В WPF команды представлены интерфейсом ICommand. В приложении он представлен в виде двух классов: CommandReference и DelegateCommand.

Класс DelegateCommand позволяет делегировать командную логику методами, передаваемыми в качестве параметров, и позволяет View связывать команды с объектами, которые не являются частью дерева элементов. Также содержит в себе класс CommandManagerHelper, содержащий методы для CommandManager, которые помогают избежать утечки памяти, используя слабые ссылки.

Класс CommandReference облегчает привязку ключа в разметке XAML к команде. Определяется в модели представления путем предоставления свойства зависимости от команды. Класс является производным от Freezable, чтобы обойти ограничение в WPF, когда привязка данных из XAML.

Внедрение паттерна Command в программное средство позволяет достичь лучшей структурированности и гибкости кода, упрощает поддержку и расширение функциональности, а также облегчает взаимодействие между различными компонентами приложения.

4.4 Реализация подключения к БД и паттерны Repository и UnitOfWork

В любом приложении, работающим с БД через Entity Framework Core, необходимо использовать контекст (класс производный от DbContext) и набор данных DbSet, через который можно взаимодействовать с таблицами из БД. В данном случае таким контекстом является класс AppConnection.

Класс ManagerDBRepository является реализацией паттерна репозиторий. Репозиторий позволяет абстрагироваться от конкретных подключений к источникам данных, с которыми работает программа, и является промежуточным звеном между классами, непосредственно взаимодействующими с данными, и остальной программой. Листинг класса представлен в Приложении Г.

Класс UnitOfWork является реализацией паттерна UnitOfWork для приложения. Листинг класса представлен в Приложении Д.

Пример создания экземпляра услуги с использованием паттернов Command, UnitOfWork и Repository представлен в приложении Е.

4.5 Реализация вспомогательных классов и методов

В проекте было реализовано несколько дополнительных вспомогательных классов для расширения функционала приложения.

Для безопас­ности хранения информации используется хэширование, в данном случае применяется встроенный класс MD5. Код метода хэширования представлен в классе SecurePassService, который продемонстрирован на рисунке 4.5.

|  |
| --- |
| public class SecurePassService  {  public static string Hash(string input)  {  byte[] hash = Encoding.ASCII.GetBytes(input);  MD5 md5 = new MD5CryptoServiceProvider();  byte[] hashenc = md5.ComputeHash(hash);  string output = "";  foreach (var b in hashenc)  {  output += b.ToString("x2");  }  return output;  }  } |
| Листинг 4.5 – Метод обеспечивающий хэширование пароля |

Для выполнения валидации и получения результатов были использованы классы ValidationResult, Validator и ValidationContext. Класс ValidationResult содержит информацию о результате валидации, включая список ошибок при нарушении правил. Класс Validator предоставляет методы для выполнения валидации объектов, а класс ValidationContext представляет контекст валидации, в котором определены правила и параметры валидации. В листинге 4.4 представлен метод для проверки вводимых значений при создании услуги.

|  |
| --- |
| bool AddProduct(string Title, Category category, string Price, string Description, string ImageLink)  {  var product = new Product  {  Title = Title,  Category = Category,  Price = Convert.ToDouble(Price),  Description = Description,  ImageLink = ImageLink,  Rating = 0  };  var results = new List<System.ComponentModel.DataAnnotations.ValidationResult>();  var context = new ValidationContext(product);  if (Validator.TryValidateObject(product, context, results, true))  {  ErrorMessage = String.Empty;  return true;  }  else  {  foreach (var error in results)  {  ErrorMessage = error.ErrorMessage;  }  }  return false;  } |
| Листинг 4.5 – Метод для проверки корректности введенных данных |

В этом методе используются атрибуты валидации, которые определены для свойств модели данных услуги. После выполнения валидации метод возвращает результат в виде объекта ValidationResult, который содержит информацию о корректности вводимых значений.

4.6 Реализация Views

Для разработки графической части приложения была выбрана технология WPF.

Windows Presentation Foundation (WPF) — это библиотека для создания пользовательских интерфейсов для интеллектуальных клиентских приложений

Одной из важных особенностей WPF является использование языка декларативной разметки интерфейса XAML, основанного на XML. Разработка с использованием XAML позволяет отделить графический интерфейс от логики приложения, а также создавать насыщенный интерфейс, используя или декларативное объявление интерфейса, или код на управляемых языках C#.

В результате применения WPF в приложении были реализованы три основных окна: стартовое окно, главное окно пользователя и окно администратора.

А также десять страниц:

1. страница для авторизации;
2. страница для регистрации;
3. главная страница с категориями услуг;
4. страница с каталогом услуг;
5. страница личного кабинета пользователя;
6. страница корзины услуг;
7. страница управления категориями услуг;
8. страница управления услугами;
9. страница для просмотра пользователей;
10. страница для просмотра заказов.

В итоге успешно выполненного этапа разработки было создано функционирующее программное средство, обладающее графическим интерфейсом, способным удовлетворить требования пользователя и предоставить удобный опыт использования.

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

Основной целью тестирования приложения было стремление доказать невозможно введения пользователем данных, которые бы могли привести приложение в неработоспособное состояние. Были проведены:

1. тестирование авторизации и регистрации;
2. тестирование управления услугами и категориями;
3. тестирование функциональности поиска, фильтрации, сортировки услуг;
4. тестирование корзины и оформления услуг;
5. тестирование страницы карточки услуги;
6. тестирование оплаты и адреса выполнения услуги.

Упомянутое тестирование будет подробно описано в последующих подпунктах, где будут представлены результаты и анализ каждого этапа тестирования.

5.1 Тестирование авторизации и регистрации

В момент регистрации возможна ситуация, когда пользователь вводит некорректные данные. Предусмотрены ограничения по количеству вводимых символов. Имя и фамилия должны быть в диапазоне от 2 до 50 символов, пароль –от 5 до 20 символов, логин – от 5 до 20 символов. Логин должен быть указан латиницей и может содержать цифры. Также логин должен быть уникальный в системе. При вводе некорректных данных будет сгенерировано исключение. Обработка данного исключения продемонстрирована на рисунке 5.1.

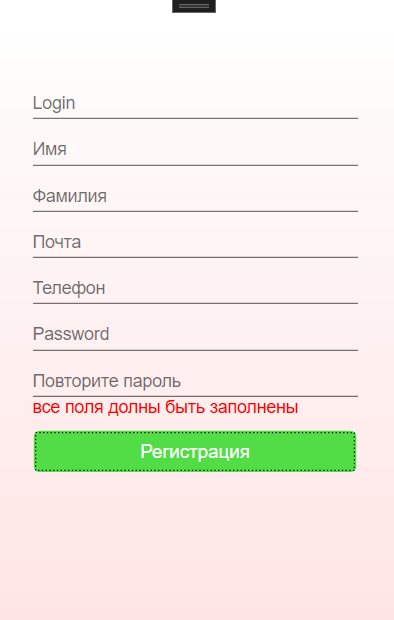


Рисунок 5.1 – Демонстрация обработки исключения

При регистрации пользователю необходимо ввести повторно ввести пароль, в случае отличия паролей будет выведено сообщение с ошибкой. Номер мобильного телефона и электронная почта должны соответствовать требуемому формату. В случае некорректного ввода также будет сгенерировано соответствующее исключение.

При попытке ввести логин несуществующего пользователя на странице «Забыли пароль» будет выведено сообщение с ошибкой, представленной на рисунке 5.2.

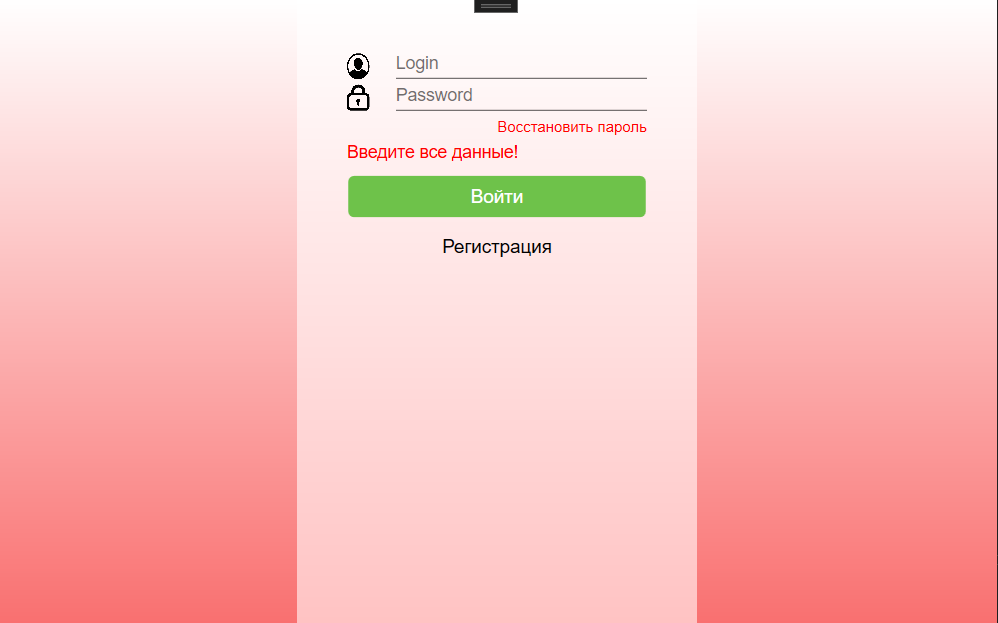


Рисунок 5.2 – Демонстрация обработки исключения

Предупреждение о неверно введённом пароле или логине продемонстрировано на рисунке 5.3.

