МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО»

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту

по дисциплине «Избранные главы информатики»

на тему: «Интернет платформа онлайн-коммерции**»**

Исполнитель: студент гр. ИП-31
П. Н. Казутин
Руководитель: ст. преподаватель
Т. Л. Романькова

Дата проверки: _____
Дата допуска к защите: _____
Дата защиты: _____
Оценка работы: _____

Подписи членов комиссии
по защите курсового проекта: _____

Гомель 2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 4
1 АНАЛИТИЧЕСКИЙ ОБЗОР 5
1.1 Постановка задачи 5
1.2 Анализ предметной области 5
1.3 Обзор существующих аналогов 7
2 Проектирование приложения 12
2.1 Описание используемых технологий 12
2.2 Концептуальное проектирование базы данных 17
2.3 Логическое проектирование базы данных 17
2.3 Физическое проектирование базы данных 20
3 Реализация приложения 26
Заключение 27
Список литературы 28
Приложение А 29
Приложение Б 30
Приложение В 31

ВВЕДЕНИЕ

С повсеместной цифровизацией у человека изменились привычки. Если раньше считалось чем-то обыденным поездки на рынок за любыми вещами, то сейчас в этом нет необходимости. В двадцать первом веке физические рынки были заменены площадками онлайн-коммерции.

У таких площадок целый ряд преимуществ. Во-первых, это просто удобно — обычным покупателям теперь не нужно часами обходить торговые ряды в поисках необходимой вещи приемлемого качества. Советы продавцов заменены реальными отзывами покупателей, система «торга» заменена скидочными купонами. Всё честно и прозрачно.

Во-вторых, следует отметить тот факт, что и для рядового продавца стало удобнее вести коммерческую деятельность. Это достигается следующими факторами: дешевизна рабочего места – в худшем случае продавец оплачивает только собственный склад; прозрачность работы системы – у платформ онлайн-коммерции есть список понятных и одинаковых для всех правил, автоматизированное получение статистики – знание о динамике продаж и т.д. позволяет сделать бизнес более гибким, восприимчивым к изменениям в спросе.

На данный момент существует множество платформ онлайн-коммерции, многие из них интернациональные. Примером такой платформы можно назвать AliExpress – лидер международной онлайн коммерции, созданный в Китае компанией Alibaba.

Целью данной курсовой работы является создание платформы онлайн-коммерции. В ходе выполнения курсового проектирования будет спроектирована и разработана онлайн платформа для автоматизированного ведения коммерческой деятельности.

1 АНАЛИТИЧЕСКИЙ ОБЗОР

1.1 Постановка задачи

Суть поставленной задачи заключается в разработке веб-приложения для ведения коммерческой деятельности множеством продавцов.

Для этого выдвигаются требования к веб-приложению:

- предусмотреть ведение справочника магазинов на платформе;
- предусмотреть ведение справочника категорий товаров на платформе;
- предусмотреть ведение справочника товаров на платформе;
- введение ролей «Администратор», «Менеджер магазина», «Простой пользователь», «Владелец магазина»;
- предусмотреть на основе указанных ролей разграничение доступа;
- предусмотреть общения между пользователями;
- предусмотреть ведения корзины товаров;
- предусмотреть ведение истории заказов пользователя;
- предусмотреть возможность получения статистики по продажам;
- предусмотреть наличие панели Администратора

Для достижения поставленной цели нужно выполнить следующие этапы:

- проведение анализа предметной области;
- рассмотреть существующие аналоги, выявить их достоинства и недостатки;
- разработать концепцию веб-приложения;
- спроектировать веб-приложение.
- Разработать и отладить веб-приложение

1.2 Анализ предметной области

На основе требований, изложенных в п. 1.1, можно заключить:

- администратор должен иметь следующие права доступа:
    1. просмотр и редактирование списка категорий;
    2. просмотр и редактирование списка пользователей;
    3. просмотр и редактирование списка магазинов;
    4. просмотр и редактирование списка товаров;
    5. просмотр списка заказов;
    6. просмотр списка пользовательских корзин
- менеджер магазина должен иметь следующие права доступа:
    1. просмотр списка сообщений;
    2. просмотр списка заказов своих и магазина;
    3. просмотр и редактирование списка товаров в магазине;

– владелец магазина должен иметь следующие права доступа:
    1. просмотр и редактирование списка собственных магазинов;
    2. просмотр сообщений;
    3. просмотр статистики;
– обычный пользователь должен иметь следующие права доступа:
    1. просмотр сообщений;
    2. просмотр своей корзины;
    3. просмотр своих заказов;
    4. просмотр списка товаров;
    5. просмотр списка категорий, товаров по категориям;

По перечисленным выше правам доступа и требованиям к веб-приложению получается диаграмма прецедентов, показанная на рисунке 1.1.

