МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение 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 ПРОЕКТА:**

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

Тема Программное средство «Магазин автозапчастей»

Исполнитель

студент 2 курса группы 4 Дащинский Максим Леонидович

(Ф.И.О.)

Руководитель работы ассистент Пахолко А.С.

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

Курсовой проект защищен с оценкой

Председатель Пахолко А.С

(подпись)

Минск 2022

**Содержание**

[ВВЕДЕНИЕ 4](#_Toc104199348)

[Глава 1. Аналитический обзор литературы и постановка задачи 5](#_Toc104199349)

[1.1 Анализ прототипов 5](#_Toc104199350)

[1.2 Постановка целей 7](#_Toc104199351)

[Глава 2. Анализ требований к программному средству и разработка функциональных требований 8](#_Toc104199352)

[2.1 Определение требований к программному средству 8](#_Toc104199353)

[2.2 Описание средств разработки 8](#_Toc104199354)

[2.3 Описание функциональности программного средства 9](#_Toc104199355)

[2.4 Спецификация функциональных требований 9](#_Toc104199356)

[Глава 3. Проектирование и создание программного средства 11](#_Toc104199357)

[3.1 Архитектура системы 11](#_Toc104199358)

[3.2 Диаграммы UML 11](#_Toc104199359)

[3.3 Проектирование логической структуры базы данных 13](#_Toc104199360)

[3.4 Структура проекта 15](#_Toc104199361)

[Глава 4. Реализация программного средства 16](#_Toc104199362)

[4.1 Реализация MVVM 16](#_Toc104199363)

[4.2 Реализация авторизации и регистрации пользователей 17](#_Toc104199364)

[4.3 Добавление и удаления товара из корзины 19](#_Toc104199365)

[4.4 Отображение оповещений внутри программного средства 20](#_Toc104199366)

[4.5 Отправка сообщений на почту и генерация чека 21](#_Toc104199367)

[4.6 Формирование, подтверждение и отмена заказа 22](#_Toc104199368)

[Глава 5. Тестирование программного средства 23](#_Toc104199369)

[Глава 6. Руководство программного средства по использованию 28](#_Toc104199370)

[6.1 Регистрация и авторизация 28](#_Toc104199371)

[6.2 Использование программного средства обычным пользователям 29](#_Toc104199372)

[6.3 Использование программного средства администратором 29](#_Toc104199373)

[ЗАКЛЮЧЕНИЕ 31](#_Toc104199374)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 32](#_Toc104199375)

[Приложение А 33](#_Toc104199376)

[Приложение Б 34](#_Toc104199377)

[Приложение В 35](#_Toc104199378)

[Приложение Г 36](#_Toc104199379)

[Приложение Д 37](#_Toc104199380)

[Приложение Е 38](#_Toc104199381)

[Приложение Ж 42](#_Toc104199382)

[Приложение И 45](#_Toc104199383)

[Приложение К 47](#_Toc104199384)

[Приложение Л 49](#_Toc104199385)

[Приложение М 51](#_Toc104199386)

# **ВВЕДЕНИЕ**

Автомобильная индустрия развивалась на протяжении своего развития. Разнообразие автомобилей стало огромно: от малолитражки до полноценного дома на колесах. Найти детали к уникальным моделям проблематично. С развитием технологий поиск деталей упростился. Приложения и площадки для поиска автозапчастей помогают найти и купить нужную деталь, а некоторые площадки сделают доставку бесплатной. Это очень сильно экономит время, вместо того, чтобы ходить в каждый магазин для покупки нужной детали.

Многие площадки и организации имеют возможность просмотра доступных деталей и их приобретение через приложения или веб-сайты. Этим и объясняется мой выбор на разработку программного обеспечения «Магазин автозапчастей». Указанное программное средство позволяет потенциальным покупателям заказать интересующую деталь на единой площадке, отслеживать статус заказа. Администратор в свою очередь имеет право добавлять новые товары, просматривать текущие заказы, а также удалять товары и поставщиков.

Для успешной реализации курсового проекта необходимо:

* выполнить анализ соответствующей литературы;
* проанализировать существующие программные средства совпадающей с моей темой;
* определить перечень основных функциональных требований;
* спроектировать структуру базы данных;
* спроектировать структуру проекта;
* осуществить релиз программного средства;
* протестировать программное средство на отказоустойчивость;
* составить руководство пользователя.

Содержание данной пояснительной записки отражает этапы выполнения курсового проекта.

# **Глава 1. Аналитический обзор литературы и постановка задачи**

Для полного понимания постановок задач к курсовому проекту, изучим и проанализируем существующие прототипы программных средств заданной темы.

# **1.1 Анализ прототипов**

ШАТЕ-М(shate-mag.by) — платформа, предоставляющая широкий выбор автомобильных запчастей, поставщиков и аксессуаров для автомобилей. Является крупнейшим оптовым центром автозапчастей на белорусском рынке (рисунок 1.1).

Преимущество:

* огромный выбор товаров;
* большой выбор поставщиков;
* имеется доставка;
* есть пункты выдачи заказов.

Недостатки: не выявлены

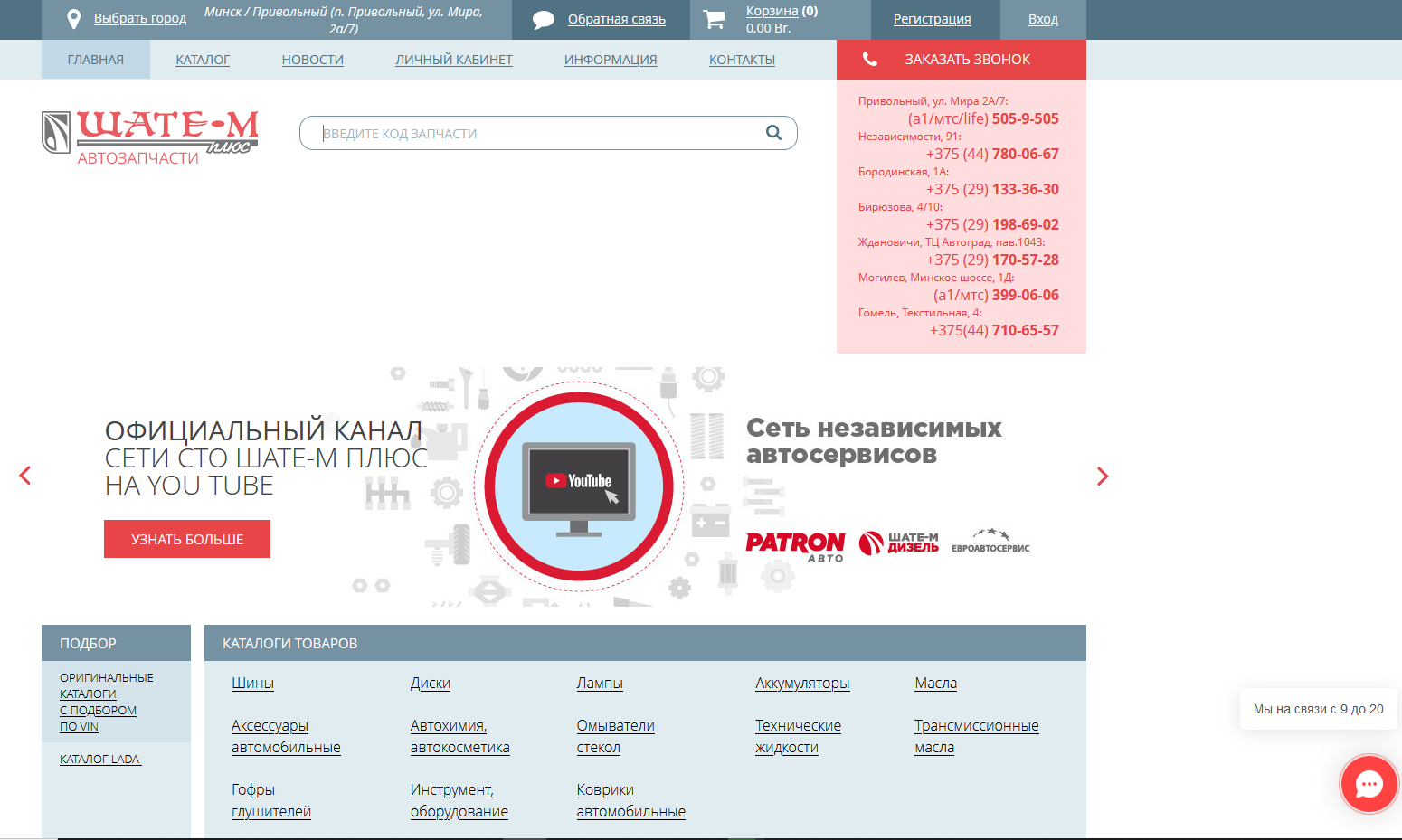


Рисунок 1.1 – Интерфейс веб-приложения «shate-mag.by»

Osaka.by — сервис для покупки автомобильных дисков и различные приспособления для тюнинга. Большая часть товаров японского происхождения или связанные с автомобильной культуры Японии (рисунок 1.2).

Преимущество:

* широкий выбор уникальных товаров;
* доставка;
* скидки постоянным клиентам;
* приятный дизайн.

Недостатки:

* для Windows существует только веб-версия;