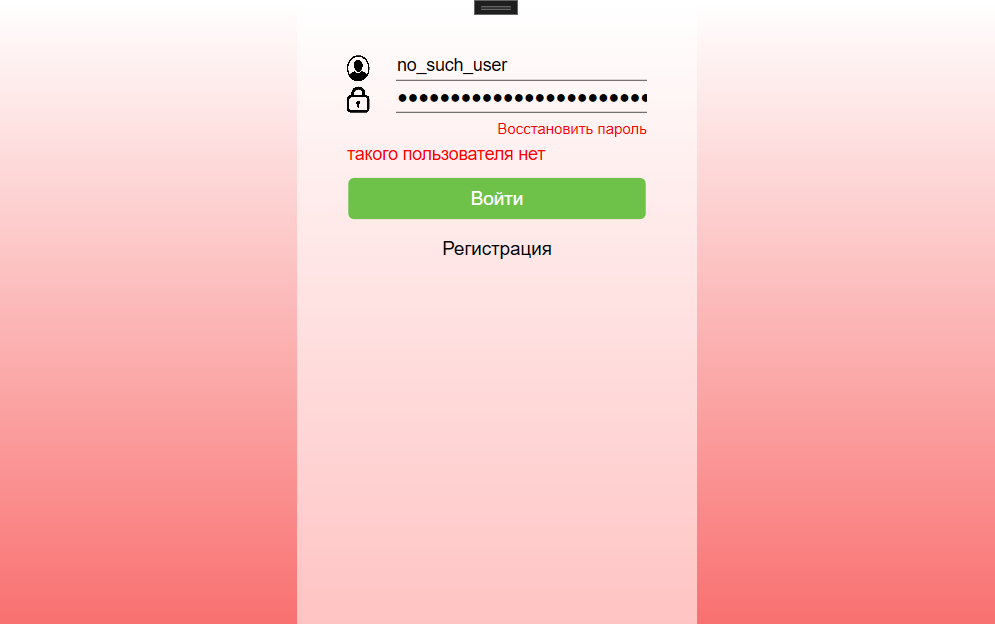


Рисунок 5.3 – Демонстрация обработки исключения

Тестирование авторизации и регистрации подтвердило, что функциональность этих модулей соответствует ожиданиям пользователей. Были проверены различные сценарии, включая успешную авторизацию, регистрацию новых пользователей и обработку ошибок. Тестовые сценарии успешно пройдены без критических проблем или ошибок. Вход в систему и регистрация работают стабильно и надежно, обеспечивая безопасность данных пользователей. Результаты тестирования гарантируют безопасный и удобный процесс входа и регистрации в приложении.

5.2 Тестирование функций администратора

Этап тестирования функций администратора включал проверку основных операций, таких как управление услугами и категориями (добавление, удаление и редактирование), поиск, изменение статуса заказа и просмотр пользователей.

В процессе тестирования функции управления услугами, была проверена возможность успешного добавления новых услуг в систему, их корректное отображение в каталоге и возможность редактирования уже существующих услуг. При вводе некорректных данных будет сгенерировано исключение. Обработка данного исключения продемонстрирована на рисунке 5.4.

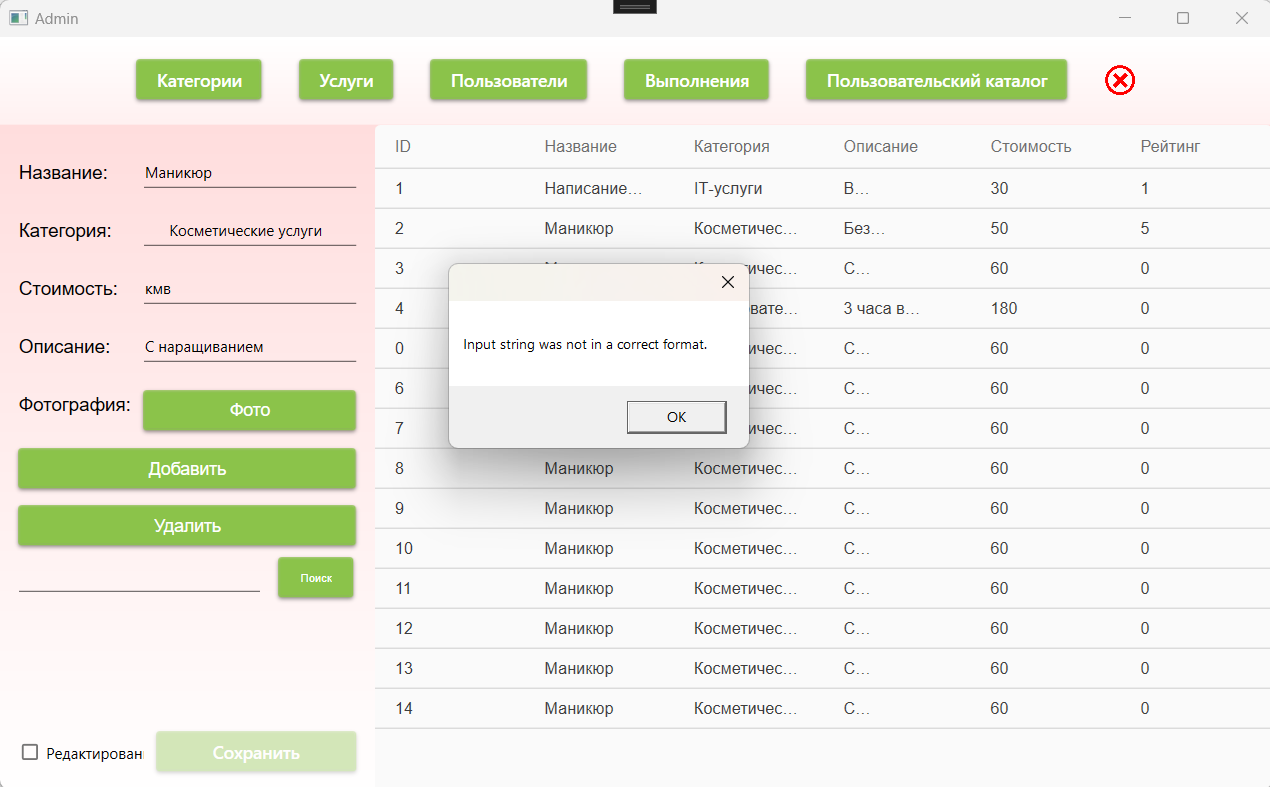


Рисунок 5.4 – Демонстрация обработки исключения

Также была проверена функция удаления услуг, удостоверившись в правильном удалении услуги из базы данных.

Тестирование функции изменения статуса заказа включало проверку возможности изменения статуса заказа на различные значения («Обрабатывается», «Выполняется» и «Выполнена»). Было также убеждено, что изменение статуса заказа корректно отражается в системе и соответствующая информация видна пользователю. Результат представлен на рисунке 5.5.

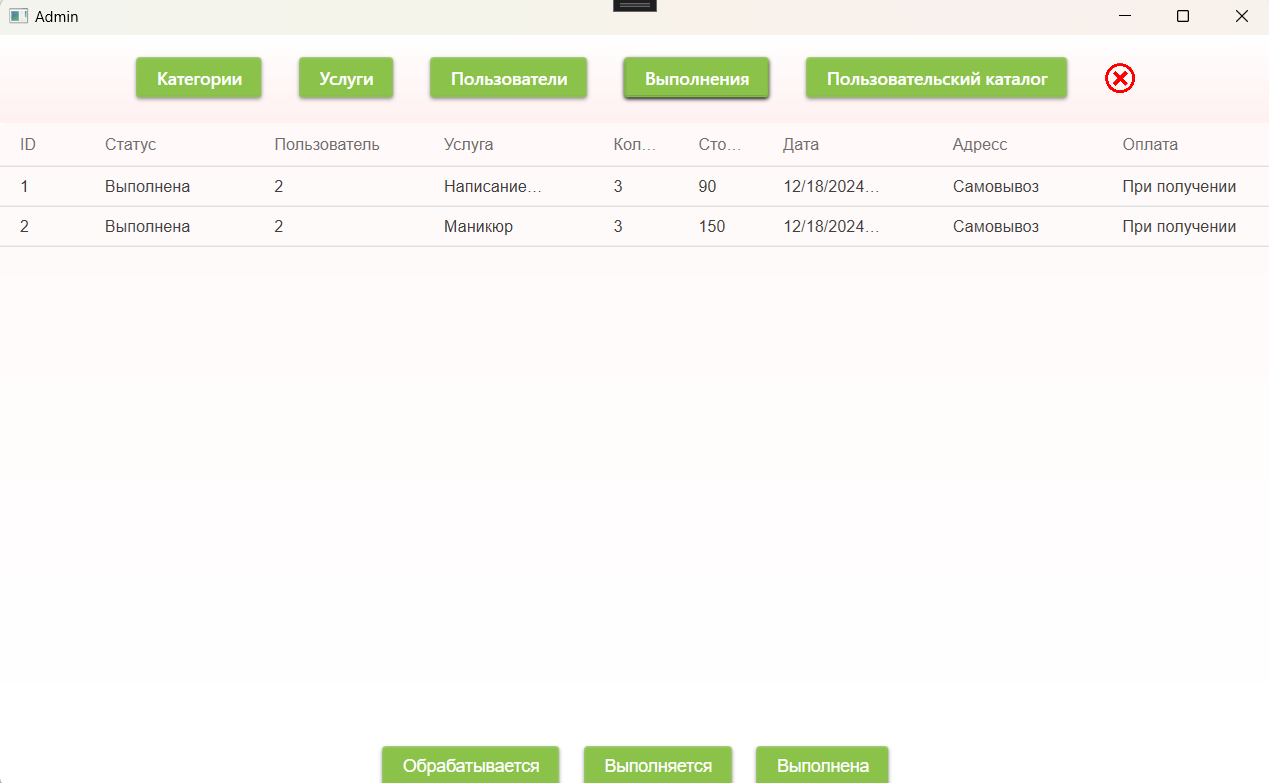


Рисунок 5.5 – Демонстрация работоспособности изменения статуса заказа

Функция просмотра и поиска пользователей была протестирована для убеждения в возможности просмотра списка зарегистрированных пользователей, их контактной информации и других соответствующих данных. Результат представлен на рисунке 5.6