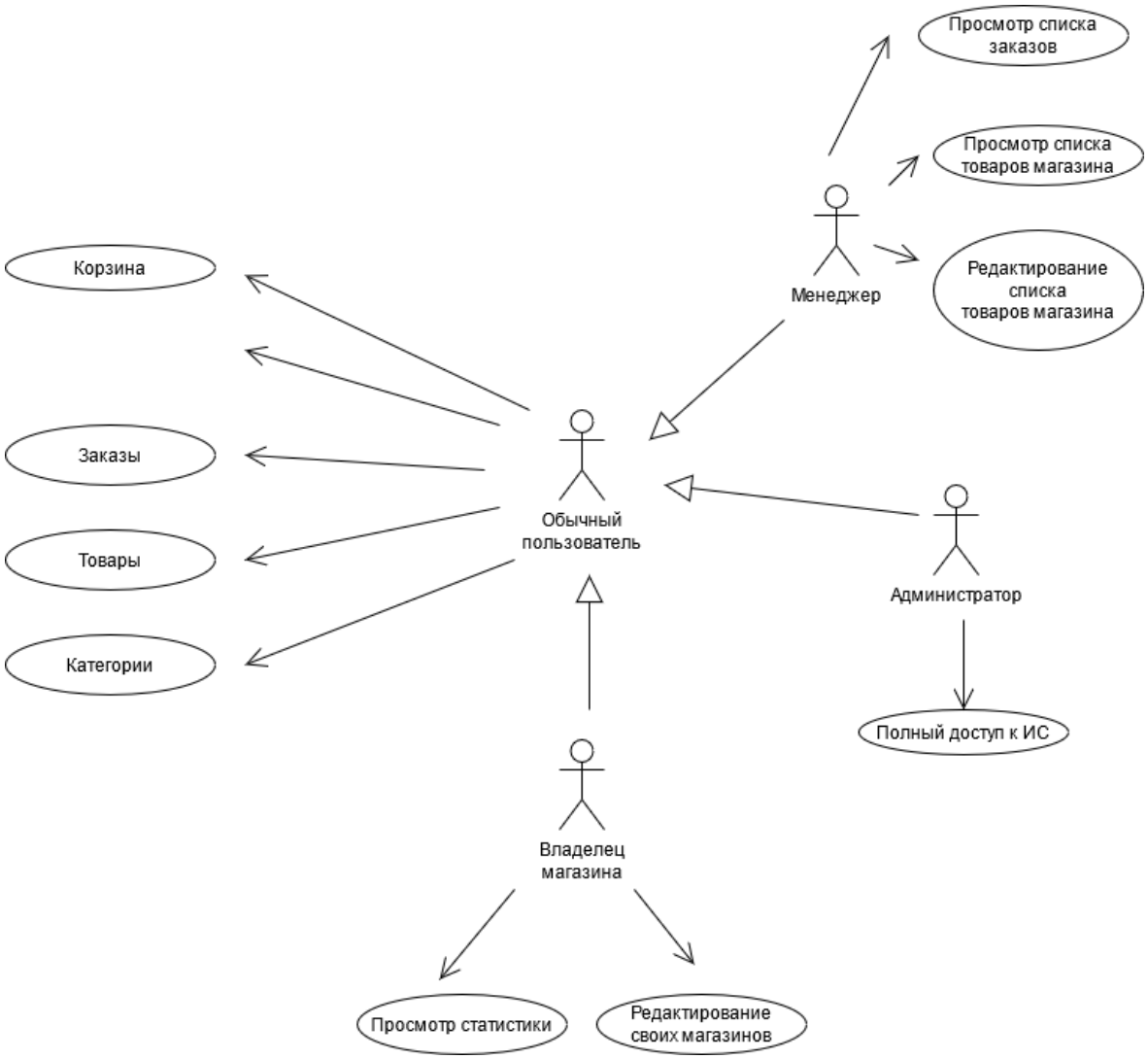

Рисунок 1.1 – Диаграмма прецедентов

1.3 Обзор существующих аналогов

В данный момент существует большое количество уже готовых сервисов, работающих в разных регионах, по разным моделям. Перечислим некоторые хорошо работающие площадки:

– Deal.by
– AliExpress.com
– Яндекс.Маркет

Попробуем дальше сравнить эти площадки, выяснить плюсы и минусы каждого из представленных примеров.

Deal.by – белорусский маркетплейс, где представлены миллионы товаров и услуг от компаний и частных лиц. По данным на 1-е апреля 2021 г. в каталоге находится 7 тысяч компаний и 14 миллионов товаров, что делает эту площадку крупнейшей в Республике Беларусь.

Плюсы площадки:

– крупнейшая площадка в Республике Беларусь;
– есть мобильное приложение, мобильная версия сайта;
– большой каталог товаров, физически находящихся в Беларуси;

Минусы площадки:

– медленный интерфейс;
– нерелевантная поисковая выдача;

Примеры пользовательского интерфейса площадки Deal.by представлены на рисунках 1.2, 1.3.

