МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Направление специальности 1-40 01 01 10 Программное обеспечение информационных технологий (программирование интернет приложений)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТ:**

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

Тема Программное средство «Площадка объявлений»

Исполнитель

студент (ка) 2 курса группы 5 Трубач Дмитрий Сергеевич

(Ф.И.О.)

Руководитель работы преп.-стажер Север А.С.

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

Курсовой проект защищен с оценкой

Председатель Пацей Н.В

(подпись)

Минск 2023

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

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023г.

**ЗАДАНИЕ**

**к курсовому проектированию**

**по дисциплине** "Объектно-ориентированные технологии программирования и стандарты проектирования"

|  |  |
| --- | --- |
| Специальность: 1-40 01 01Программное обеспечение информационных технологий  Студент: Трубач Дмитрий Сергеевич | Группа:\_\_5\_\_ |
| **Тема: Программное средство «Площадка для размещения объявлений»** | |

**1. Срок сдачи студентом законченной работы**: "20 мая 2023 г."

**2. Исходные данные к проекту:**

**2.1**. Функционально ПС поддерживает:

* Функции администратора:
  + Поддерживать работу c базой данных;
  + Просмотр данных всех пользователей;
  + Изменение привилегий пользователей;
  + Одобрение объявлений от пользователей.
* Функции клиента:
  + Выполнять регистрацию и авторизацию;
  + Редактирование профиля;
  + Просмотр информации;
  + Загружать файлы/изображения;
  + Выполнять поисковые запросы.

**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. Использовать архитектурные и шаблоны проектирования.

**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)
* Руководство пользователя
* 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 в приложении.
  + Пояснительную записку, листинги, проект (инсталляцию проекта) необходимо загрузить диск, указанный преподавателем.

***Календарный план***

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Введение | 19.02.2023 |  |
| 2 | Аналитический обзор литературы по теме проекта. Изучение требований, определение вариантов использования | 12.03.2023 |  |
| 3 | Анализ и проектирование архитектуры приложения (построение диаграмм, проектирование бизнес-слоя, представления и данных) | 26.03.2023 |  |
| 4 | Проектирование структуры базы данных. Разработка дизайна пользовательского интерфейса | 02.04.2023 |  |
| 5 | Кодирование программного средства | 23.04.2023 |  |
| 6 | Тестирования и отладка программного средства | 30.04.2023 |  |
| 7 | Оформление пояснительной записки | 07.05.2023 |  |
| 8 | Защита проекта | 20.05.2023 |  |

**5. Дата выдачи задания** 12.02.2023

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Север А.С.*

(подпись)

Задание принял к исполнению 12.02.2023\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Трубач Д.С.*

(дата и подпись студента)

Содержание

[ВВЕДЕНИЕ 6](#_Toc135098256)

[1 Аналитический обзор литературы и формирование требований 7](#_Toc135098257)

[1.1 Анализ прототипов 7](#_Toc135098258)

[1.1.1 Onliner 7](#_Toc135098259)

[1.1.2 Kufar 8](#_Toc135098260)

[1.1.3 Avito 8](#_Toc135098261)

[1.1.4 Барахолка 9](#_Toc135098262)

[1.2 Требования к проекту 10](#_Toc135098263)

[2 Анализ требований к программному средству и разработка функциональных требований 11](#_Toc135098264)

[2.1 Описание средств разработки 11](#_Toc135098265)

[2.2 Технологии и инструменты разработки 11](#_Toc135098266)

[2.3 Спецификация функциональных требований к программному средству 13](#_Toc135098267)

[2.4 Спецификация функциональных требований 13](#_Toc135098268)

[3 Проектирование программного средства 15](#_Toc135098269)

[3.1 Общая структура 15](#_Toc135098270)

[3.2 Разработка общей схемы приложения 21](#_Toc135098271)

[3.3 Взаимоотношение между классами 25](#_Toc135098272)

[3.4 Модель базы данных 25](#_Toc135098273)

[3.5 Проектирование архитектуры приложения 28](#_Toc135098274)

[3.6 Проектирование последовательностей проекта 28](#_Toc135098275)

[4 Реализация программного средства 29](#_Toc135098276)

[4.1 Основные классы программного средства 29](#_Toc135098277)

[4.2 Выполнение входа 29](#_Toc135098278)

[4.3 Регистрация 29](#_Toc135098279)

[4.4 Просмотр объявлений 30](#_Toc135098280)

[4.5 Заполнение формы размещения объявления 31](#_Toc135098281)

[4.6 Просмотр собственных объявлений 31](#_Toc135098282)

[4.7 Функционирование панели администратора 32](#_Toc135098283)

[5 Тестирование, проверка работоспособности и анализ полученных результатов 33](#_Toc135098284)

[5.1 Тестирование авторизации и регистрации 33](#_Toc135098285)

[5.2 Тестирование заполнения формы объявления 36](#_Toc135098286)

[6 Руководство по использованию 40](#_Toc135098287)

[ЗАКЛЮЧЕНИЕ 45](#_Toc135098288)

[Список использованных источников 46](#_Toc135098289)

[ПРИЛОЖЕНИЕ А 47](#_Toc135098290)

[ПРИЛОЖЕНИЕ Б 48](#_Toc135098291)

[ПРИЛОЖЕНИЕ В 49](#_Toc135098292)

[ПРИЛОЖЕНИЕ Г 50](#_Toc135098293)

[ПРИЛОЖЕНИЕ Д 52](#_Toc135098294)

# ВВЕДЕНИЕ

Торговля постепенно все больше и больше перемещается из оффлайн в онлайн-пространство, и это открывает практически безграничные перспективы как для компаний, так и для обычных людей.

Многие от случая к случаю пытаются продать что-то – ненужные вещи, которые просто стоят без дела, или результаты собственного творчества.

Удобнее всего это делать через специальные приложения-маркетплейсы, которые помогают продавцам найти покупателей, а покупателям – продавцов. С помощью разрабатываемого программное средство можно продать практически все, что угодно.

Программное средство «Площадка объявлений» – это программное средство, где каждый желающий может вывесить своё объявление, а все пользователи приложения — прочитать его.

Также это приложение может вести сортировку предложений, и в зависимости от выбора критериев, предлагать подходящие варианты.

Приложение имеет 2 типа пользователей: администратор и пользователь. Они имеют разные полномочия. Пользователи могут создавать объявления, следить за списком предложений, выполнять сортировку по названию, цене, продавцу, региону. Администраторы имеют полные права. Они могут удалять любые объявления, а также пользователей.

Также будет реализована авторизация всех пользователей. Это значит, что перед тем, как предложить свой товар, нужно будет зарегистрироваться. Хранение всех пользователей будет осуществлено в базе данных, и при авторизации или регистрации будет непосредственное обращение к ней. Также вся информация о товаре будет хранится в базе данных.

Реализована возможность входа/регистрация в программу через социальную сеть «Вконтакте».

В ходе использования этой программы, пользователи смогут получать информацию о интересующих их товарах и продавцах (характеристика товара, номер телефона продавца).

# Аналитический обзор литературы и формирование требований

## Анализ прототипов

Были проанализированы цели и задачи, поставленные в данном курсовом проекте, а также рассмотрены аналогичные примеры их решений. На основании анализа всех достоинств и недостатков данных альтернативных решений были сформулированы требования к данному программному средству.

### Onliner

Самое популярное аналогичное решение — интернет-ресурс «Onliner» [1], крупнейшая в Беларуси площадка для объявлений, на которой пользователи могут размещать объявления о продаже, покупке или обмене товаров и услуг. Основными преимуществами данного ресурса является его удобный интерфейс, большой ассортимент товаров и услуг, а также возможность быстрого и удобного поиска нужного объявления. Пользователи могут выбрать нужный раздел, подраздел и фильтр по региону, цене, типу товара и другим параметрам для уточнения поиска, а также узнать о скидках и акциях на различные товары и услуги.

Однако, как и у многих других площадок для объявлений, есть некоторые недостатки. Например, на площадке может быть большое количество объявлений с недостаточной информацией о товаре или услуге, что может затруднить поиск и сравнение. Кроме того, некоторые продавцы могут предлагать товары с недостаточной информацией о состоянии или качестве, что может привести к недовольству покупателей.

Интерфейс интернет-ресурса представлен на рисунке 1.1.

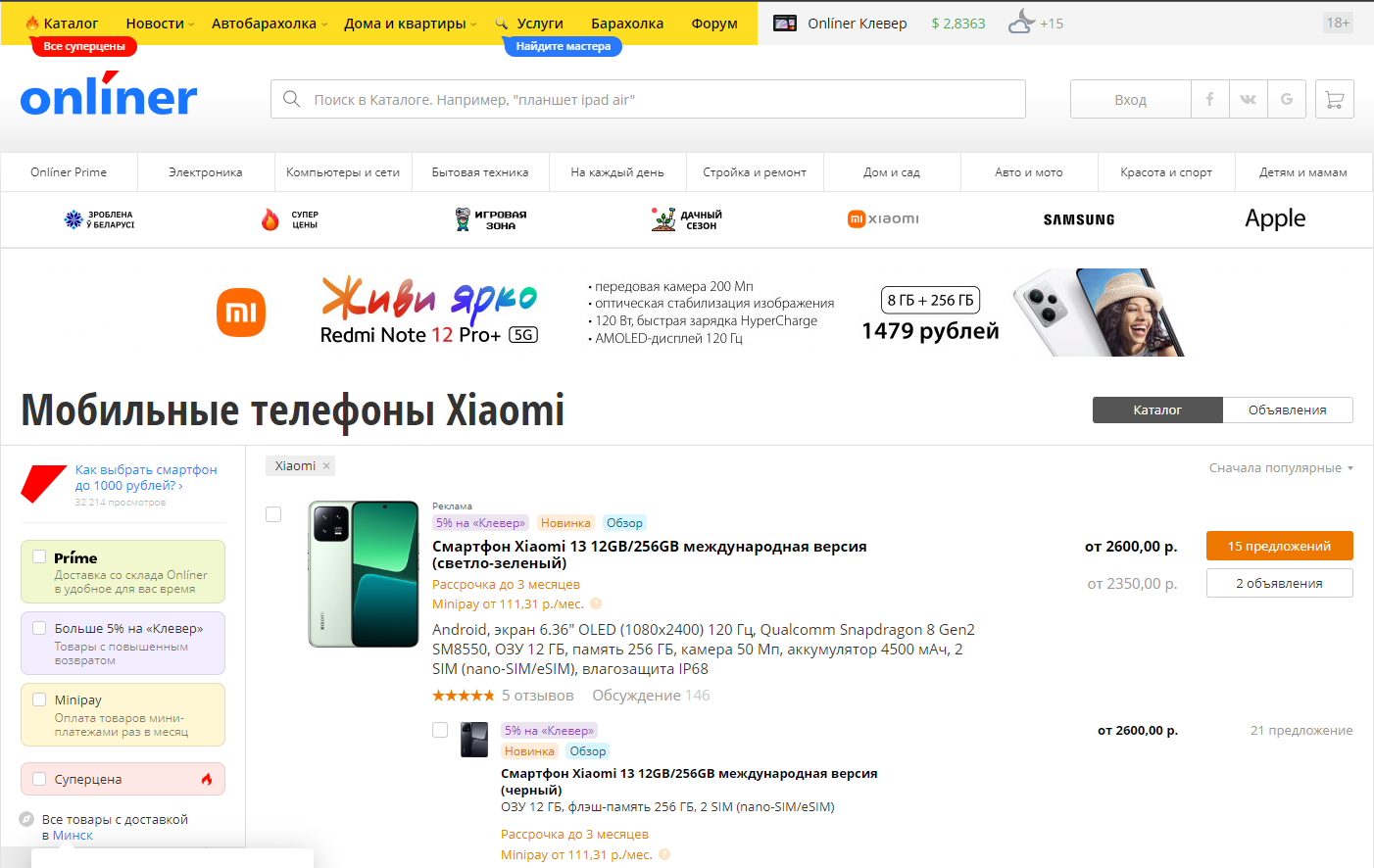


Рисунок 1.1 – Интернет-ресурс «Onliner»

В целом, интернет-ресурс «Onliner» представляет собой удобную и популярную площадку для размещения объявлений о продаже и покупке товаров и услуг в Беларуси.

### Kufar

Еще одним альтернативным решением, которое может использоваться для поиска и покупки товаров и услуг, является интернет-ресурс «Kufar» [2]. Он позволяет пользователям размещать объявления о продаже, покупке или обмене различных товаров и услуг. Кроме того, на сайте есть возможность поиска по ключевым словам, фильтрации результатов по региону, цене, типу товара и другим параметрам.

Одним из главных преимуществ Kufar является наличие большого количества объявлений о различных товарах и услугах, что позволяет быстро найти нужный товар или услугу. Также на сайте представлены различные категории товаров, что упрощает навигацию и поиск. Кроме того, пользователи могут получить информацию о продавцах, задать им вопросы и оставить отзывы.

Однако, как и у других площадок для объявлений, есть некоторые недостатки. Например, на сайте могут быть объявления с недостаточной информацией о товаре или услуге, что может затруднить поиск и сравнение. Также есть риск столкнуться с мошенниками, поэтому перед совершением сделки необходимо внимательно изучить информацию о продавце и товаре.

Интерфейс интернет-ресурса представлен на рисунке 1.2.

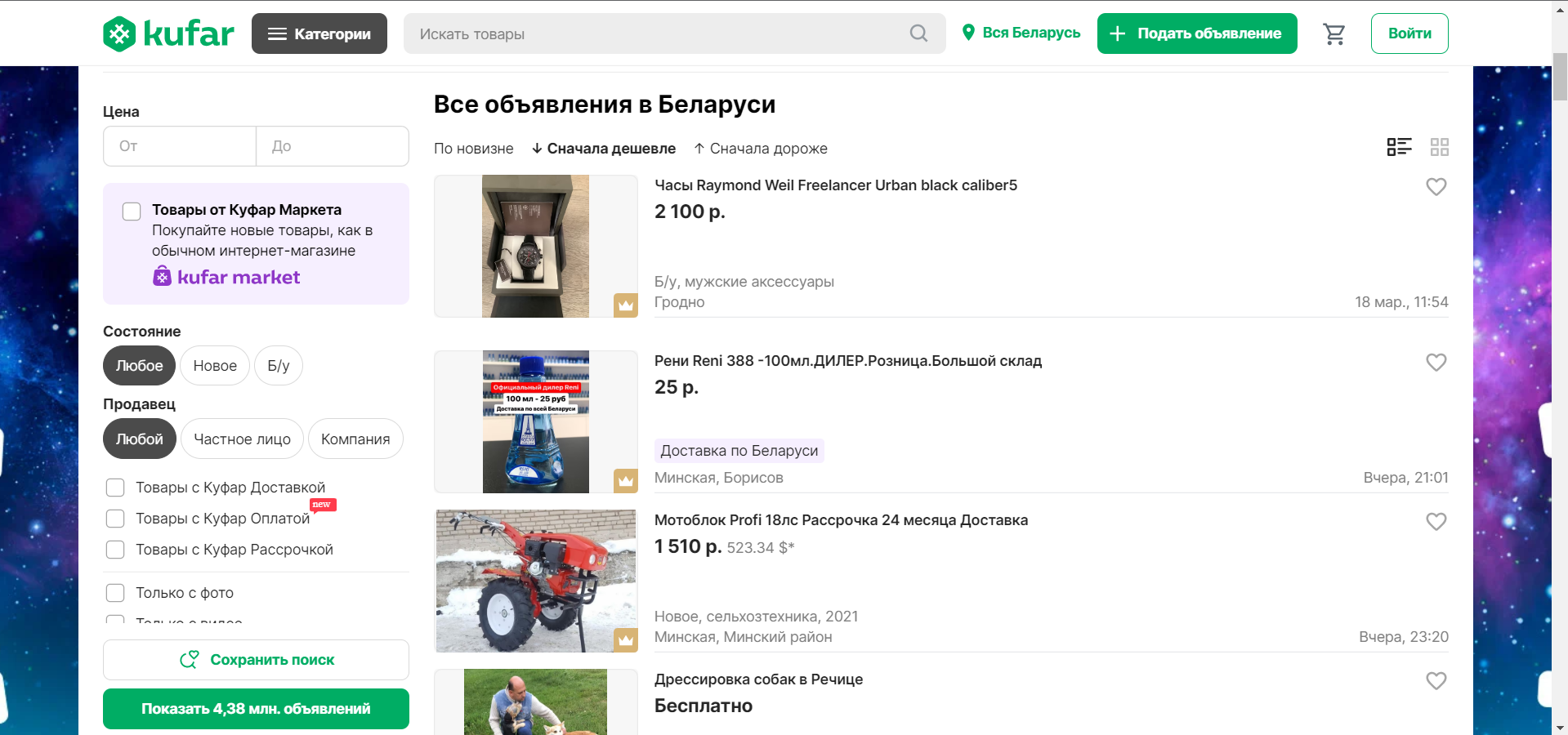


Рисунок 1.2 – Интернет-ресурс «Kufar»

Kufar является удобным и популярным ресурсом для поиска и покупки товаров и услуг на ряду с Onliner, но пользователи должны быть осторожными и бдительными при совершении сделок.