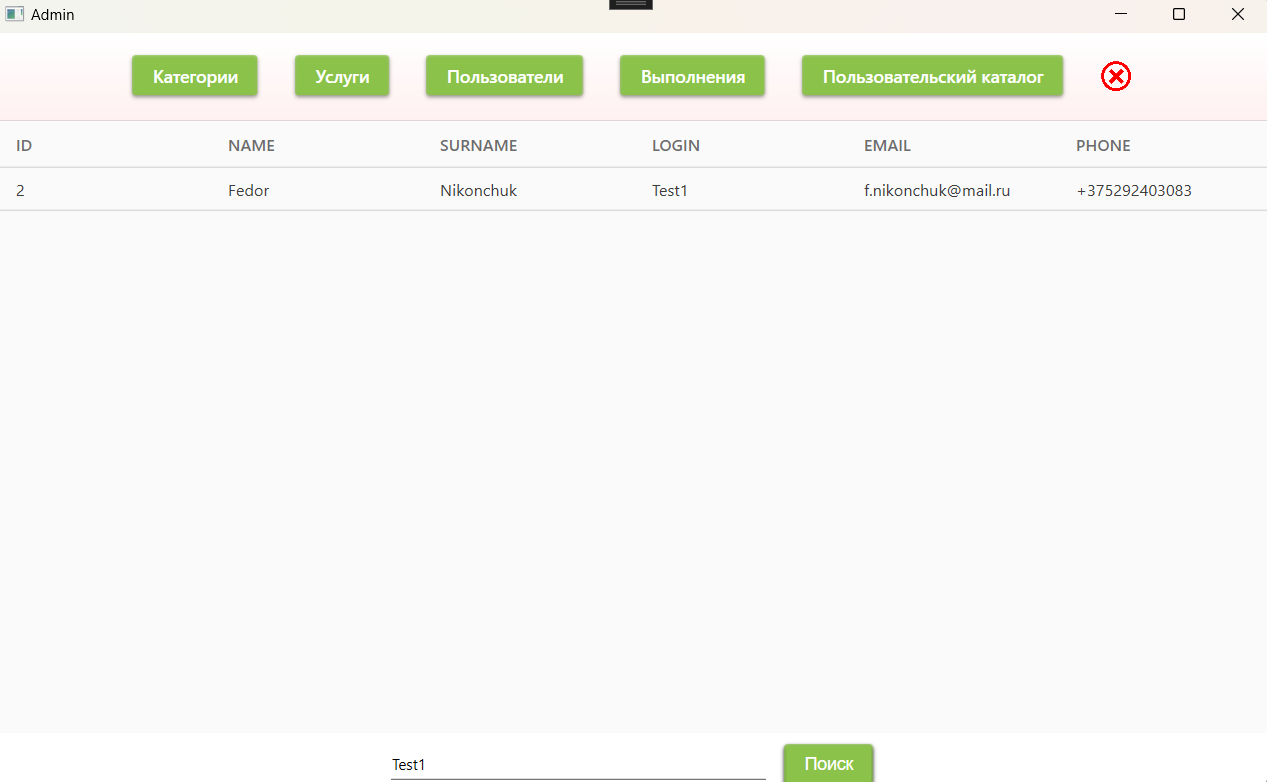


Рисунок 5.6 – Демонстрация работы поиска

В результате тестирования функций администратора было подтверждено, что все указанные функции работают корректно и соответствуют требованиям.

5.2 Тестирование функций пользователя

В ходе тестирования функциональности поиска, фильтрации и сортировки услуг была проверена возможность точного поиска услуг по заданным критериям, правильная фильтрация услуг по различным атрибутам (например, цене, категории) и корректная сортировка услуг по заданным параметрам (например, по возрастанию или убыванию цены). Результаты поиска, фильтрации и сортировки, представленные на рисунке 5.7, соответствуют ожиданиям пользователя.

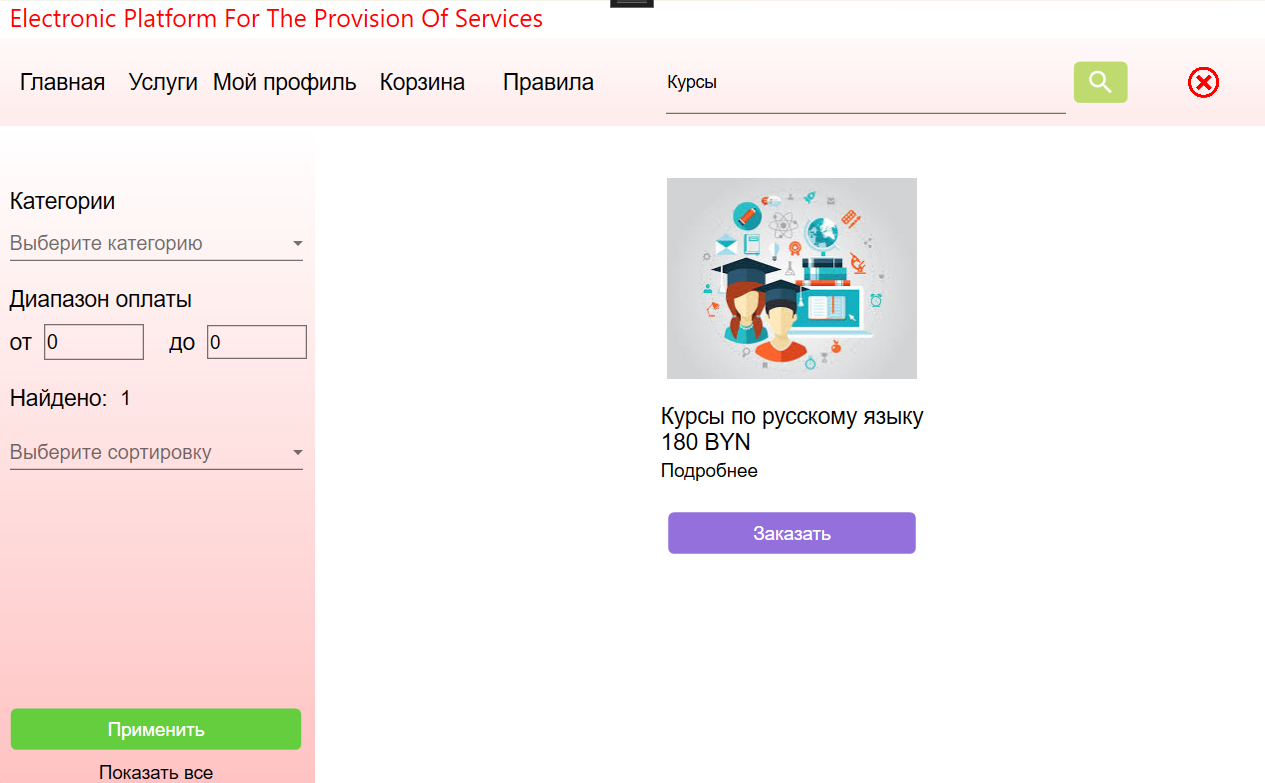


Рисунок 5.7 – Демонстрация корректной работы поиска

Тестирование функциональности корзины и оформления услуги включало проверку возможности добавления услуг в корзину, корректное отображение выбранных услуг и их количества, а также успешное оформление услуги с правильным учетом выбранных услуг, адреса выполнения и способа оплаты. Все изменения с услугами происходили корректно, при изменении услуг количество и стоимость изменяются корректно, что можно увидеть на рисунке 5.8.

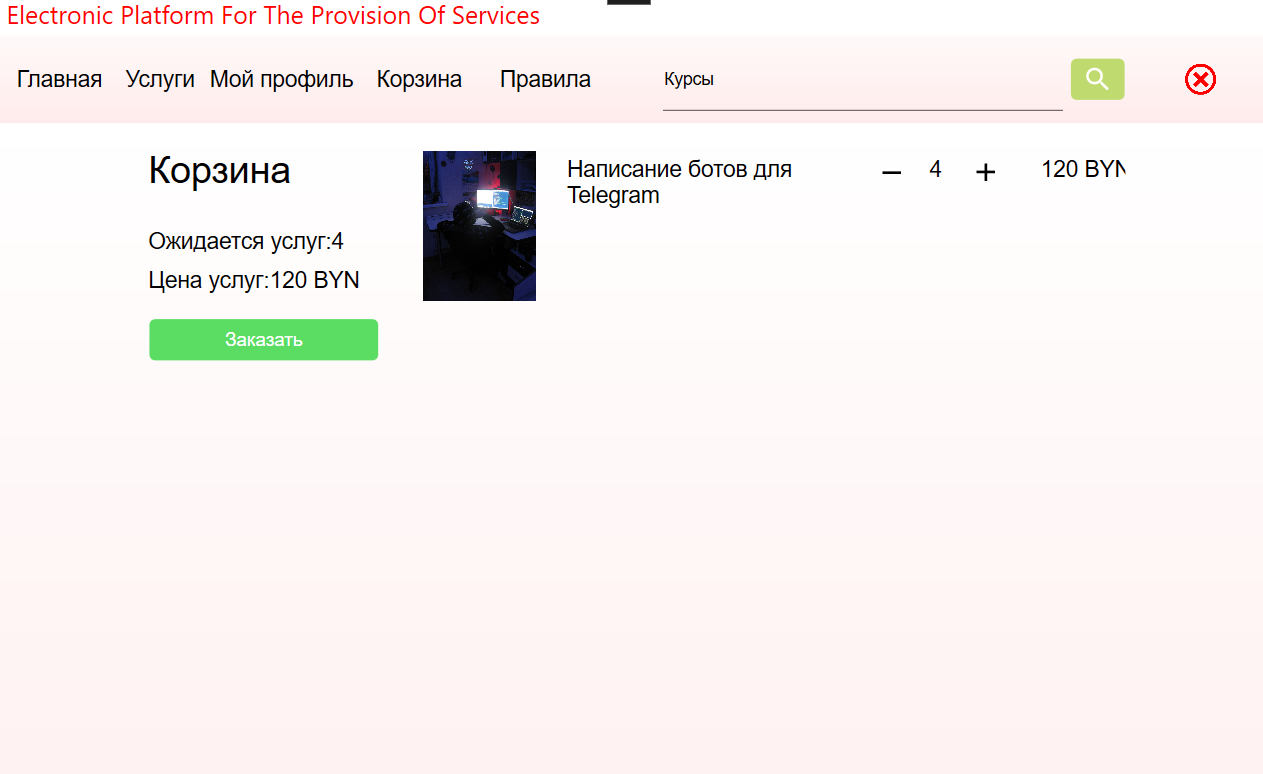


Рисунок 5.8 – Демонстрация корректной работы функций корзины

Тестирование страницы карточки услуги включало проверку правильного отображения информации об услуге, его изображения, описания и других атрибутов. Добавления услуги в корзину происходит корректно.

