МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Информационных систем и технологий

Специальность 1-40 05 01-03 «Информационные системы и технологии»

Направление специальности 1-40 05 01-03 Информационные системы и технологии

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине **«**Объектно-ориентированные технологии программирования и стандарты проектирования»

Тема «Агрегатор услуг»

Исполнитель

студентка 2 курса группы 4 Шабуня А.С

(Ф.И.О.)

Руководитель работы Ст. преподаватель Северинчик Н.А.

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

Курсовой проект защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Председатель Пацей Н.В.

(подпись)

Минск 2020

Оглавление

[Введение 4](#_Toc73127336)

[1. Аналитический обзор литературы 5](#_Toc73127337)

[1.1 Обзор и анализ аналогов 5](#_Toc73127338)

[1.2 Аналитический обзор источников 7](#_Toc73127339)

[2.Анализ требований к программному средству и разработка функциональных требований 8](#_Toc73127340)

[2.1 Функциональность системы 8](#_Toc73127341)

[2.2 Спецификация функциональных требований 10](#_Toc73127342)

[3.Проектирование программного средства 11](#_Toc73127343)

[3.1 Проектирование базы данных 11](#_Toc73127344)

[3.2 Структура проекта 13](#_Toc73127345)

[3.3 Проектирование архитектуры проекта 14](#_Toc73127346)

[3.4 Представления 15](#_Toc73127347)

[4. Реализация программного средства 17](#_Toc73127348)

[4.1 Описание разработанных классов 17](#_Toc73127349)

[4.2 Взаимодействие представления и модели представления 18](#_Toc73127350)

[4.3 Работа с базой данных 18](#_Toc73127351)

[5. Тестирование, проверка работоспособности и анализ полученных результатов 21](#_Toc73127352)

[5.1 Тестирование авторизации и регистрации 21](#_Toc73127353)

[5.2 Тестирование функций приложения 23](#_Toc73127354)

[5.3 Демонстрация некоторых возможностей приложения 24](#_Toc73127355)

[6. Руководство по установке и использованию 26](#_Toc73127356)

[6.1 Требования к системе и установке программного средства 26](#_Toc73127357)

[6.2 Руководство использования 26](#_Toc73127358)

[Заключение 27](#_Toc73127359)

[Список использованных источников 28](#_Toc73127360)

[Приложение А 29](#_Toc73127361)

[Листинг кода класса PasswordHash 29](#_Toc73127362)

[Приложение Б 30](#_Toc73127363)

[Пример хэширования с использованием алгоритма SHA256 30](#_Toc73127364)

## Введение

В современном обществе, когда человечество обеспечено ресурсами и нет необходимости думать о пропитании, когда человек может быть достаточно обеспечен чтобы не работать или заниматься только тем, к чему лежит его душа складывается ситуация, в которой человек современного общества не должен обладать навыками по ремонту сантехники или фурнитуры, ведь есть люди, которые этими навыками обладают.

Агрегатор Услуг - площадка, которая является связующим мостом между людьми, которые нуждаются в помощи и людьми, которые эту помощь могут оказать. Основной идеей приложения является упрощение поиска нанимателей для людей, которые обладают навыками, не требующими постоянного использования и обратное – поиск исполнителей в определённой сфере людьми, которые в этой области не разбираются.

Например, есть человек, который очень хорошо умеет мыть окна, и он размещает соответствующее объявление на площадке. Существует и противоположность – человек, который не представляет, как мыть окна, заказывать уборку всего помещения дороже, чем нанять работника для мойки окон, воспользовавшись приложением наниматель найдёт специалиста для конкретной задачи, а специалист применение своим навыкам.

# 1. Аналитический обзор литературы

## 1.1 Обзор и анализ аналогов

Для создания приложения не требуется изобретать велосипед, концептуально – это обычный агрегатор, в основе которого лежит простая задача – собирать и группировать объявления, товары, новости.

В качестве анализа и разбора анализов использовались агрегаторы разных категорий: новостей, товаров, услуг.

**Deal.by**

Агрегатор товаров от Deal.by – удобная площадка, где продавцы могут разместить свои товары в определённых категориях, что упрощает поиск со стороны пользователя. Основным отличием площадки от схожих является более бытовое направление и отсутствие, в пределах разумного, ограничений на товары для размещения.