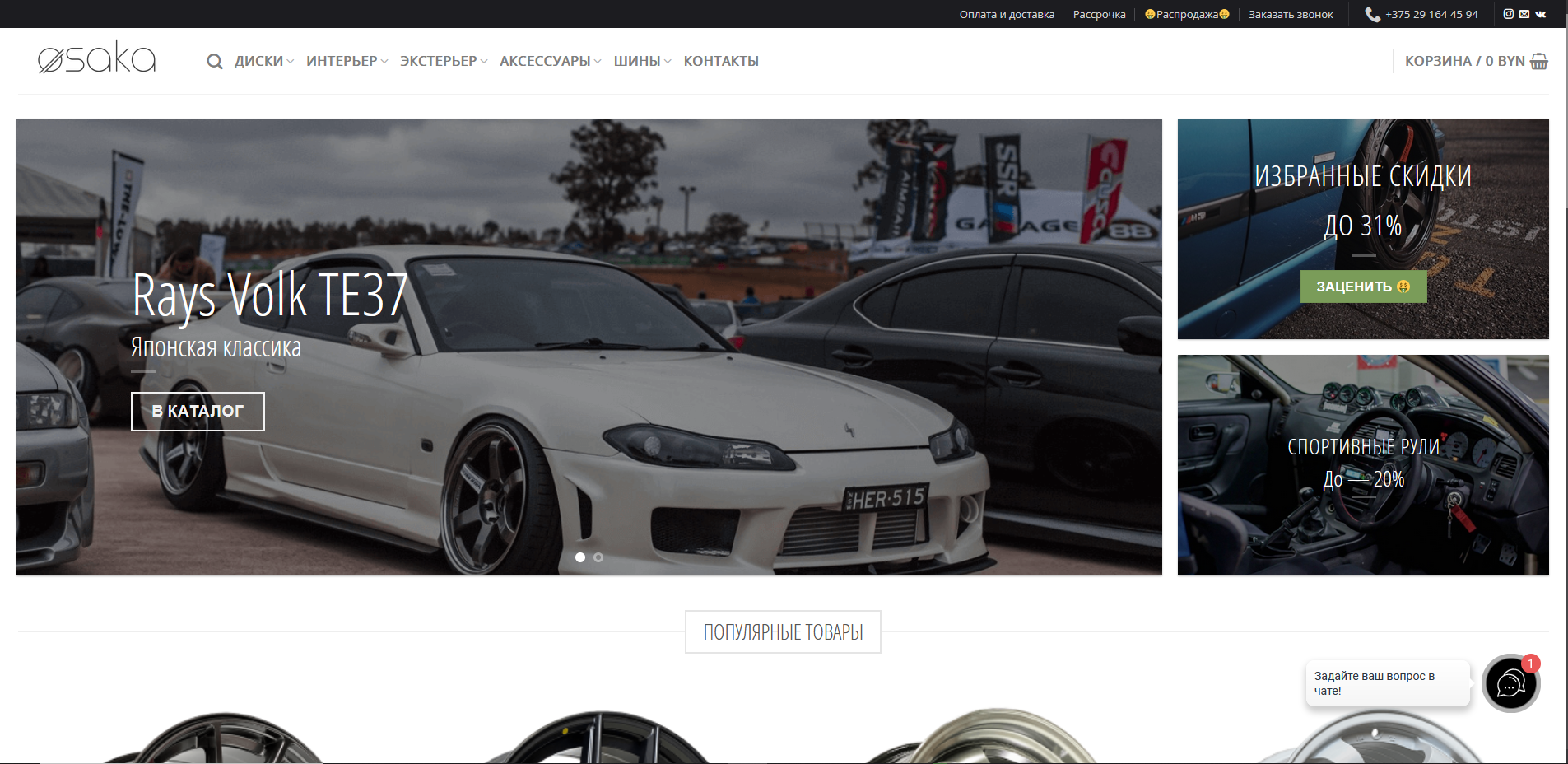


Рисунок 1.2 – Интерфейс веб-приложения «Osaka.by»

Auto1.by – сервис предоставляющий широкий выбор автозапчастей и услуги по обслуживанию автомобилей. Основан компанией «АвтоСпейс» – одним из крупнейших поставщиков запчастей для легковых автомобилей и легких коммерческих грузовиков в Республике Беларусь (рисунок 1.3).

Преимущество:

* современный дизайн;
* огромный выбор розничных магазинов;
* огромный выбор товаров;
* множество точек продаж;
* поиск по номеру необходимой запчасти;
* каталог по VIN номеру автомобиля;
* каталог по номеру на подкапотной табличке;
* рекомендации по подбору автозапчастей к каждой категории.

Недостатки: не выявлены

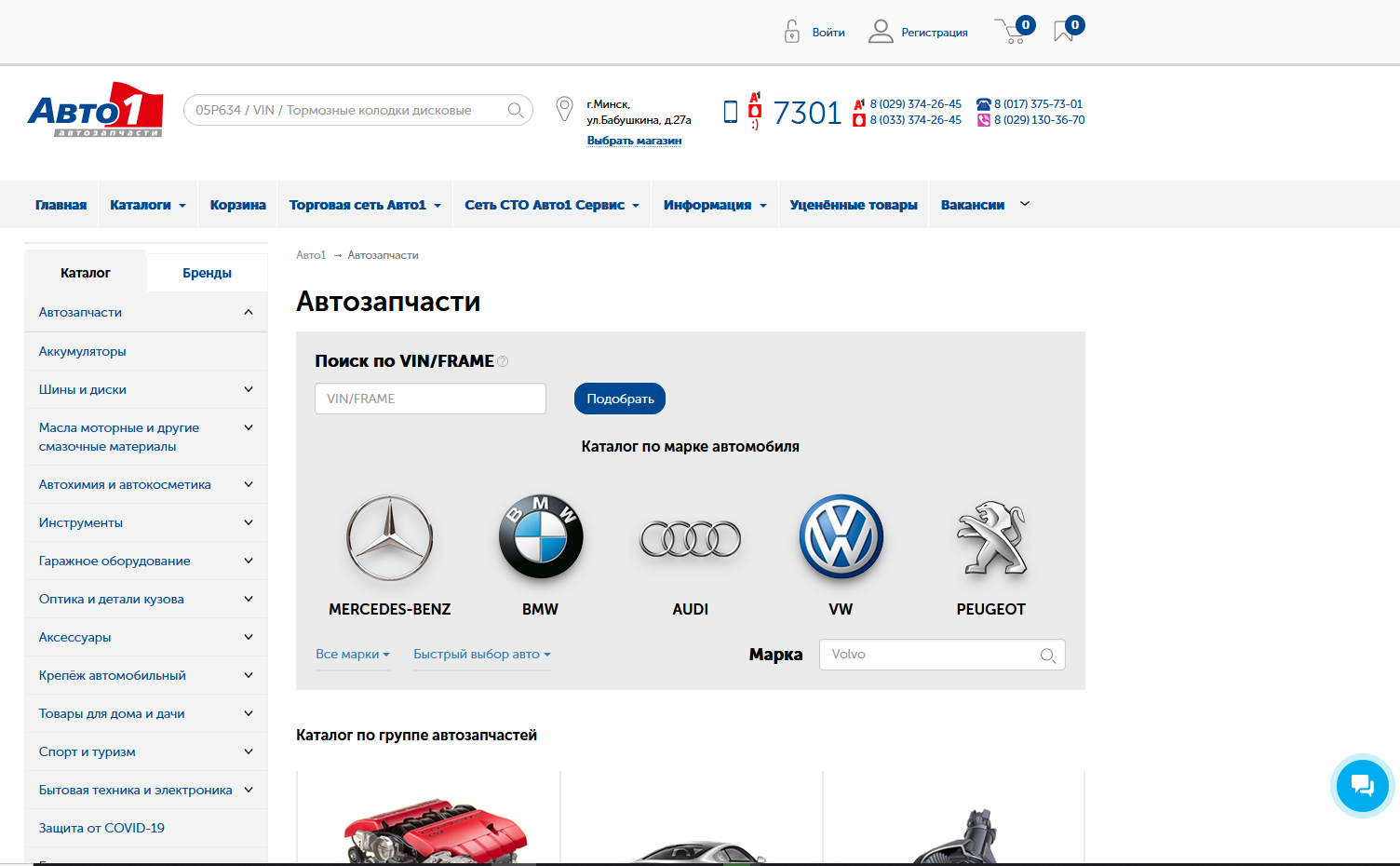


Рисунок 1.3 – Интерфейс веб-приложения «auto1.by»

# **1.2 Постановка целей**

Изучив и проанализировав прототипы, были выделены основная цель проекта и функциональные требования проекта.

Основная цель курсового проекта является разработка приложения, позволяющего выполнять авторизацию и регистрацию пользователей. Администратор может добавлять товары с информацией о них с возможностью удаления. Пользователь в лице клиента просматривать все продукты, добавлять их в корзину, удалять и оформлять заказ с последующей возможностью отмены.

Функциональные требования описаны в главе 2.

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

Сформулируем функциональные требования к программному средству для этапа разработки проекта.

# **2.1 Определение требований к программному средству**

Программное средство должно выполнять следующие цели:

* регистрация и авторизация пользователей;
* сохранение данных в централизованной базе данных;
* администратор может добавлять товар с возможность его удаления;
* администратор может просматривать всн имеющиеся товары и пользователей с возможность удаления;
* клиент может добавлять товар в корзину с возможность удаления;
* клиент может заказать товары из корзины и подтвердит их;
* отправка кода подтверждения заказа посредством отправки электронного письма.

# **2.2 Описание средств разработки**

При разработке приложения были использованы:

* интегрированная среда разработки Microsoft Visual Studio 2022;
* программная платформа .NET Framework 4.7.2;
* язык программирования C#;
* расширяемый язык разметки XAML;
* технология WPF;
* технология Entity Framework 6.4.4;
* Microsoft SQL Server 2019.

В качестве интерфейса прикладного программирования был выбран обширный API-интерфейс – Windows Presentation Foundation (WPF), предназначенный для создания настольных программ с графически насыщенным пользовательским интерфейсом. Мощная инфраструктура основана на DirectX, которая является основой для графической технологии WPF и позволяет использовать аппаратное ускорение графики. Это является существенным различием от своего предшественника создания пользовательских интерфейсов – Windows Forms.

Для работы с WPF использовался объектно-ориентированный язык программирования с С-подобным синтаксисом – С#, разработанный для создания приложений на платформе Microsoft .NET Framework.

Для получения связи между базой даннох и приложением на C# нужен посредник. Моим посредником является технология Entity Framework. Он предоставляет собой объектно-ориентированную технологию доступа к данным является object-relational mapping (ORM) решением для платформы .NET Framework. Entity Framework дает возможность взаимодействия с объектами посредством LINQ to Entities. Основная концепция Entity Framework является понятие сущности или entity. Сущность представляет набор данных, ассоциированных с определенным объектом. Следовательно, мы работаем с объектами и их наборами, а не с таблицами.

# **2.3 Описание функциональности программного средства**

Описание функциональности программного средства представлено с помощью UML-диаграммы вариантов использования показано в приложение А.

Спецификация функциональности программного средства представлена в следующем пункте.

# **2.4 Спецификация функциональных требований**

Для работоспособности программного средства необходимо создание базы данных для хранения информации приложения. Подробно о базе данных описано в главе 3.

При запуске программного средства необходимо реализовать регистрацию и авторизацию пользователей для последующего использования приложения. Для входа в приложения нам нужно ввести логин и пароль пользователя, которые должно быть в базе данных. Иначе пользователь должен зарегистрироваться. Для регистрации необходимо ввести желаемый логин, пароль, адрес электронную почту, имя и фамилию. Если данные прошли валидацию, то данные заносятся в базу данных.

При успешной авторизации, пользователь окажется на странице «Главная». На ней можно просмотреть доступные товары, осуществить поиск товара путем ввода имени товара или по выбору категории товара. На странице «Поиск» пользователь может выполнять поиск по названию, подробный поиск в заданном диапазоне цены и выбранной категории товара. На странице «Корзина» пользователь может увидеть выбранные им товары для заказа. При желании пользователь может удалять товары из корзины, если он изменил свое решение. На странице «Настройки» пользователь имеет возможность выбирать язык интерфейса. На странице «О нас» пользователь может найти ответы на часто задаваемые вопросы и контакты для обратной связи. На странице «Профиль» пользователь может увидеть данные о своем аккаунте, просмотреть историю заказов, привязать или отвязать карту, пополнить баланс счета.