### Avito

Другой интернет-ресурс, который также предлагает услуги по покупке, продаже и обмену товаров и услуг, - это «Avito» [3]. С помощью этой платформы пользователи могут размещать объявления о продаже б/у и новых товаров, а также услуг, предлагаемых различными компаниями и физическими лицами. На сайте можно найти множество категорий товаров, что упрощает поиск и выбор необходимого товара. Кроме того, пользователи могут оставлять отзывы и задавать вопросы продавцам, что позволяет получить дополнительную информацию о товаре и продавце.

Однако, как и у любой площадки для объявлений, есть определенные риски. Например, мошенники могут размещать ложные объявления, чтобы обмануть покупателей и продавцов. Кроме того, может быть сложно выбрать лучшее предложение, так как на сайте может быть много одинаковых или похожих товаров.

Интерфейс интернет-ресурса представлен на рисунке 1.3.

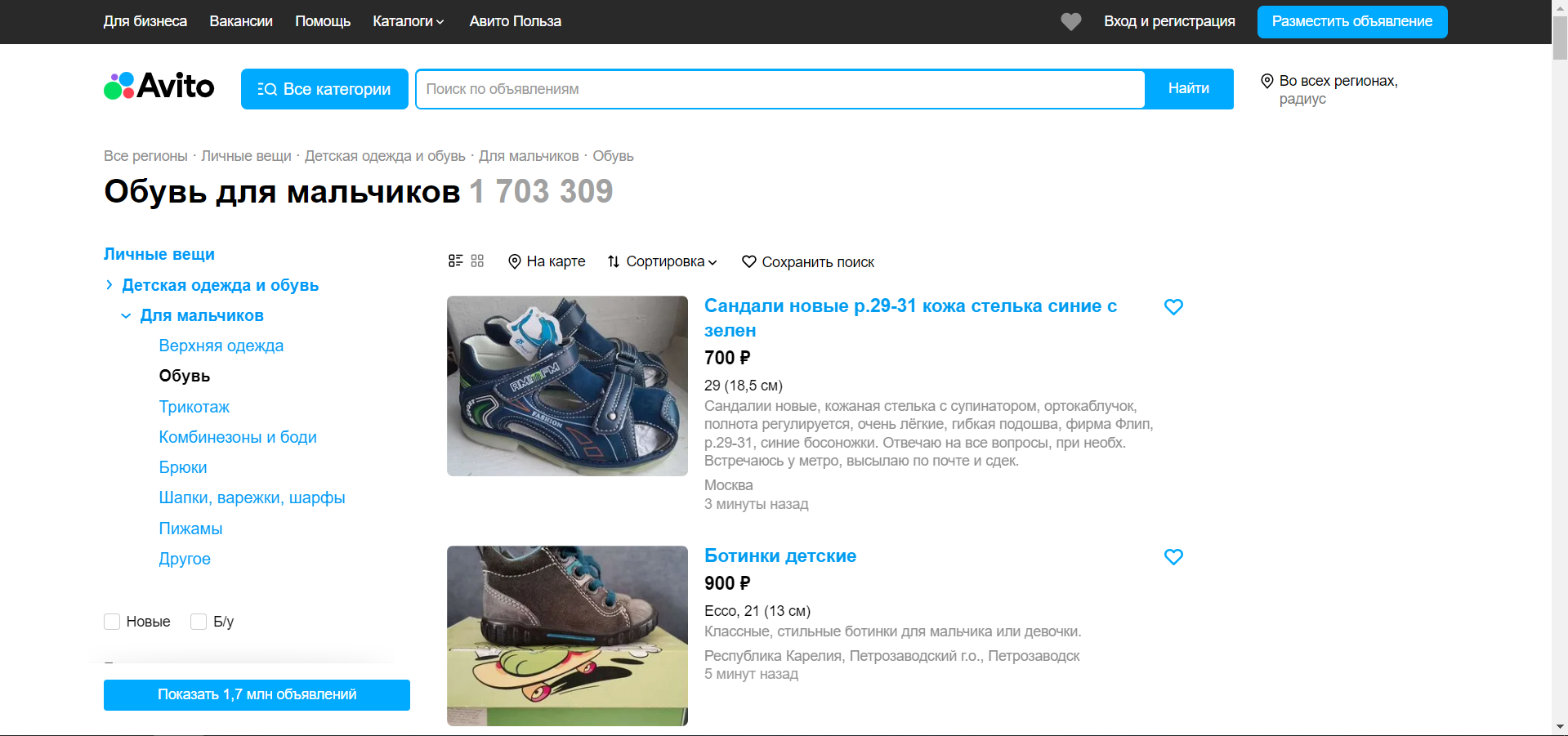


Рисунок 1.3 – Интернет-ресурс «Avito»

По итогу, «Avito» является полезным и удобным ресурсом для покупки и продажи товаров и услуг, но перед совершением сделки нужно быть осторожным и проверять информацию о товаре и продавце.

### Барахолка

Еще одной альтернативой решения поставленных задач может быть посещение барахолки. Основным преимуществом данного выбора является возможность личного ознакомления с товаром, его осмотра и проверки перед покупкой.

Среди недостатков данной альтернативы можно выделить ограниченный выбор товаров и услуг, возможность попадания на подделку или товар с низким качеством, а также неудобство в поиске нужного товара, особенно если речь идет о редком и необычном предмете.

Несмотря на это, для любителей ретро и винтажных вещей, барахолки могут стать прекрасным местом для нахождения уникальных предметов и оригинальных подарков.

## Требования к проекту

Обзор вышеперечисленных известных аналогов позволяет проанализировать все преимущества и недостатки альтернативных возможностей и позволяет сформулировать список требований, предъявляемых к программному средству, разрабатываемому в данном курсовом проекте. Программное средство должно обеспечивать возможность выполнения перечисленных ниже функций:

* управление администратором базой данных;
* возможность пользователю зарегистрироваться или войти в существующую учетную запись;
* возможность авторизированным пользователям просмотра информации о фильмах, сеансах и заказа билетов;
* возможность просмотра личной страницы пользователя с отображением всех объявлений;
* уведомление пользователя о принятии объявления.

Таким образом, можно сказать, что требования к разрабатываемому программному средству основаны на анализе прототипов и аналогов, и представляют собой список функций, необходимых для управления базой данных и обеспечения комфортного пользовательского опыта.

Кроме того, важным критерием является безопасность пользователей при совершении сделок, поэтому необходимо уделять внимание проверке информации о товарах и продавцах. Разрабатываемое программное средство должно быть удобным и простым в использовании, а также иметь функциональность для уведомления пользователя о принятии объявления.

# Анализ требований к программному средству и разработка функциональных требований

Для начала этапа разработки необходимо сформулировать функциональные требования к программному средству.

## Описание средств разработки

При разработке приложения были использованы:

* интегрированная среда разработки Microsoft Visual Studio 2022;
* программная платформа .NET 6.0;
* язык программирования C#;
* технология WPF;
* расширяемый язык разметки XAML;
* технология Entity Framework Core;
* технология VkNet API;
* MS SQL Server.

Данный набор средств разработки и технологий был выбран с учетом их широкой функциональности, эффективности и поддержки, чтобы обеспечить разработку качественного и масштабируемого программного средства. Использование интегрированной среды разработки Microsoft Visual Studio 2022, программной платформы .NET 6.0, языка программирования C# и других технологий позволило нам эффективно реализовать требования и достичь желаемого функционального результата.

## Технологии и инструменты разработки

Microsoft Visual Studio 2022 - это программный продукт, который объединяет инструменты для написания, тестирования и сборки кода, а также для публикации разработанных приложений. Он позволяет создавать различные типы приложений, включая консольные и десктопные, с использованием технологий, таких как WinForms или WPF [4].

Платформа .NET Framework - это технология, разработанная компанией Microsoft в 2002 году, которая обеспечивает возможность создания и выполнения веб-служб и приложений Windows. Она предоставляет среду программирования для локального сохранения и выполнения объектного кода, а также его выполнения в Интернете или удаленно. Основой платформы является общеязыковая среда Common Language Runtime, которая может выполнять код на разных языках программирования [5].

Для разработки в платформе .NET применяется язык программирования C#, который является основным в данной технологии. Это объектно-ориентированный язык, который характеризуется строгой статической типизацией, поддержкой перегрузки операторов, указателей на функции-члены классов, атрибутов, событий и свойств, а также исключений. Кроме того, C# используется в качестве основного языка разработки приложений на технологии WPF [6].

Microsoft WPF - это технология, используемая для создания клиентских приложений Windows с интерактивным пользовательским интерфейсом и разделением дизайна и бизнес-логики. Она является альтернативой WinForms и включена в состав .NET Framework. WPF использует язык разметки XAML для описания элементов пользовательского интерфейса и обеспечивает более высокую гибкость и масштабируемость в сравнении с WinForms [7].

WPF позволяет разрабатывать визуальный интерфейс, используя язык разметки XAML (eXtensible Application Markup Language). Он предоставляет различные элементы управления, возможности привязки данных, макеты, 2D и 3D графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление. XAML - это декларативный язык описания интерфейса на основе XML, который позволяет отделить дизайн интерфейса от его реализации и бизнес-логики [8].

Entity Framework Core является расширяемой технологией, предоставляющей высокий уровень абстракции для взаимодействия с базами данных в приложении WPF. В отличие от традиционных средств ADO.NET, которые требуют создания подключений и объектов для взаимодействия с базами данных, Entity Framework Core позволяет абстрагироваться от деталей реализации базы данных и работать с данными на уровне объектов. Entity Framework Core также является кроссплатформенной и поддерживает .NET Core.

Существуют три возможных способа использования Entity Framework Core в приложении WPF: Database First, Model First и Code First.

При использовании Database First, Entity Framework Core создает классы, отражающие модель базы данных.

При использовании Model First, сначала создается модель базы данных, а затем на ее основе создается реальная база данных на сервере.

При использовании Code First, разработчик создает классы модели данных, которые будут храниться в базе данных, а затем Entity Framework Core генерирует базу данных и ее таблицы на основе этих классов [9].

VkNet API - это библиотека для работы с социальной сетью ВКонтакте (VK), разработанная на языке C# и предназначенная для использования в приложениях на платформе .NET. Она предоставляет удобный и простой интерфейс для доступа к функционалу VK через её API, такой как доступ к данным пользователей, групп, аудиозаписей, видео, фото, сообщениям и другим объектам.

VkNet API позволяет получать и отправлять данные через VK API, используя OAuth авторизацию и работу с токенами. Также в библиотеке присутствуют средства работы с асинхронными методами, событиями и обработкой ошибок [10].

MS SQL Server использует реляционную модель баз данных, где данные хранятся в таблицах, состоящих из строк и столбцов. Каждая строка содержит отдельный объект, а атрибуты объектов хранятся в столбцах. Для взаимодействия с базой данных используется язык SQL (Structured Query Language). Клиент отправляет запрос на языке SQL, который база данных интерпретирует и выполняет, а затем возвращает результат выполнения запроса клиенту. Основной используемый язык запросов в MS SQL Server - Transact-SQL, который основан на SQL с дополнительными расширениями [11].

## Спецификация функциональных требований к программному средству

Программное средство должно предоставлять следующие функциональные возможности:

Для пользователя:

* регистрация;
* авторизация;
* вход через VKontakte;
* изменение данных своего профиля;
* размещение объявлений разного типа;
* просмотр доступных объявлений;
* удаление собственных объявлений;
* выборка объявлений по различным критериям;
* управление своими объявлениями (редактирование, удаление).

Для администратора:

* авторизация;
* изменение данных своего профиля;
* просмотр информации о пользователях;
* изменение привелегий у пользователей;
* проверка поданных объявлений;
* размещение объявлений разного типа;
* просмотр доступных объявлений;
* удаление объявлений;
* выборка объявлений по различным критериям;
* управление всеми объявлениями (редактирование, удаление);
* отправление электронного письма пользователю при одобрении объявления.

Реализация этих требований позволит создать удобное и функциональное программное средство, которое будет полезным для пользователей и обеспечит эффективное управление объявлениями.

По итогу работы над продуктом, его необходимо протестировать и отладить, проанализировать его возможности и, по необходимости, внести дополнения или улучшения проекта, с возможностью дальнейшего расширения проекта.

## Спецификация функциональных требований

Для функциональности ПС необходимо создание базы данных для хранения информации приложения. Подробно база данных описано в следующем разделе.

В программном средстве необходимо реализовать регистрацию и авторизацию пользователей для доступа ко всем возможностям приложения. Для авторизации входными параметрами являются логин и пароль пользователя, которые содержатся в базе данных. Для регистрации входными данными являются имя, фамилия, e-mail и пароль. Введенные данные, успешно прошедшие валидацию, заносятся в базу данных.

Пользователь может просматривать объявления, создавать объявления, производить поиск и фильтрацию. Для каждого пользователя создается его профиль, в котором он может просмотреть информацию о своем профиле. Все сведения пользователе должны автоматически обновляться в базе данных.

Администратор обладает теми же возможностями, что и пользователь, а также обладает возможностями одобрения, просмотра, редактирования и удаления информации о объявлениях всех пользователей.

Описание функциональности программного средства представлено на UML-схеме, изображённой на рисунке 2.1.

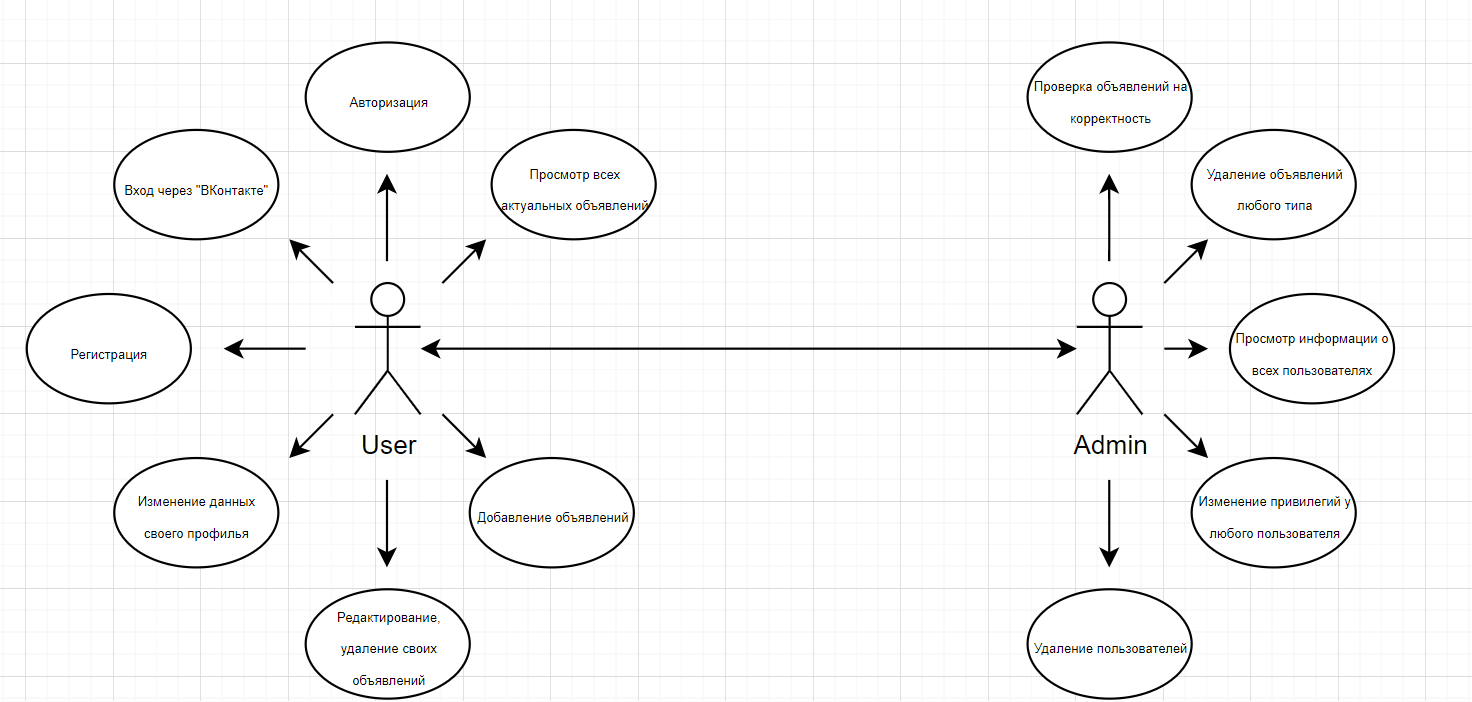


Рисунок 2.1 – UML-диаграмма возможностей

Таким образом, программа должна обеспечивать удобный интерфейс для пользователя и администратора, а также функционал для работы с базой данных, VKontakte API и другими необходимыми технологиями.

# Проектирование программного средства

## Общая структура

Программное средство «Площадка объявлений» имеет следующею структуру, представленную на рисунке 3.1.

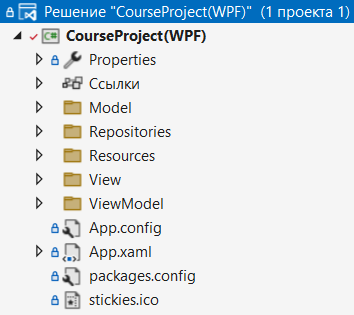


Рисунок 3.1 – Структура проекта

Описание структуры основных папок и файлов проекта представлено в таблице 3.1.