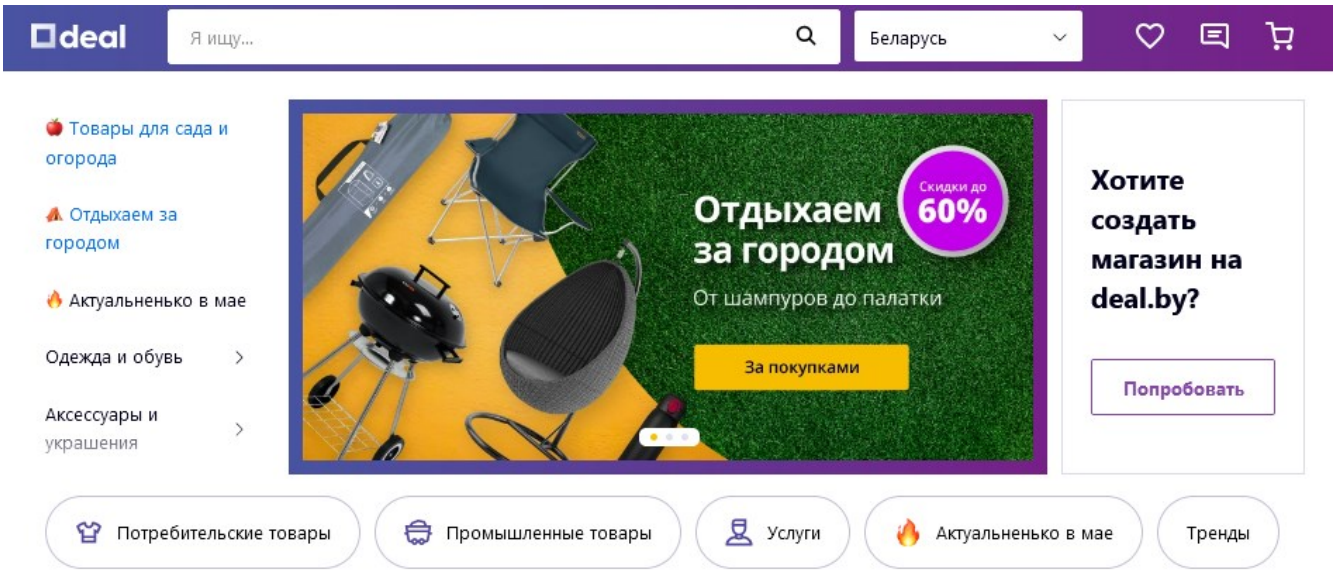

Рисунок 1.2 – Пример пользовательского интерфейса площадки Deal.by

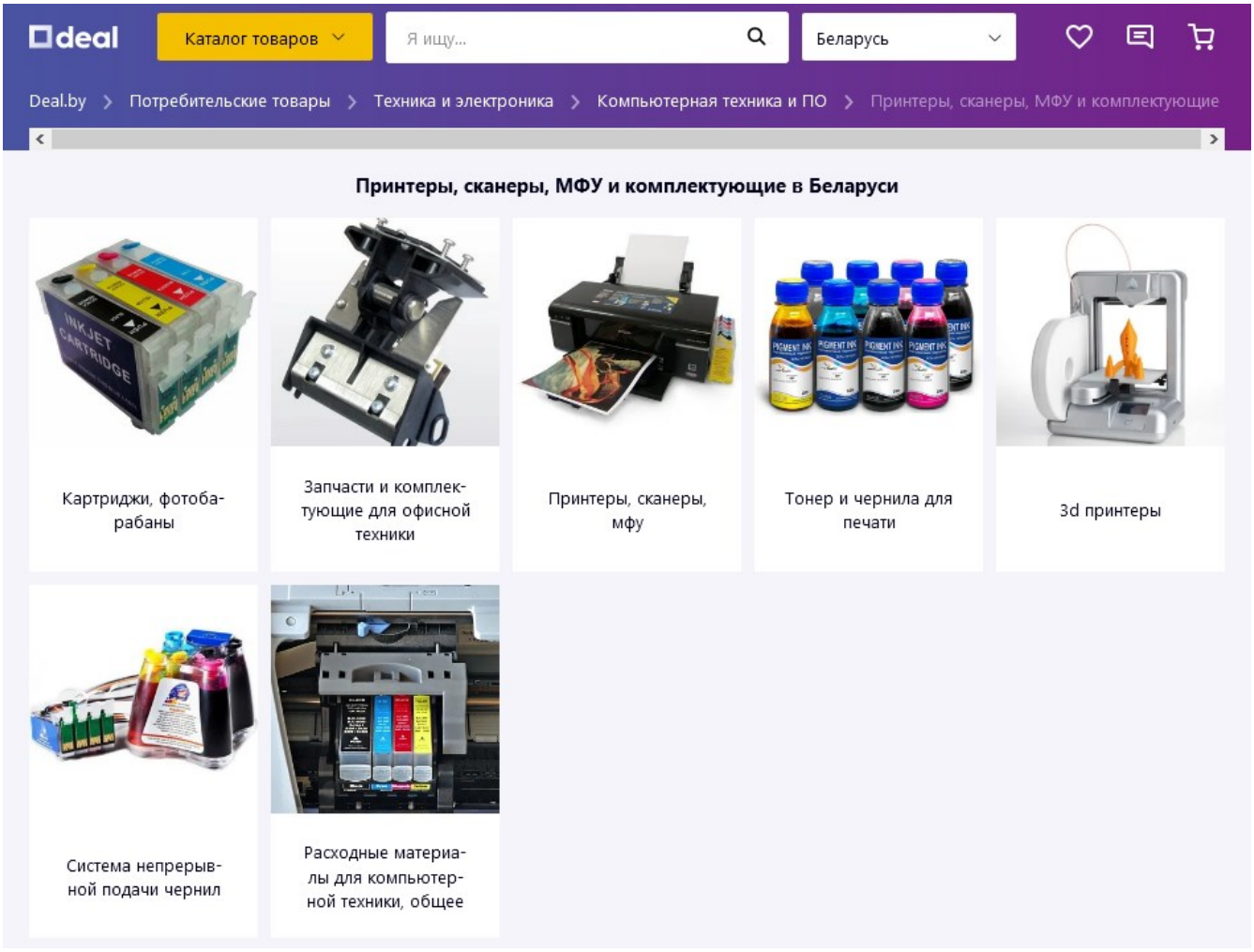

Рисунок 1.3 – Пример отображения категорий на площадке Deal.by

AliExpress — глобальная виртуальная торговая площадка, предоставляющая возможность покупать товары производителей из КНР (Китайская Народная Республика), а также России, Европы, Турции и других стран. Товары на площадке продаются в розницу и мелким оптом (в отличие от оптовой Alibaba). Платформа не работает в КНР — там её заменяет Taobao.