При авторизации администратора на странице «Пользователи» он может ознакомиться со списком всех пользователей, а так же изменять их имя, фамилию, электронную почту или удалять. На странице «Заказы» администратор может ознакомиться со списком всех заказов. На странице «Поставщики» администратор имеет право просматривать список всех поставщиков, добавлять и удалять поставщиков. На странице «Запчасти» администратор может увидеть все товары, добавлять и удалять запчасти. На странице «Доставки» администратор может просмотреть список всех доставщиков, добавлять и удалять. На странице «Бренды» администратор имеет право просматривать список всех брендов, добавлять и удалять. На странице «Категории» администратор может ознакомиться со списком всех категорий, добавлять и удалять. На странице «Карты» администратор в праве просматривать список привязанных карт и удалять их. На странице «Изображения» администратор в праве добавлять изображения, удалять и ознакомиться со списком изображений.

Выход из приложения должен возвращать пользователя к окну авторизации.

Вся информация о запчастях, пользователях и заказах обязаны заноситься в базу данных в соответствующие таблицы.

# **Глава 3. Проектирование и создание программного средства**

Проектирование программного средства — процесс создания проекта программного обеспечения. Целью проектирования является определение внутренних свойств системы и детализации её внешних свойств на основе исходных условий задача. Исходные условия уже были сформулированы во втором разделе данной пояснительной записки. Этап проектирования означает их анализ.

# **3.1 Архитектура системы**

В данном программном средстве используется архитектурный паттерн MVVM (MODEL-VIEW-VIEWMODEL). Он позволяет разделить логику приложения от визуальной части. MVVM состоит из трех частей:

* модель описывает используемые в приложении данные;
* представление определяет визуальный интерфейс, через который пользователь взаимодействуют с приложением;
* модель представления соединяет модель и представление через механизм привязки данных. ViewModel также имеет логику по получению данных из модели, которые потом передаются в представление.

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



Рисунок 3.1 – Структура шаблона MVVM

# **3.2 Диаграммы UML**

UML — унифицированный язык моделирования (Unified Modeling Language) — это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования.

Имеет возможность для визуализации, спецификации, конструирования и документирования программных средств.

Диаграмма — это графическое представление набора элементов, чаще всего изображенного в виде связного графа вершин (сущностей) и путей (связей).

Диаграммы последовательностей используются для уточнения диаграмм прецедентов, более детального описания логики сценариев использования.

С точки зрения сценариев использования это очень хорошее средство документирования проекта.

Диаграмма последовательности взаимодействия с окном авторизации приведена в приложении Б.

Диаграмма деятельности, как и диаграмма состояний, показывает динамические аспекты поведения системы. На практике, эта диаграмма представляет собой блок-схему, которая наглядно изображает, как поток управления переходит от одной деятельности к другой.

Диаграммы деятельности применяются при моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений.

Часть диаграммы деятельности данного приложения приведена в приложении В, данная диаграмма приведена для части программы связанной со входом пользователя в приложение. Она разбита на 2 части: авторизация и регистрация, для каждой из этих частей приведен свой алгоритм, осуществляющий все проверки и этапы обработки информации необходимые для корректной работы данной части приложения.

Диаграммы использования показывает актеров и возможные прецеденты, при работе с приложением. Диаграмма использования приведена в приложении Г.

Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования.

Диаграмма классов ViewModel представлена на рисунке 3.2.

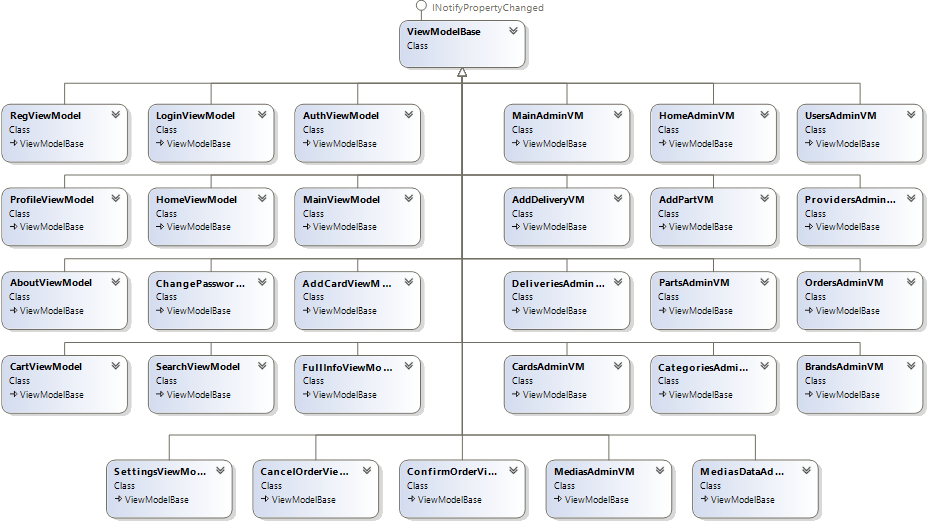


Рисунок 3.2 – Диаграмма классов

# **3.3 Проектирование логической структуры базы данных**

Для реализации поставленной цели была создана база данных. Для её создания использовалась система управления реляционными базами данных Microsoft SQL Server 2019.

База данных — совокупность данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования данными, независимо от прикладных программ.

Система управления базами данных — совокупность программ и языковых средств, предназначенных для управления данными в базе данных, ведения базы данных и обеспечения взаимодействия ее с прикладными программами. База данных нашей программной подсистемы состоит из 11 таблиц, ее схема изображена на рисунке 3.3.

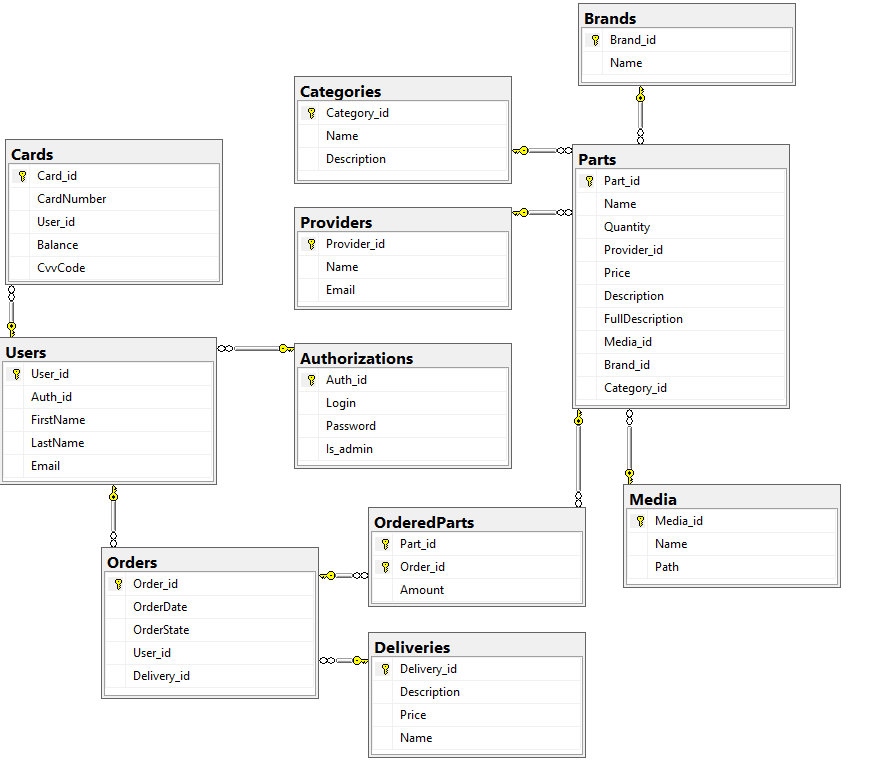


Рисунок 3.3 – Логическая модель базы данных

Таблица Authorizations используется для хранения данных об авторизации. Включает 4 столбца: Auth\_id — id авторизации, Login — логин пользователя, Password — пароль пользователя, при записи в базу данных хэшируется, Is\_admin — 0(False) — пользователь, 1(True) - администратор.

Таблица Users используется для хранения данных о пользователях. Включает 5 столбцов: User\_id — id пользователя, Auth\_id — id авторизации, FirstName — имя пользователя, LastName — фамилия пользователя, Email — адрес электронной почты.

Таблица Cards используется для хранения данных о привязанной карте для оплаты. Состоит из 5 столбцов: Card\_id — id карты, CardNumber — номер карты, User\_id — id пользователя, Balance — баланс карты, CvvCode — код cvv карты.

Таблица Providers используется для хранения данных о поставщиках запчастей. Включает 3 столбца: Provider\_id — id поставщика, Name — имя поставщика, Email — адрес электронной почты поставщика.

Таблица Parts используется для хранения запчастей. Включает 10 столбцов: Part\_id — id продукта, Name — название продукта, Quantity — количество товара, доступного для покупки, Provider\_id — id поставщика продукт, Price — цена покупки товара, Description – краткое описание товара, FullDescription — полное описание товара, Media\_id — id изображение товара, Brand\_id — id марки бренда, к которому принадлежит запчасть, Category\_id — id категории товара.

Таблица Orders используется для хранения информации о заказах. Включает 5 столбцов: Order\_id — id заказа, OrderDate — дата открытия заказа, OrderState — статус заказа, User\_id — id пользователя, который сделал заказ, Delivery\_id — id службы доставки.

Таблица OrderedParts используется для связи между таблицами Orders и Parts. Используется для определения того, какие товары и какое их количество входит в заказ. Состоит из 3 столбцов: Part\_id — id товара, Order\_id — id заказа, Amount — количество единиц определенного товара в заказе.

Таблица Brands используется для хранения данных о брендах. Включает 2 столбца: Brand\_id — id марки автомобиля, Name — наименование марки автомобиля.

Таблица Deliveries используется для хранения информации о службах доставки. Состоит из 4 столбцов: Delivery\_id — id службы доставки, Description — описание службы доставки, Price — цена услуги доставки, Name — наименование службы доставки.

Таблица Categories хранит информацию о категориях запчастях. Состоит из 3 столбцов: Category\_id — id категории, Name — имя категории, Description — описание категории.

Таблица Media используется для хранения информации об изображении запчастей. Состоит из 3 столбцов: Media\_id — id медиа, Name — имя изображения, Path — путь к изображению.

# **3.4 Структура проекта**

Структура проекта представлена на рисунке 3.4.

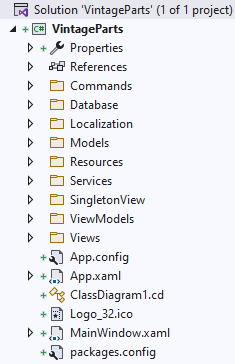


Рисунок 3.4 – Структрура проекта

Папка «Database» содержит классы для работы с базой данных.

Папка «Models» содержит классы моделей.

Папка «Localization» содержит класс для работы с локализацией приложения.

Папка «Resources» содержит стили и шаблоны элементов управления, а так же каталог с «Images» в котором содержатся все необходимые графические элементы.

Папка «Services» содержит классы для поддержки некоторых функций приложения.

Папка «ViewModels» содержит классы моделей представлений.