Тестирование оплаты и адреса выполнения включало проверку возможности успешной оплаты заказа, правильного отображения информации о способах оплаты и адреса выполнения. При вводе некорректного номера карты выведено соответствующее сообщение, что отображено на рисунке 5.9.



Рисунок 5.9 – Демонстрация обработки исключения

Процесс оплаты и доставки проходит без проблем, и информация сохраняется корректно.

В результате тестирования было подтверждено, что все они работают корректно, соответствуют требованиям и обеспечивают удобство использования для пользователя.

**6. Руководство по использованию**

Окно входа содержит два поля для ввода логина и пароля. После ввода верных данных, необходимо нажать на кнопку вход. Окно входа в аккаунт представлено на рисунке 6.1.

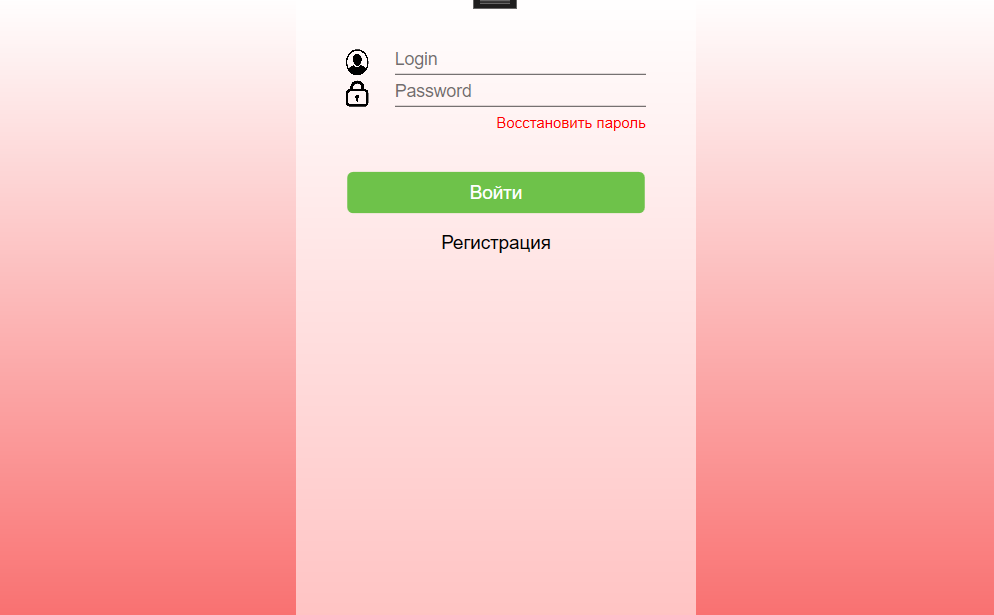


Рисунок 6.1 – Окно входа в аккаунт

Если пользователь забыл пароль, то по нажатию на кнопку «Забыли пароль» откроется страница, представленная на рисунке 6.2. Для смены пароля необходимо ввести логин и новый пароль.



Рисунок 6.2 – Окно входа в аккаунт

Окно регистрации представлено на рисунке 6.3.

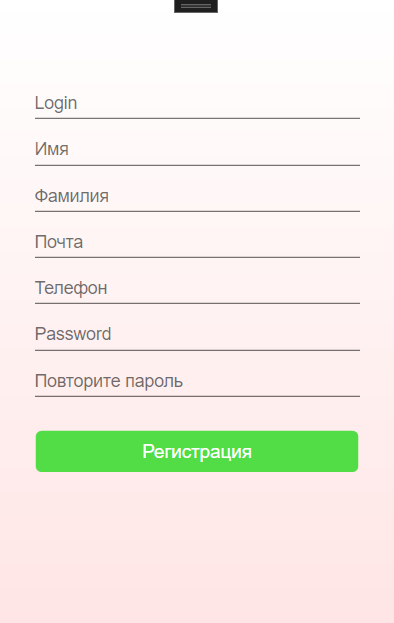


Рисунок 6.2 – Окно регистрации

На поля регистрации накладываются следующие ограничения:

1) имя и фамилия: должны содержать только буквы, должны начинаться с заглавной буквы, длина должна быть от 2 до 30 символов;

2) логин: длина должна быть от 5 до 20 символов, должен быть уникальным в системе, может содержать только английские буквы, цифры, дефисы и подчеркивания;

3) номер телефона и почта: должны соответствовать формату;

4) пароль: длина должна быть от 5 до 20 символов, может содержать буквы английского алфавита, цифры, нижнее подчеркивание и дефис;

После успешной авторизации открывается главная страница, изображенная на рисунке 6.3.

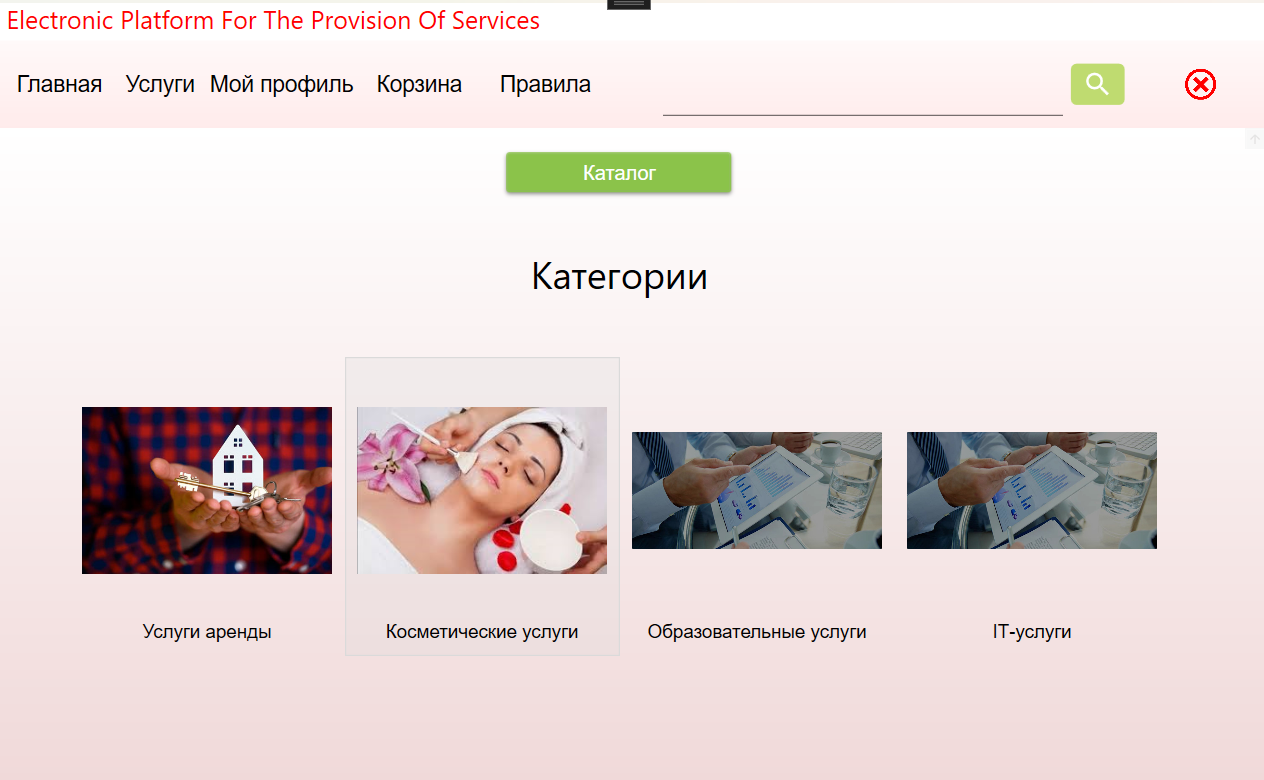


Рисунок 6.3 – Главная страница

Доступна навигация по страницам «Главная», «Услуги», «Мой профиль», «Корзина» и «Правила». Ниже на главной странице расположены категории услуг, что можно увидеть на рисунке 6.4. Для отображения услуг определенной категории необходимо нажать на соответствующую категорию.