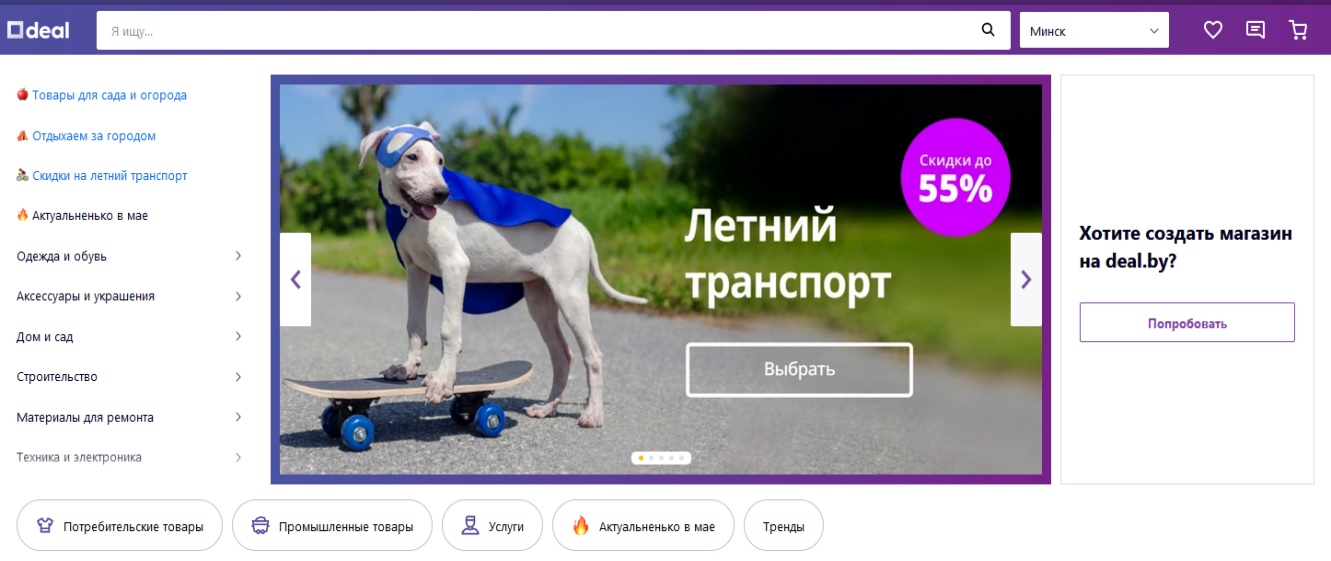


Рисунок 1.1 - Deal.by

**Onliner.by**

Более известный агрегатор товаров, удобство которого заключается в удобном каталоге и грамотном распределении по категориям, на данной площадке в основном размещаются интернет магазины. На данной площадке уже введены ограничения на товары, товар должен обладать определённой степенью востребованности и иметь продавца для размещения на площадке

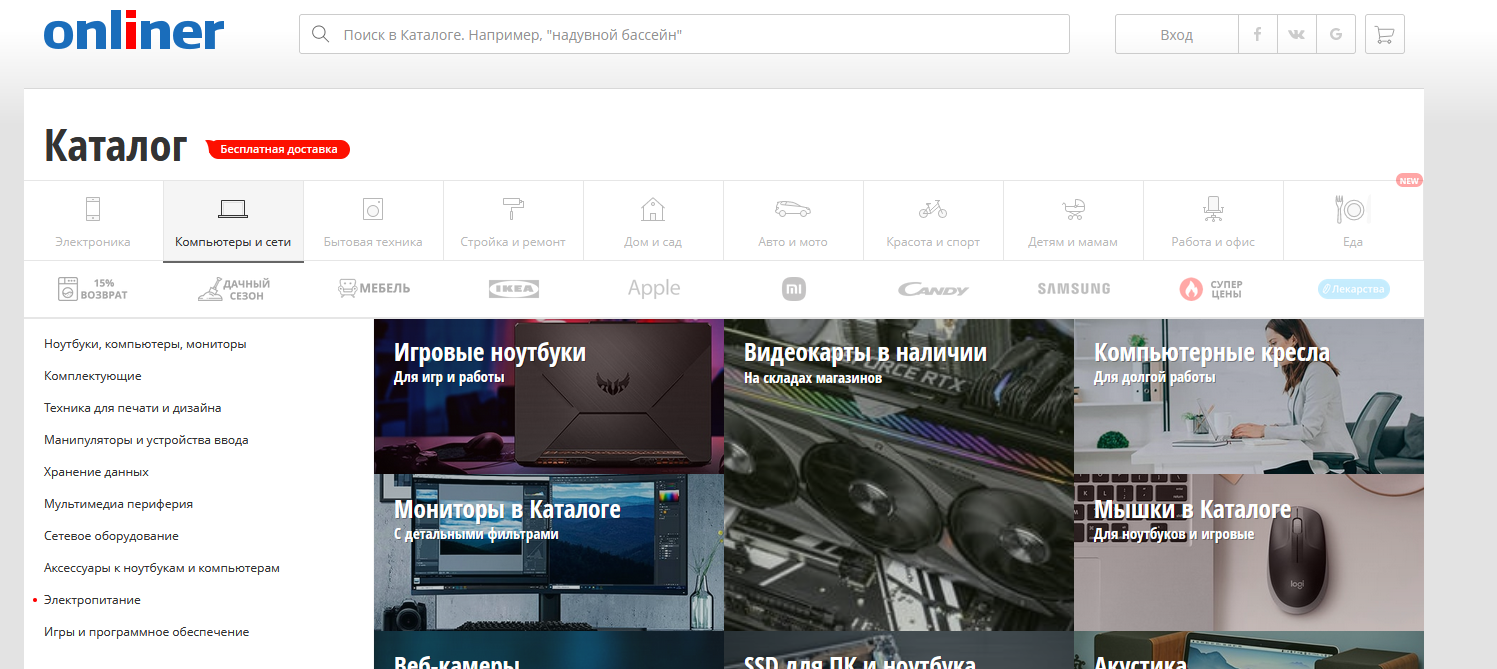


Рисунок 1.1 – Onliner.by

**Яндекс Такси**

Пример агрегатора услуг, в данном случае – услуги такси, особенность заключается в том, что Яндекс Такси сотрудничает как с таксопарками, так и с обычными водителями. Клиент размещает объявление – заказывает поездку из точки А в точку Б, приложения на основе этих данных и данных об общей загрузке формирует маршрут и стоимость, после чего рассылает предложение водителям по близости, пока кто-то из них не примет предложение. Преимуществами данного агрегатора являются: формирование стоимости до момента заказа, удобные карты, система отзывов, что стимулирует водителей придерживаться определённых правил.

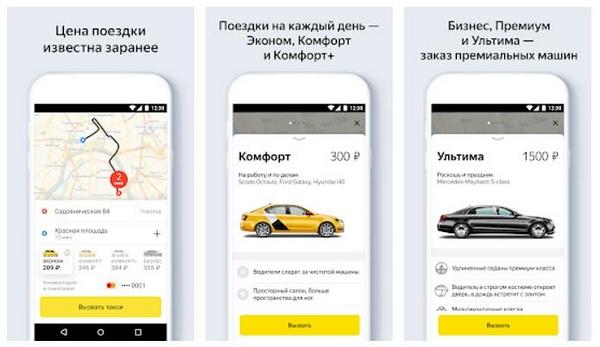


Рисунок 1.1 – Приложение «Яндекс такси»

## 1.2 Аналитический обзор источников

В ходе подготовки перед созданием приложения были изучены различные источники информации, как специализированные: техническая, учебно-методическая и справочная литература, так и менее узконаправленные: статьи и материалы, опубликованные в сети интернет.