Папка «Views» содержит все представления. Программное средство включает в себя 7 окон и 24 пользовательских элементов управления, выступающих в качестве страниц. Связи между ними представлены в приложении Д.

# **Глава 4. Реализация программного средства**

# **4.1 Реализация MVVM**

Паттерн MVVM реализуется через класс ViewModelBase, который реализует интерфейс INotifyPropertyChanged и от которого наследуются все страницы. Для реализации паттерна файлы программы были распределены по соответствующим пространствам имен (см. 3.2) и реализованы следующие функции. На рисунке 4.1 замечаем, что класс MainViewModel наследует класс ViewModelBase (рис. 4.2).

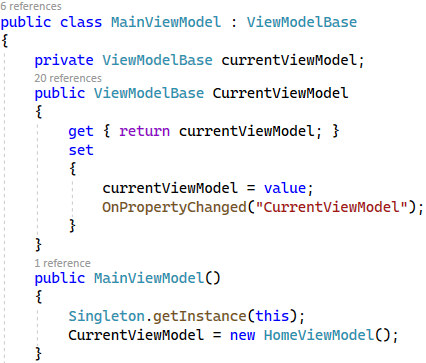


Рисунок 4.1 – Пример использования ViewModelBase

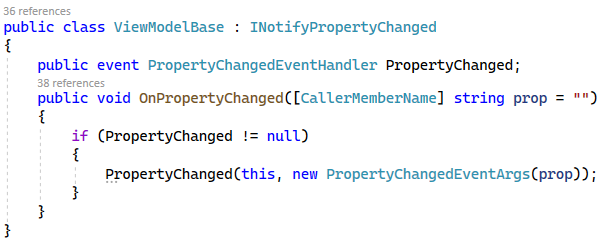


Рисунок 4.2 – Структура класса ViewModelBase

В программном средстве используется паттерн Command который позволяет инкапсулировать запрос на выполнение определенного действия в виде отдельного объекта. В WPF команды представлены интерфейсом ICommand. Класс реализует два метода:

* CanExecute: определяет, может ли команда выполняться;
* Execute: собственно, выполняет логику команды.

В программном средстве он представлен в виде собственной команды Command код которой приведен в рисунке 4.3.



Рисунок 4.3 – Структура класс Command

# **4.2 Реализация авторизации и регистрации пользователей**

Для того чтобы пользоваться приложением каждый пользователь должен войти в систему, предварительно зарегистрировавшись. Реализация представлена в приложении Е.

Для безопасности хранения информации используется хэширование в данном случае применяется встроенный класс MD5. Код метода хэширования приведен на рисунке 4.4.

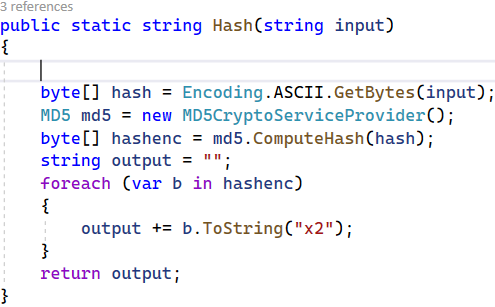


Рисунок 4.4 – Пример метода обеспечивающего хэширование пароля

При регистрации пользователю задается тип аккаунта user. Если все данные были введены верно и логин пользователя, почта еще не заняты, то осуществляется регистрация и переход на страницу авторизации.

Данные при регистрации проходят валидацию. Валидация реализована с использованием регулярных выражений в аннотациях к полям модели авторизации и пользователя. Пример кода реализующего валидацию приведен на рисунке 4.5 и 4.6.

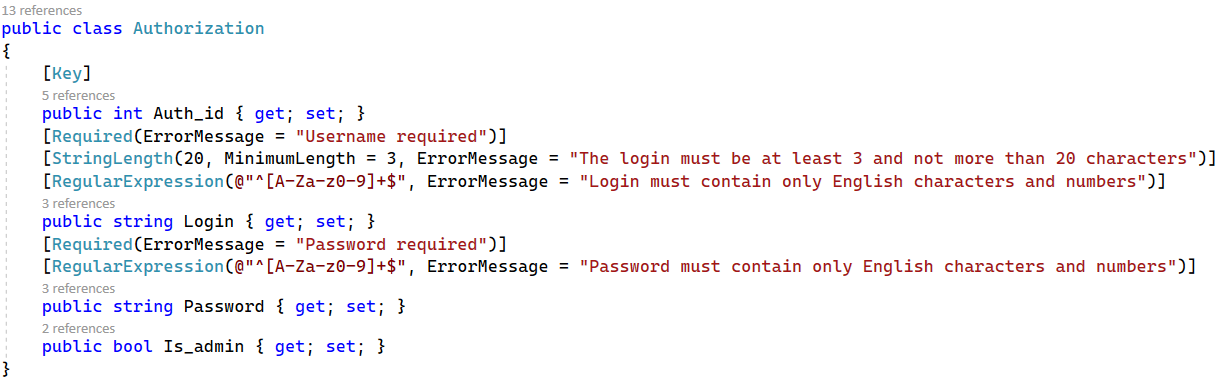


Рисунок 4.5 – Пример валидации с использованием аннотаций в авторизации

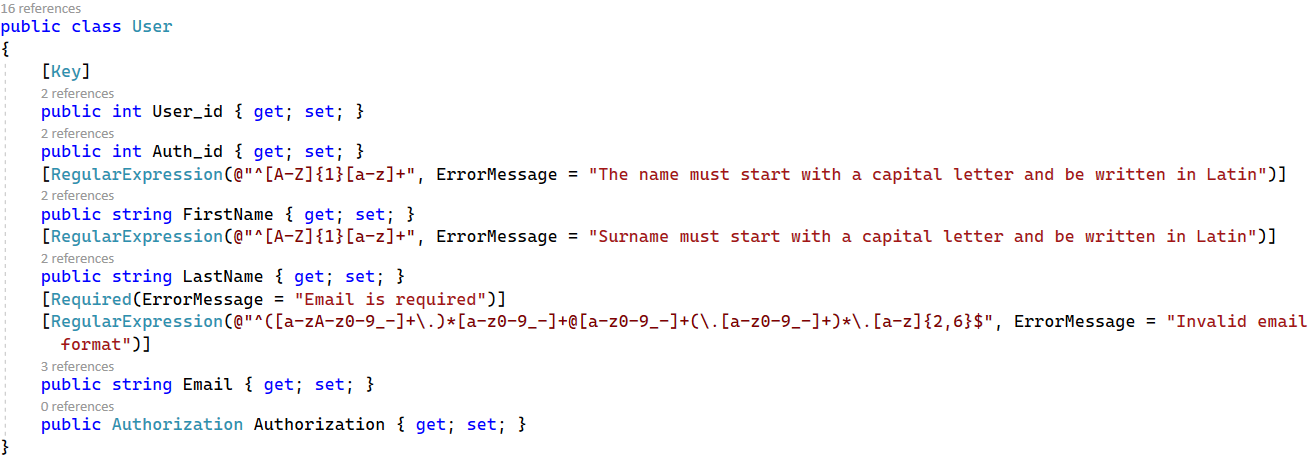


Рисунок 4.6 – Пример валидации с использованием аннотаций в пользователях

Авторизация проверяет есть ли пользователь с такими данными и проверяется значения хэш-функций паролей. При неудаче появляется надпись с ошибкой. Иначе создается окно пользователя или администратора в зависимости от значения в поле Is\_admin в базе данных. Реализация представлена в приложении Ж.

# **4.3 Добавление и удаления товара из корзины**

Добавление товара в корзину пользователя производится с помощью команды привязанной на странице полной информации о товаре FullInfoView. Пример команды, реализующей добавление товара в корзину пользователя приведен на рисунке 4.7.



Рисунок 4.7 – Команда, реализующая добавление товара в корзину

Удаление товара из корзины пользователя производится с помощью команды привязанной на странице корзины пользователя. Команда, реализующее удаление приведен на рисунке 4.8.

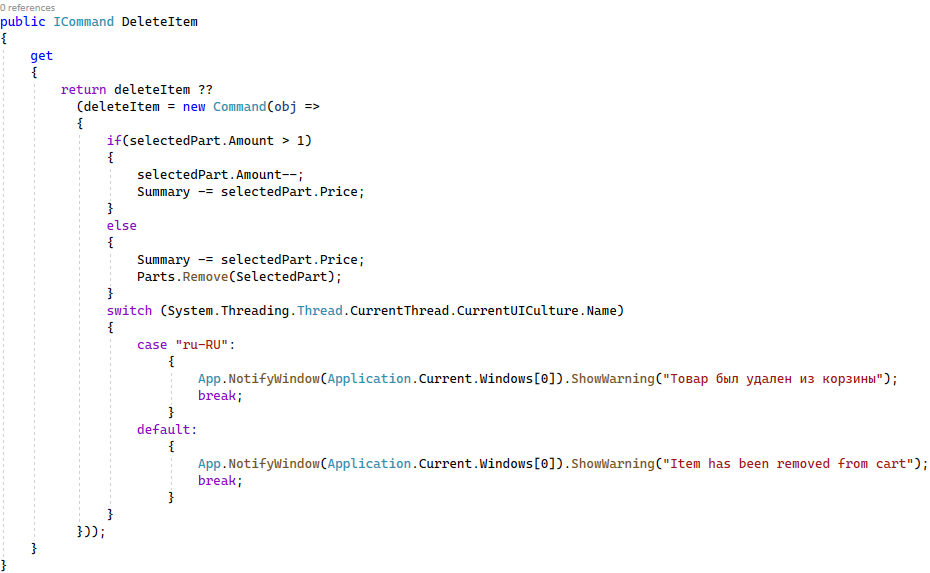


Рисунок 4.8 – Команда, реализующая удаление товара из корзины

# **4.4 Отображение оповещений внутри программного средства**

Для отображения оповещений внутри программного средства используется библиотека ToastNotifications, которая предоставляет класс Notification и методы для отображения оповещений различного типа. В классе App реализован статический метод, который возвращает объект типа Notifier. Пример метода приведен на рисунке 4.9.

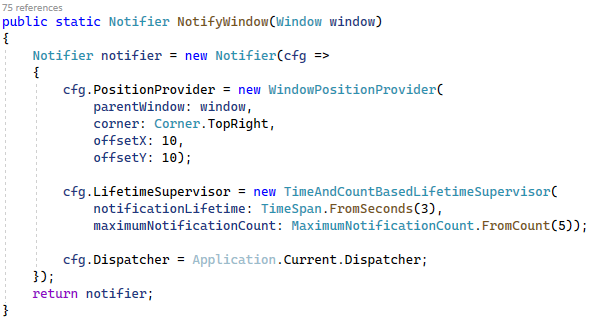


Рисунок 4.9 – Метод предоставляющий объект уведомлений

В объекте уведомлений задаются такие параметры как:

− окно родитель;

− позиция отображения уведомления;

− отступы от краев окна, в котором отображается уведомление;

− время существования уведомления;

− максимальное количество отображаемых уведомлений.