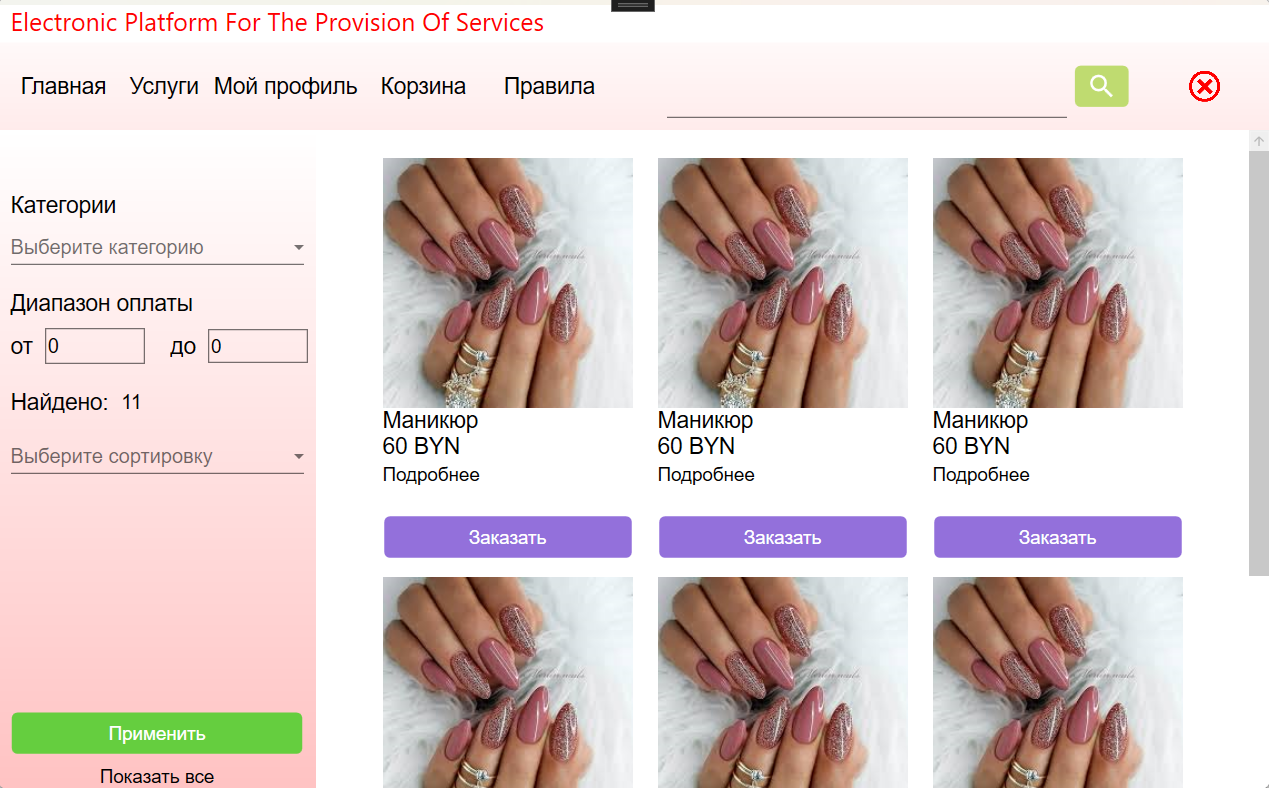


Рисунок 6.4 – Продолжение главной страницы

Страница «Правила» содержит контактную информацию владельцев электронной площадки.

После введения поискового запроса пользователь попадает на страницу с результатами поиска, где он может просматривать услуги и добавлять их в корзину, применять фильтры и сортировку. Страница с каталогом представлена на рисунке 6.5.

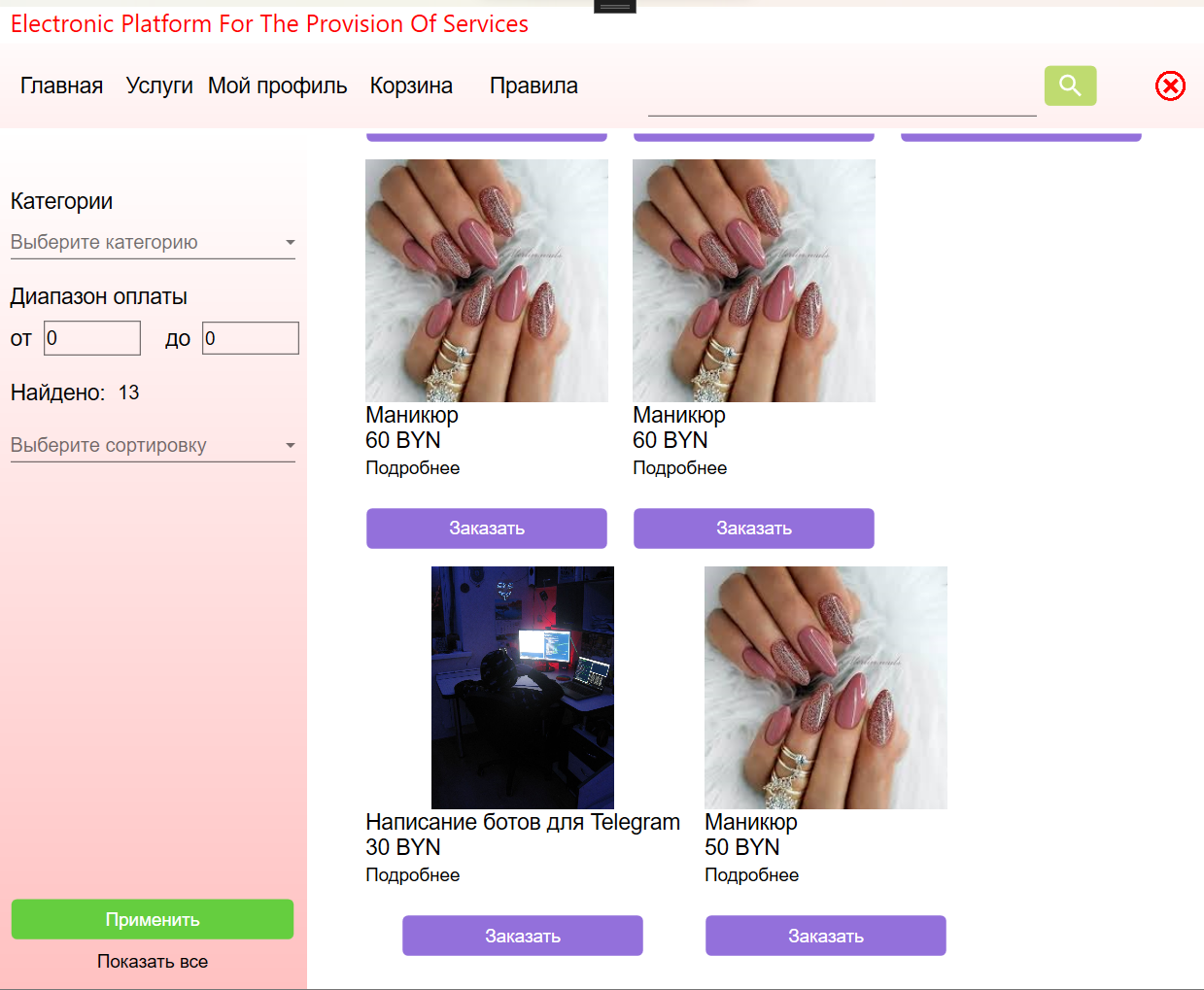


Рисунок 6.5 – Страница каталога услуг

По нажатию на карточку услуги открывается персональная страница услуги, на которой находятся подробная информация об услуге, представленная на рисунке 6.6. Также имеется возможность добавить услугу в корзину, просматривать комментарии других пользователей.

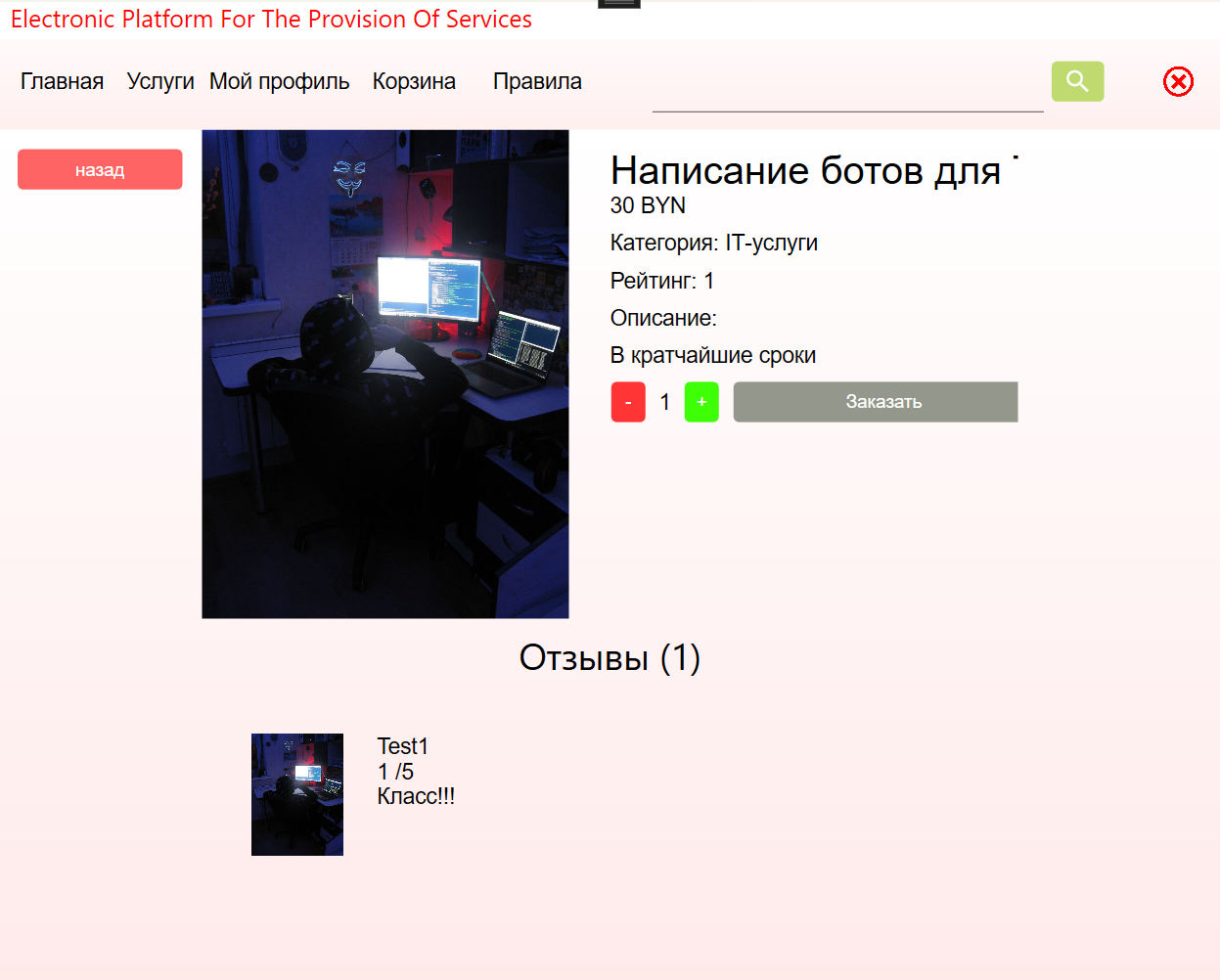


Рисунок 6.6 – Страница услуги

Страница «Профиль» содержит персональные данные пользователя, которые он может изменить. На данной странице располагаются активные доставки, статус которых пользователь может просмотреть, а также список ранее заказанных услуг. Для заказанных услуг при нажатии на кнопку «Отзыв» может оставить комментарий и поставить оценку. Длинна комментария ограничена 300 символами. На рисунке 6.7 изображена страница «Мой профиль».