AliExpress начал работу в 2010 году, как площадка для продажи китайских товаров в другие страны. Сначала эта площадка работала в сфере B2B (Business to Business) для покупки и продажи. С тех пор портал расширился до B2C (Business to Consumer), C2C (Consumer to Consumer), облачных вычислений и платежных сервисов.

Изначально AliExpress специализировался на продаже товаров только от китайских продавцов. С ноября 2018 года в ассортименте AliExpress появились товары турецких брендов с площадки Trendyol, которую приобрела Alibaba.

AliExpress в настоящее время доступен на русском, английском, испанском, нидерландском, французском, итальянском, польском, арабском и португальском языках (информация о товарах автоматически переводится на разные языки). Клиенты вне границ стран для этих языков автоматически обслуживаются на английской версии сайта.

Сайт занимает 32-е место в списке самых посещаемых веб-ресурсов в мире.

Попытаемся выяснить плюсы и минусы этой площадки.

Плюсы:

– глобальность данной площадки обеспечивает широчайший ассортимент товаров на виртуальном «прилавке»;

– наличие у площадки собственного логистического сервиса, складов по территории Европы, России;

– общая дешевизна товаров, обусловленная местом их происхождения;

Минусы:

– большая часть товаров всё же находится на территории КНР, что обуславливает долгую доставку заказов конечному потребителю;

– слабая защита конечного пользователя от контрафактной, низкого качества и испорченной продукции;

– очень сложный интерфейс, неочевидный во многих местах;

Примеры пользовательского интерфейса площадки Deal.by представлены на рисунках 1.4, 1.5.