# **4.5 Отправка сообщений на почту и генерация чека**

Для отправки сообщений на электронную почту пользователей используются классы, предоставленные пространством имен System.Net.Mail. Данное пространство имен предоставляет классы для работы с SMTP протоколом.

В качестве параметров в первом случае передается адрес электронной почты пользователя, код, тема и тело сообщения, а во втором – адрес электронной почты, тема и тело сообщения.

Примеры отправки кода подтверждения и чека заказа приведены на рисунках 4.10 и 4.11.



Рисунок 4.10 – Метод асинхронной отправки кода подтверждения



Рисунок 4.11 – Метод асинхронной отправки чека заказа

Для создания чека используется статический метод GenerateTicket, в котором используется объединение нескольких таблиц для получение наиболее детальной информации о заказе и товарах в нем. В качестве параметра в метод передается объект заказа. Реализация представлена в приложении И.

# **4.6 Формирование, подтверждение и отмена заказа**

Заказ формируется с помощью метода AddOrder в который передается коллекция объектов из корзины. В данном методе так же проводится проверка на наличие средств на счете, наличие выбранной доставки и наличие достаточного количества товаров на складе для оформления заказа.

Так как таблицы Order и Parts связаны как один-ко-многим, то в методе формируется новая коллекция объектов промежуточной таблицы, которая прикрепляется к заказу. Реализация представлена в приложении К.

Подтверждение заказа осуществляется с помощью команды SubmitCode, которая получает код из поля представления и сравнивает его с кодом, сгенерированным приложением. Реализация представлена в приложении Л.

Отмена заказа осуществляется командой CancelOrder, которая получает код из поля представления и сравнивает его с кодом, сгенерированным приложением. В ходе работы заказ получает метку отмены, товары возвращаются в базу данных, а деньги возвращаются на счет пользователя. Реализация представлена в приложении М.

# **Глава 5. Тестирование программного средства**

В курсовом проекте задействуется обработка ошибок, таким образом, что пользователь будет уведомлен о неудачном выполнении запроса к базе данных, , или недоступности данных в формах. Присутствуют различные элементы, всплывающие окна уведомляющие пользователя.

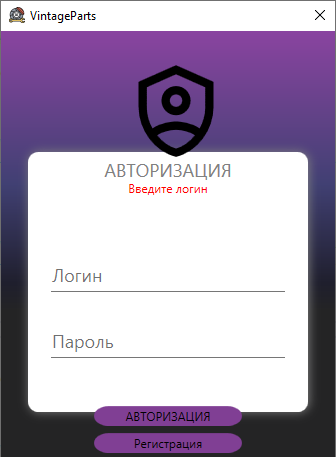


Рисунок 5.1 – Отсутствие введенного логина и пароля

Введен правильный логин, но без пароля. Введен пароль, но без логина.

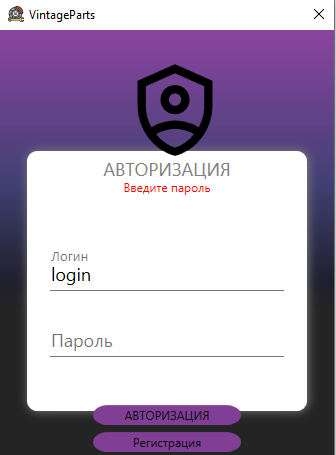


Рисунок 5.2 – Отсутствие введенного пароля

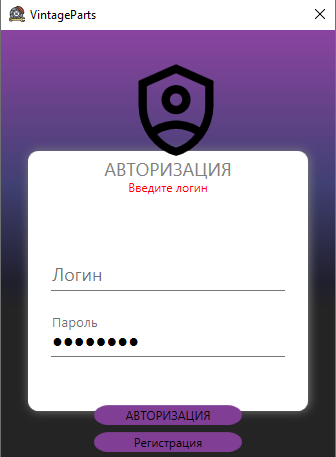


Рисунок 5.3 – Отсутствие введенного логина

Введен правильный пароль, но неправильный логин. Во втором слишком короткий логин при регистрации.

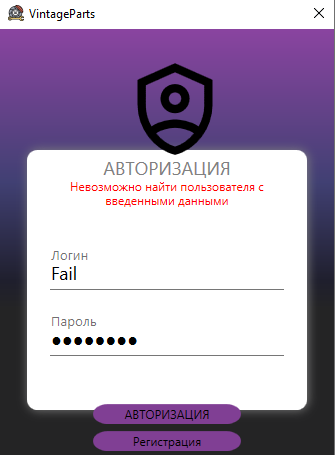


Рисунок 5.4 – Неверный логин

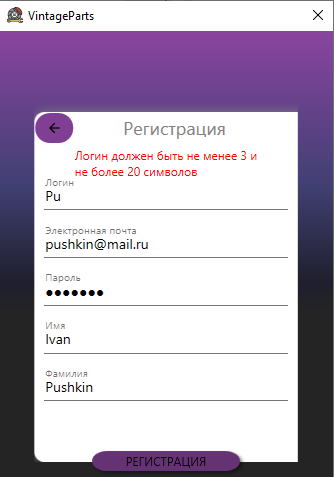


Рисунок 5.5 – Логин слишком короткий

Все ввели, кроме имени. Во втором пользователь с такими данными уже создан.

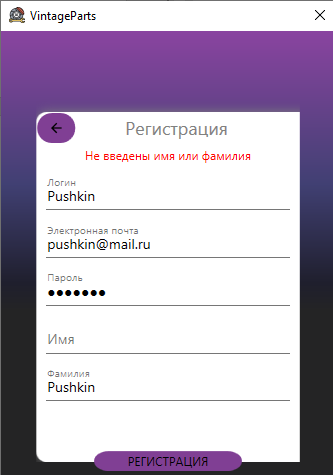


Рисунок 5.6 – Отсутствие имени

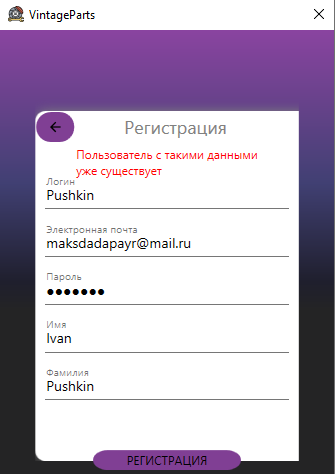


Рисунок 5.7 – Введенная почта уже зарегистрирована

Введен логин на кириллице. Попытка заказать больше запчастей, чем есть на складе.

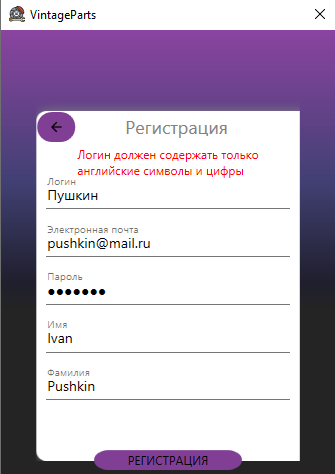


Рисунок 5.8 – Логин на кириллице

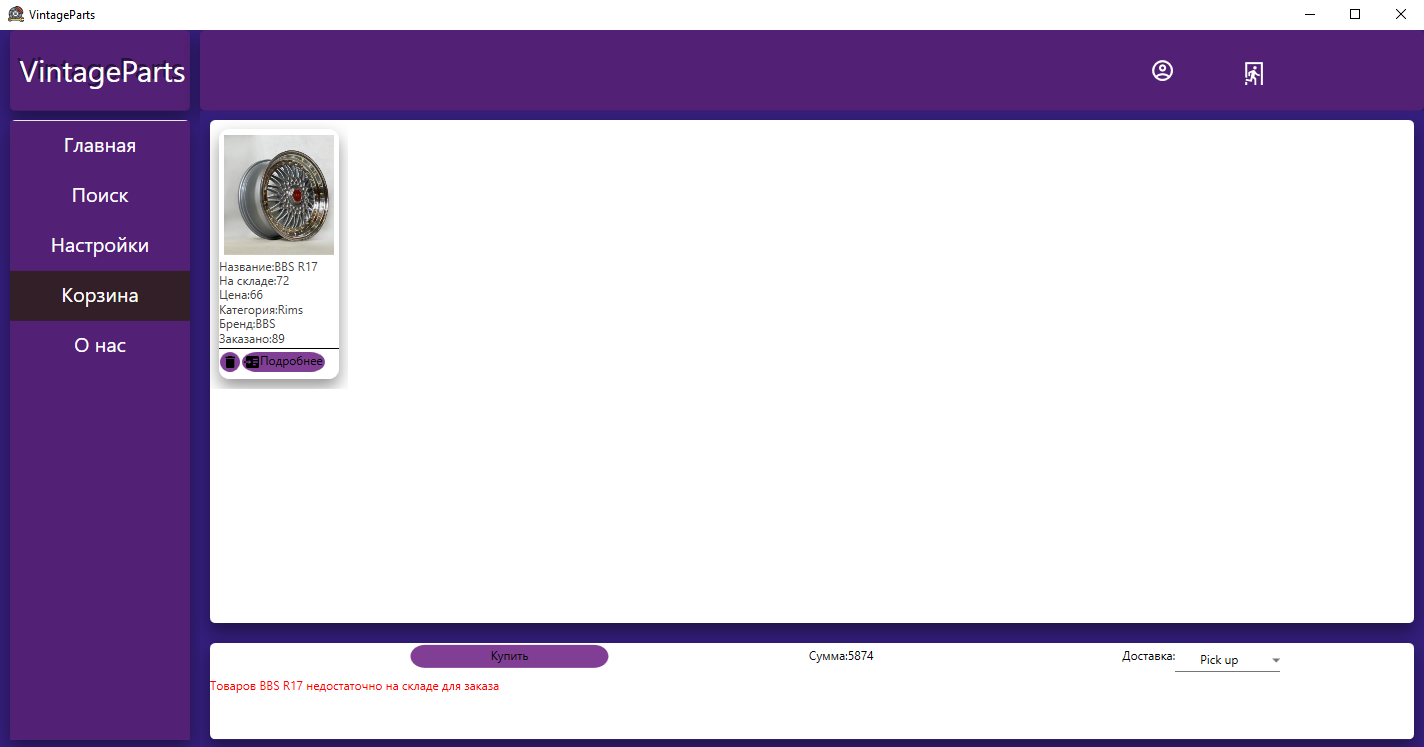


Рисунок 5.9 – Попытка заказать больше запчастей, чем есть на складе

# **Глава 6. Руководство программного средства по использованию**

# **6.1 Регистрация и авторизация**

После запуска приложения пользователь видит главное окно приложения, на котором он может авторизоваться. Окно представлено на рисунке 6.1.