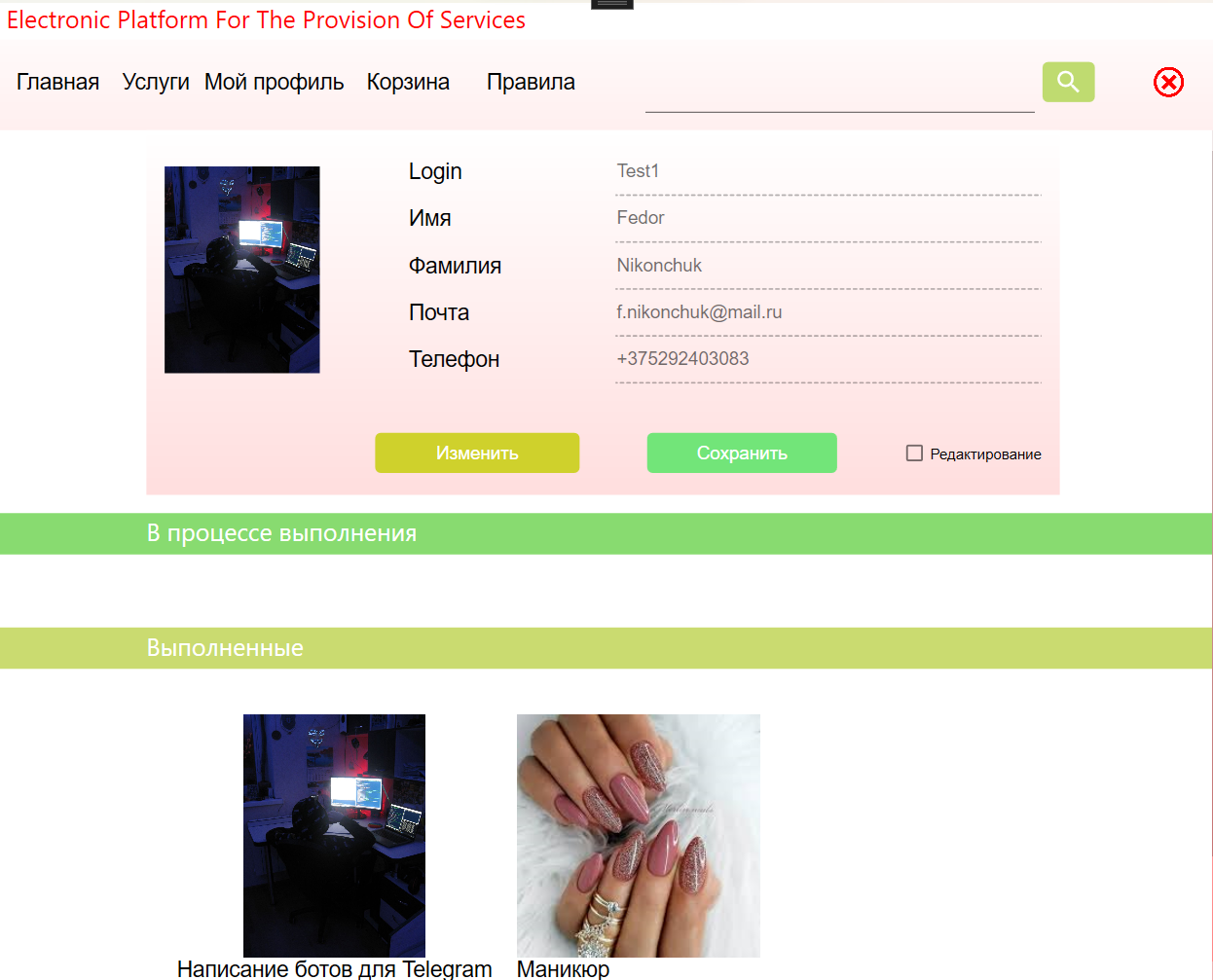


Рисунок 6.7 – Страница «Мой профиль»

На странице «Корзина» пользователь может изменять количество конкретной услуги, удалять её, просмотреть итоговую стоимость. Данная страница изображена на рисунке 6.8.

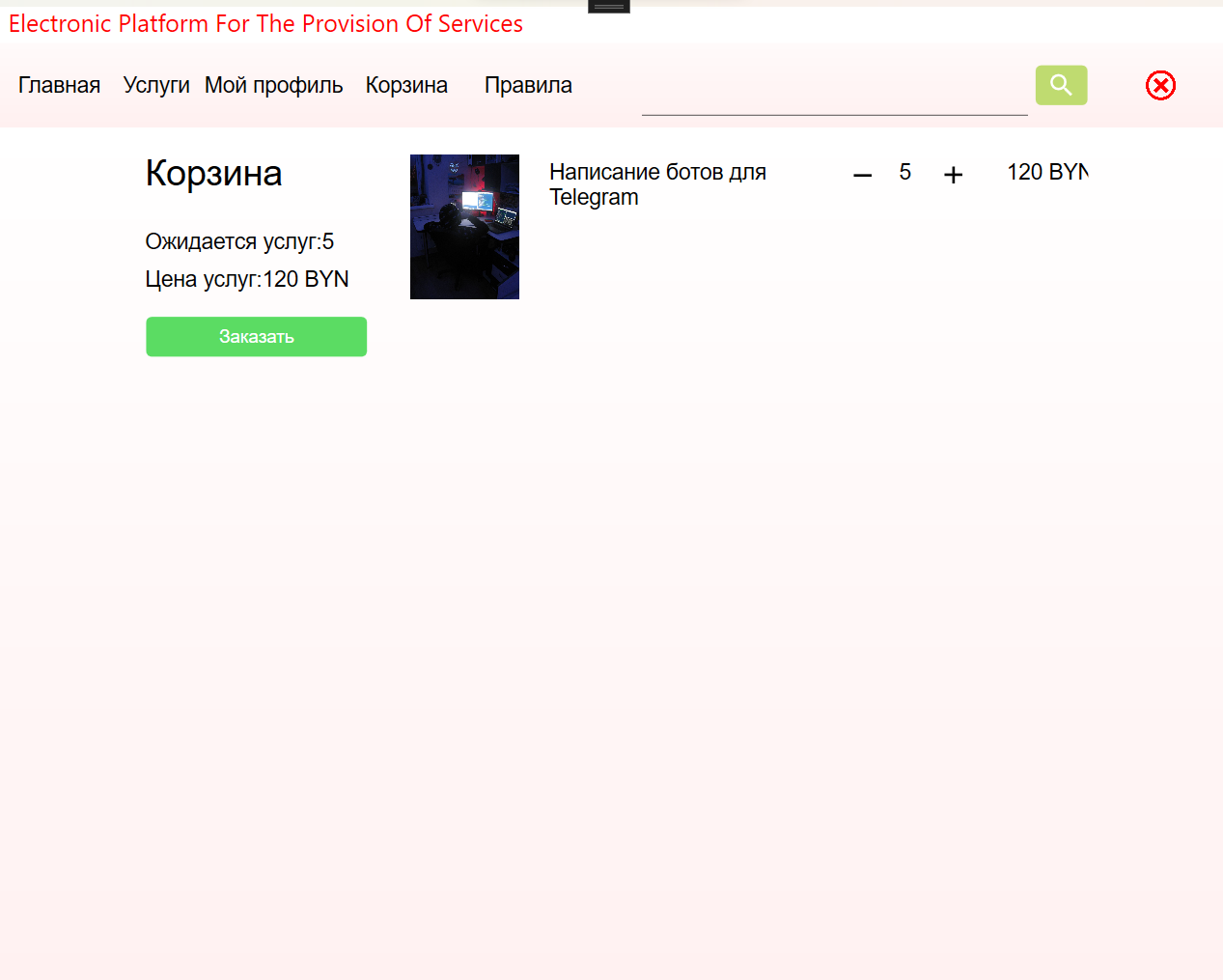


Рисунок 6.8 – Страница «Корзина»

По нажатию на кнопку оформить услугу пользователя перенаправляет на страницу, где он может выбрать удобный для себя способ оплаты и адрес выполнения. При выборе способа оплаты «Картой онлайн» откроется окно для привязки карты, в которой нужно будет ввести соответствующие данные. При выборе доставки курьером появится поле для ввода адреса доставки. На рисунке 6.9 продемонстрирована страница оформления заказа.