Основным источник при создании приложения была Документация Microsoft – основное пособие для языка C#, в качестве дополнительных использовались отдельные главы из других справочников и учебников.

Дополнительная информации о принципах работы с WPF была получена из интернет-источника «Metanit», содержащего практические советы по работе с технологией.

Работа с базой данных была взята из статьи «Подходы для работы с Entity Framework», где были приведены различные подходы работы с базой данных, откуда был выбран наиболее подходящий под задачу вариант.

Дополнительная информация о сочетании цветов была UI/UX дизайне была взята из статьи «WPF C# Professional Modern Flat UI Tutorial», в который были рассмотрены и показаны на примерах основные подходы в современном дизайне корпоративных приложений.

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

# 2.1 Функциональность системы

Функционал приложения зависит от роли пользователя, в зависимости от роли функционал системы должен отвечать следующим требованиям.

Заказчик:

* регистрация;
* авторизация;
* просмотр доступных услуг;
* заказ услуги;
* просмотр и управление заказами;
* просмотр профилей исполнителей;
* оставить отзыв о работе исполнителя.

Диаграмма использования покупателя представлена на рисунке 2.1



Рисунок 2.1 - Диаграмма использования заказчика

Если пользователь размещает услугу, то он получает функции исполнителя.

Исполнитель:

* редактировать услугу;
* управление заказами;
* удалить услугу.

Диаграмма использования исполнителя представлена на рисунке 2.2

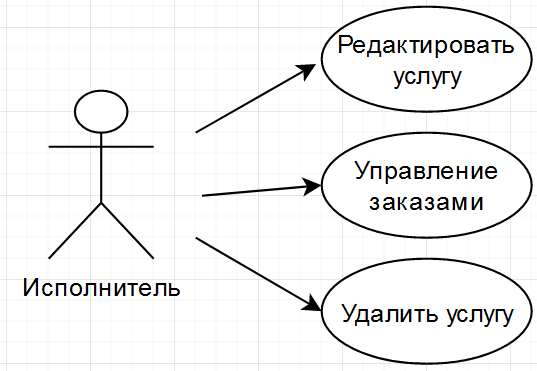


Рисунок 2.2 – Диаграмма использования исполнителя

В случае, когда пользователь обладает правами администратора ему доступны функции администратора, при этом функции заказчика и исполнителя у него не пропадают.

Администратор:

* просмотр пользователей;
* блокировка пользователей;
* управление правами администратора.

Диаграмма использования администратора представлена на рисунке 2.3

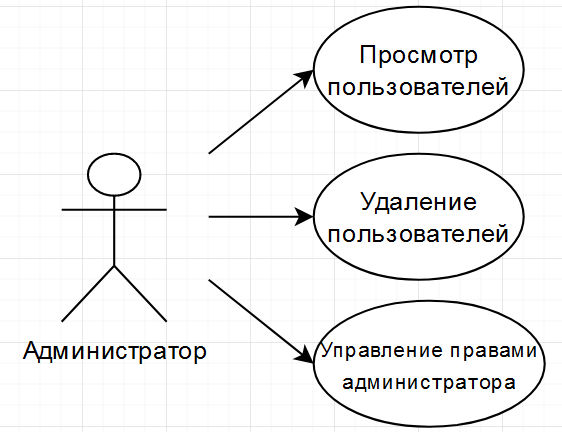


Рисунок 2.3 – Диаграмма использования администратора

## 2.2 Спецификация функциональных требований

Спроектированная система отвечает следующим функциональным требованиям:

* при запуске приложения пользователь может как авторизоваться, так и зарегистрироваться;
* в случае ошибок при регистрации или авторизации пользователь получает соответствующее уведомление;
* в зависимости от роли пользователя ему должен быть предоставлен свой функционал, определенный с учетом диаграмм использования;

# 3.Проектирование программного средства

## 3.1 Проектирование базы данных

Для проекта балы спроектирована база данных под названием «ServiceAgreagtor». Для управления базой данных используется ПО Microsoft SQ Server 2018. Логическая схема данных представлена на рисунке 3.1

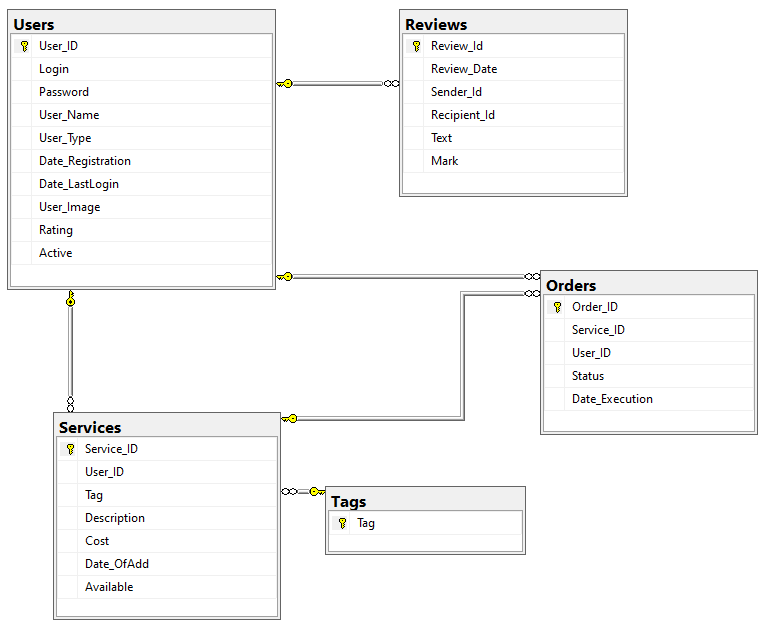


Рисунок 3.1 – Логическая схема данных

Таблица Users содержит всю информацию о пользователях приложения:

* идентификатор пользователя (User\_ID int PK)
* логин (Login nvarchar (25))
* пароль (Password nvarchar(95))
* имя пользователя (User\_Name)
* роль (User\_Type smallint)
* дата регистрации (Date\_Registrtion date)
* дата последнего входа (Date\_LastLogin date)
* изображение профиля (User\_Image varbinary(MAX))
* рейтинг (Rating float)
* статус, активен или заблокирован (Active bit)

Таблица Services содержит в себе информацию об услугах, которые размещают пользователи, а именно:

* идентификатор услуги (Service\_ID int PK)
* идентификатор исполнителя (User\_ID int FK)
* категория, в которой разместили услугу (Tag nvarchar(15))
* описание (Description ntext)
* стоимость (Cost nmoney)
* дата размещения (Date\_OfAdd date)
* доступна услуга или удалена (Available bit)

Таблица Orders содержит информацию о размещенных заказах:

* идентификатор услуги (Order\_ID int PK)
* идентификатор услуги (Service\_ID int FK)
* идентификатор заказчика (User\_ID int FK)
* статус заказа (Status nvarchar(25))
* дата, на которую сделан заказ (Date\_Execution date)

Таблица Reviews содержит в себе информацию о отзывах которые пользователи оставили под профилями исполнителей:

* идентификатор отзыва (Review\_Id int PK)
* дата отзыва (Review\_Date date)
* идентификатор отправителя (Sender\_Id int FK)
* идентификатор получателя (Recepient\_Id int FK)
* текст отзыва (Text text)
* рейтинг, выставленный исполнителю Mark(float)

Таблица Tags содержит категории, по которым можно добавить услугу:

* Категория (Tag nvarchar(15))

## 3.2 Структура проекта

Структура проекта представлена на рисунке 3.2.

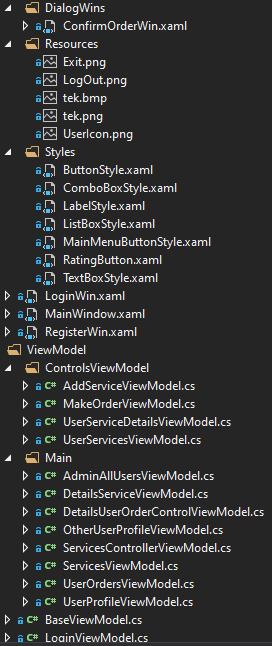
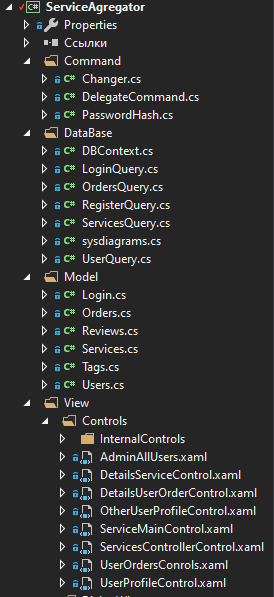
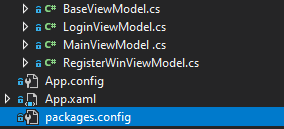
 

Рисунок 3.2 – Структура проекта

Проект создан с использованием паттерна проектирования MVVM, согласно которому элементы управления разделены на три группы:

Model – группа, содержащая в себе классы, которые хранят информацию.

ViewModel – модель представления. Именно она связывает модель и представление через механизм привязки данных.

View – представления, отвечает за отображение информации и включает в себя окна и страницы проекта.

В проекте, также присутствуют группы файлов, которые не входят в MVVM, но также являются частью бизнес-логики или визуальной состосвляющей.

DataBase – содержит классы для работы с базой данных: контекст данных и классы с запросами к определённым таблицам.

Resources - используемые проектом ресурсы и языки.

Styles – используемы в проекте стили.

## 3.3 Проектирование архитектуры проекта

Для работы с базой данных используется технология Entity Framewrok – объектно-ориентированное средство работы с базой данных при помощи сущностей без необходимости писать сложные запросы.

Работы с Entity Framework расположена в группе DataBase и представлена в следующих классах:

* DBContext – основной класс – контекст базы данных, содержит в себе модель таблицы.
* LoginQuery – класс, который содержит запросы к базе данных, связанные с авторизацией пользователя.
* RegisterQuery – содержит в себе инструменты для регистрации нового пользователя.
* UserQuery – отвечает за запросы на получение информации связанной с пользователями и работу с таблицами Users и Reviews.
* ServiceQuery – отвечает за запросы на получение информации связанной с услугами и работу с таблицами Services и Tags.
* OrdersQuery – отвечает за запросы на получение информации связанной с заказами и работу с таблицей Orders.

Модель данных MVVM представлена в группе Models и представлена следующими классами:

* Login – класс, который содержит свойства, необходимые для авторизации;
* Users – класс, представляющий информацию о пользователях;
* Services – представляет информацию об услугах;
* Orders – представляет информацию об заказах;
* Reviews – класс, представляющий информацию об отзывах;
* Tag – класс, представляющий информацию о категориях, в которых могут быть добавлены услуги.

DelegateCommand – класс, который располагается в группе Command, требуемый для реализации команд. С помощью него элемент управления может быть привязана не к обработчику, а к действию, определенному во ViewModel.

Классы для работы с моделью представлены в группе ViewModel, вот основные из них:

* LoginViewModel – авторизация пользователя.
* RegisterViewModel – регистрация пользователя;
* MainViewModel – работает с меню и отвечает за переходы между окнами приложения;
* ServicesViewModel – работа со списком доступных услуг;
* DetailsServiceViewModel – работа с функциями, досутпными при просмотре одной услуги. Связан с побочным классом MakeOrderViewModel, который отвечает за заказ услуги;
* ServiceControllerViewModel – класс, отвечающий за работу исполнителя с заказами, которые он добавил, связан с классами редактирования и деталей заказа – UserServiceDetailsViewModel, AddServiceViewModel – добавление новой услуги;
* UserOrdersViewModel – работа с заказами пользователя;
* UserProfileViewModel - работа профилем текущего пользователя;
* OtherUserProfile – работа с профилем исполнителя.