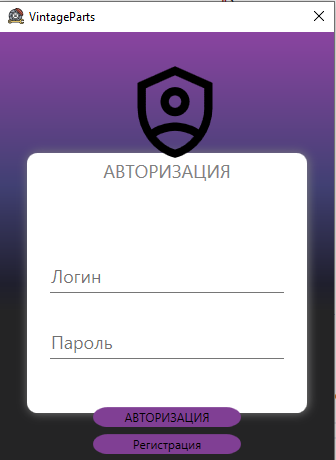


Рисунок 6.1 – Окно входа в аккаунт

Если у пользователя нет своего аккаунта, то по нажатию на соответствующую кнопку можно перейти к окну регистрации. Пользователь должен придумать логин и пароль, которые будет использовать для авторизации в приложении. Окно регистрации показано на рисунке 6.2.

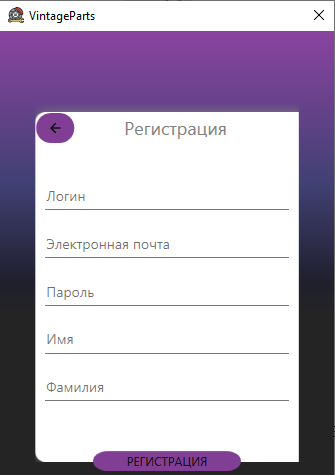


Рисунок 6.2 – Окно регистрации

# **6.2 Использование программного средства обычным пользователям**

После успешной авторизации пользователь попадает на главный экран, где находятся все товары магазина, а так же предоставлен поиск по названию товара и по выбранной категории. Так же на главном окне имеется 7 кнопок для навигации по страницам и выхода из приложения.

Стартовое окно пользователя приведено на рисунке 6.3.

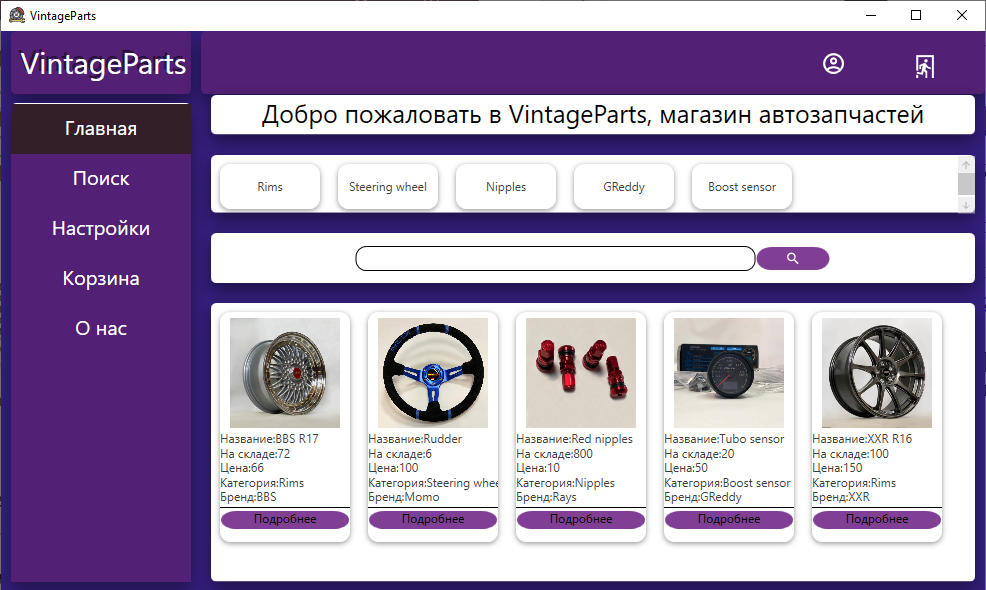


Рисунок 6.3 – Главное окно пользователя

Пользователь в праве просматривать и добавлять товары в корзину, а так же оформлять заказ.

# **6.3 Использование программного средства администратором**

После успешной авторизации администратор попадает в главное окно администратора, где видит 10 для навигации по страницам и выхода их приложения. Администратор имеет возможность добавлять такие объекты как:

− товары;

− доставки;

− категории;

− марки.

Так же администратор имеет возможность вносить изменения данных хранящихся в базе данных.

Пример главного окна администратора приведен на рисунке 6.4.

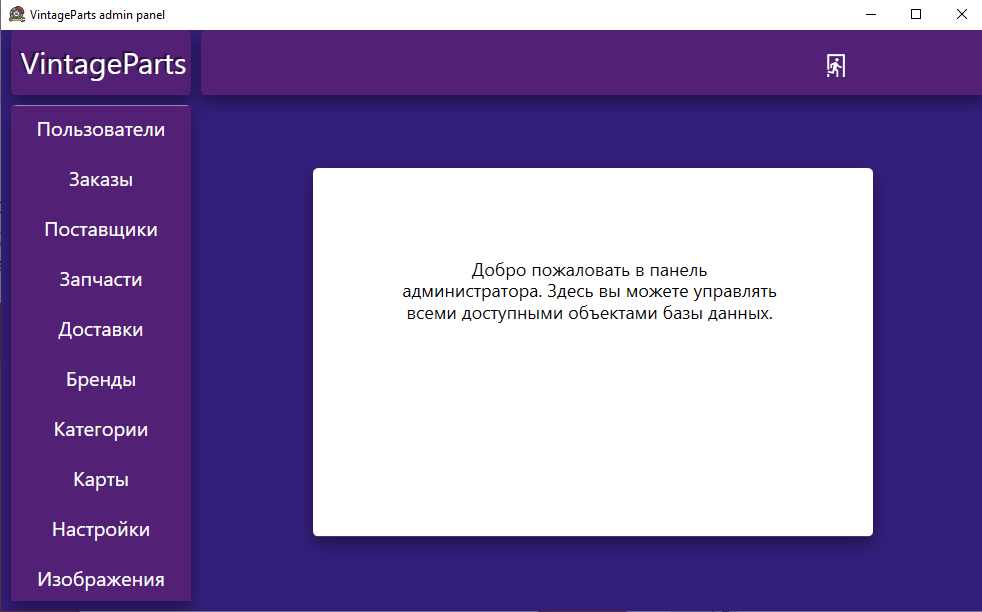


Рисунок 6.4 – Главное окно администратора

# **ЗАКЛЮЧЕНИЕ**

В итоге выполнения курсового проекта было разработано программное средство «Магазин автозапчастей» имеющее название «VintageParts», предоставляющее возможность добавлять товары в корзину, а так же косвенно покупать товары, находящиеся в данной корзине. Так же приложение поддерживает отправку кода подтверждения по электронной почте и смену языка. Основные достоинства – простота использования, оригинальный дизайн и гибкий функционал.

В процессе реализации программного средства были достигнуты выше указанные требования и задачи. При выполнении курсового проекта использовались принципы и приемы ООП.

Приложение отлично работает с базой данных и не нарушает работу и целостность базы дынных.

Для разработки дизайна использовалась библиотека MaterialDesign, которая имеет огромное количество красивых и удобных элементов управления. Также использовалась библиотека ToastNotifications, которая предоставляет красивые и удобные оповещения.

Приложение было успешно протестировано, что означает пригодность этого приложения для своей цели.

Также в процессе выполнения данного курсового проекта были закреплены навыки в программировании на языке C#, создании приложений на WPF, использование Entity Framework 6, работа с современным паттерном MVVM, проектирование базы данных и реализация их в СУБД MS SQL Server 2019.

Был получен опыт взаимодействия со сторонними библиотеками, изучения их и внедрения в свой проект.

# **СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

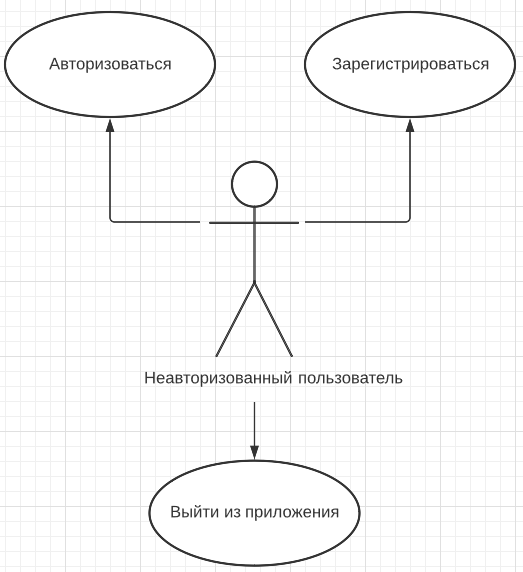
1) MSDN сеть разработчиков в Microsoft [Электронный ресурс] / Режим доступа: https://docs.microsoft.com/ru-ru/

2) WPF | Определение паттерна MVVM [Электронный ресурс] / Режим доступа: https://metanit.com/sharp/wpf/22.1.php

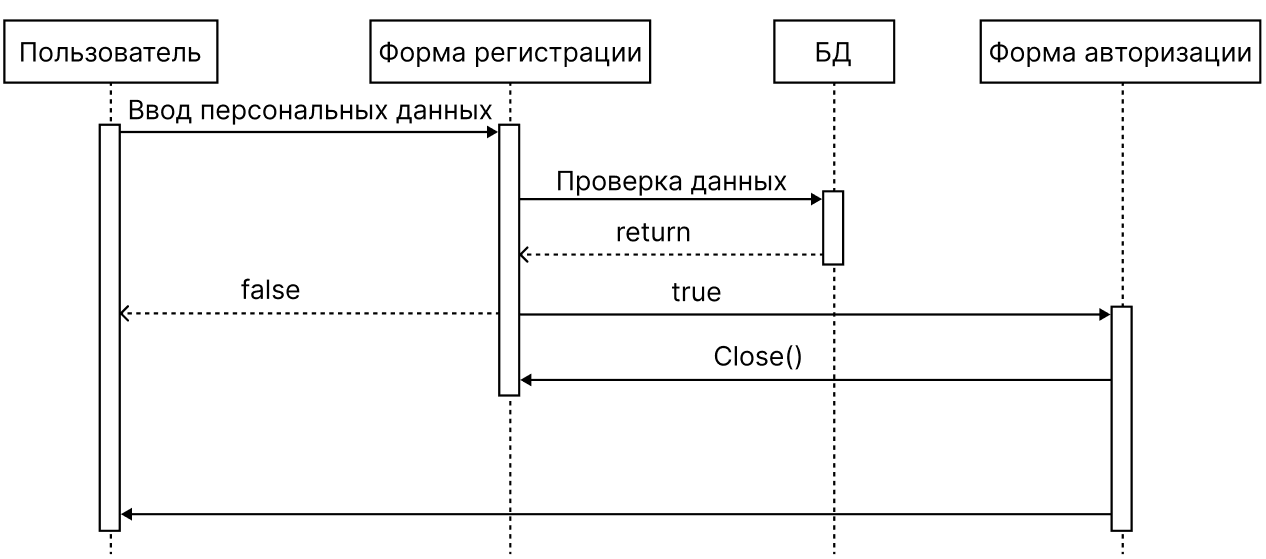
3) Пацей, Н.В. Курс лекций по языку программирования С# / Н.В. Пацей. – Минск: БГТУ, 2021. – 175 с.