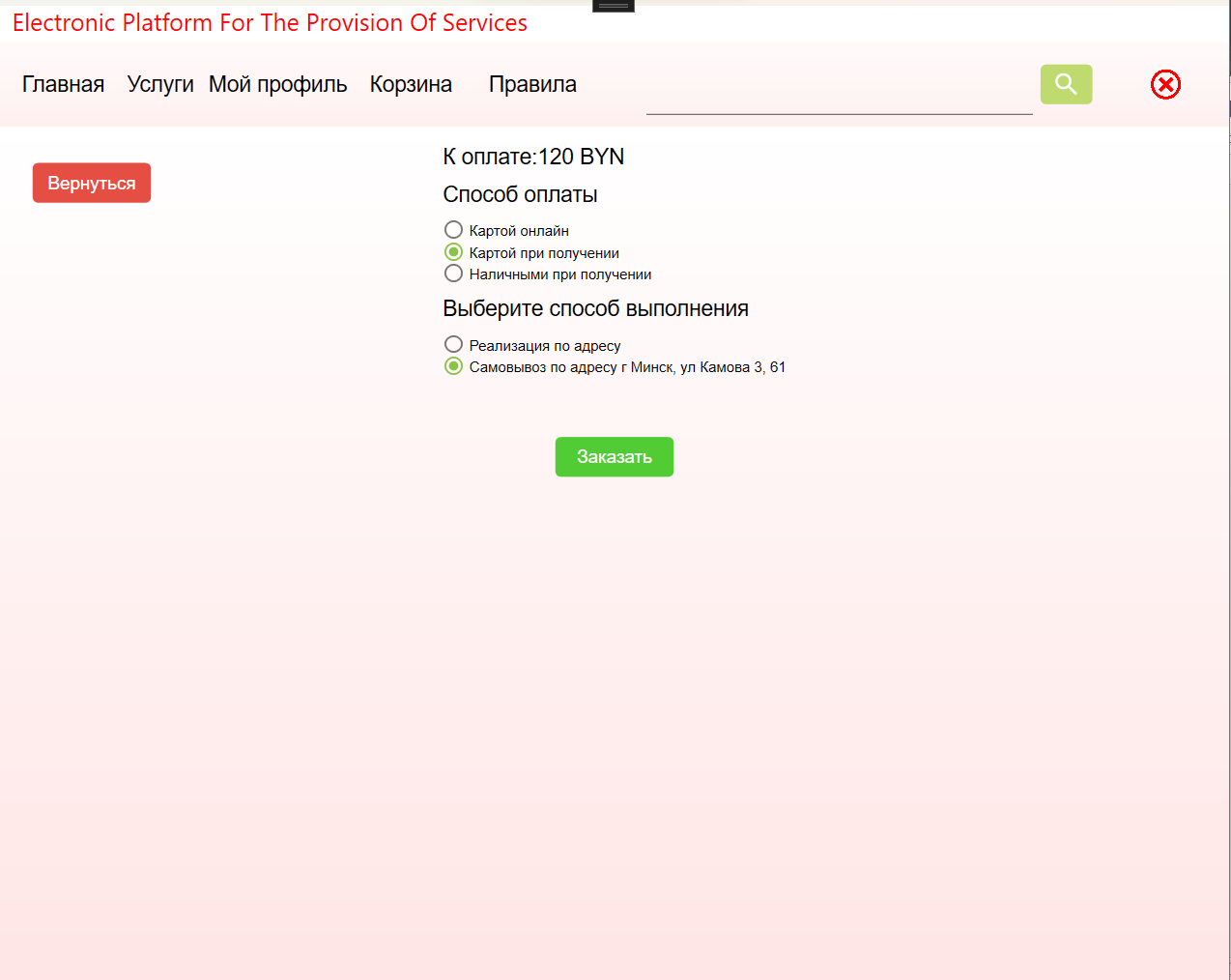


Рисунок 6.9 – Страница «Оформление услуги»

При входе в приложение под аккаунтом администратора присутствует возможность редактирования всех таблиц базы данных приложения, в чём и заключается особенность работы с приложением в роли администратора. Администратору доступны функции добавления, удаления, редактирования услуг и категорий. Страница администрирования представлена на рисунке 6.10.

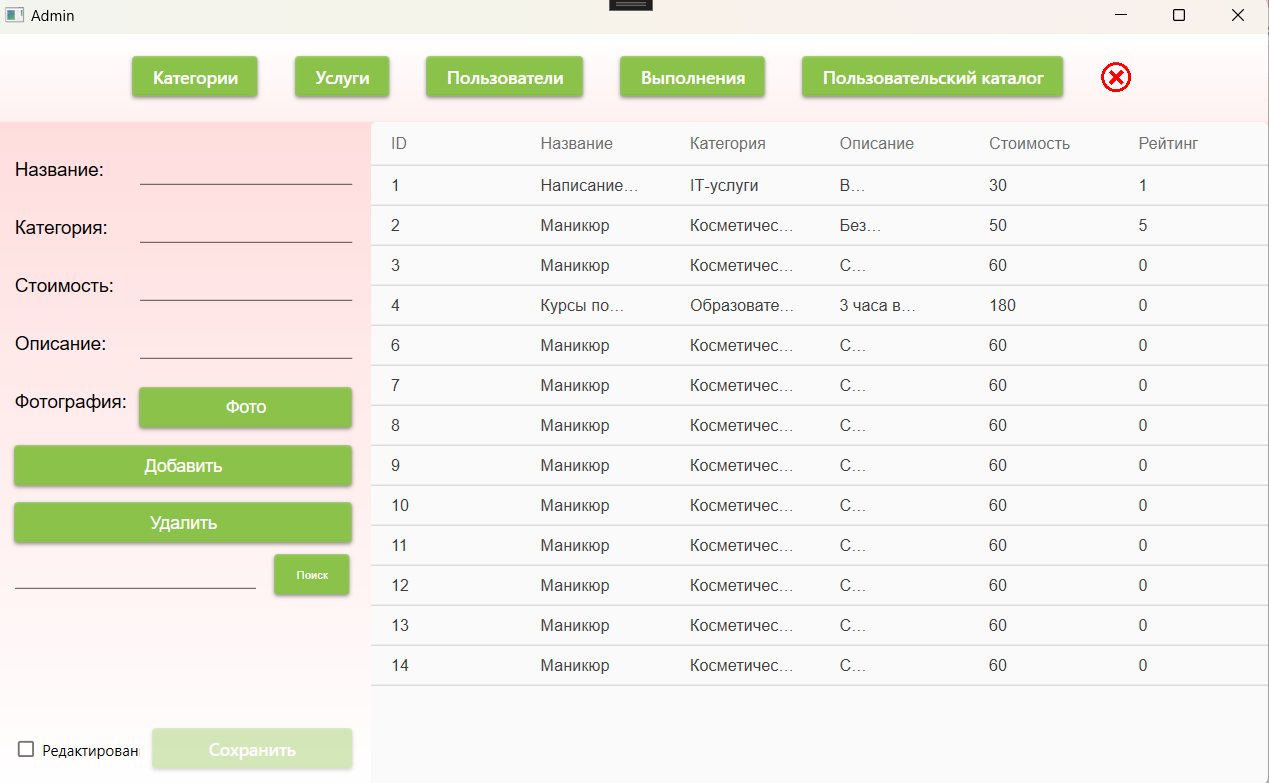
****

Рисунок 6.10 – Окно администратора

Также администратор имеет возможность просматривать базу данных пользователей и изменять статус заказа.

# **Заключение**

Основной задачей данного проекта было освоение ключевых технологий разработки веб-приложений для платформы по предоставлению услуг. Разработанное программное средство стало практической реализацией знаний, полученных в ходе изучения работы с базами данных, а также технологий веб-разработки, таких как WPF и Entity Framework Core.

В процессе работы была достигнута цель по созданию приложения для электронной площадки, предоставляющей услуги. Это приложение не только иллюстрирует применение теоретических знаний, но и представляет собой готовый продукт, который может быть использован для практических нужд.

В программном обеспечении реализованы следующие функции:

* регистрация, авторизация и смена пароля;
* выполнение поисковых запросов, фильтрация и сортировка услуг;
* добавление услуг в корзину;
* заполнение формы заказа;
* изменение данных пользователя;
* просмотр статуса заказа;
* возможность оставлять отзывы о качествах предоставленных услуг; создание, редактирование и удаление услуг и категорий;
* изменение статуса выполнения услуг.

Разработанная программа имеет завершённый вид, работает верно и готова к использованию.

**Список литературы**

1. YouDo [Электронный ресурс] / Режим доступа: URL <https://youdo.com/>

- Дата доступа: 26.04.2024

2. TaskRabbit [Электронный ресурс] / Режим доступа: URL <https://www.taskrabbit.com/> - Дата доступа: 26.04.2024

3. Yandex Услуги [Электронный ресурс] / Режим доступа: URL <https://uslugi.yandex.ru/> - Дата доступа: 26.04.2024

4. SQL и реляционные базы данных: учебное пособие / М. П. Баранов, С. М. Бабенко, Ю. В. Михайленко. – М.: Издательский центр «Академия», 2019. - 200 с.

5. Habr.com Связи между таблицами базы данных [Электронный ресурс] /

Режим доступа: <https://habr.com/ru/articles/488054/> - Дата доступа: 23.09.2024

6. Habr.com. Основы нормализации базы данных [Электронный ресурс] / Режим доступа: https://habr.com/ru/articles/488065/ - Дата доступа: 23.09.2024.

7. Habr.com. Как проектировать эффективные связи между таблицами базы данных [Электронный ресурс] / Режим доступа: https://habr.com/ru/articles/487900/ - Дата доступа: 23.09.2024.

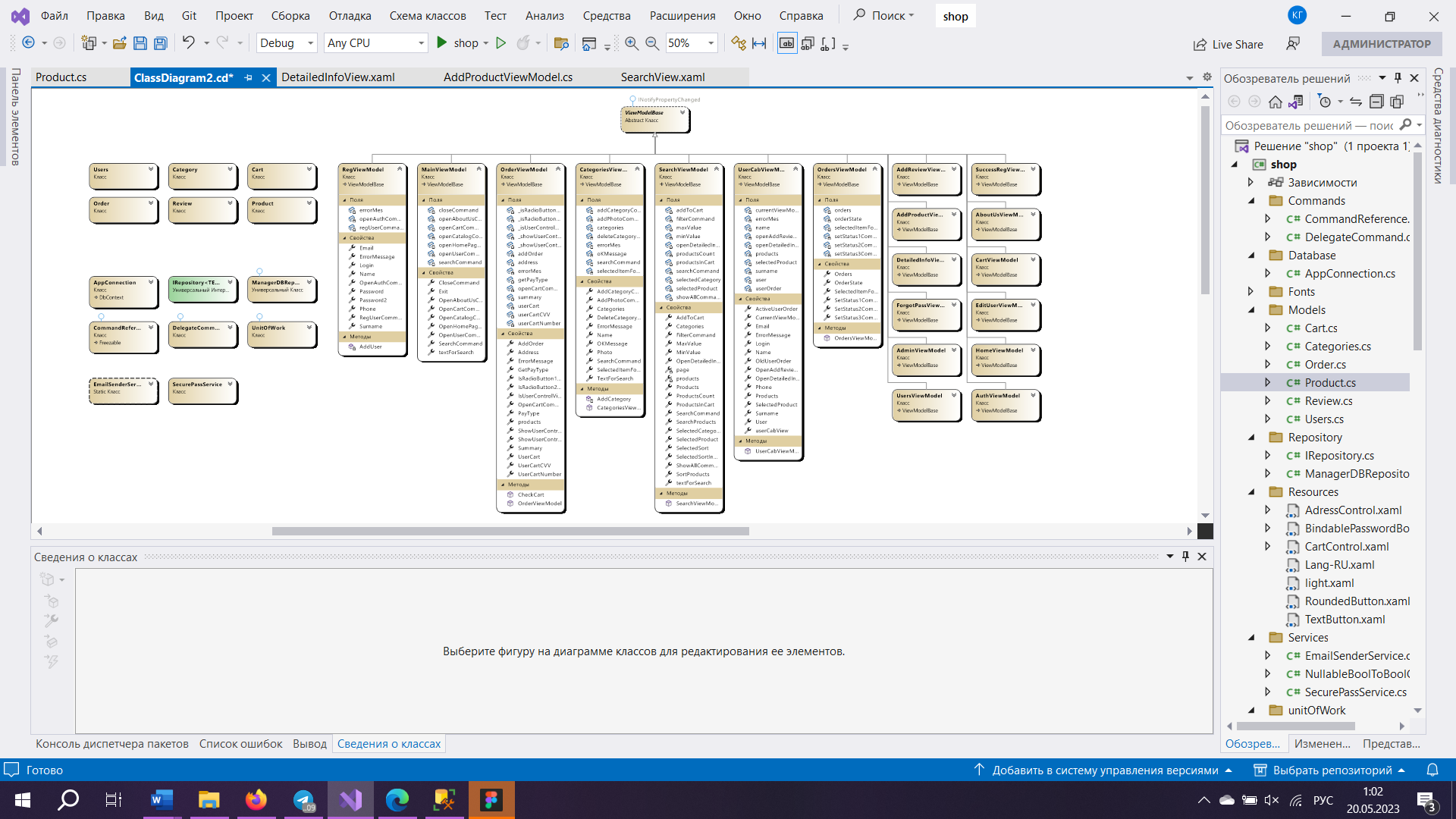
8. Habr.com. Реляционные базы данных: основные принципы и связи между таблицами [Электронный ресурс] / Режим доступа: https://habr.com/ru/articles/488200/ - Дата доступа: 23.09.2024.