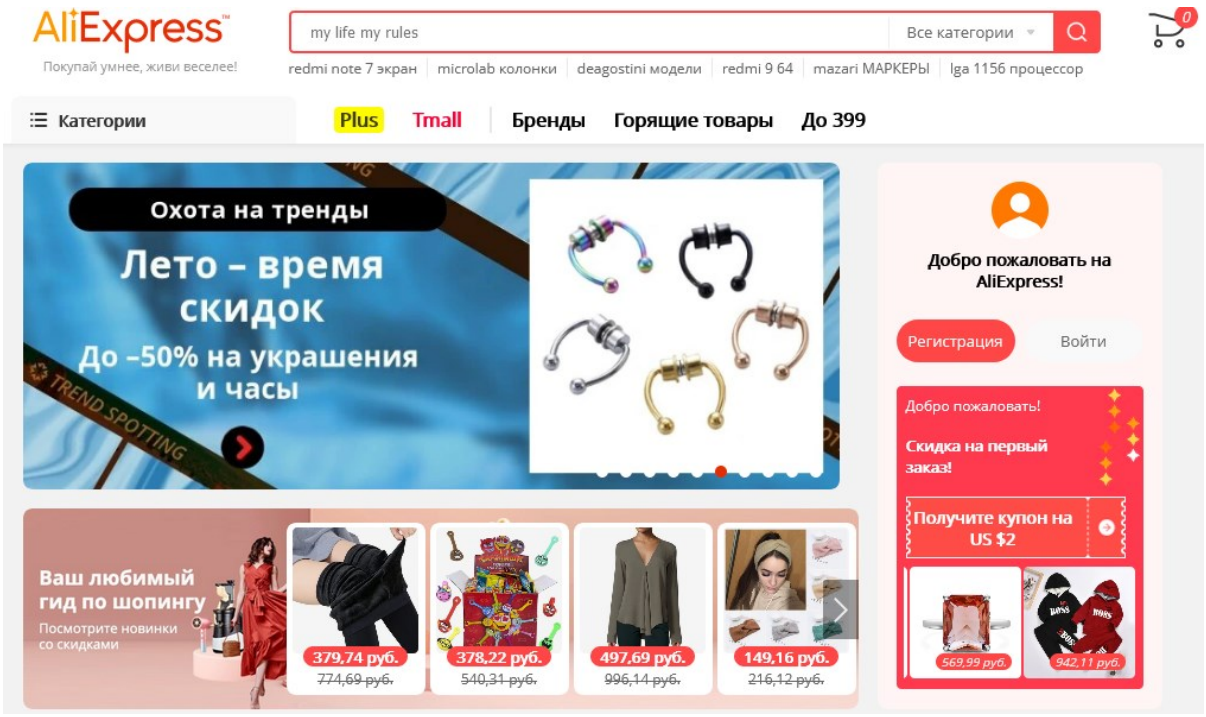

Рисунок 1.4 – Пример пользовательского интерфейса площадки AliExpress

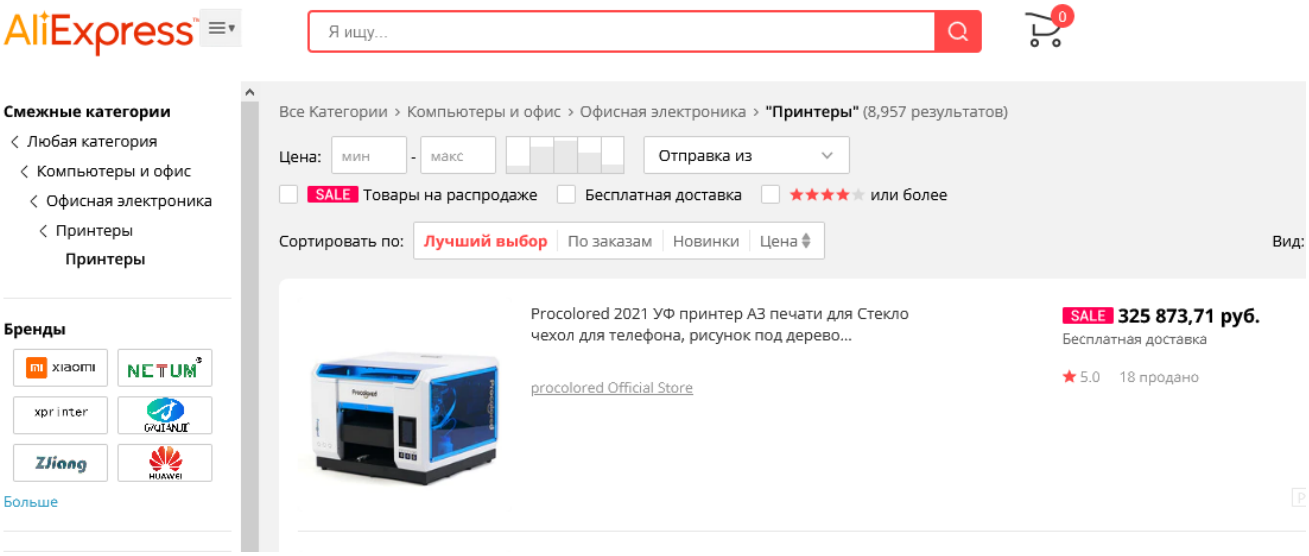

Рисунок 1.5 – Пример просмотра товаров категории на площадке AliExpress

«Яндекс. Маркет» — сервис для выбора и покупки товаров. Пользователям сервиса доступны более 230 млн предложений от 25 тыс. российских и зарубежных интернет-магазинов. Сервис позволяет пользователям сравнивать товары по параметрам и ценам, изучать отзывы и обзоры, а также задавать вопросы другим посетителям сайта, магазинам и производителям. Сервис берет на себя обработку, доставку заказа и общение с покупателями. Приложение Яндекс. Маркет доступно для iOS и Android.

С 1 октября 2020 года «Яндекс. Маркет» провел ребрендинг маркетплейса «Беру», который перестал существовать как бренд и был перемещен в раздел «Покупки» на «Яндекс. Маркете». Таким образом, «Яндекс. Маркет» был переквалифицирован в площадку не только для сравнения и выбора товаров, но и для их покупки.

Плюсы данной площадки:

– высокий технический уровень площадки;
– хорошие рекомендательные алгоритмы;
– хорошие поисковые алгоритмы;
– современный дизайн;
– отзывчивый веб-интерфейс;

Минусы площадки:

– малое количество беларусских продавцов.

Примеры интерфейса отображены на рисунках 1.6, 1.7.