## 3.4 Представления

Представления – визуальная часть проекта – окна и страницы проекта, они осуществляют работу с моделями представления и через них работают с базой данных.

Окна и страницы:

* LoginWin – стартовое окно приложения в котором происходит авторизация и переключение на окно регистрации RegisterWin
* MainWindow – главное окно приложения, в нём реализовано меню навигации, в нём расположено большинство страниц приложения;
* ServicesMainControl – является стартовой страницей в Main Window на данной странице отображаются услуги, размещённые исполнителями в открытый доступ, с этой страницы можно перейти в меню конкретной услуги;
* DetailsServiceControl – меню конкретной услуги, в котором ожно посмотреть детали или заказать услугу;
* ServiceControllerControl – страница, через которую исполнитель занимается менеджментом своих размещённых услуг, может редактировать их, добавить новые и проводить операции с заказами на услуги;
* UserOrdersControl – страница через которую пользователь может просмотреть информацию о заказах которые он совершил и проводить над ними операции;
* UserProfileСontrol – отображение информации о текущем польщователе;
* OtherUserProfile – страница просмотра профиля другого пользователя, на которой можно просмотреть информацию о нём, отзывы о его работе и оставить отзыв.

Диаграмма последовательности, представленная на рисунке 3.3, отражает процесс авторизации, выбор нужной книги, заказа и отмену его.

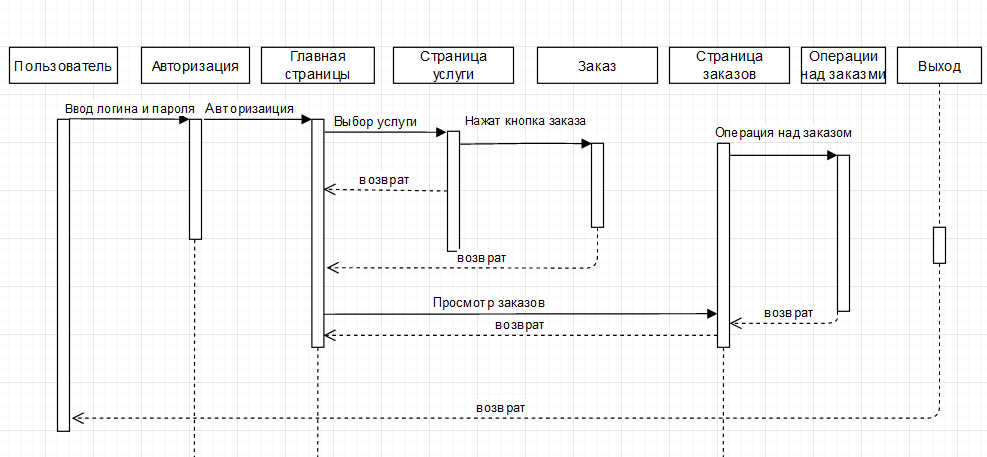


Рисунок 3.4 – Диаграмма последовательности пользователя.

# 4. Реализация программного средства

## 4.1 Описание разработанных классов

В основе разработанного приложения лежит паттерн проектирования MVVM, составляющие которого были описаны выше. В соответствии с паттерном файлы приложения были разделены по категориям: модель, модель представления, представления. Классы моделей содержат в себе свойства, отслеживаемые при помощи интерфейса INotifyPropertyChanged. Данный интерфейс реализован в классе BaseViewModel, от которого наследуются остальные классы. Данный интерфейс добавляет свойствам метод OnPropertyChanged(), который позволяет отслеживать изменения данных свойств. Классы моделей представления используют свойства связанных с ними моделей. Частичное дублирование кода неизбежно, но это позволяет создавать привязки при использовании данных моделей представления в окнах. Также недостатком MVVM является неизбежность нарушения паттерна в некоторых ситуациях. Паттерн предполагает использование привязки данных и исполнение команд без использования Code Behind, но из-за ограничений привязки могут возникать ситуации, в которых приходится нарушать MVVM.

Переключение страниц приложения в MainWindow реализована через переключение пользовательских элементов управления (User Control), при переключении мы меняем текущий контекст данных, это осуществлено через создание класса Changer с использованием – паттерна Singleton, который гарантирует что данный класс будет иметь всего один экземпляр и что он будет доступен из любой точки программы, этот объект содержит в себе свойство класса MainViewModel, через которое и осуществляется смена страниц. Также класс Changer содержит статические свойства IsAdmin – значение, которое характеризует права текущего пользователя и CurrentUserID – идентификатор текущего пользователя.

Разработанное приложение по правилам должно отвечать определённым требованиям отказоустойчивости и безопасности, одним из аспектов которой является правильное хранение паролей пользователей. Для хранения пользовательских паролей был выбран криптографический алгоритм SHA256, хэш-функция находится в классе PasswordHashи реализована за счёт встроенных средств C# из библиотеки System.Security.Cryptography. В классе PasswordHash, помимо хэш-функции находится функция, которая делает запрос в базу данных с целю проверить соответствует ли хэш введённого пароля значению из базы данных, в случае положительного ответа пользователь получает доступ в систему и устанавливает для пользователя новую дату последнего входа.

Основой многих классов является коллекция ObservableCollection<T>, где T – имя соответствующе модели. В ходе разработки приложения были разработаны методы расширения для ObservableCollection<T> для возможности записать в коллекцию данного типа данные, полученные через Entity Framework из базы данных.

## 4.2 Взаимодействие представления и модели представления

Каждое окно и страница используют привязку контекста данных к модели представления. Благодаря этому окно имеет возможность отображать связанные данные и выполнять над ними действия в соответствии с командами, определенными в привязанной к контексту данных модели представления.

Для корректного использования данных модели представления конструкторы некоторых окон и страниц в качестве параметров принимают модель представления другого окна или же связанные с моделью представления экземпляры моделей. Это используется, например, в окне редактирования, просмотра страницы пользователя или просмотре страницы отдельной книги.

Большинство действий осуществляется при нажатии элементов управления, причем кнопки связаны не с обработчиками, а с командами используемой модели представления при помощи Command=”{Binding }”. Это также является одним из элементов паттерна MVVM.