Таблица 3.1 – Описание структуры папок и файлов проекта

|  |  |
| --- | --- |
| Имя пакета | Содержание |
| Папка Model | Здесь описаны модели, с которыми происходит вся работа в приложении:  Пользователь  Актуальное объявление  Непроверенное объявление  Регион |
| Папка Repositories | Содержит классы и интерфейсы для работы с данными БД. Они являются связующим элементом между данными в БД и теми данными, с которыми работает приложение |
| Папка Resources | Файлы стилей |

Продолжение таблицы 3.1

|  |  |
| --- | --- |
| Имя пакета | Содержание |
| Папка View | Содержит все представления, которые позволяют пользователю работать с приложением. Они описывают графическую составляющую приложения. |
| Папка ViewModel | Содержит логику, которая позволяет получить данные при помощи View, обработать их, использую при этом Model, после чего передать в базу данных. |
| App.config | Файл конфигурации приложения |
| App.xaml | Файл, который определяет ресурсы, которые являются общие для приложения |
| packages.config | Файл для для управления зависимостями и установки сторонних библиотек |
| stickies.ico | Логотип приложения для отображения в панели задач |

Таким образом, сформированная таблица помогает понять общую структуру проекта проектируемого программного средства.

Более подробная структура содержимого папок «Model» и Repositories программного средства показана на рисунке 3.2.

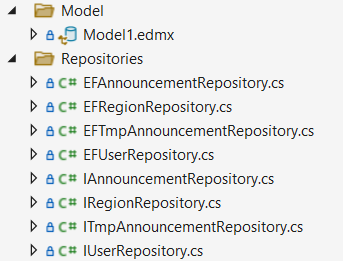


Рисунок 3.2 — Подробная структура папок «Model» и «Repositories»

Описание файлов папок представлено в таблице 3.2.

Таблица 3.2 – Описание файлов папок «Model» и «Repositories»

|  |  |
| --- | --- |
| Имя файла | Содержание |
| Model1.edmx | Файл модели Entity Framework, который содержит информацию об объектно-ориентированной модели базы данных, включая сущности, атрибуты и отношения между ними |
| EFAnnouncementRepository.cs | Файл репозитория для доступа к сущностям Announcement (объявления) в базе данных. Он содержит методы для создания, чтения, обновления и удаления объектов Announcement. |
| EFRegionRepository.cs | Файл репозитория для доступа к сущностям Region (регионы) в базе данных. Он содержит методы для создания, чтения, обновления и удаления объектов Region. |
| EFTmpAnnouncementRepository.cs | Файл репозитория для доступа к временным сущностям TmpAnnouncement в базе данных. Он содержит методы для создания, чтения, обновления и удаления объектов TmpAnnouncement. |
| EFUserRepository.cs | Файл репозитория для доступа к сущностям User (пользователи) в базе данных. Он содержит методы для создания, чтения, обновления и удаления объектов User |
| IAnnouncementRepository.cs | Интерфейс для репозитория доступа к сущностям Announcement в базе данных. Он содержит объявления методов для создания, чтения, обновления и удаления объектов Announcement. |
| IRegionRepository.cs | Интерфейс для репозитория доступа к сущностям Region в базе данных. Он содержит объявления методов для создания, чтения, обновления и удаления объектов Region. |
| ITmpAnnouncementRepository.cs | Интерфейс для репозитория доступа к временным сущностям TmpAnnouncement в базе данных. Он содержит объявления методов для создания, чтения, обновления и удаления объектов TmpAnnouncement. |

Продолжение таблицы 3.2

|  |  |
| --- | --- |
| Имя файла | Содержание |
| IUserRepository.cs | Интерфейс для репозитория доступа к сущностям User в базе данных. Он содержит объявления методов для создания, чтения, обновления и удаления объектов User. |

Рассмотрим папки «Resources» и «View». Их подробная структура показана на рисунке 3.3.

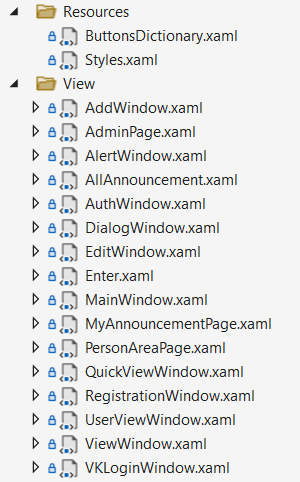


Рисунок 3.3 — Подробная структура папок «Resources» и «View»

Описание файлов папок представлено в таблице 3.3.

Таблица 3.3 – Описание файлов папок «Model» и «Repositories»

|  |  |
| --- | --- |
| Имя файла | Содержание |
| ButtonsDictionary.xaml | Файлы стилей кнопок. |
| Styles.xaml | Стили для других элементов управления. |

Продолжение таблицы 3.3

|  |  |
| --- | --- |
| Имя файла | Содержание |
| AddWindow.xaml | Окно добавления объявления, в котором указывается информация о будущем объявлении. |
| AdminPage.xaml | Окно для управления работы приложения и ведения проверки данных. |
| AlertWindow.xaml | Окно уведомления. |
| AllAnnouncement.xaml | Страница, отображающая все актуальные объявления. |
| AuthWindow.xaml | Окно авторизации. Требует ввода логина и пароля или выбор способа авторизации. |
| DialogWindow.xaml | Окно, в котором нужно выбрать вариант ответа «Да» или «Нет» на заданный там вопрос. |
| EditWindow.xaml | Окно изменения свойств объявления. |
| Enter.xaml | Окно ввода кода подтверждения . |
| MainWindow.xaml | Главное окно приложения. |
| MyAnnouncementPage.xaml | Страница, на которой показаны все ваши объявления и выбор действий над ними. |
| PersonAreaPage.xaml | Страница профиля, в которой можно посмотреть или изменить свои личные данные. |
| QuickViewWindow.xaml | Окно для закрепления объявления или его быстрого просмотра. |
| RegistrationWindow.xaml | Окно регистрации нового пользователя. |
| UserViewWindow.xaml | Окно просмотра информации о пользователе (доступно для модератора/админа). |
| ViewWindow.xaml | Основное окно для просмотра информации о объявлениях. |
| VKLoginWindow.xaml | Окно авторизации через ВКонтакте, в котором требуется ввести логин и пароль от аккаунта ВКонтакте. |

Рассмотрим папку «ViewModel». Её подробная структура показана на рисунке 3.4.

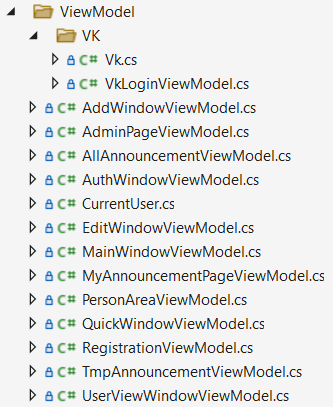


Рисунок 3.4 — Подробная структура папки «ViewModel»

Описание файлов папок представлено в таблице 3.4.

Таблица 3.4 – Описание файлов папки «ViewModel»

|  |  |
| --- | --- |
| Имя файла | Содержание |
| Vk.cs | Класс для работы с VkNet API. |
| VkLoginViewModel.cs | Класс ViewModel для страницы авторизации через VK. |
| AddWindowViewModel.cs | Класс ViewModel для окна добавления объявления. |
| AdminPageViewModel.cs | Класс ViewModel для страницы администратора. |
| AllAnnouncementViewModel.cs | Класс ViewModel для страницы со всеми объявлениями. |
| AuthWindowViewModel.cs | Класс ViewModel для окна авторизации. |
| CurrentUser.cs | Класс, представляющий текущего пользователя. |
| EditWindowViewModel.cs | Класс ViewModel для окна редактирования объявления. |
| MainWindowViewModel.cs | Класс ViewModel для главного окна приложения. |
| MyAnnouncementPageViewModel.cs | Класс ViewModel для страницы с объявлениями текущего пользователя. |

Продолжение таблицы 3.4

|  |  |
| --- | --- |
| Имя файла | Содержимое |
| PersonAreaViewModel.cs | Класс ViewModel для личного кабинета пользователя. |
| QuickWindowViewModel.cs | Класс ViewModel для закрепления объявления или его быстрого просмотра. |
| RegistrationViewModel.cs | Класс ViewModel для страницы регистрации. |
| TmpAnnouncementViewModel.cs | Класс ViewModel для неподтверждённого объявления. |
| UserViewWindowViewModel.cs | Класс ViewModel для окна просмотра объявления пользователя. |

В целом, описание структуры проекта позволяет лучше понимать, как устроено программное средство и какие компоненты в нем присутствуют.

## Разработка общей схемы приложения

Рассмотрим общую схему программного средства «Площадка объявлений». На рисунке 3.5 представлена общая схема работы приложения.

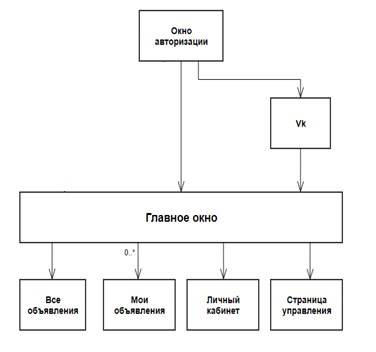


Рисунок 3.5 — Общая схема приложения

Из нее видно, что при запуске приложения у нас появляется окно авторизации, которое требует ввода логина и пароля для начала работы.

Если у нас нет учетной записи в системе, мы можем создать ее, нажав на кнопку "Регистрация" в окне авторизации. При регистрации нам нужно заполнить все поля, после чего вернуться в окно авторизации и ввести данные, указанные при регистрации. Мы также можем войти в систему через социальную сеть "ВКонтакте", нажав соответствующую кнопку в окне авторизации и введя данные для входа в учетную запись через социальную сеть.

После ввода логина и пароля в соответствующих полях, проверяется правильность ввода данных.

В случае неверного ввода данных, выведется соответствующее сообщение. В ином случае окно авторизации закроется и откроется главное окно приложения.

В главном окне работа идёт в страничном и оконных режимах.

Рассмотрим каждую страницу в отдельности.

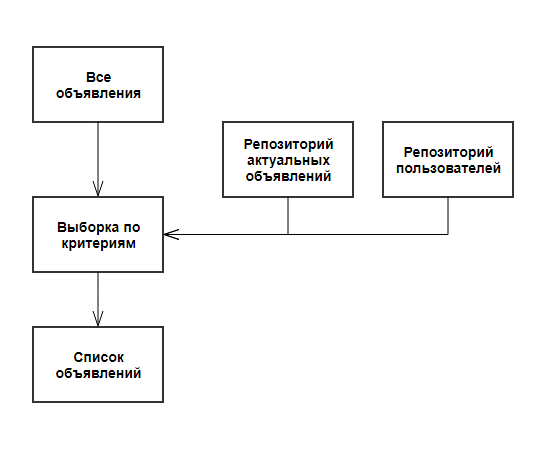


Рисунок 3.6 — Страница «Все объявления»

Эта страница содержит список всех актуальных объявлений.

Также она содержит механизм поиска объявления по определённым критериям.

Осуществить поиск можно по таким критериям:

* слово в названии;
* продавец;
* цена;
* регион;
* категорий;
* комбинации вышеперечисленных.

На основе критериев поиска, формируется результирующий набор. При нажатии мы видим более подробную информацию о объявлении и получаем возможность его закрепления, чтобы была возможность сравнить несколько объявлений между собой.

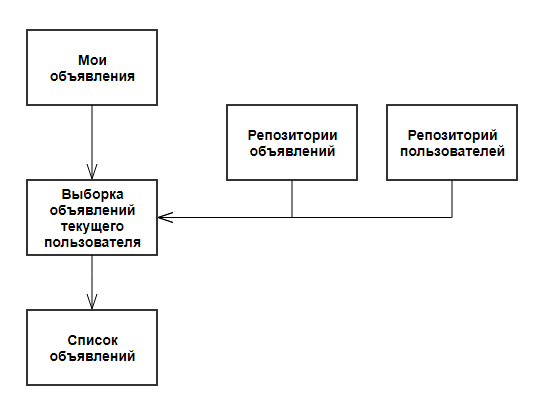


Рисунок 3.7 — Страница «Мои объявления»

На рисунке 3.7 показана схема работы страницы «Мои объявления».

На этой странице ведётся редактирование, удаление, просмотр только тех объявлений, которые размещал вошедший пользователь.

Есть переключатель, который позволяет переключатся между объявлениями, которые ещё не прошли проверку и теми, которые уже находятся в разделе «Актуальные».

Редактирование происходит в отдельном окне, которое появляется при нажатии на соответствующую кнопку. В этом окне есть ограничения и коррекция недопустимых значений. На название отводится от 5 до 50 символов, а на описание от 10 до 1000 символов. Также нельзя установить отрицательную цену т.к. она, при некорректно введённых значениях, будет обращаться в нуль или в то значение, которое было изначально.

При просмотре своего объявления, открывается окно, в котором выводится вся информация о данном объявлении.

Если происходит изменение объявления из раздела «Актуальные», то после изменения, оно автоматически попадает в раздел «Непроверенные» и становится невидимым для других пользователей.

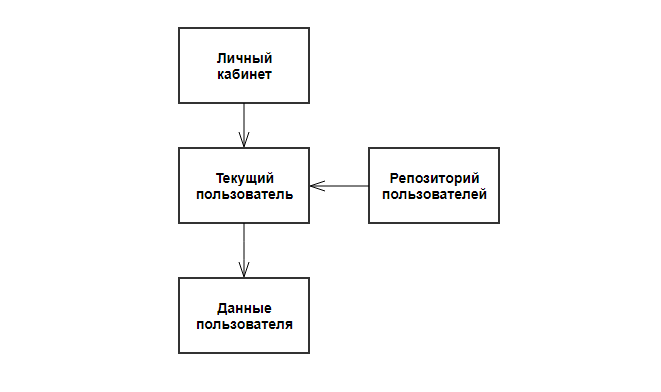


Рисунок 3.8 — Страница «Личный кабинет»

На рисунке 3.8 показана схема страницы «Личный кабинет».

На этой странице можно редактировать свои данные, такие как:

* имя;
* фамилия;
* mail;
* телефон;
* описание (о себе).
* картинка профиля.

Под картинкой профиля находится кнопка «Загрузить», нажатие на которую приводит к открытию окна с выбором файла картинки профиля, которую вы хотите установить в качестве основой.

На этой странице есть 2 кнопки: «Сохранить изменения», «Удалить свой профиль».

При нажатии на 1-ю кнопку происходит обновление информации о текущем пользователе в базе данных.

При на нажатии на 2-ю кнопку – удаление всех объявлений текущего пользователя, последующее удаление самого профиля и перенаправление на окно авторизации.

Также на эту страницу можно попасть, нажав на элемент, находящийся в верху самого окна чуть правее середины, и содержащий имя и фамилию текущего пользователя, а также инициалы.

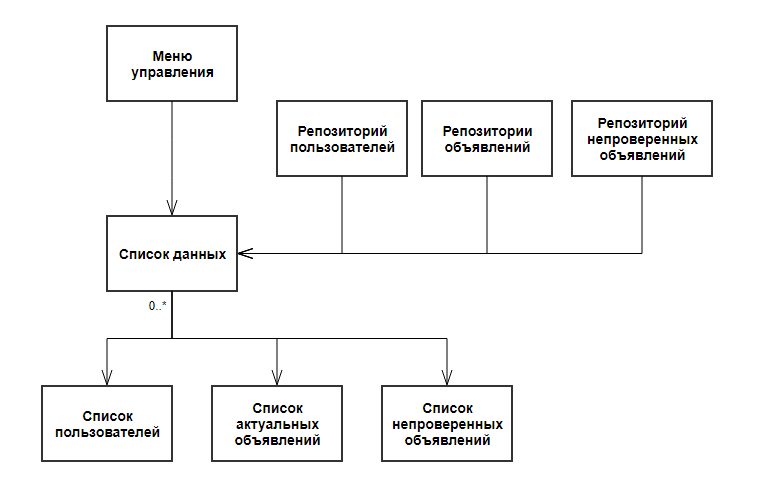


Рисунок 3.9 — Страница «Панель управления»

В этом окне у нас производится администрирование нашего приложения.

Всего есть 2 типа привилегий: пользователь и администратор.

Пользователь не имеет доступ к этой странице. При попытке зайти в это меню, у него будет выведено сообщение о недостатке прав для доступа к этой странице.

Администратор имеет возможность проверки всех объявлений, удаление объявлений как из разряда «Актуальные», так и из разряда «Непроверенные», просматривать информацию о всех пользователях, а также возможность назначать и изменять привилегии пользователей и удалять пользователей из приложения.

## Взаимоотношение между классами

Для визуализации взаимосвязей между классами используется диаграмма UML – графическое представление набора элементов, изображаемое чаще всего в виде связанного графа с вершинами (сущностями) и ребрами (отношениями).

Для представления внутренней структуры программы в виде классов и связей между ними используется диаграмма классов. Приложение спроектировано таким образом, что каждый класс выполняет свои функции и практически не зависит от других. Диаграмма классов представлена в [приложении А](#_ПРИЛОЖЕНИЕ_А).

## Модель базы данных

Для реализации поставленной задачи была создана база данных CW\_MarketPlace\_OOP. Для ее создания использовалась система управления реляционными базами данных MS SQL Server. База данных состоит из таблиц, представленных на рисунке 3.10. Скрипт для создания базы данных представлен в [приложении Г](#_ПРИЛОЖЕНИЕ_Г).