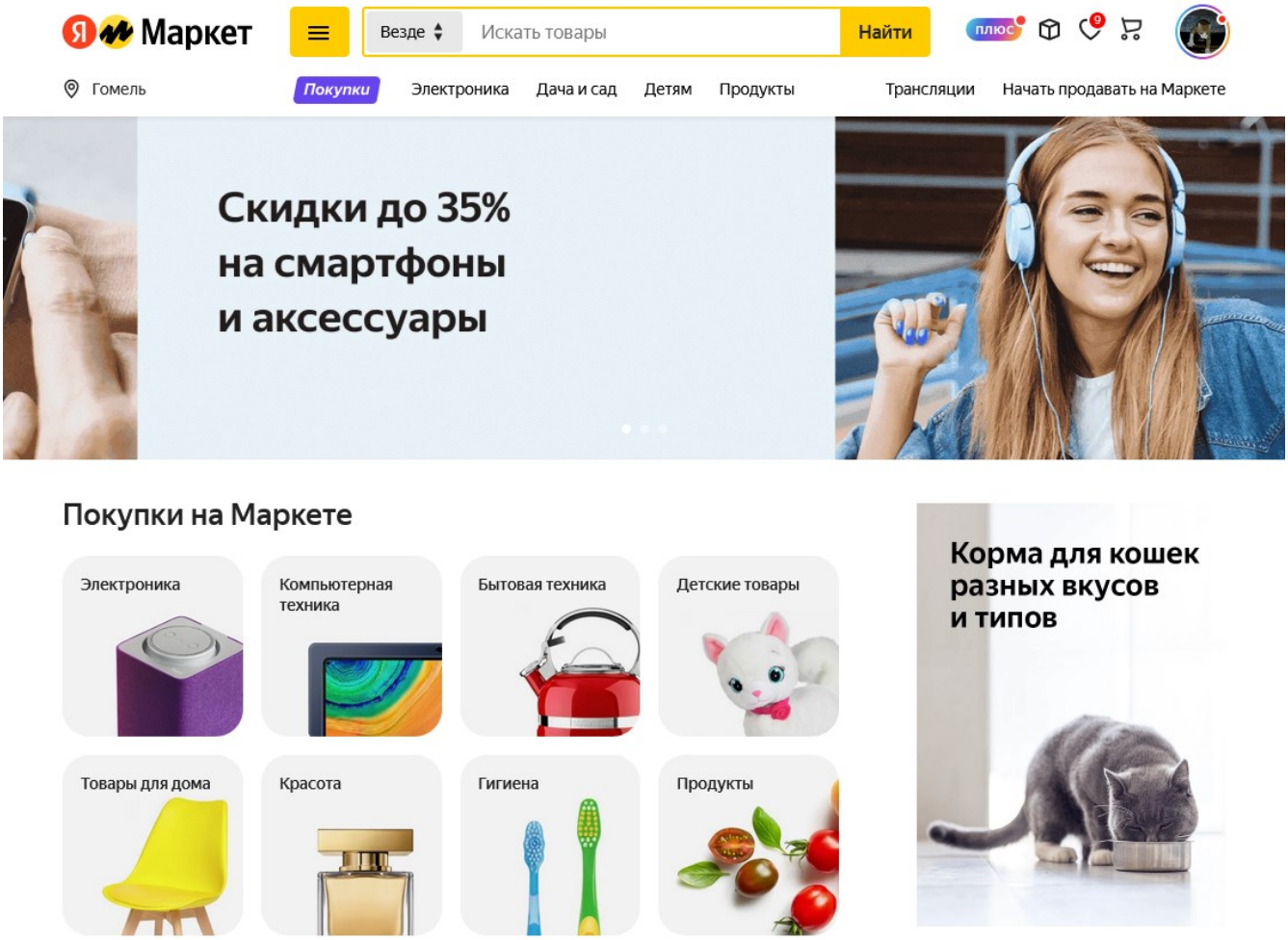

Рисунок 1.6 – Пример пользовательского интерфейса платформы Яндекс.Маркет

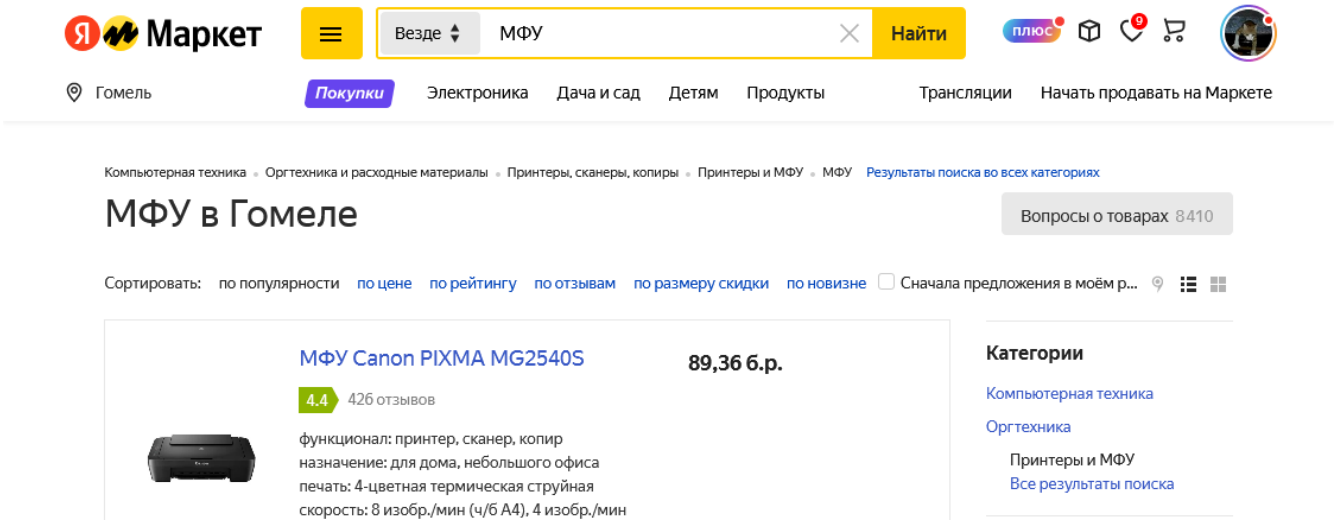

Рисунок 1.7 – Результат поиска на платформе Яндекс.Маркет

2 Проектирование приложения

2.1 Описание используемых технологий

Для создания веб-приложения были использованы следующие технологии:

– язык программирования C# версии 9.0;
– платформа .Net Core 5;
– платформа ASP.Net Core 5 MVC;
– СУБД PostgreSQL 12.

Опишем каждую из приведённых выше технологий.

C# (произносится как "си шарп") — современный объектно-ориентированный и типобезопасный язык программирования. C# позволяет разработчикам создавать множество типов безопасных и надежных приложений, работающих в экосистеме .NET. C# относится к широко известному семейству языков C, и покажется хорошо знакомым любому, кто работал с C, C++, Java или JavaScript.

C# — это объектно- и компонентно-ориентированный язык программирования. C# предоставляет языковые конструкции для непосредственной поддержки такой концепции работы. Благодаря этому C# подходит для создания и применения программных компонентов. С момента создания язык C# обогатился функциями для поддержки новых рабочих нагрузок и современными рекомендациями по разработке ПО.

Вот лишь несколько функций языка C#, которые позволяют создавать надежные и устойчивые приложения.