## 4.3 Работа с базой данных

Работа с базой данных осуществляется с использованием технологии Entity Framework. В соответствии с подходом Code-First, вначале пишутся классы, на основе которых будут созданы таблицы. Затем создается класс Context. В этом классе публичные свойства DbSet<T>, где T – модель, в соответствии с которой должна быть создана таблица. Необходимо также учитывать, что изменение модели приведет к изменению базы данных, поэтому для предотвращения ошибок выполняются миграции. Функции запросов к базе данных и работы с таблицами расположены в группе файлов DataBase, в проекте работа с базой данных – CRUD функции и некоторые специфические запросы, устроена так что все возможные запросы написаны в классах группы DataBase с использованием LINQ to Entity, таким образом модели представления взаимодействуют с базой данных через эти классы.

В качестве пример работы с базой данных можно привести в пример процесс авторизации, представленный на риснуке 4.1

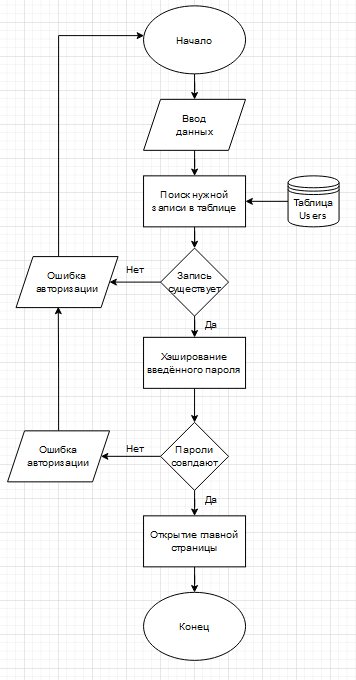


Рисунок 4.1 – Блок-схема авторизации

Пользователь вводит данные для авторизации – логин и пароль, после чего через класс LoginQuery происходит запрос на поиск пользователя по логину, в случае если пользователь найден функция возвращает эту запись, после чего управление передаётся в описанный класс PasswordHash, который проверяет на соответствие хэш введённого пароля со значением пароля из записи таблицы, в случае если данные равны, то происходит открытие главной страницы приложения.

При работе с базой данных через Entity Framework, также, является встроенная валидация, которая не позволит добавить в базу данных объект модели, в случае если её свойства не соответствуют ограничениям полей в таблице. Это преимущество Entity Framework, но ошибка валидации вызывает остановку приложения, так что это преимущество не избавляет от необходимости собственной валидации, одна из которых показана на рисунке 4.2 на примере добавления записи о услуге.

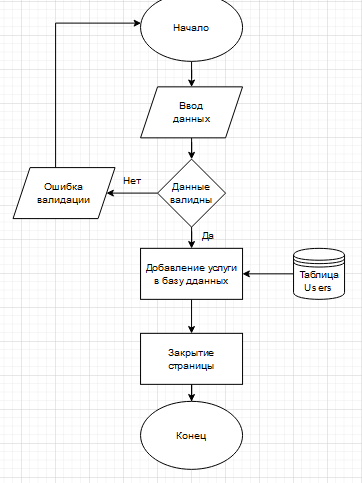


Рисунок 4.2 – Блок-схема добавления записи об услуге в базу данных

# 5. Тестирование, проверка работоспособности и анализ полученных результатов

## 5.1 Тестирование авторизации и регистрации

Окно авторизации является стартовым окном в приложении, в нём пользователь должен ввести логин и пароль для авторизации, в случае если пользователь вводит не верные данные или пользователь заблокирован ему покажет соответствующее сообщение об ошибке.

Примеры ошибок авторизации и блокировки пользователя показаны на рисунках 5.1 и 5.2 соответственно.

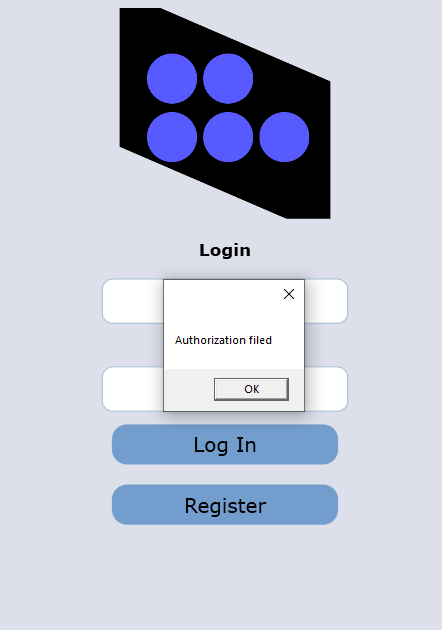


Рисунок 5.1 – ошибка авторизации, неверные данные

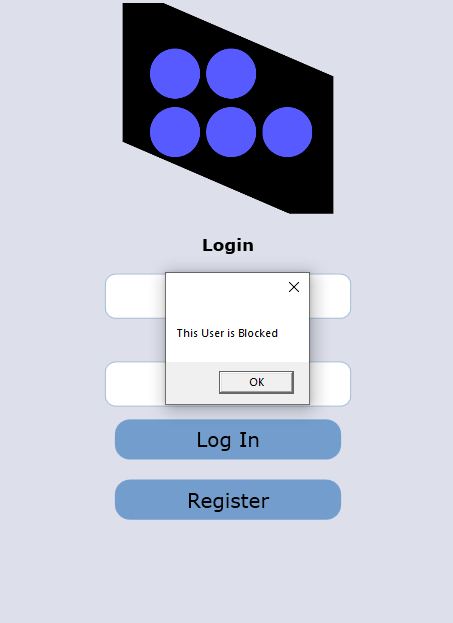


Рисунок 5.2 – ошибка авторизации, пользователь заблокирован.