4) Entity Framework 6 [Электронный ресурс] / Режим доступа: https://metanit.com/sharp/entityframeworkcore/3.1.php

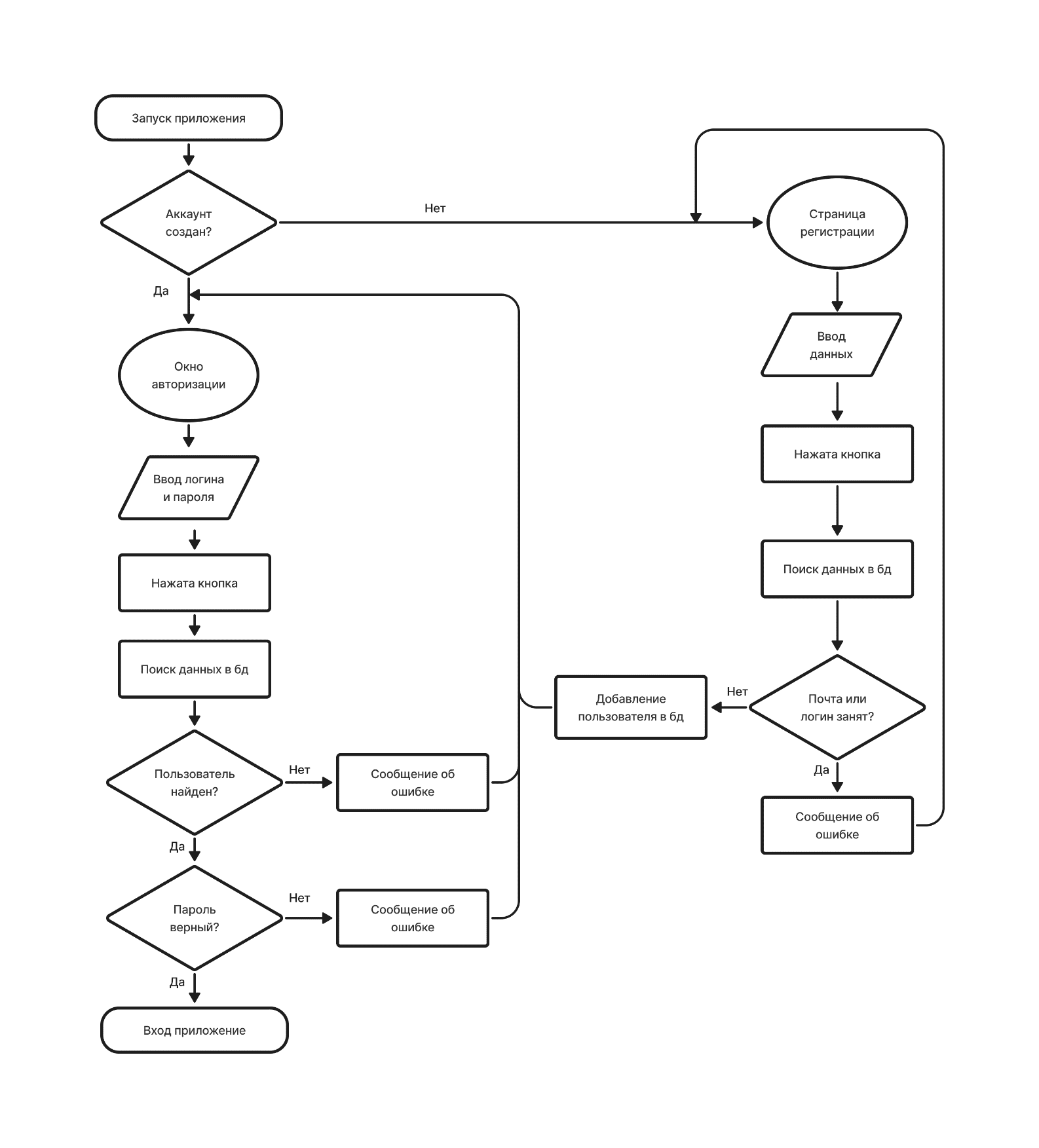
# **Приложение А**



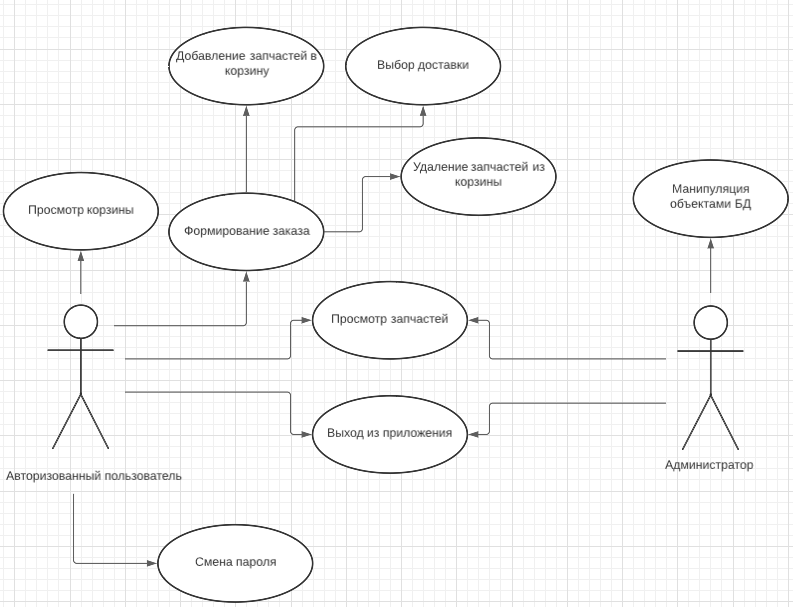
# **Приложение Б**



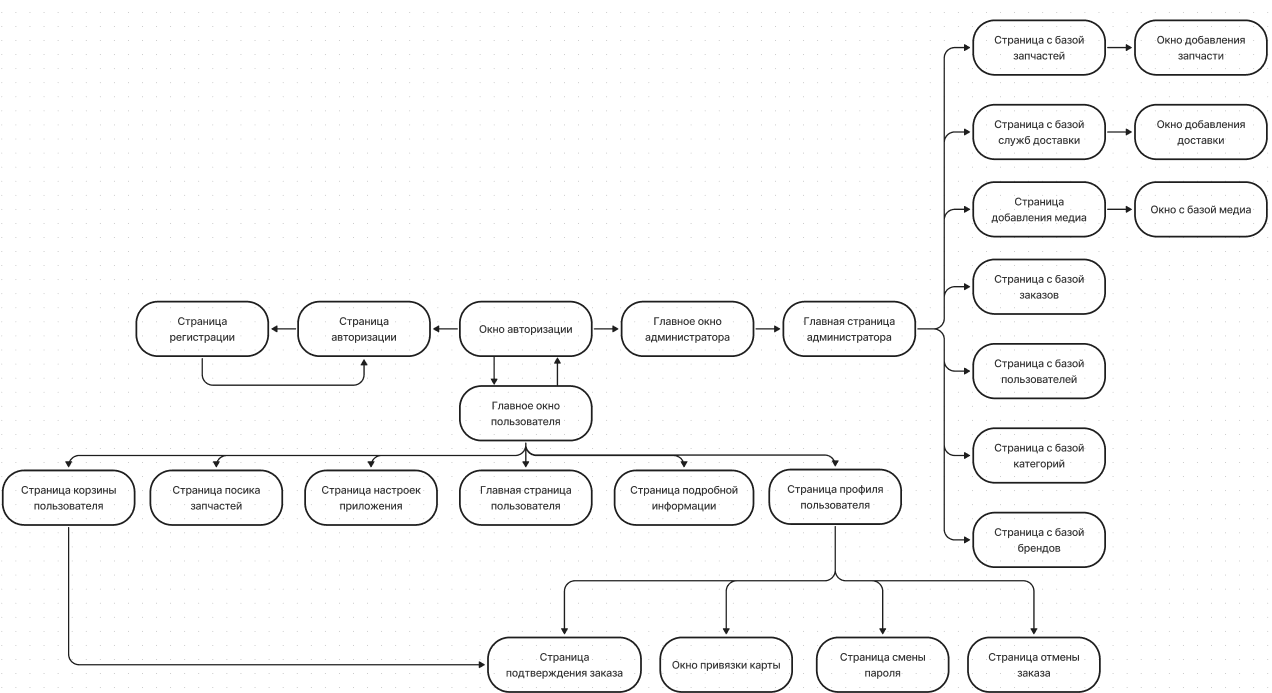
# **Приложение В**



# **Приложение Г**



# **Приложение Д**



# **Приложение Е**

public Command regCommand;

public ICommand RegCommand

{

get

{

return regCommand ??

(regCommand = new Command(obj =>

{

try

{

using (PartShopDbContext db = new PartShopDbContext())

{

Authorization auth = new Authorization();

User user = new User();

if(String.IsNullOrEmpty(login))

{

switch (System.Threading.Thread.CurrentThread.CurrentUICulture.Name)

{

case "ru-RU":

{

throw new Exception("Введите логин");

}

default:

{

throw new Exception("Enter login");

}

}

}

auth.Login = login;

if (String.IsNullOrEmpty(password))

{

switch (System.Threading.Thread.CurrentThread.CurrentUICulture.Name)

{

case "ru-RU":

{

throw new Exception("Введите пароль");

}

default:

{

throw new Exception("Enter password");

}

}

}

Листинг – команда, реализующая регистрацию

if (password != null & password[0] != ' ')

{

if (password.Length < 4 || password.Length > 21)

{

switch (System.Threading.Thread.CurrentThread.CurrentUICulture.Name)

{

case "ru-RU":

{

throw new Exception("Пароль должен быть не менее 4 и не более 20 символов");

}

default:

{

throw new Exception("The password must be at least 4 and not more than 20 characters");

}

}

}

auth.Password = SecurePassService.Hash(password);

}

else

{

switch (System.Threading.Thread.CurrentThread.CurrentUICulture.Name)

{

case "ru-RU":

{

throw new Exception("Неверный формат пароля");

}

default:

{

throw new Exception("Invalid password format");

}

}

}

if (String.IsNullOrEmpty(firstname) || String.IsNullOrEmpty(lastname))

{

switch (System.Threading.Thread.CurrentThread.CurrentUICulture.Name)

{

case "ru-RU":

{

throw new Exception("Не введены имя или фамилия");

}

default:

{

throw new Exception("First or last name not entered");

}

Листинг(продолжение) – команда, реализующая регистрацию

}

}

user.FirstName = firstname;

user.LastName = lastname;

auth.Is\_admin = false;

user.Email = mail;

if (login != null && password != null)

{

if (db.Authorizations.Any(a => a.Login == login) || db.Users.Any(a => a.Email == mail))

{

switch (System.Threading.Thread.CurrentThread.CurrentUICulture.Name)

{

case "ru-RU":

{

throw new Exception("Пользователь с такими данными уже существует");

}

default:

{

throw new Exception("User with such data already exists");

}

}

}

else

{

db.Authorizations.Add(auth);

db.Users.Add(user);

db.SaveChanges();

SingletonAuth.getInstance(null).StartViewModel.CurrentViewModel = new LoginViewModel();

switch (System.Threading.Thread.CurrentThread.CurrentUICulture.Name)

{

case "ru-RU":

{

App.NotifyWindow(Application.Current.Windows[0]).ShowSuccess("Регистрация прошла успешно");

break;

}

default:

{

App.NotifyWindow(Application.Current.Windows[0]).ShowSuccess("Registration completed successfully");

break;

}

}

Листинг(продолжение) – команда, реализующая регистрацию

}

}

}

}

catch(DbEntityValidationException e)

{

foreach(DbEntityValidationResult validationRes in e.EntityValidationErrors)

{

foreach(DbValidationError err in validationRes.ValidationErrors)

{

ErrorMessage = err.ErrorMessage;

}

}

}

catch(Exception e)

{

ErrorMessage = e.Message;

}

}));

}

}

Листинг(продолжение) – команда, реализующая регистрацию

# **Приложение Ж**

public Command authCommand;

public ICommand AuthCommand

{

get

{

return authCommand ??