Сборка мусора автоматически освобождает память, занятую недоступными неиспользуемыми объектами. Типы, допускающие значение null, обеспечивают защиту от переменных, которые не ссылаются на выделенные объекты. Обработка исключений предоставляет структурированный и расширяемый подход к обнаружению ошибок и восстановлению после них. Лямбда-выражения поддерживают приемы функционального программирования. Синтаксис LINQ создает общий шаблон для работы с данными из любого источника. Поддержка языков для асинхронных операций предоставляет синтаксис для создания распределенных систем.

В C# действует единая система типов. Все типы C#, включая типы-примитивы, такие как int и double, наследуют от одного корневого типа object. Все типы используют общий набор операций, а значения любого типа можно хранить, передавать и обрабатывать схожим образом. Более того, C# поддерживает как определяемые пользователями ссылочные типы, так и типы значений. C# позволяет динамически выделять объекты и хранить упрощенные структуры в стеке. C# поддерживает универсальные методы и типы, обеспечивающие повышенную безопасность типов и производительность. C# предоставляет итераторы, которые позволяют разработчикам классов коллекций определять пользовательские варианты поведения для клиентского кода.

В языке особое внимание уделяется управлению версиями для обеспечения совместимости программ и библиотек при их изменении. Вопросы управления версиями существенно повлияли на такие аспекты разработки C#, как раздельные модификаторы virtual и override, правила разрешения перегрузки методов и поддержка явного объявления членов интерфейса.

Программы C# выполняются в .NET, виртуальной системе выполнения, вызывающей общеязыковую среду выполнения (Common Language Runtime, CLR) и набор библиотек классов. Среда CLR — это реализация общеязыковой инфраструктуры языка (Common Language Infrastructure, CLI), являющейся международным стандартом, от корпорации Майкрософт. CLI является основой для создания сред выполнения и разработки, в которых языки и библиотеки прозрачно работают друг с другом.

Исходный код, написанный на языке C#, компилируется в промежуточный язык (Intermediate Language, IL), который соответствует спецификациям CLI. Код на языке IL и ресурсы, в том числе растровые изображения и строки, сохраняются в сборке, обычно с расширением .dll. Сборка содержит манифест с информацией о типах, версии, языке и региональных параметрах для этой сборки.

При выполнении программы C# сборка загружается в среду CLR. Среда CLR выполняет JIT (Just In Time)-компиляцию из кода на языке IL в инструкции машинного языка. Среда CLR также выполняет другие операции, например, автоматическую сборку мусора, обработку исключений и управление ресурсами. Код, выполняемый средой CLR, иногда называют "управляемым кодом", чтобы подчеркнуть отличия этого подхода от "неуправляемого кода", который сразу компилируется в машинный язык для определенной платформы.

Обеспечение взаимодействия между языками является ключевой особенностью .NET. Код IL, созданный компилятором C#, соответствует спецификации общих типов (Common Type System, CTS). Код IL, созданный из кода на C#, может взаимодействовать с кодом, созданным из версий .NET для языков F#, Visual Basic, C++ и любых других из более чем 20 языков, совместимых с CTS. Одна сборка может содержать несколько модулей, написанных на разных языках .NET, и все типы могут ссылаться друг на друга, как если бы они были написаны на одном языке.

В дополнение к службам времени выполнения .NET также включает расширенные библиотеки. Эти библиотеки поддерживают множество различных рабочих нагрузок. Они упорядочены по пространствам имен, которые предоставляют разные полезные возможности: от операций файлового ввода и вывода до управления строками и синтаксического анализа XML (eXtensible Markup Language), от платформ веб-приложений до элементов управления Windows Forms.

ASP.NET Core является кроссплатформенной, высокопроизводительной средой с открытым исходным кодом для создания современных облачных

приложений, подключенных к Интернету. ASP.NET Core позволяет выполнять следующие задачи:

– создавать веб-приложения и службы, приложения Интернета вещей (IoT) и серверные части для мобильных приложений;
– использовать избранные средства разработки в Windows, macOS и Linux;
– выполнять развертывания в облаке или локальной среде;
– запускать в .NET Core.

Миллионы разработчиков использовали и продолжают использовать ASP.NET 4.x для создания веб-приложений. ASP.NET Core — это модификация ASP.NET 4.x с архитектурными изменениями, формирующими более рациональную и более модульную платформу.

ASP.NET Core предоставляет следующие преимущества:

– единое решение для создания пользовательского веб-интерфейса и веб-API;
– разработано для тестируемости;
– Razor Pages упрощает написание кода для сценариев страниц и повышает его эффективность;
– Blazor позволяет использовать в браузере язык C# вместе с JavaScript. совместное использование серверной и клиентской логик приложений, написанных с помощью .NET;
– возможность разработки и запуска в ОС Windows, macOS и Linux;
– открытый исходный код и ориентация на сообщество;
– интеграция современных клиентских платформ и рабочих процессов разработки;
– поддержка размещения служб удаленного вызова процедур (RPC) с помощью gRPC;
– облачная система конфигурации на основе среды;
– встроенное введение зависимостей;
– упрощенный высокопроизводительный модульный конвейер HTTP-запросов;
– инструментарий, упрощающий процесс современной веб-разработки.

ASP.NET Core MVC (Model View Controller) представляет собой упрощенную, эффективно тестируемую платформу с открытым исходным кодом, оптимизированную для использования с ASP.NET Core.

ASP.NET Core MVC предоставляет основанный на шаблонах способ создания динамических веб-сайтов с четким разделением задач. Она обеспечивает полный контроль разметки, поддерживает согласованную с TDD (Test Driven Development) разработку и использует новейшие веб-стандарты.