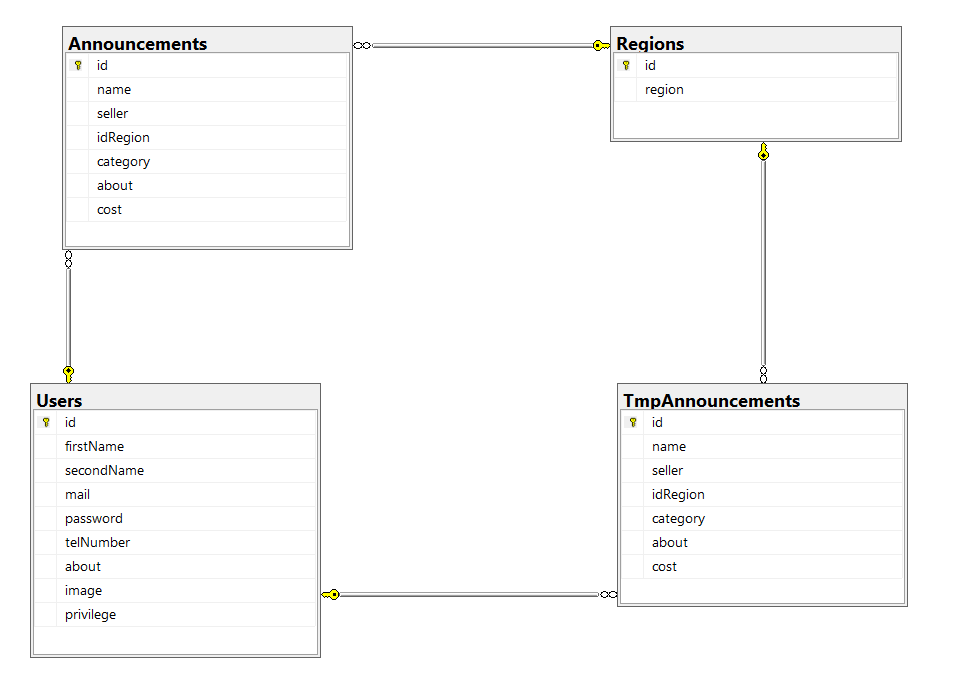


Рисунок 3.10 — База данных CW\_MarketPlace\_OOP

На рисунке 3.11 проиллюстрирована структура таблицы «Users», которая содержит информацию о пользователях. В данной таблице поле ID, представленное целочисленным значением, является первичным ключом. Поле mail хранит электронную почту пользователя, password – пароль, telNumber — номер телефона, firstName, secondName, и priviege – соответственно имя, фамилию и роль (admin для администратора, user для пользователя).

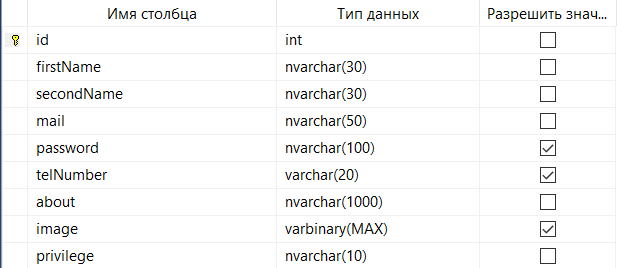


Рисунок 3.11 — Структура таблицы «Users»

На рисунке 3.12 изображена структура таблицы «Announcements», содержащая информацию о объявлениях, которые прошли проверку. Первичный ключ ID хранит уникальный идентификатор объявления, seller – идентификатор продавца, idRegion – с какой области страны объявление, category – категория объявления, about – вторичная информация о товаре, cost – стоимость товара.

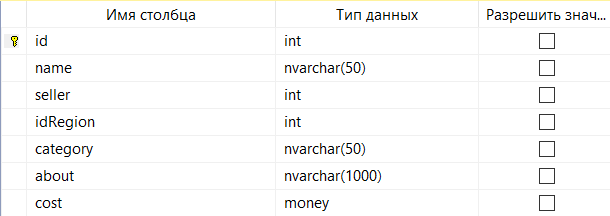


Рисунок 3.12 — Структура таблицы «Announcements»

На рисунке 3.13 изображена структура таблицы «TmpAnnouncements», содержащая информацию о объявлениях, которые находятся на проверке. Первичный ключ ID хранит уникальный идентификатор объявления, seller – идентификатор продавца, idRegion – с какой области страны объявление, category – категория объявления, about – вторичная информация о товаре, cost – стоимость товара.

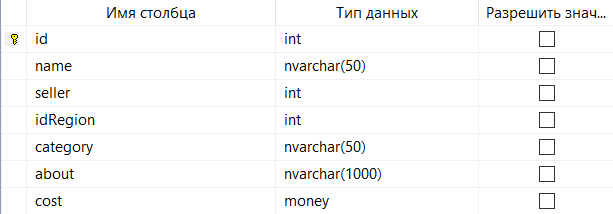


Рисунок 3.13 — Структура таблицы «TmpAnnouncements»

На рисунке 3.14 изображена структура таблицы «Regions», содержащая информацию о областях страны, которые находятся на проверке. Первичный ключ ID хранит уникальный идентификатор региона, region – его название.

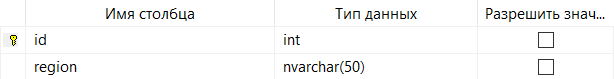


Рисунок 3.14 — Структура таблицы «Regions»

Создание базы данных позволит эффективно хранить и управлять информацией о пользователях, объявлениях и регионах, необходимой для функционирования интернет-магазина.

## Проектирование архитектуры приложения

Для общего представления функционального назначения системы используется диаграмма использования, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей. На диаграмме использования применяются два типа основных сущностей: варианты использования и группы пользователей.

Разные группы пользователей в диаграмме называются актёрами, и обозначают любые сущности, использующие систему. Любая функция системы называется вариантом использования. Каждый вариант использования обозначает набор действий, который может быть использован актёром для взаимодействия с системой, и определяет набор действий, выполняемых этой системой.

Диаграмма использования представлена в [приложении Б](#_ПРИЛОЖЕНИЕ_Б).

## Проектирование последовательностей проекта

В целях визуализации взаимодействия объектов системы между собой во времени в едином сценарии использования используется ещё одна UML-диаграмма – диаграмма последовательностей. Данная диаграмма иллюстрирует, как различные части системы взаимодействуют друг с другом для выполнения функции, а также порядок, в котором происходит взаимодействие при выполнении конкретного случая использования.

Для отображения течения времени используется линия жизни объекта, которая изображается с помощью штриховой линии, которая проводится вертикально вниз. С помощью линии жизни показывается период, в течение которого объект существует в системе. Сами объекты изображаются в виде прямоугольников, а сообщения, которыми они обмениваются – в виде линий со стрелками.

Диаграмма последовательностей представлена в [приложении В](#_ПРИЛОЖЕНИЕ_В).

Таким образом, благодаря проектированию программного средства удалось создать эффективное и удобное приложение, соответствующее всем требованиям и спецификациям, описанным в предыдущих разделах.

# Реализация программного средства

## Основные классы программного средства

Для выполнения технических задач программного средства «Площадка объявлений» должны быть реализованы следующие функции и соответствующие им классы и методы:

* выполнение входа;
* регистрация;
* просмотр объявлений;
* заполнение формы размещения объявления;
* просмотр собственных объявлений;
* функционирование панели администратора.

Далее подробно рассмотрены каждые из необходимых для выполнения технических задач функции, а также созданные для их выполнения классы и методы и их функционал и реализация.

## Выполнение входа

Графическая часть формы авторизации описана в классе AuthWindow. После нажатия кнопки «Войти» сработает метод compareDataOfUser(), преобразующий введённый пользователем пароль в строковое значение для последующей проверки на соответствие с сохранённым в базе данных паролем. Данный метод описан в классе AuthWindowViewModel – логика модели представления для окна авторизации.

При успешной валидации введённых пользователем входных данных выполняется проверка на существование введённого логина в базе данных, и при нахождении значения выполняется проверка на соответствие введённого пароля и хранящегося в базе данных пароля. При успешном прохождении всех проверок пользователь перенаправляется на главную страницу, а противном случае – получает сообщение о соответствующей ошибке.

Код метода compareDataOfUser представлен в [приложении Д.](#_ПРИЛОЖЕНИЕ_Д)

Таким образом, в классе AuthWindowViewModel определены свойства, методы и события, которые обеспечивают взаимодействие с базой данных и обработку действий пользователя при авторизации.

## Регистрация

Для совершения пользователем регистрации нового аккаунта реализован класс RegistrationWindow, в котором описана графическая составляющая окна «Регистрации».

При переходе к окну регистрации с окна входа создается новый экземпляр класса RegistrationWindow, а также выполняется метод registrationWindow.Show() - отображает окно регистрации на экране.

При нажатии пользователя на кнопку «Регистрация» вызывается основной метод registration(). Данный метод находится в классе RegistrationViewModel – логика модели представления для окна регистрации.

В начале все введённые пользователем данные проходят проверку: для валидации e-mail адреса вызывается метод IsNullOrEmpty(), а также идёт проверка по регулярному выражению, имя и фамилия также проверяется на корректность с использованием метода IsNullOrEmpty(), поле с паролем и его подтверждением проверяются с помощью метода Equals().

После успешного прохождения вышеописанных проверок происходит попытка добавить пользователя в базу данных и перенаправление пользователя на главную страницу. Если логин, адрес электронной почты или номер телефона уже зарегистрирован в базе данных, то пользователь получает соответствующее уведомление об ошибке.

Код метода registration() представлен в [приложении Д.](#_ПРИЛОЖЕНИЕ_Д)

Таким образом, класс RegistrationViewModel содержит логику и данные, связанные с окном регистрации, и предоставляет методы для выполнения регистрации и уведомлений об изменении значений свойств.

## Просмотр объявлений

После успешной авторизации пользователь перенаправляется на основную страницу программного средства, графическая часть которой представлена классом AllAnnouncement.

В конструкторе класса происходит загрузка всех объявлений, изображений продавцов и прочих данных. Также создаются объекты окон, которые будут использоваться для отображения информации о выбранном объявлении и быстрого просмотра.

В классе реализованы методы для поиска, добавления и удаления объявлений, а также для переключения между объявлениями и сортировки их по цене. После нажатия на кнопку «Поиск», вызывается метод search() – метод находится в классе AllAnnouncementViewModel – логика модели представления для страницы со всеми объявлениями. В классе также реализованы методы для обработки событий, например, изменения выбранного объявления.

Все данные хранятся в виде коллекций, а также изображения продавцов загружаются из базы данных и преобразуются в формат BitmapImage. Класс также реализует интерфейс INotifyPropertyChanged, что позволяет уведомлять об изменении свойств и обновлять UI элементы.

Код метода search() представлен в [приложении Д.](#_ПРИЛОЖЕНИЕ_Д)

Класс AllAnnouncementViewModel реализует функциональность, связанную с отображением, поиском, фильтрацией, сортировкой и управлением объявлениями. Он также обрабатывает события изменения свойств и реализует уведомление об изменении свойств с помощью интерфейса INotifyPropertyChanged.

## Заполнение формы размещения объявления

Заполнение формы для размещения объявления реализовано в классе AddWindow, который представляет графическое отображение окна.

Он использует несколько репозиториев (tmpAnnouncementRepository, announcementRepository, regionRepository), чтобы получить и сохранить данные.

AddWindowViewModel имеет свойства Name, Category, About, Region, Cost и Info, каждое из которых имеет геттер и сеттер, а также два списковых свойства - Categories и Regions. Кроме того, этот класс реализует методы IsCorrected(), addAnnouncement() и clear(), каждый из которых выполняет свои задачи.

Данный класс также имеет событие PropertyChanged, которое генерируется при изменении свойства объекта. Если это событие подписано на экземпляр AddWindowViewModel, то оно вызовет метод OnPropertyChanged(), который сообщит подписчикам об изменении свойства. После успешной валидации метод вносит объявление в таблицу с объявлениями, которые поставлены на проверку администратором.

Листинг реализации класса AddWindowViewModel представлен в [приложении Д.](#_ПРИЛОЖЕНИЕ_Д)

В итоге можно сказать, что класс AddWindowViewModel отвечает за взаимодействие представления окна добавления объявления с соответствующими репозиториями (tmpAnnouncementRepository, announcementRepository и regionRepository) и обеспечивает валидацию и обработку данных перед их сохранением.

## Просмотр собственных объявлений

Графическая часть просмотра собственных объявлений реализован в классе MyAnnouncementPage.

Класс MyAnnouncementPageViewModel является моделью представления для страницы собственных объявлений. Внутри класса определены:

* поля для работы с хранилищем данных через репозитории;
* две коллекции объявлений tmpAnnouncements и tmpTmpAnnouncements, представленные в виде ObservableCollection<T>, которые используются для отображения списков актуальных и на проверке объявлений соответственно;
* поле selectedItem для хранения выбранного пользователем элемента в списке;
* свойства CountActual и CountTmp для вывода информации о количестве актуальных и на проверке объявлений;
* методы для обновления, добавления, редактирования и удаления объявлений, а также для отображения подробной информации об объявлении;
* реализация интерфейса INotifyPropertyChanged, который позволяет уведомлять представление об изменениях в модели представления;
* конструктор класса, который вызывает метод update(), чтобы заполнить коллекции объявлениями при инициализации класса.

Листинг реализации класса MyAnnouncementPageViewModel представлен в [приложении Д.](#_ПРИЛОЖЕНИЕ_Д)

В общем, класс предоставляет функционал для управления объявлениями на странице, включая добавление, редактирование и удаление объявлений, а также отображение подробной информации об объявлении.

## Функционирование панели администратора

Все необходимые администратору методы инкапсулированы в классе AdminPage, имеющем методы для управления пользователями, всеми объявлениями, а также принятием объявлений от всех пользователей.

В соответствующей ViewModel используются различные репозитории для доступа к данным из базы данных, такие как EFUserRepository, EFAnnouncementRepository и EFTmpAnnouncementRepository.

ViewModel содержит три коллекции объектов: tmpUsers, tmpAnnouncements и tmpTmpAnnouncements, которые используются для отображения данных в представлении.

ViewModel также имеет свойства SelectedItem и Message, которые используются для отслеживания выбранного элемента в представлении и сообщения, которое будет отображаться в диалоговом окне.

Конструктор ViewModel вызывает метод update(), который обновляет коллекции объектов из репозиториев.

ViewModel также содержит методы transferToAnnouncement(), accept() и delete(), которые выполняют различные действия, такие как перенос временного объявления в постоянное объявление, изменение привилегий пользователя и удаление объектов из базы данных.

Изменения свойств в ViewModel реализованы через реализацию интерфейса INotifyPropertyChanged, который позволяет уведомлять представление об изменениях в свойствах объектов ViewModel.

Листинг реализации класса AdminPageViewModel представлен в [приложении Д.](#_ПРИЛОЖЕНИЕ_Д)

Каждый класс имеет свою реализацию, которая описывает функциональность класса и методы, используемые для достижения этой функциональности. Были использованы различные репозитории для доступа к данным из базы данных, такие как EFUserRepository, EFAnnouncementRepository и EFTmpAnnouncementRepository. ViewModel использует интерфейс INotifyPropertyChanged, который позволяет уведомлять представление об изменениях в свойствах объектов ViewModel.

Таким образом, в разделе подробно описаны основные классы и методы, используемые для реализации функциональности программного средства «Площадка объявлений».

# Тестирование, проверка работоспособности и анализ полученных результатов

## Тестирование авторизации и регистрации

При тестировании данного приложения, были применены сценарии, которые могли бы привести к ошибке. В этой главе мы рассмотрим некоторые такие сценарии и посмотрим на их обработку.

В момент авторизации, возможна такая ситуация, в которой пользователь ничего не ввёл. Обработка данного сценария приведена на рисунке 5.1.

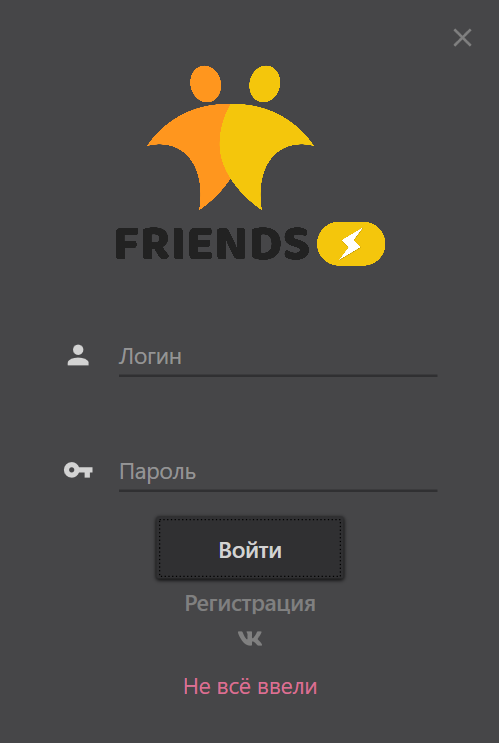


Рисунок 5.1 — Обработка пустых полей при авторизации

Как видно с рисунка, в статус-поле вывело сообщение «Не всё ввели».

При вводе некорректных данных, возникает ошибка, представленная на рисунке 5.2.