Из окна авторизации пользователь может попасть в окно регистрации, после ввода данных и нажатия кнопки регистрации в случае, если пользователь с введённым логином уже существует, то высвечивается соответствующее сообщение об ошибке – рисунок 5.3

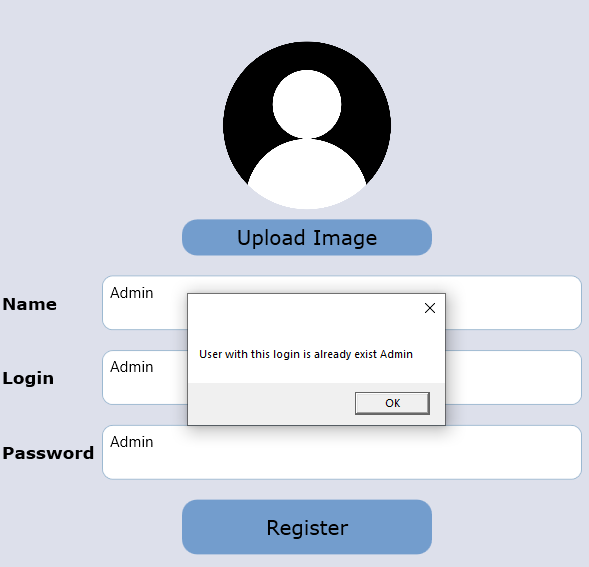


Рисунок 5.3 – ошибка регистрации – пользователь уже существует.

## 5.2 Тестирование функций приложения

При переходе на страницу конкретной услуги пользователь может нажать на кнопку совершения заказа, но было бы не очень правильно если бы пользователь мог бы заказывать одну и ту же услугу несколько раз, если она уже заказана, при попытке заказать уже заказанную услугу пользователь вызовет срабатывание соответствующей ошибки – рисунок 5.4

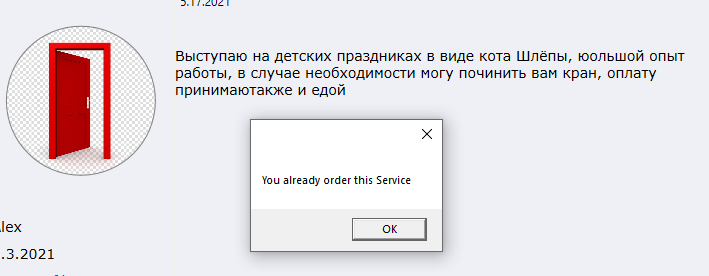
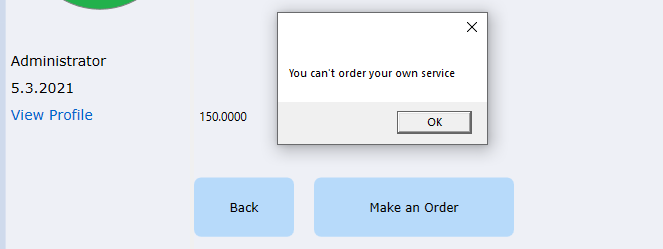
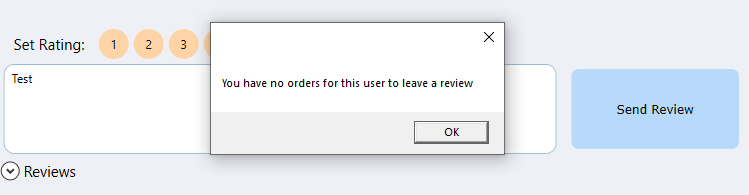


Рисунок 5.4 – ошибка повторного заказа услуги

При попытке заказа собственной услуги приложение также оповестит пользователя о неправильности его действий – рисунок 5.5



Также в приложении из списка услуг, заказов или страницы услуги можно перейти в профиль исполнителя и оставить отзыв о работе исполнителя, отзыв можно оставить только в случае, если у вас имеется заказ на конкретную услугу, при попытке оставить отзыв без заказа польщователь увидит сообщение об ошибке – рисунок 5.6

Рисунок 5.6 – сообщение об ошибке, попытка оставить отзыв без заказа

## 5.3 Демонстрация некоторых возможностей приложения

При переходе пользователя на страницу его профиля он может увидеть информацию, которую он оставил в приложении, свой текущий рейтинг исполнителя, а также отзывы, в случае если они есть, в случае если их нет, нижняя часть страницы остаётся пустой -рисунки 5.7 и 5.8

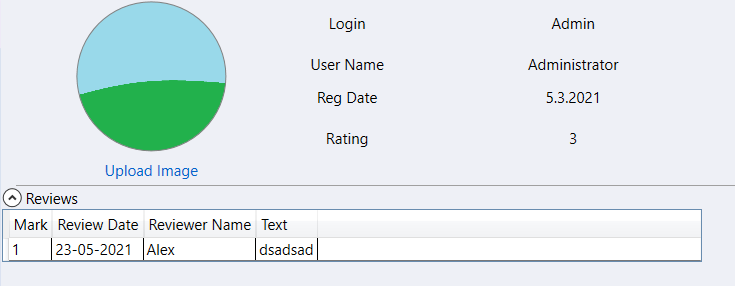


Рисунок 5.7 – информация на странице пользователя

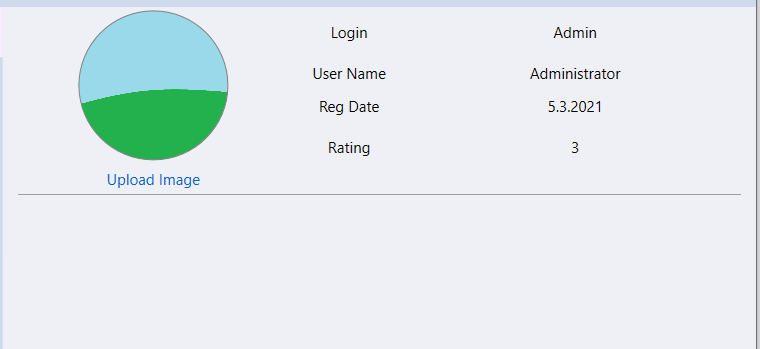


Рисунок 5.8 – информация на странице пользователи, если отзывы отсутствуют

При переходя на страницу управления размещёнными услугами пользователь может видеть информацию о его услугах, а также может добавить новую, после добавления его услуга появляется на странице размещённых услуг, видна всем пользователям и доступна для заказа – рисунки 5.9, 5.10 и 5.11.