9. Habr.com. Индексы и связи между таблицами в SQL [Электронный ресурс] / Режим доступа: https://habr.com/ru/articles/487911/ - Дата доступа: 23.09.2024.

10. Habr.com. Технологии связи таблиц и их оптимизация в реляционных СУБД [Электронный ресурс] / Режим доступа: https://habr.com/ru/articles/488050/ - Дата доступа: 23.09.2024.

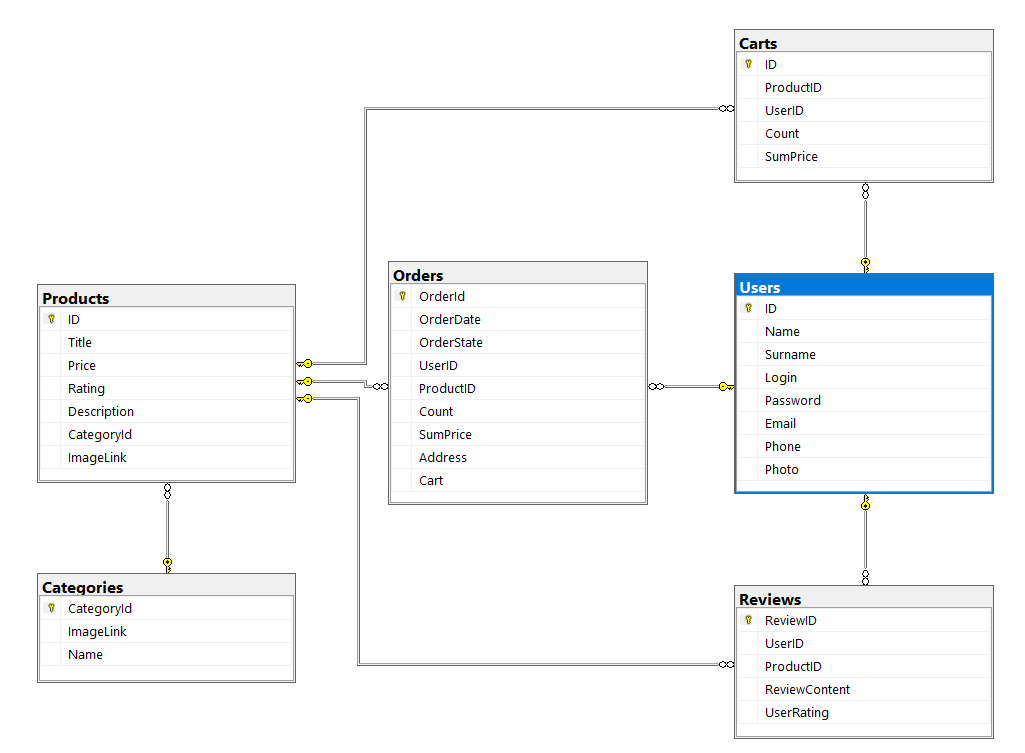
**Приложение А**

**Диаграмма классов**



**Приложение Б**

**Структура базы данных**



**Приложение В**

**Класс UnitOfWork**

|  |
| --- |
| public class UnitOfWork : IDisposable  {  private AppConnection db = new AppConnection();  private ManagerDBRepository<Cart> cartRepository;  private ManagerDBRepository<Category> categoryRepository;  private ManagerDBRepository<Order> orderRepository;  private ManagerDBRepository<Product> productRepository;  private ManagerDBRepository<Review> reviewRepository;  private ManagerDBRepository<Users> userRepository;  public ManagerDBRepository<Cart> CartRepository  {  get  {  if (cartRepository == null)  cartRepository = new ManagerDBRepository<Cart>(db);  return cartRepository;  }  }  public ManagerDBRepository<Category> CategoryRepository  {  get  {  if (categoryRepository == null)  categoryRepository = new ManagerDBRepository<Category>(db);  return categoryRepository;  }  }  public ManagerDBRepository<Order> OrderRepository  {  get  {  if (orderRepository == null)  orderRepository = new ManagerDBRepository<Order>(db);  return orderRepository;  }  }  public ManagerDBRepository<Product> ProductRepository  {  get  {  if (productRepository == null)  productRepository = new ManagerDBRepository<Product>(db);  return productRepository;  }  }  public ManagerDBRepository<Review> ReviewRepository  {  get  {  if (reviewRepository == null)  reviewRepository = new ManagerDBRepository<Review>(db);  return reviewRepository;  }  }  public ManagerDBRepository<Users> UserRepository  {  get  {  if (userRepository == null)  userRepository = new ManagerDBRepository<Users>(db);  return userRepository;  }  }  public void Save()  {  db.SaveChanges();  }    private bool disposed = false;  public virtual void Dispose(bool disposing)  {  if (!this.disposed)  {  if (disposing)  {  db.Dispose();  }  this.disposed = true;  }  }  public void Dispose()  {  Dispose(true);  GC.SuppressFinalize(this);  }  } |

**Приложение Г**

**Класс ManagerDBRepository<TEntity>**

|  |
| --- |
| public class ManagerDBRepository<TEntity> : IRepository<TEntity> where TEntity : class  {  DbContext \_context;  DbSet<TEntity> \_dbSet;  public ManagerDBRepository(DbContext context)  {  \_context = context;  \_dbSet = context.Set<TEntity>();  }  public IEnumerable<TEntity> Get()  {  return \_dbSet.AsNoTracking().ToList();  }  public IEnumerable<TEntity> Get(Func<TEntity, bool> predicate)  {  return \_dbSet.AsNoTracking().Where(predicate).ToList();  }  public TEntity FindById(int id)  {  return \_dbSet.Find(id);  }  public void Create(TEntity item)  {  \_dbSet.Add(item);  }  public void Update(TEntity item)  {  \_context.Entry(item).State = EntityState.Modified;  }  public void Remove(TEntity item)  {  \_dbSet.Remove(item);  }  public IEnumerable<TEntity> GetWithInclude(params Expression<Func<TEntity, object>>[] includeProperties)  {  return Include(includeProperties).ToList();  }  public IEnumerable<TEntity> GetWithInclude(Func<TEntity, bool> predicate,  params Expression<Func<TEntity, object>>[] includeProperties)  {  var query = Include(includeProperties);  return query.Where(predicate).ToList();  }  private IQueryable<TEntity> Include(params Expression<Func<TEntity, object>>[] includeProperties)  {  IQueryable<TEntity> query = \_dbSet.AsNoTracking();  return includeProperties  .Aggregate(query, (current, includeProperty) => current.Include(includeProperty));  }  public void RemoveRange(IEnumerable<TEntity> entities)  {  \_dbSet.RemoveRange(entities);  }  } |

**Приложение Д**

**Команда для добавления услуги**

|  |
| --- |
| private DelegateCommand addProducеCommand;  public ICommand AddProductCommand  {  get  {  if (addProducеCommand == null)  {  addProducеCommand = new DelegateCommand(() =>  {  try  {  ErrorMessage = string.Empty;  if (Title != null || SelectedCategory != null || Price != null || Description != null || ImageLink != null)  {  if (AddProduct(Title, SelectedCategory, Price, Description, ImageLink))  {  using (UnitOfWork unit = new UnitOfWork())  {  Product product = new Product();  product.Title = Title;  product.Category = unit.CategoryRepository.FindById(SelectedCategory.CategoryId) ;  product.Price = Convert.ToDouble(Price);  product.Rating = 0;  product.Description = Description;  product.ImageLink = ImageLink;  unit.ProductRepository.Create(product);  unit.Save();  }  ErrorMessage = string.Empty;  OKMessage = "услуга добавлена";  }  }  else throw new Exception("все поля долны быть заполнены");  }  catch (DbEntityValidationException e)  {  foreach (DbEntityValidationResult validationRes in e.EntityValidationErrors)  {  foreach (DbValidationError err in validationRes.ValidationErrors)  {  OKMessage = string.Empty;  MessageBox.Show(err.ErrorMessage);  ErrorMessage = err.ErrorMessage;  }  }  }  catch (Exception e)  {  OKMessage = string.Empty;  ErrorMessage = e.Message;  }  });  }  return addProducеCommand;  }  } |