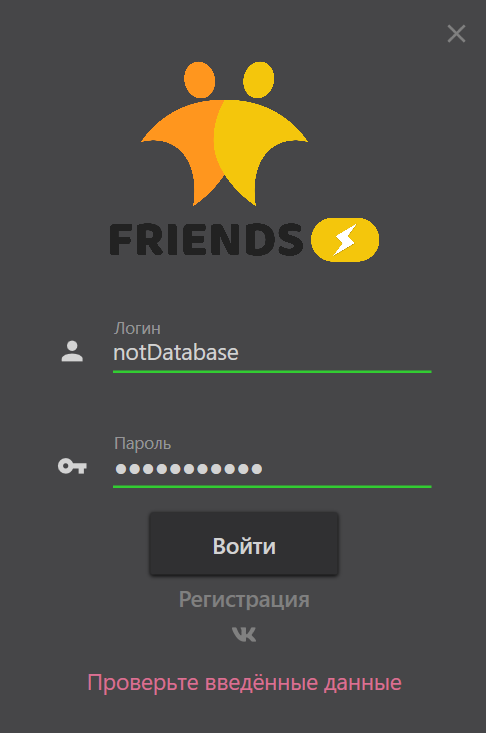


Рисунок 5.2 — Обработка некорректных данных

Как видно с рисунка, в статус-поле вывело сообщение «Проверьте введённые данные».

Также возможен такие случаи:

Пользователь хочет зарегистрироваться, при этом не указав никаких данных (рисунок 5.3).

Пользователь указывает электронную почту, которая уже привязана к учётной записи (рисунок 5.4)

Пользователь, когда осуществляет ввод паролей, указывает несовпадающие пароли (рисунок 5.5).

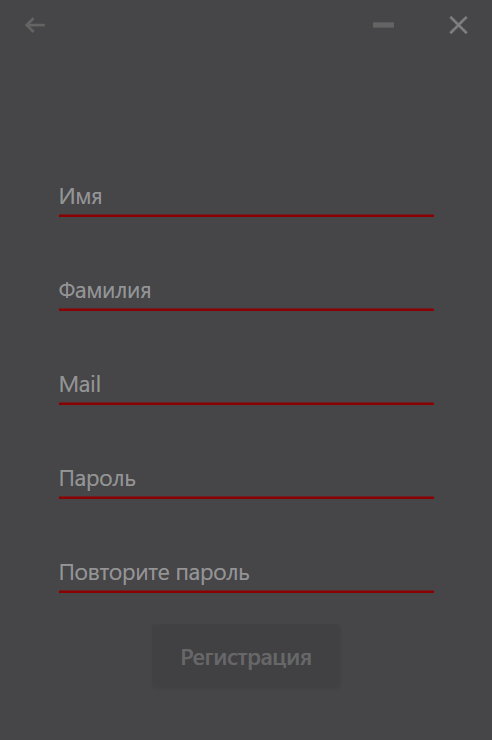


Рисунок 5.3 — Обработка пустых полей при регистрации

Как видно с рисунка, кнопка «Регистрация» не доступна для клика.

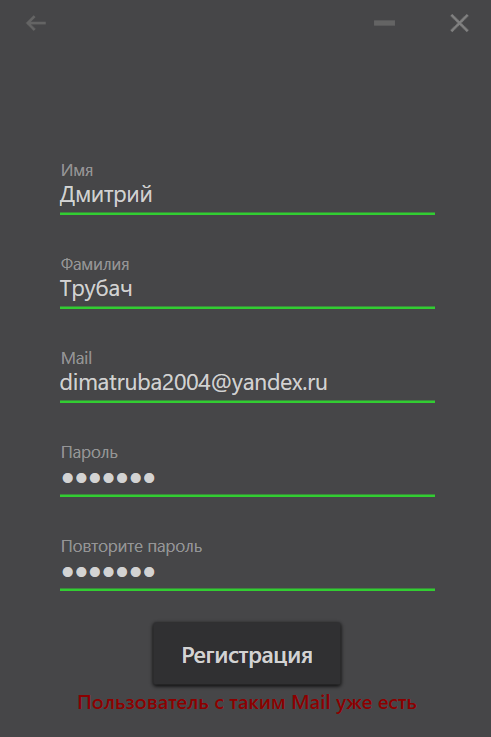


Рисунок 5.4 — Обработка ввода уже зарегистрированного Mail

Как видно с рисунка, в статус-поле вывело сообщение «Пользователь с таким Mail уже есть».

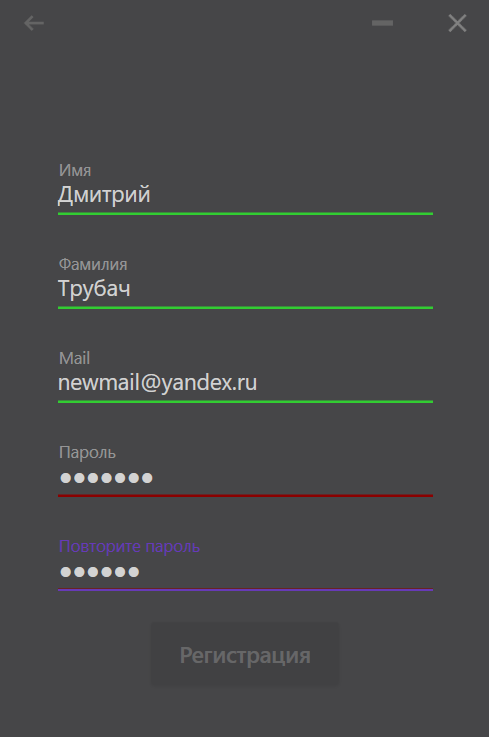


Рисунок 5.5 — Обработка подтверждения пароля

Как видно с рисунка, кнопка «Регистрация», как и в случае пустых полей, не доступна для клика.

Поскольку в приложении имеется вход через социальную сеть «ВКонтакте», имеет смысл также протестировать данный тип авторизации.

Возможны два исхода:

Пользователь не ввёл необходимые данные (рисунок 5.6).

Пользователь ввёл некорректный логин либо пароль (рисунок 5.7).

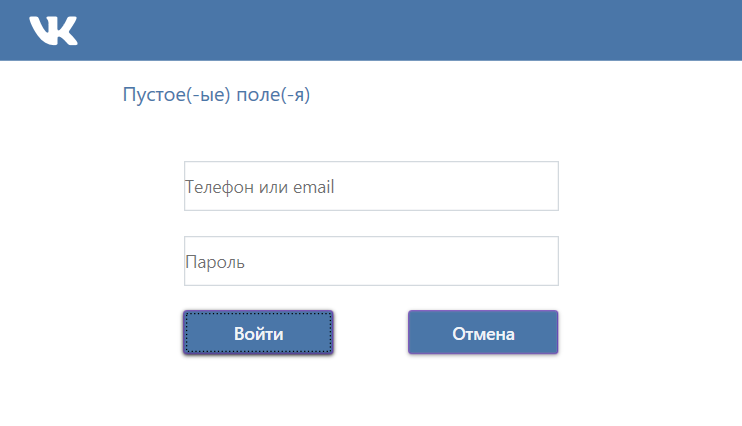


Рисунок 5.6 — Обработка пустых полей

Как видно с рисунка, в статус-поле вывело сообщение «Пустое(-ые) поле(-я)».

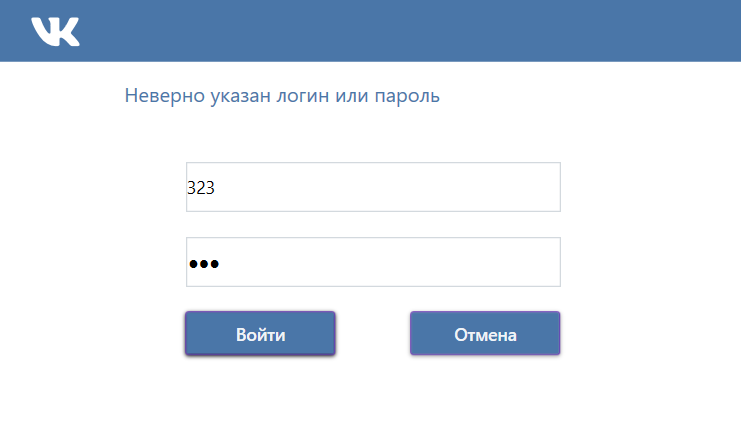


Рисунок 5.7 — Обработка некорректных данных

Как видно с рисунка, в статус-поле вывело сообщение «Неверно указан логин или пароль».

Также может возникать такая ситуация, когда пользователь, который не имеет достаточные привилегии, хочет зайти в меню управления. Ему не даст туда зайти ограничение по привилегиям. Пример показан на рисунке 5.8.

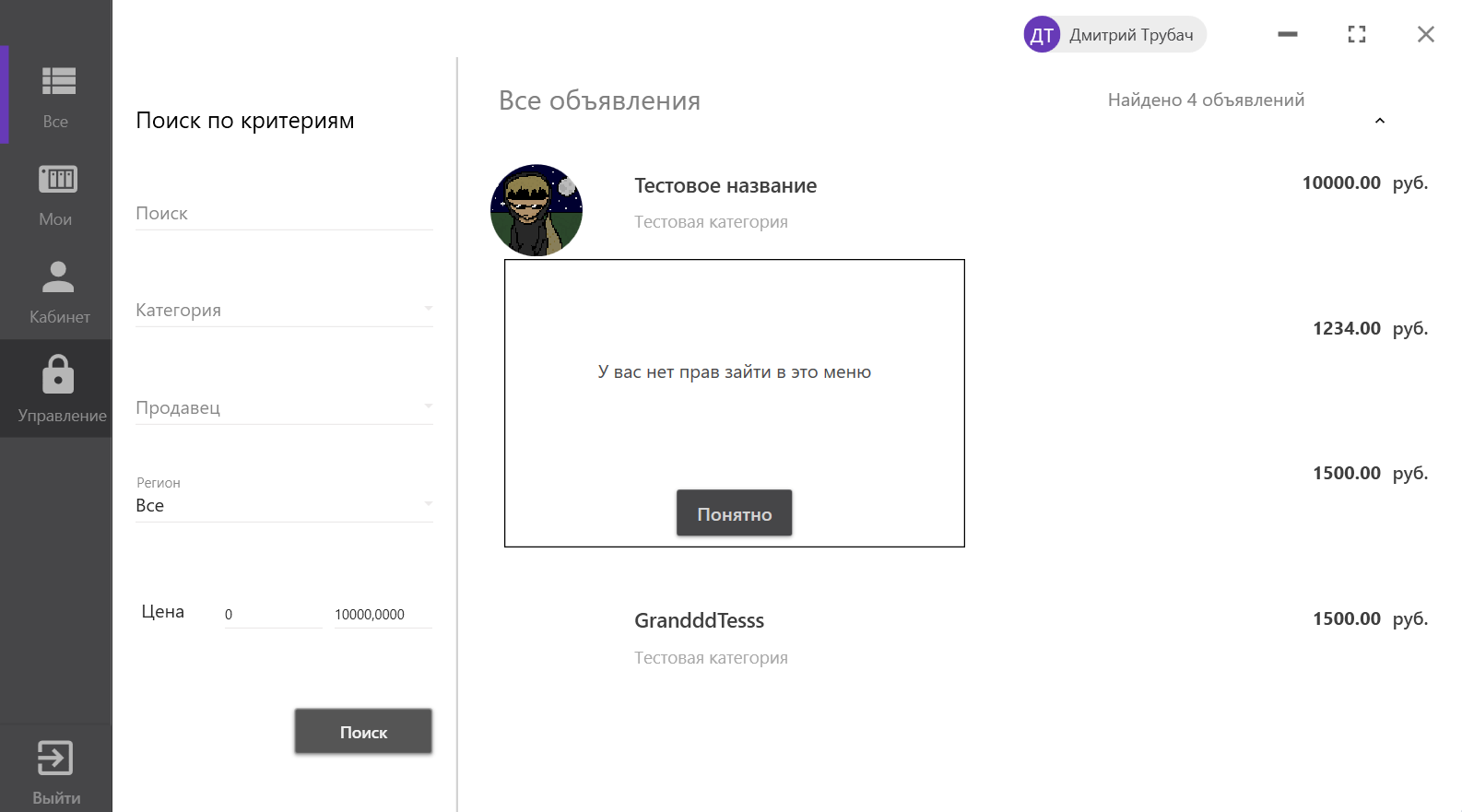


Рисунок 5.8 — Обработка ограничений по привилегиям

Как можно видеть, при попытке входа в админ-панель обычному пользователю, будет показано окно с текстом «У вас нет прав зайти в это меню».

## Тестирование заполнения формы объявления

При заполнении формы объявления возможно возникновение исключительных ситуаций. Название объявления должно быть длинной от 5 до 50 символов, должна быть выбрана существующая категория, либо вписать новую, корректно ввести цену и наличие вторичной информации о объявлении. Примеры обработки исключительных ситуаций представлены на рисунках 5.9 – 5.14.



Рисунок 5.9 — Обработка длины названия объявления

Как видно с рисунка, в статус-поле вывело сообщение «Название слишком короткое».



Рисунок 5.10 — Обработка длины названия объявления

Как видно с рисунка, в статус-поле вывело сообщение «Название слишком длинное».



Рисунок 5.11 — Обработка выбора категории

Как видно с рисунка, в статус-поле вывело сообщение «Название слишком длинное».

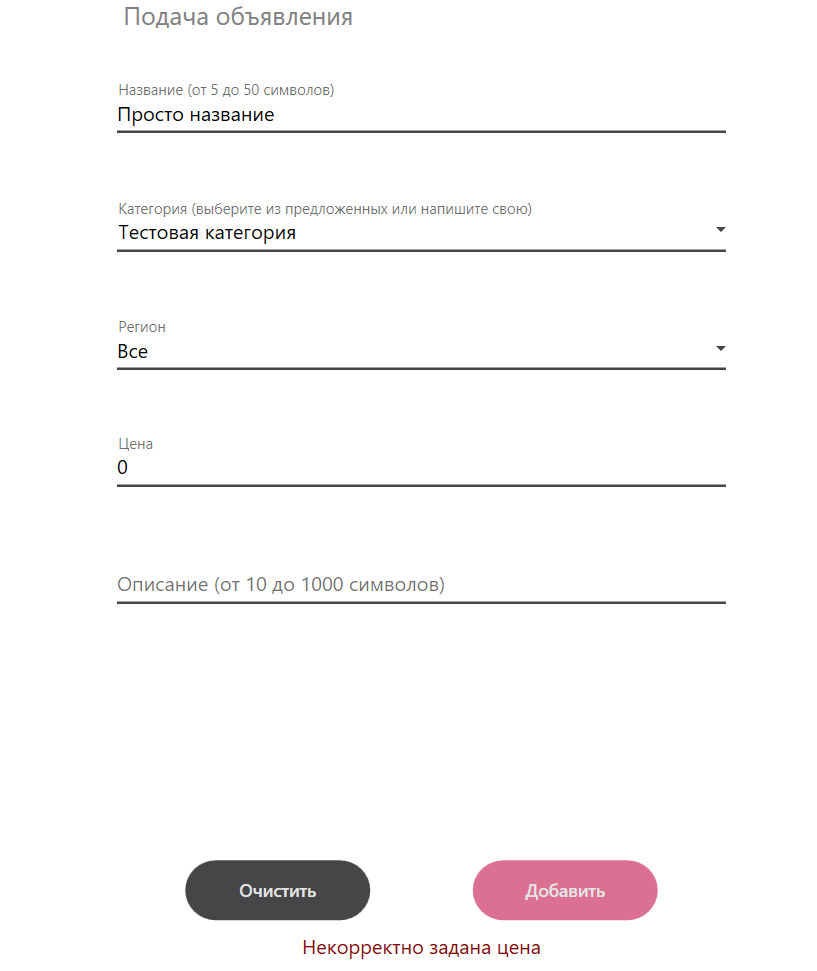


Рисунок 5.12 — Обработка корректности цены

Как видно с рисунка, в статус-поле вывело сообщение «Некорректно задана цена».

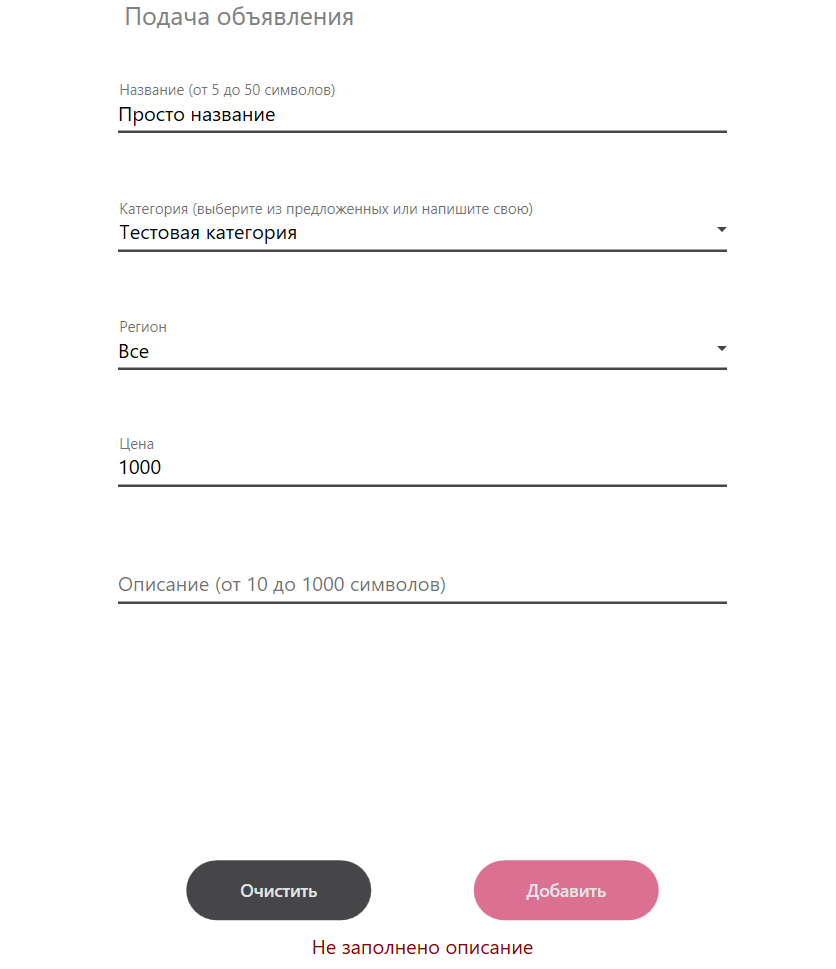


Рисунок 5.13 — Обработка ввода описания

Как видно с рисунка, в статус-поле вывело сообщение «Не заполнено описание».