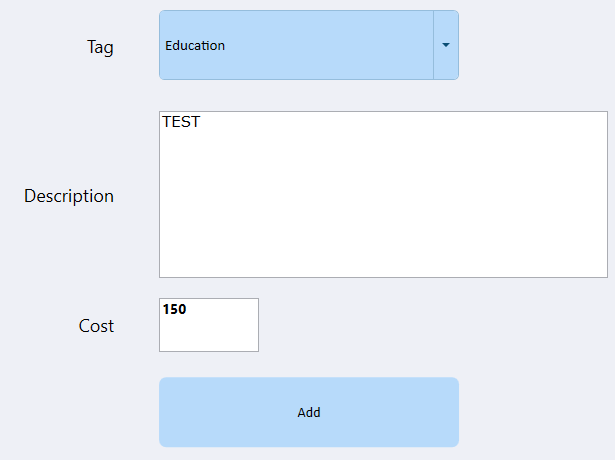


Рисунок 5.9 – пример добавления услуги пользователем

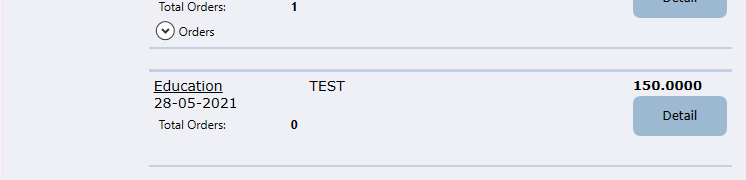


Рисунок 5.10 – добавленная услуга в списке услуг пользователя

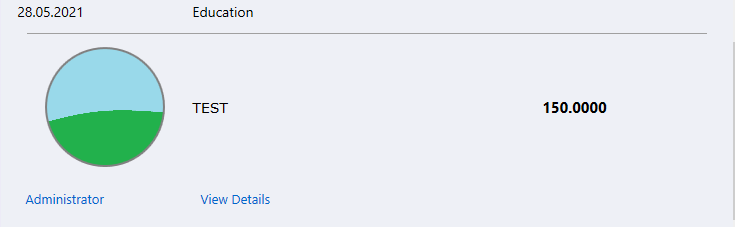


Рисунок 5.11 – добавленная услуга в общем списке услуг

# 6. Руководство по установке и использованию

## 6.1 Требования к системе и установке программного средства

Минимальные системные требования представлены в таблице 6.1.

Таблица 6.1 - Минимальные системные требования

|  |  |
| --- | --- |
| **Операционная система** | Windows XP и выше |
| **Версия .NET Framework** | .NET Framework 4.6.1 |
| **Процессор** | Минимум 1 Ггц |
| **ОЗУ** | 512 МБ |
| **Минимум дискового**  **пространства** | x64 – 4,5 Гб x32 – 4,5 Гб |

Программное средство представляет собой уже собранный проект, не требующий инсталляции. Запуск осуществляется через файл КП.exe.

## 6.2 Руководство использования

Вход в систему начинается с окна Авторизации. Пользователь вводит логин и пароль, после чего нажимает кнопку «Войти». При корректном вводе данных пользователь попадает на главное окно. В случае, если пользователь не зарегистрирован в системе, то в окне Авторизации он может нажать кнопку «Регистрация». Пользователь может зарегистрироваться, но для этого ему будет необходим код, присылаемый на почту.

# Заключение

В ходе курсового проектирования была спроектирована автоматизированная система – агрегатор услуг. Были изучены системы-аналоги и их особенности, в соответствии с которыми разработано ТЗ для проектируемой системы. Функционал достаточен для системы с не слишком большим количеством книг и пользователей.

Проектирование системы проведено с учетом некоторых наиболее популярных паттернов. Связи между проектируемыми классами отображены на диаграммах.

Структура таблиц создана с учетом возможных связей с другими таблицами. Для хранения данных разработана локальная база данных, работа с которыми велась с использованием Entity Framework.

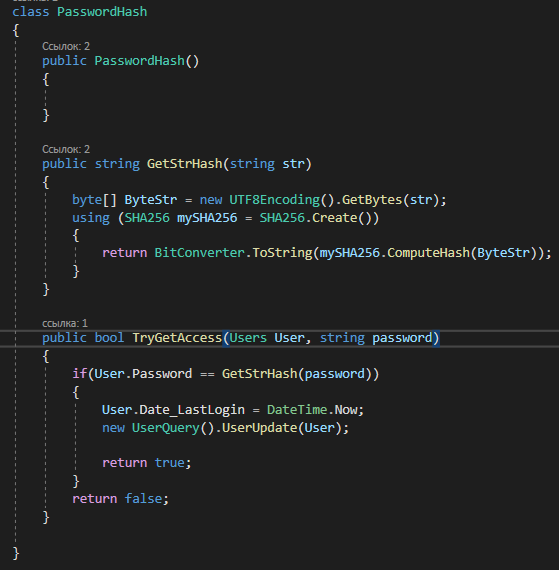
Система была отлажена и протестирована, определены минимальные параметры аппаратного и программного обеспечения.

# Список использованных источников

1. Блинова Е.А. Курс лекций по базам данных / Е.А. Блинова
2. Microsoft OLE DB [Электронный ресурс]/ msdn.microsoft.com –https://msdn.microsoft.com/en-us/library/ms722784(v=vs.85).aspx
3. ProfessorWeb .NET & Web Programming [Электронный ресурс] / <https://professorweb.ru>
4. Deal.by [Электронный ресурс]/ https://minsk.deal.by/
5. Onliner.by [Электронный ресурс]/ https://www.onliner.by/

# Приложение А

## Листинг кода класса PasswordHash



# Приложение Б

## Пример хэширования с использованием алгоритма SHA256