(authCommand = new Command(obj =>

{

try

{

using (PartShopDbContext db = new PartShopDbContext())

{

if (String.IsNullOrEmpty(login))

{

switch (System.Threading.Thread.CurrentThread.CurrentUICulture.Name)

{

case "ru-RU":

{

throw new Exception("Введите логин");

}

default:

{

throw new Exception("Enter login");

}

}

}

if (String.IsNullOrEmpty(password))

{

switch (System.Threading.Thread.CurrentThread.CurrentUICulture.Name)

{

case "ru-RU":

{

throw new Exception("Введите пароль");

}

default:

{

throw new Exception("Enter password");

}

}

}

if (password.Length < 4 || password.Length > 21)

{

switch (System.Threading.Thread.CurrentThread.CurrentUICulture.Name)

Листинг – команда, реализующая авторизацию

{

case "ru-RU":

{

throw new Exception("Пароль должен быть не менее 4 и не более 20 символов");

}

default:

{

throw new Exception("The password must be at least 4 and not more than 20 characters");

}

}

}

password = SecurePassService.Hash(password);

Authorization authUser = db.Authorizations.Where(a => a.Login == login && a.Password == password).FirstOrDefault();

if(authUser == null)

{

switch (System.Threading.Thread.CurrentThread.CurrentUICulture.Name)

{

case "ru-RU":

{

throw new Exception("Невозможно найти пользователя с введенными данными");

}

default:

{

throw new Exception("Unable to find user with entered data");

}

}

}

if (authUser != null)

{

if (authUser.Is\_admin == false)

{

MainWindow main = new MainWindow();

main.WindowStartupLocation = WindowStartupLocation.CenterScreen;

main.Show();

AuthViewModel.Close();

}

else

{

MainAdminView mainAdmin = new MainAdminView();

mainAdmin.WindowStartupLocation = WindowStartupLocation.CenterScreen;

mainAdmin.Show();

AuthViewModel.Close();

}

Листинг(продолжение) – команда, реализующая авторизацию

User user = db.Users.Where(u => u.Auth\_id == authUser.Auth\_id).FirstOrDefault();

Settings.Default.AuthId = authUser.Auth\_id;

Settings.Default.UserMail = user.Email;

Settings.Default.UserId = user.User\_id;

}

}

}

catch(Exception e)

{

ErrorMessage = e.Message;

}

}));

}

}

Листинг(продолжение) – команда, реализующая авторизацию

# **Приложение И**

public static string GenerateTicket(Order order)

{

using (PartShopDbContext db = new PartShopDbContext())

{

double sum = 0;

var orderParts = from o in db.Orders

join op in db.OrderedParts

on o.Order\_id equals op.Order\_id

join p in db.Parts

on op.Part\_id equals p.Part\_id

where o.Order\_id == order.Order\_id

select new

{

Name = p.Name,

Amount = op.Amount,

Price = p.Price,

Brand = p.Brand.Name

};

StringBuilder body = new StringBuilder();

switch (System.Threading.Thread.CurrentThread.CurrentUICulture.Name)

{

case "ru-RU":

{

body.Append($"Чек заказа {order.Order\_id} из магазина автозапастей VintageParts");

body.Append("\n" + order.OrderDate);

body.Append("\n-----------------------------------");

foreach (var p in orderParts)

{

body.Append($"\nЗапчасть: {p.Name}, Количество: {p.Amount}, Цена: {p.Price}, Бренд: {p.Brand}");

sum += p.Price \* p.Amount;

}

body.Append("\n-----------------------------------");

body.Append($"\nДоставка: {order.Delivery.Name}");

body.Append($"\nИтоговая сумма: {sum + order.Delivery.Price}");

body.Append("\n-----------------------------------");

body.Append("\nСпасибо за заказ в нашем магазине!");

return body.ToString();

}

default:

{

body.Append($"Order receipt {order.Order\_id} from VintageParts auto parts store.");

Листинг – метод генерации чека

body.Append("\n" + order.OrderDate);

body.Append("\n-----------------------------------");

foreach (var p in orderParts)

{

body.Append($"\nPart: {p.Name}, Quantity: {p.Amount}, Price: {p.Price}, Brand: {p.Brand}");

sum += p.Price \* p.Amount;

}

body.Append("\n-----------------------------------");

body.Append($"\nDelivery: {order.Delivery.Name}");

body.Append($"\nTotal amount: {sum + order.Delivery.Price}");

body.Append("\n-----------------------------------");

body.Append("\nThank you for ordering in our store!");

return body.ToString();

}

}

}

}

Листинг(продолжение) – метод генерации чека

# **Приложение К**

public void AddOrder(ObservableCollection<Part> parts)

{

using (PartShopDbContext db = new PartShopDbContext())

{

try

{

if(Card.Balance < Summary)

{

switch (System.Threading.Thread.CurrentThread.CurrentUICulture.Name)

{

case "ru-RU":

{

throw new Exception("Недостаточно средств на счете");

}

default:

{

throw new Exception("Insufficient funds on the account");

}

}

}

else if( selectedDelivery == null)

{

switch (System.Threading.Thread.CurrentThread.CurrentUICulture.Name)

{

case "ru-RU":

{

throw new Exception("Должна быть выбрана доставка");

}

default:

{

throw new Exception("Delivery must be selected");

}

}

}

Order order = new Order();

order.OrderDate = DateTime.Now;

order.OrderState = Resources.waiting;

List<OrderedParts> details = new List<OrderedParts>();

foreach (Part i in Parts)

{

if(i.Amount > db.Parts.Where(x => x.Part\_id == i.Part\_id).FirstOrDefault().Quantity)

{

switch (System.Threading.Thread.CurrentThread.CurrentUICulture.Name)

Листинг– метод, реализующий формирование заказа

{

case "ru-RU":

{

throw new Exception($"Товаров {i.Name} недостаточно на складе для заказа");

}

default:

{

throw new Exception($"There are not enough {i.Name} items in stock to order");

}

}

}

details.Add(new OrderedParts()

{

Order\_id = order.Order\_id,

Part\_id = i.Part\_id,

Amount = i.Amount

});

db.Parts.Where(x => x.Part\_id == i.Part\_id).FirstOrDefault().Quantity -= i.Amount;

}

order.Parts = details;

order.User\_id = Settings.Default.UserId;

order.Delivery\_id = selectedDelivery.Delivery\_id;

db.Orders.Add(order);

Card.Balance -= Summary;

Parts.Clear();

Summary = 0;

db.SaveChanges();

ConfirmOrderViewModel.orderId = order.Order\_id;

Singleton.getInstance(null).MainViewModel.CurrentViewModel = new ConfirmOrderViewModel();

}

catch(Exception e)

{

ErrorMessage = e.Message;

}

}

}

Листинг(продолжение)– метод, реализующий формирование заказа

# **Приложение Л**

public Command submitCode;

public ICommand SubmitCode

{

get

{

return submitCode ??

(submitCode = new Command(obj =>

{

try

{

if (code == Convert.ToInt32(codeFromBox))

{

App.db.Orders.Where(x => x.Order\_id == orderId).FirstOrDefault().OrderState = Resources.acepted;

App.db.SaveChanges();

switch (System.Threading.Thread.CurrentThread.CurrentUICulture.Name)

{

case "ru-RU":

{

App.NotifyWindow(Application.Current.Windows[0]).ShowSuccess("Ваш заказ был подтвержден");

EmailSenderService.SendTicket(Settings.Default.UserMail, "Чек заказа", EmailSenderService.GenerateTicket(order)).GetAwaiter();

break;

}

default:

{

App.NotifyWindow(Application.Current.Windows[0]).ShowSuccess("Your order has been confirmed");

EmailSenderService.SendTicket(Settings.Default.UserMail, "Order receipt", EmailSenderService.GenerateTicket(order)).GetAwaiter();

break;

}

}

}

else

{

switch (System.Threading.Thread.CurrentThread.CurrentUICulture.Name)

{

case "ru-RU":

{

App.NotifyWindow(Application.Current.Windows[0]).ShowError("Введен неверный код");

break;

Листинг – команда подтверждения заказа

}

default:

{

App.NotifyWindow(Application.Current.Windows[0]).ShowError("Invalid code entered");

break;

}

}

}

}

catch(Exception e)

{

MessageBox.Show(e.Message);

}

}));

}

}

Листинг(продолжение)– команда подтверждения заказа

# **Приложение М**

private Command cancelOrder;

public ICommand CancelOrder

{

get

{

return cancelOrder ??

(cancelOrder = new Command(obj =>

{

try

{

if (Convert.ToInt32(codeFromView) == code)

{

List<OrderedParts> prts = new List<OrderedParts>(App.db.OrderedParts.Where(x => x.Order\_id == orderId));

foreach (var p in prts)

{

App.db.Parts.Where(x => x.Part\_id == p.Part\_id & p.Order\_id == orderId).FirstOrDefault().Quantity += p.Amount;

Sum += App.db.Parts.Where(x => x.Part\_id == p.Part\_id).FirstOrDefault().Price \* p.Amount;

}

Sum += orderForCancelation.Delivery.Price;

orderForCancelation.OrderState = Resources.canceled;

userCard.Balance += Sum;

App.db.SaveChangesAsync();

switch (System.Threading.Thread.CurrentThread.CurrentUICulture.Name)

{

case "ru-RU":

{

App.NotifyWindow(Application.Current.Windows[0]).ShowSuccess("Ваш заказ отменен");

break;

}

default:

{

App.NotifyWindow(Application.Current.Windows[0]).ShowSuccess("Your order has been cancelled");

break;

}

}

}

else

{

Листинг – команда отмены заказа

switch (System.Threading.Thread.CurrentThread.CurrentUICulture.Name)

{

case "ru-RU":

{

App.NotifyWindow(Application.Current.Windows[0]).ShowError("Введен неверный код");

break;

}

default:

{

App.NotifyWindow(Application.Current.Windows[0]).ShowError("Invalid code entered");

break;

}

}

}

}

catch(Exception e)

{

MessageBox.Show(e.Message);

}

}));

}

}

Листинг(продолжение)– команда отмены заказа