Рисунок 5.14 — Обработка длины описания

Как видно с рисунка, в статус-поле вывело сообщение «Описание слишком короткое».

Таким образом, в данном разделе были рассмотрены обработки всех исключений, которые могут быть вызваны пользователем при использовании программного средства.

# Руководство по использованию

При запуске приложения у нас появляется окно авторизации, которое требует ввода логина и пароля для начала работы (рисунок 6.1).

В случае, если вас нету своей учётной записи, её можно создать, нажав на кнопку «Регистрация» в окне авторизации. Вы будете направлены в окно регистрации, где вам нужно будет заполнить все поля. После успешной регистрации, вам нужно будет вернутся в окно авторизации и ввести ваши данные, указанные при регистрации. Также поддерживается вход через социальную сеть «ВКонтакте».

Нажав на соответствующие кнопки в окне авторизации, у вас появится окно, где нужно будет ввести данные для входа в учётную запись, через которую производится вход.

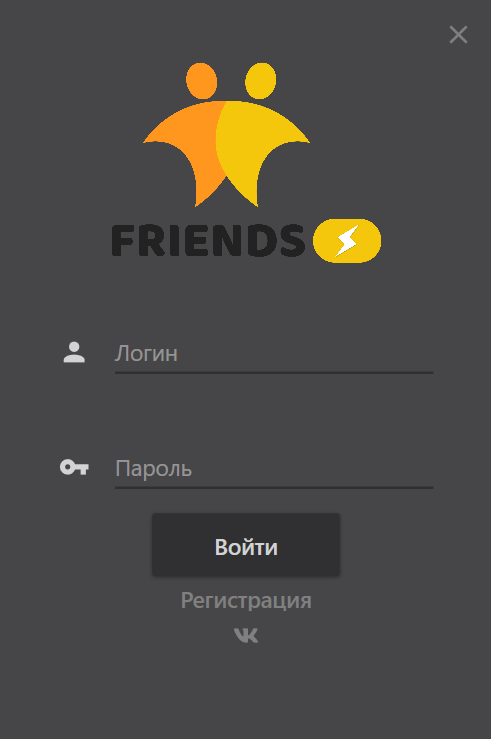


Рисунок 6.1 – Окно авторизации

После успешного входа, мы попадаем в главное окно, которое изображено на рисунке 6.2.

В этом окне мы может по нескольким критериям выбрать набор тех объявлений, которые нам подходя. Эти критерии находятся слева страницы. Поиск производится по нажатию кнопки «Поиск».

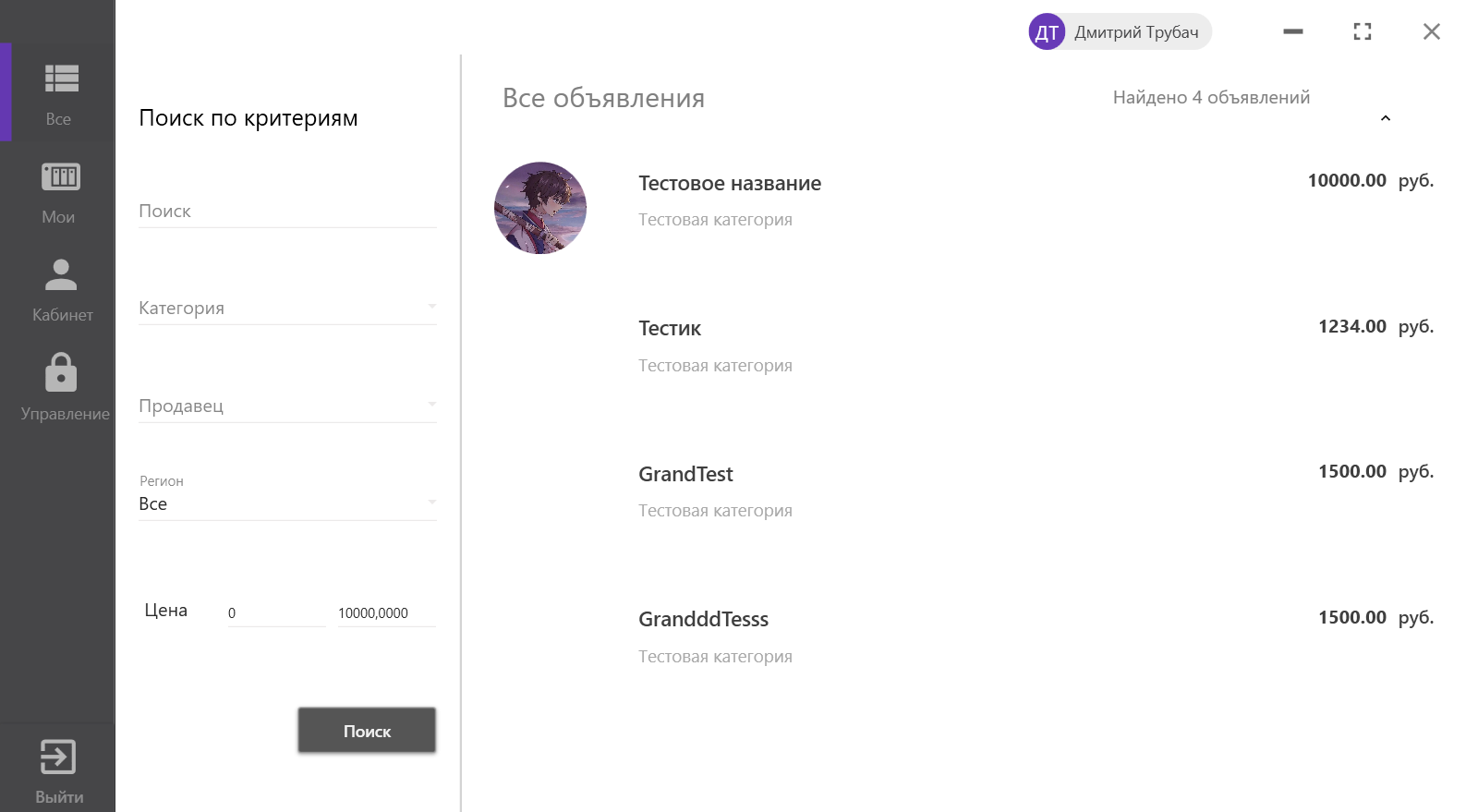


Рисунок 6.2 – Главное окно

Для добавления своего объявления нам потребуется перейти на страницу «Мои» и нажать на кнопку «+» в верхнем правом углу экрана. После чего у нас появится окно, в котором нам необходимо будет ввести данные нового объявления, а именно:

* название;
* категория;
* цена;
* регион;
* описание.

После этого, объявление будет показано в разделе «Неактуальные» на странице «Мои».

После проверки этого объявления модератором или админом, оно будет показано как актуальное и видно в общем списке объявлений.

Ниже показан рисунок 6.3, на котором показана страница «Мои».

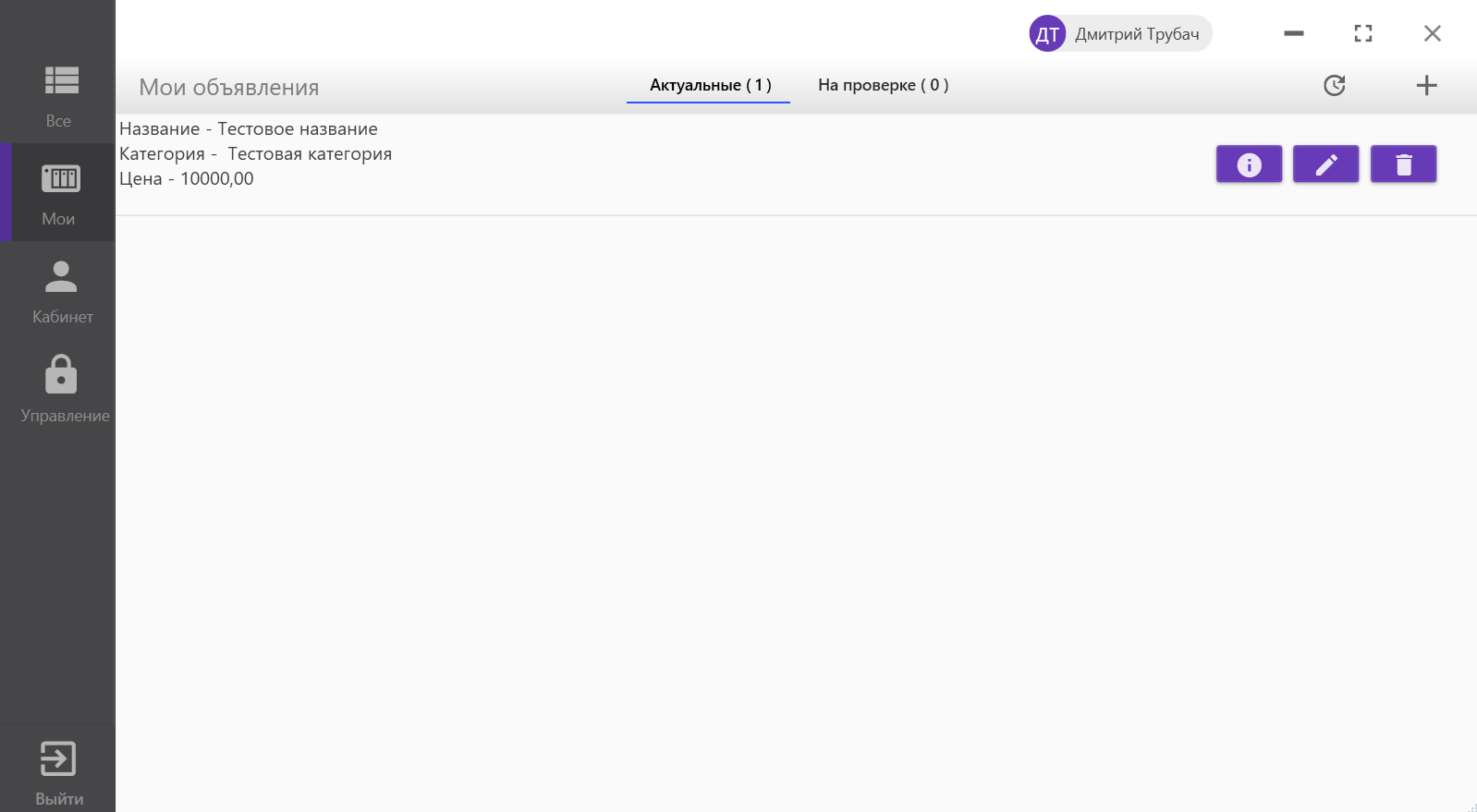


Рисунок 6.3 – Страница «Мои»

Ниже показана страница добавления объявления (рисунок 6.4).

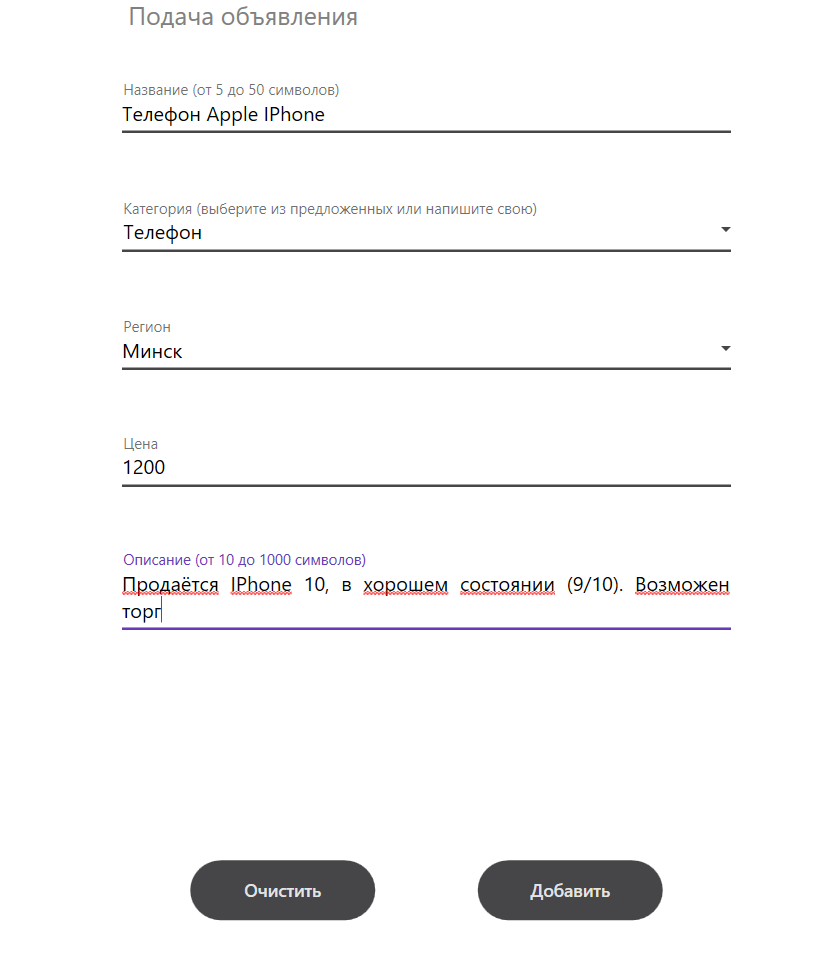


Рисунок 6.4 – Окно добавления объявления

Также на странице «Мои» есть кнопки, нажав на которые будут открыты соответствующие иконкам окна:

* быстрый просмотр;
* редактирование;
* удаление.

При переходе на страницу «Кабинет» мы попадаем на страницу, в которой отображены наши личные данные, которые можем изменять. Если был выполнен вход из «ВКонтакте», то все эти поля заполнятся теми данными, которыми были заполнены соответствующие поля в социальной сети.

Ниже, на рисунке 6.5, представлена страница «Кабинет».

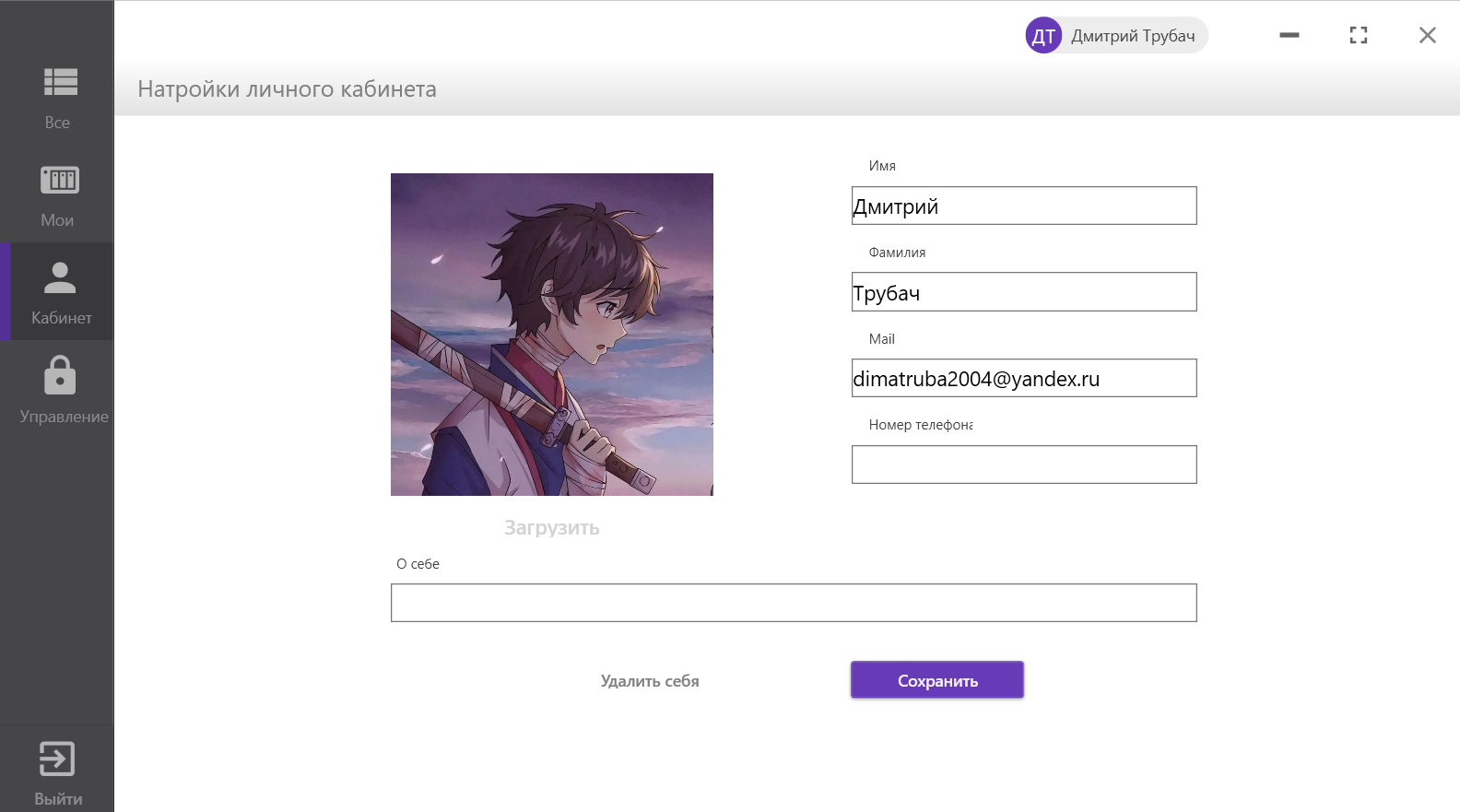


Рисунок 6.5 – Страница «Кабинет»

На рисунке 6.6 показана страница администратора.

Есть 3 вкладки: Пользователи, Актуальные, Неактуальные.

На этой странице можно совершать действия над пользователями, временными объявлениями, актуальными объявлениями.

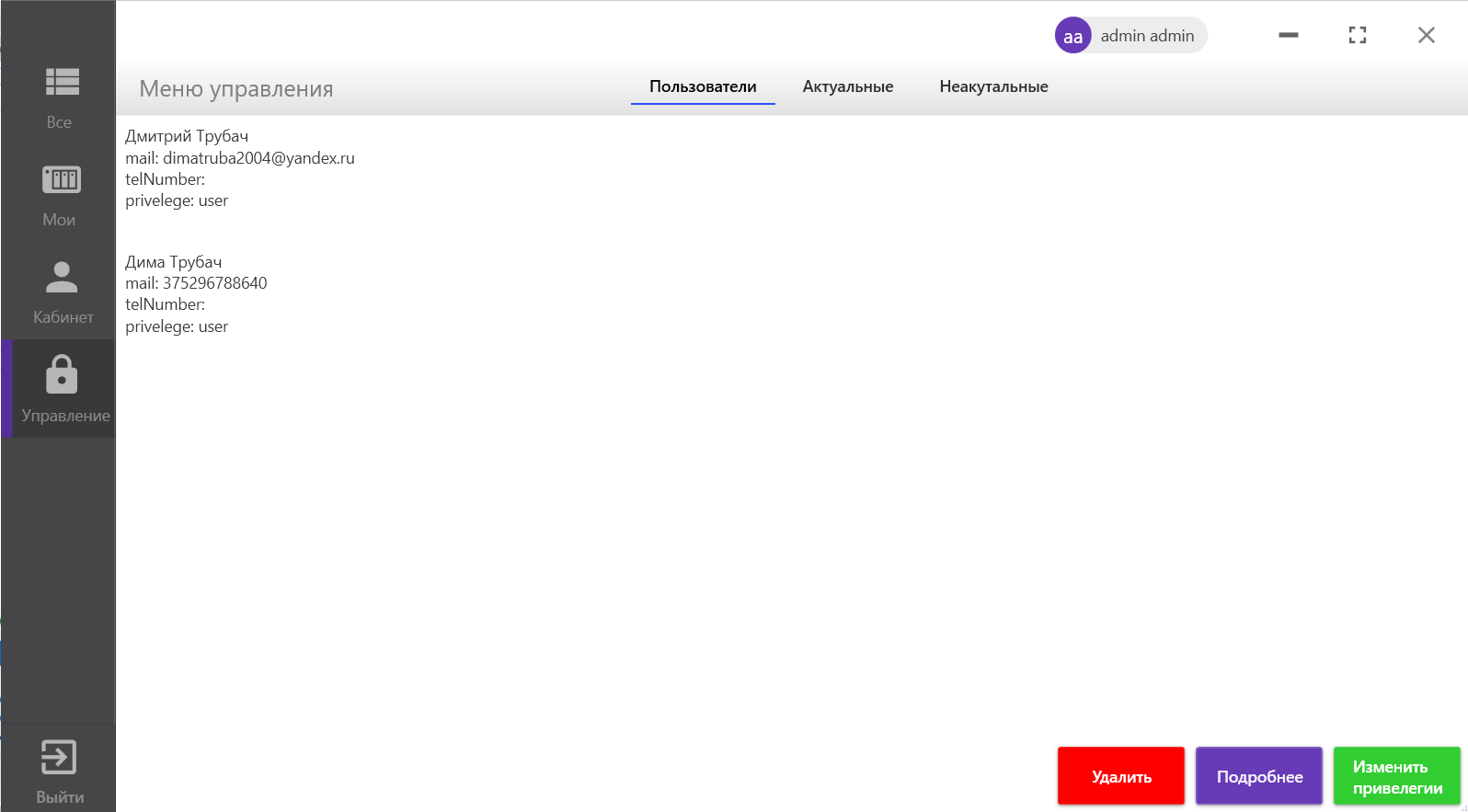


Рисунок 6.6 – Панель администратора

В целом, данный предоставляет всю необходимую информацию, чтобы пользователи могли успешно использовать приложение, добавлять свои объявления, осуществлять поиск и управлять своими учетными записями, а администраторы имели полный контроль над системой и пользователями.

# ЗАКЛЮЧЕНИЕ

В данном курсовом проекте было разработано программное средство «Площадка объявлений» при помощи языка программирования C#, API-интерфейса Windows Presentation Foundation, технологии Entity Framework и базы данных Microsoft SQL Server. При выполнении курсового проекта использовались принципы и приемы ООП.

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

Для пользователя:

* регистрация;
* авторизация;
* вход через VKontakte;
* изменение данных своего профиля;
* размещение объявлений разного типа;
* просмотр доступных объявлений;
* удаление собственных объявлений;
* выборка объявлений по различным критериям;
* управление своими объявлениями (редактирование, удаление).

Для администратора:

* авторизация;
* изменение данных своего профиля;
* просмотр информации о пользователях;
* изменение привелегий у пользователей;
* проверка поданных объявлений;
* размещение объявлений разного типа;
* просмотр доступных объявлений;
* удаление объявлений;
* выборка объявлений по различным критериям;
* управление всеми объявлениями (редактирование, удаление);
* отправление электронного письма пользователю при одобрении объявления.

Тестирование программы показало, что она работает корректно и выполняет все свои функции.

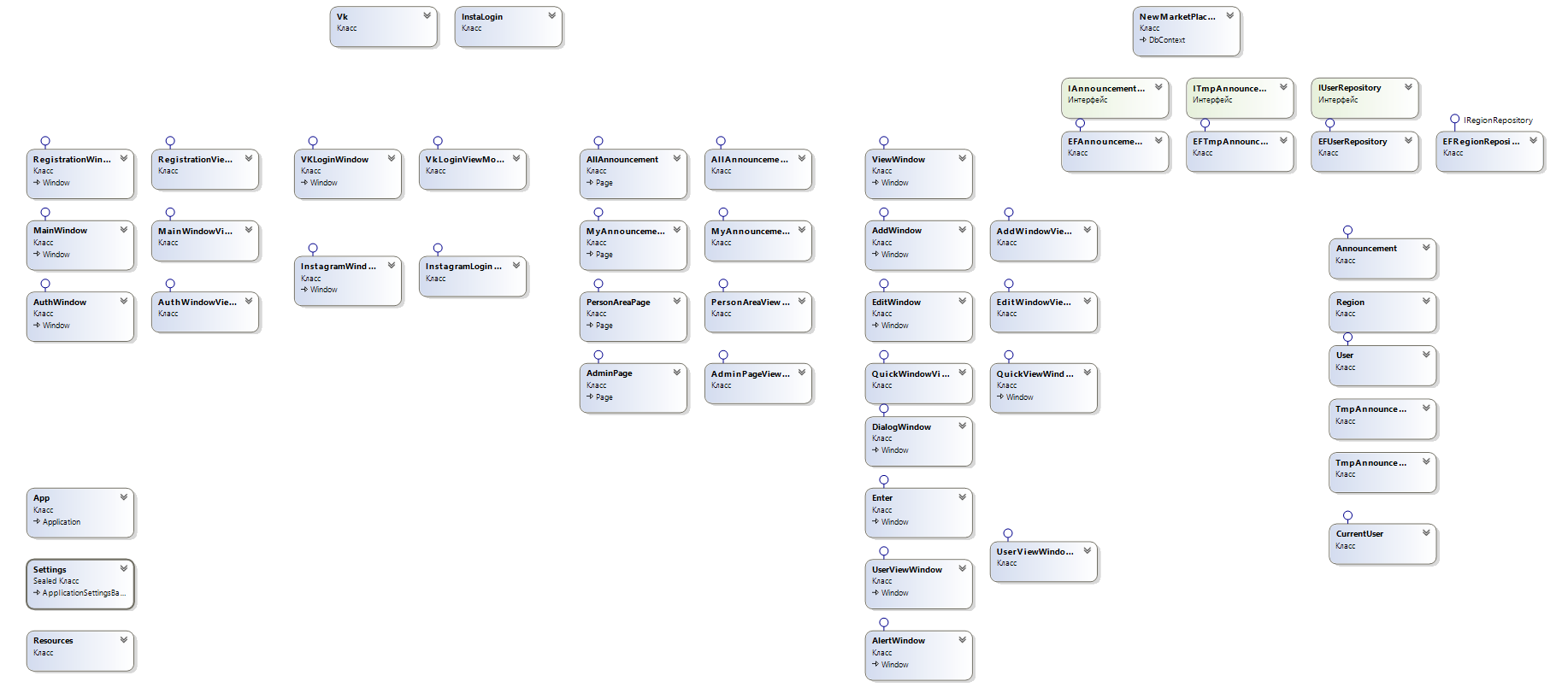
В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объеме.

# Список использованных источников

1. Onliner [Электронный ресурс] – https://catalog.onliner.by/ – Дата доступа 12.03.2023
2. Kufar [Электронный ресурс] – https://www.kufar.by/l – Дата доступа 13.03.2023
3. Avito [Электронный ресурс] – https://www.avito.ru/– Дата доступа 15.03.2023
4. Microsoft Visual Studio [Электронный ресурс] – https://ru.wikipedia.org/wiki/Microsoft\_Visual\_Studio – Дата доступа 23.04.2023
5. Полное руководство по языку программирования C# 7.0 и платформе .NET 4.7. Режим доступа: https://metanit.com/sharp/tutorial/ – Дата доступа: 23.04.2023
6. Пацей, Н.В. Курс лекций по языку программирования C# / Н. В. Пацей. – Минск: БГТУ, 2018. – 175 с.
7. Руководство по WPF // [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/wpf/ – Дата доступа: 25.04.2023
8. Руководство по XAML // [Электронный ресурс]. – Режим доступа: https://www.tutorialspoint.com/xaml/index.htm– Дата доступа: 25.04.2023
9. Работа с Entity Framework Core [Электронный ресурс] – https://professorweb.ru/my/entity-framework/6/level1/ – Дата доступа 26.04.2023
10. Руководство по VkNet API // [Электронный ресурс]. – Режим доступа: https://vknet.github.io/vk/ – Дата доступа: 28.04.2023
11. Блинова, Е.А. Курс лекций по Базам данным / Е.А. Блинова. – Минск: БГТУ, 2019. – 175 с.

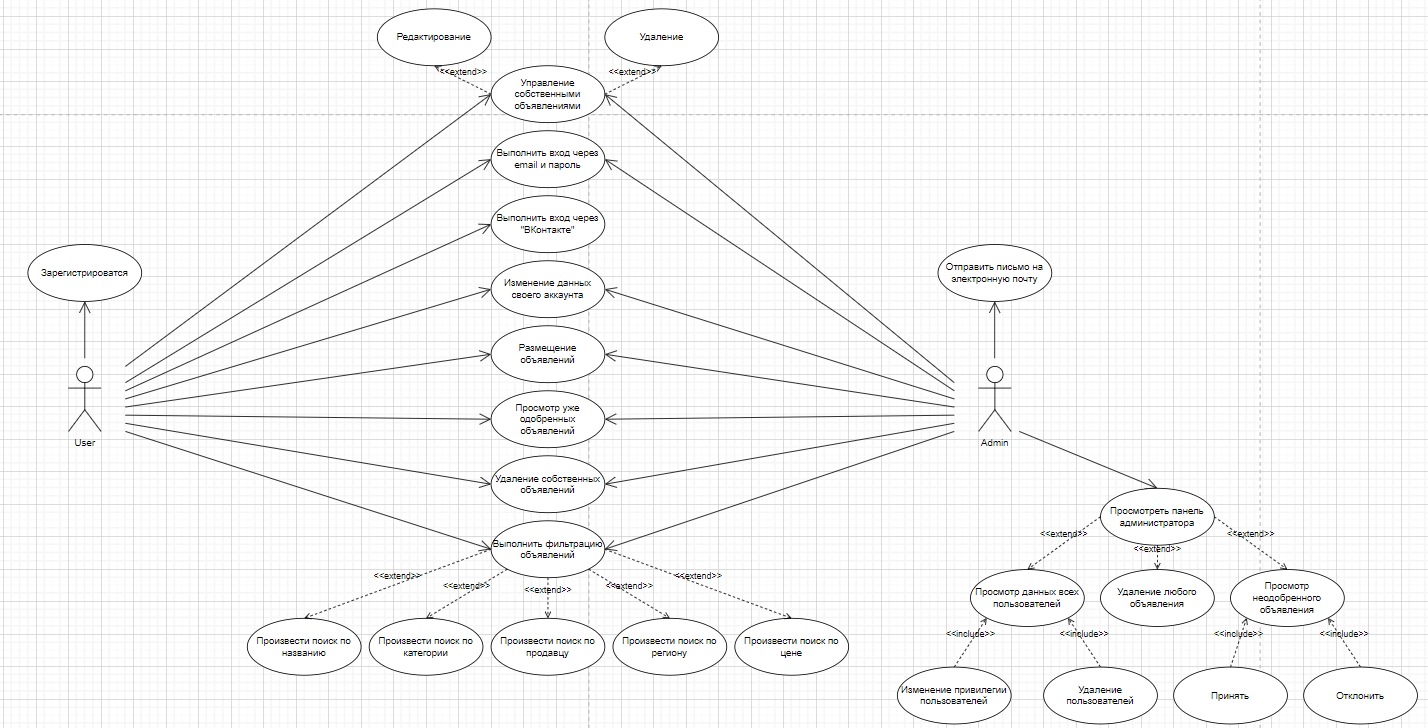
# ПРИЛОЖЕНИЕ А

Диаграмма классов



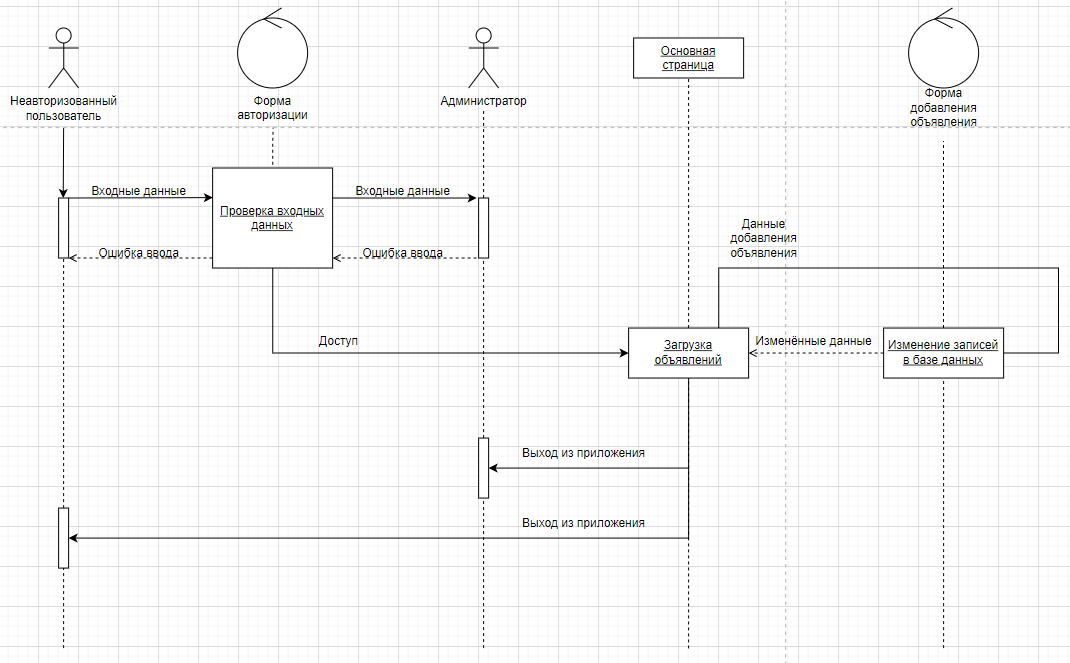
# ПРИЛОЖЕНИЕ Б

Диаграмма использований



# ПРИЛОЖЕНИЕ В

Диаграмма последовательностей



# ПРИЛОЖЕНИЕ Г

Листинг – База данных CW\_MarketPlace\_OOP

|  |
| --- |
| create database CW\_MarketPlace\_OOP;  use CW\_MarketPlace\_OOP  create table Users  (  id int primary key identity(1,1),  firstName nvarchar(30) not null,  secondName nvarchar(30) not null,  mail nvarchar(50) not null,  password nvarchar(100) null,  telNumber varchar(20),  about nvarchar(1000) not null,  image varbinary(MAX),  privilege nvarchar(10) default 'user' not null  );  create table Regions  (  id int primary key identity(0,1),  region nvarchar(50) not null  );  create table TmpAnnouncements  (  id int primary key identity(1,1),  name nvarchar(50) not null,  seller int foreign key references Users(id),  idRegion int foreign key references Regions(id),  category nvarchar(50) not null,  about nvarchar(1000) not null,  cost money not null  );  create table Announcements  (  id int primary key identity(1,1),  name nvarchar(50) not null,  seller int foreign key references Users(id) not null,  idRegion int foreign key references Regions(id) not null,  category nvarchar(50) not null,  about nvarchar(1000) not null,  cost money not null  );  insert into Regions (region) values('Все');  insert into Regions (region) values('Брестская');  insert into Regions (region) values('Витебская');  insert into Regions (region) values('Гомельская');  insert into Regions (region) values('Гродненская');  insert into Regions (region) values('Минская');  insert into Regions (region) values('Могилёвская');  insert into Regions (region) values('Минск');  insert into Users (firstName, secondName, mail, password, telNumber, about, privilege)  values ('admin', 'admin', 'ForTestAdm1@gmail.com', 'ISMvKXpXpadDiUoOSoAfww==', '+375291234567', 'Admin from database', 'admin'); |

# ПРИЛОЖЕНИЕ Д

Листинг – Код метода compareDataOfUser

|  |
| --- |
| public bool compareDataOfUser(string password)  {  if (!String.IsNullOrEmpty(Login) && !String.IsNullOrEmpty(password))  {  User tmp = eFUserRepository.getByMail(Login);  if (tmp != null)  {  if (User.getHash(password).Equals(tmp.password))  {  CurrentUser.User = tmp;  App.mainWindow = new MainWindow();  App.mainWindow.Show();  return true;  }  else  {  Info = "Проверьте введённые данные";  return false;  }    }  else  {  Info = "Проверьте введённые данные";  return false;  }  }  else  {  Info = "Не всё ввели";  return false;  }  } |

Листинг – Код метода registration

|  |
| --- |
| public bool registration(string password1, string password2)  {  Password1 = password1;  Password2 = password2;  if (!String.IsNullOrEmpty(FirstName) && !String.IsNullOrEmpty(SecondName) && !String.IsNullOrEmpty(Mail) && !String.IsNullOrEmpty(Password1) && !String.IsNullOrEmpty(Password2))  {  if (Password1.Equals(Password2))  {  User tmp = userRepository.getByMail(Mail);  if(tmp == null)  {  userRepository.add(new User(FirstName, SecondName, Mail, Password1));  Info = "Зареган!";  return true;  }  else  {  Info = "Пользователь с таким Mail уже есть";  return false;  }    }  else  {  Info = "Пароли должны совпадать";  return false;  }  }  else  {  Info = "Проверьте данные!";  return false;  }  } |

Листинг – Код метода search

|  |
| --- |
| public void search()  {  selectedItem = null;  Announcements.Clear();  Regex regex = new Regex(@"(\w\*)(?i)" + SearchText + @"(\w\*)");  int regionId = SelectedIndex;  HashSet<Announcement> tmp1 = new HashSet<Announcement>();  HashSet<Announcement> tmp2 = new HashSet<Announcement>();  if (SelectedIndex != 0)  {  foreach (Announcement announcement in announcementRepository.getByRegionId(SelectedIndex))  tmp1.Add(announcement);  }  else  {  foreach (Announcement announcement in announcementRepository.getAll())  tmp1.Add(announcement);  }  if (!String.IsNullOrEmpty(Category))  {  foreach (Announcement announcement in tmp1.Where(x=> x.category.Equals(Category)))  tmp2.Add(announcement);  tmp1.Clear();  }  else  {  foreach (Announcement announcement in tmp1)  tmp2.Add(announcement);  tmp1.Clear();  }  if (!String.IsNullOrEmpty(Seller))  {  int id = userRepository.getByName(Seller).id;  foreach (Announcement announcement in tmp2.Where(x=>x.seller == id))  tmp1.Add(announcement);  tmp2.Clear();  }  else  {  foreach (Announcement announcement in tmp2)  tmp1.Add(announcement);  tmp2.Clear();  }  if (!String.IsNullOrEmpty(SearchText))  {  foreach (Announcement announcement in tmp1)  {  if (regex.IsMatch(announcement.about) || regex.IsMatch(announcement.name))  {  if (maxCost == 0)  {  if (announcement.cost <= maxCost && announcement.cost >= MAX\_COST)  tmp2.Add(announcement);  }  else  {  if (announcement.cost <= maxCost && announcement.cost >= minCost)  tmp2.Add(announcement);  }  }  }  }  else  {  foreach (Announcement announcement in tmp1)  {  if (maxCost == 0)  {  if (announcement.cost <= maxCost && announcement.cost >= MAX\_COST)  tmp2.Add(announcement);  }  else  {  if (announcement.cost <= maxCost && announcement.cost >= minCost)  tmp2.Add(announcement);  }  }  }  foreach (Announcement announcement in tmp2)  Announcements.Add(announcement);  tmp1.Clear();  tmp2.Clear();  Info = $"Найдено {Announcements.Count}";  } |

Листинг – Код класса AddWindowViewModel

|  |
| --- |
| public class AddWindowViewModel :INotifyPropertyChanged  {  EFTmpAnnouncementRepository tmpAnnouncementRepository = new EFTmpAnnouncementRepository();  EFAnnouncementRepository announcementRepository = new EFAnnouncementRepository();  EFRegionRepository regionRepository = new EFRegionRepository();  string name;  string category;  string about;  string region;  decimal cost;  string info;  string statusName = "Не заполнено название";  string statusCategory = "Не заполнена категория";  string statusCost = "Не заполнена цена";  string statusAbout = "Не заполнено описание";  List<string> tmpCategories = new List<string>();  List<string> tmpRegions = new List<string>();  public string Name  {  get { return name; }  set  {  if (value.Length >= 5 && value.Length <= 50)  {  name = value;  statusName = "";  }  else if (value.Length > 50) statusName = "Название слишком длинное";  else statusName = "Название слишком короткое";  OnPropertyChanged("Name");  }  }  public string Category  {  get { return category; }  set  {  if (!String.IsNullOrEmpty(value))  {  category = value;  statusCategory = "";  }  else  statusCategory = "Проверьте категорию";  OnPropertyChanged("Category");  }  }  public string About  {  get { return about; }  set  {  if (value.Length >= 10 && value.Length < 1000)  {  about = value;  statusAbout = "";  }  else  statusAbout = "Описание слишком короткое";  OnPropertyChanged("About");  }  }  public string Region  {  get { return region; }  set { region = value; }  }  public string Cost  {  get { return cost.ToString(); }  set  {  if (Decimal.TryParse(value.ToString(), out cost) && Decimal.Parse(value) >= 0)  {  if (Decimal.Parse(value) > Decimal.MaxValue)  cost = Decimal.MaxValue;  else  cost = Decimal.Parse(value);  statusCost = "";  }  else  {  statusCost = "Некорректно задана цена";  cost = 0;  }  OnPropertyChanged("Cost");  }  }  public string Info  {  get { return info; }  set  {  info = value;  OnPropertyChanged("Info");  }  }  public List<string> Categories  {  get { return tmpCategories; }  }  public List<string> Regions  {  get { return tmpRegions; }  }  public AddWindowViewModel()  {  tmpCategories = announcementRepository.getCategories().Distinct().ToList();  tmpRegions = regionRepository.getRegions();  }  bool IsCorrected()  {  if (!String.IsNullOrEmpty(statusName))  {  Info = statusName;  return false;  }  else if (!String.IsNullOrEmpty(statusCategory))  {  Info = statusCategory;  return false;  }  else if (!String.IsNullOrEmpty(statusCost))  {  Info = statusCost;  return false;  }  else if (!String.IsNullOrEmpty(statusAbout))  {  Info = statusAbout;  return false;  }  else  return true;  }  public bool addAnnouncement()  {  if(IsCorrected())  {  tmpAnnouncementRepository.add(new TmpAnnouncement(Name, CurrentUser.User.id, Regions.IndexOf(Region),Category, About, cost));  return true;  }  return false;  }  public void clear()  {  name = "";  cost = 0;  about = "";  category = "";  }  public event PropertyChangedEventHandler PropertyChanged;  void OnPropertyChanged(string propertyName)  {  if (PropertyChanged != null)  PropertyChanged(this, new PropertyChangedEventArgs(propertyName));  }  } |

Листинг – Код класса MyAnnouncementViewModel

|  |
| --- |
| public class MyAnnouncementPageViewModel : INotifyPropertyChanged  {  EFUserRepository userRepository = new EFUserRepository();  EFAnnouncementRepository announcementRepository = new EFAnnouncementRepository();  EFTmpAnnouncementRepository tmpAnnouncementRepository = new EFTmpAnnouncementRepository();  ObservableCollection<Announcement> tmpAnnouncements = new ObservableCollection<Announcement>();  ObservableCollection<TmpAnnouncement> tmpTmpAnnouncements = new ObservableCollection<TmpAnnouncement>();  object selectedItem;  int countActual;  int countTmp;  public ObservableCollection<Announcement> Announcements  {  get { return tmpAnnouncements; }  }  public ObservableCollection<TmpAnnouncement> TmpAnnouncements  {  get { return tmpTmpAnnouncements; }  }  public string CountActual  {  get { return $" Актуальные ( {countActual} )"; }  set  {  countActual = Int32.Parse(value);  OnPropertyChanged("CountActual");  }  }  public string CountTmp  {  get { return $"На проверке ( {countTmp} )"; }  set  {  countTmp = Int32.Parse(value);  OnPropertyChanged("CountTmp");  }  }  public object SelectedItem  {  get { return selectedItem; }  set  {  selectedItem = value;  }  }  public MyAnnouncementPageViewModel()  {  update();  }    public void update()  {  tmpAnnouncements.Clear();  tmpTmpAnnouncements.Clear();    foreach (Announcement announcement in announcementRepository.getBySellerId(CurrentUser.User.id))  tmpAnnouncements.Add(announcement);  foreach (TmpAnnouncement announcement in tmpAnnouncementRepository.getBySellerId(CurrentUser.User.id))  tmpTmpAnnouncements.Add(announcement);  CountActual = announcementRepository.getBySellerId(CurrentUser.User.id).Count().ToString();  CountTmp = tmpAnnouncementRepository.getBySellerId(CurrentUser.User.id).Count().ToString();  }  public void deleteAnnouncement()  {  if (SelectedItem is Announcement)  announcementRepository.delete(SelectedItem as Announcement);  else if (SelectedItem is TmpAnnouncement)  tmpAnnouncementRepository.delete(SelectedItem as TmpAnnouncement);  update();  }  public void addAnnouncement()  {  AddWindow addWindow = new AddWindow();  addWindow.ShowDialog();  }  public void showInfo()  {  if(SelectedItem != null)  {  QuickViewWindow quickViewWindow = new QuickViewWindow(SelectedItem);  quickViewWindow.ShowDialog();  }    }  public void editAnnouncement()  {  EditWindow editWindow = new EditWindow(SelectedItem);  editWindow.ShowDialog();  update();  }  public event PropertyChangedEventHandler PropertyChanged;  void OnPropertyChanged(string propertyName)  {  if (PropertyChanged != null)  PropertyChanged(this, new PropertyChangedEventArgs(propertyName));  }  } |

Листинг – Код класса AdminPageViewModel

|  |
| --- |
| public class AdminPageViewModel : INotifyPropertyChanged  {  EFUserRepository userRepository = new EFUserRepository();  EFAnnouncementRepository announcementRepository = new EFAnnouncementRepository();  EFTmpAnnouncementRepository tmpAnnouncementRepository = new EFTmpAnnouncementRepository();  ObservableCollection<User> tmpUsers = new ObservableCollection<User>();  ObservableCollection<Announcement> tmpAnnouncements = new ObservableCollection<Announcement>();  ObservableCollection<TmpAnnouncement> tmpTmpAnnouncements = new ObservableCollection<TmpAnnouncement>();  ObservableCollection<object> tmp = new ObservableCollection<object>();  object selectedItem;  string message;  public object SelectedItem  {  get { return selectedItem; }  set  {  if (value != null)  selectedItem = value;  OnPropertyChanged("SelectedItem");  }  }  public string Message  {  get { return message; }  set  {  message = value;  OnPropertyChanged("Message");  }  }  public ObservableCollection<User> Users  {  get  {  selectedItem = null;  return tmpUsers;  }  }  public ObservableCollection<Announcement> Announcements  {  get  {  selectedItem = null;  return tmpAnnouncements;  }  }  public ObservableCollection<TmpAnnouncement> TmpAnnouncements  {  get  {  selectedItem = null;  return tmpTmpAnnouncements;  }  }  public AdminPageViewModel()  {  update();  }  public void update()  {  tmpUsers.Clear();  tmpAnnouncements.Clear();  tmpTmpAnnouncements.Clear();  foreach (User user in userRepository.getAll())  {  if(user.id != CurrentUser.User.id)  tmpUsers.Add(user);  }    foreach (Announcement announcement in announcementRepository.getAll())  tmpAnnouncements.Add(announcement);  foreach (TmpAnnouncement announcement in tmpAnnouncementRepository.getAll())  tmpTmpAnnouncements.Add(announcement);  }  public void transferToAnnouncemet(TmpAnnouncement tmp)  {  if (tmp is TmpAnnouncement && tmp != null)  {  Announcement announcement = new Announcement();  announcement.name = tmp.name;  announcement.seller = tmp.seller;  announcement.category = tmp.category;  announcement.idRegion = tmp.idRegion;  announcement.about = tmp.about;  announcement.cost = tmp.cost;  announcementRepository.add(announcement);  tmpAnnouncementRepository.delete(tmp);  update();  }  }  public void accept()  {  if (SelectedItem is User)  {  if (CurrentUser.isAdmin())  {  if ((SelectedItem as User).privilege.Equals("admin"))  userRepository.changePrivelege((SelectedItem as User), "user");  else if ((SelectedItem as User).privilege.Equals("user"))  userRepository.changePrivelege((SelectedItem as User), "moderator");  else if ((SelectedItem as User).privilege.Equals("moderator"))  userRepository.changePrivelege((SelectedItem as User), "admin");  AlertWindow alertWindow = new AlertWindow($"Пользователь {(SelectedItem as User).firstName} {(SelectedItem as User).secondName} теперь {(SelectedItem as User).privilege}");  alertWindow.ShowDialog();  }  else  {  AlertWindow alertWindow = new AlertWindow("У вас недостаточно прав для совершения данного действия");  alertWindow.ShowDialog();  }    }  else if (SelectedItem is TmpAnnouncement)  transferToAnnouncemet(SelectedItem as TmpAnnouncement);  else  {  AlertWindow alertWindow = new AlertWindow($"Выберите объект");  alertWindow.ShowDialog();  }  update();  }  void deleteUser(User user)  {  foreach (TmpAnnouncement announcement in tmpAnnouncementRepository.getBySellerId(user.id))  tmpAnnouncementRepository.delete(announcement);  foreach (Announcement announcement in announcementRepository.getBySellerId(user.id))  announcementRepository.delete(announcement);  userRepository.delete(user);  }  public void delete()  {  if (SelectedItem is User)  {  if (CurrentUser.isAdmin())  {  DialogWindow dialogWindow = new DialogWindow();  dialogWindow.DataContext = this;  Message = $"Уверены, что хотите удалить пользователя {(SelectedItem as User).firstName} {(SelectedItem as User).secondName}?";  dialogWindow.ShowDialog();  if (dialogWindow.DialogResult == true)  deleteUser(SelectedItem as User);  }  else  {  AlertWindow alertWindow = new AlertWindow("У вас недостаточно прав для совершения данного действия");  alertWindow.ShowDialog();  }    }  else if (SelectedItem is Announcement)  {  DialogWindow dialogWindow = new DialogWindow();  dialogWindow.DataContext = this;  Message = $"Уверены, что хотите удалить объявление \"{(SelectedItem as Announcement).Name}\" ?";  dialogWindow.ShowDialog();  if (dialogWindow.DialogResult == true)  announcementRepository.delete(SelectedItem as Announcement);  }  else if (SelectedItem is TmpAnnouncement)  {  DialogWindow dialogWindow = new DialogWindow();  dialogWindow.DataContext = this;  Message = $"Уверены, что хотите удалить объявление \"{(SelectedItem as TmpAnnouncement).Name}\" ?";  dialogWindow.ShowDialog();  if (dialogWindow.DialogResult == true)  tmpAnnouncementRepository.delete(SelectedItem as TmpAnnouncement);  }  else  {  AlertWindow alertWindow = new AlertWindow($"Выберите объект");  alertWindow.ShowDialog();  }    update();  }  public void showInfo()  {  if(SelectedItem != null)  {  if(SelectedItem is User)  {  UserViewWindow userViewWindow = new UserViewWindow(SelectedItem);  userViewWindow.ShowDialog();  }  else  {  QuickViewWindow quickViewWindow = new QuickViewWindow(SelectedItem);  quickViewWindow.ShowDialog();  }    }    }  public event PropertyChangedEventHandler PropertyChanged;  void OnPropertyChanged(string propertyName)  {  if (PropertyChanged != null)  PropertyChanged(this, new PropertyChangedEventArgs(propertyName));  }  } |