Структура архитектуры MVC разделяет приложение на три основных группы компонентов: модели, представлении и контроллеры. Это позволяет реализовать

принципы разделения задач. Согласно этой структуре, запросы пользователей направляются в контроллер, который отвечает за работу с моделью для выполнения действий пользователей и получение результатов запросов. Контроллер выбирает представление для отображения пользователю со всеми необходимыми данными модели.

На рисунке 2.1 показаны три основных компонента и существующие между ними связи.

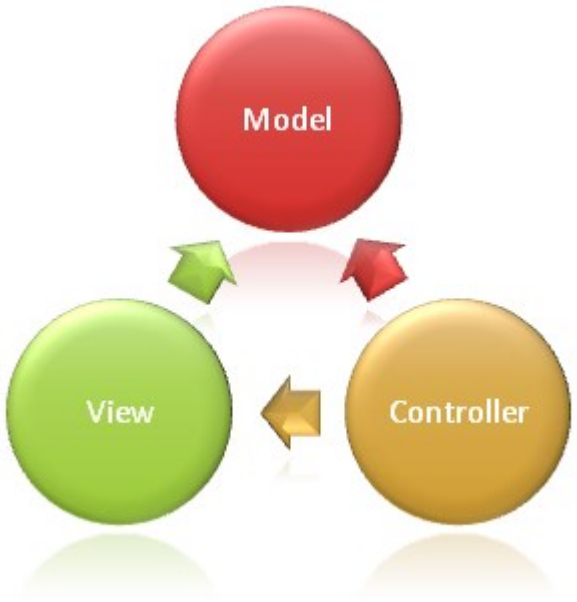

Рисунок 2.1 – Схема структуры MVC

Такое распределение обязанностей позволяет масштабировать приложение в контексте сложности, так как проще писать код, выполнять отладку и тестирование компонента (модели, представления или контроллера) с одним заданием. Гораздо труднее обновлять, тестировать и отлаживать код, зависимости которого находятся в двух или трех этих областях. Например, логика пользовательского интерфейса, как правило, подвергается изменениям чаще, чем бизнес-логика. Если код представления и бизнес-логика объединены в один объект, содержащий бизнес-логику, объект необходимо изменять при каждом обновлении пользовательского интерфейса. Это часто приводит к возникновению ошибок и необходимости повторно тестировать бизнес-логику после каждого незначительного изменения пользовательского интерфейса.

Модель в приложении MVC представляет состояние приложения и бизнес-логику или операций, которые должны в нем выполняться. Бизнес-логика должна быть включена в состав модели вместе с логикой реализации для сохранения состояния приложения. Как правило, строго типизированные представления используют

типы ViewModel, предназначенные для хранения данных, отображаемых в этом представлении. Контроллер создает и заполняет эти экземпляры ViewModel из модели.

Представления отвечают за представление содержимого через пользовательский интерфейс. Они используют Razor обработчик представлений для внедрения кода .NET в разметку HTML. Представления должны иметь минимальную логику, которая должна быть связана с представлением содержимого. Если есть необходимость выполнять большую часть логики в представлении для отображения данных из сложной модели, рекомендуется воспользоваться компонентом представления, ViewModel или шаблоном представления, позволяющими упростить представление.

Контроллеры — это компоненты для управления взаимодействием с пользователем, работы с моделью и выбора представления для отображения. В приложении MVC представление служит только для отображения информации. Обработку введенных данных, формирование ответа и взаимодействие с пользователем обеспечивает контроллер. В структуре MVC контроллер является начальной отправной точкой и отвечает за выбор рабочих типов моделей и отображаемых представлений (именно этим объясняется его название — он контролирует, каким образом приложение отвечает на конкретный запрос).

PostgreSQL — свободная объектно-реляционная система управления базами данных (СУБД).

PostgreSQL создана на основе некоммерческой СУБД Postgres, разработанной как open-source проект в Калифорнийском университете в Беркли. К разработке Postgres, начавшейся в 1986 году, имел непосредственное отношение Майкл Стоунбрейкер, руководитель более раннего проекта Ingres, на тот момент уже приобретённого компанией Computer Associates. Название расшифровывалось как «Post Ingres», и при создании Postgres были применены многие ранние наработки.

Стоунбрейкер и его студенты разрабатывали новую СУБД в течение восьми лет с 1986 по 1994 год. За этот период в синтаксис были введены процедуры, правила, пользовательские типы и другие компоненты. В 1995 году разработка снова разделилась: Стоунбрейкер использовал полученный опыт в создании коммерческой СУБД Illustra, продвигаемой его собственной одноимённой компанией (приобретённой впоследствии компанией Informix), а его студенты разработали новую версию Postgres — Postgres95, в которой язык запросов POSTQUEL — наследие Ingres — был заменен на SQL.

Разработка Postgres95 была выведена за пределы университета и передана команде энтузиастов. Новая СУБД получила имя, под которым она известна и развивается в текущий момент — PostgreSQL.

2.2 Концептуальное проектирование базы данных

Определим сущности, которые необходимы для создания базы данных приложения, адекватно описывающий заданную предметную область.

Перечислим их:

– Пользователь
– Роли пользователя
– Категория товара
– Товары
– Магазины
– Заказы
– Пользовательская корзина
– Сообщения

Концептуальная модель базы данных приведена на рисунке 2.2.

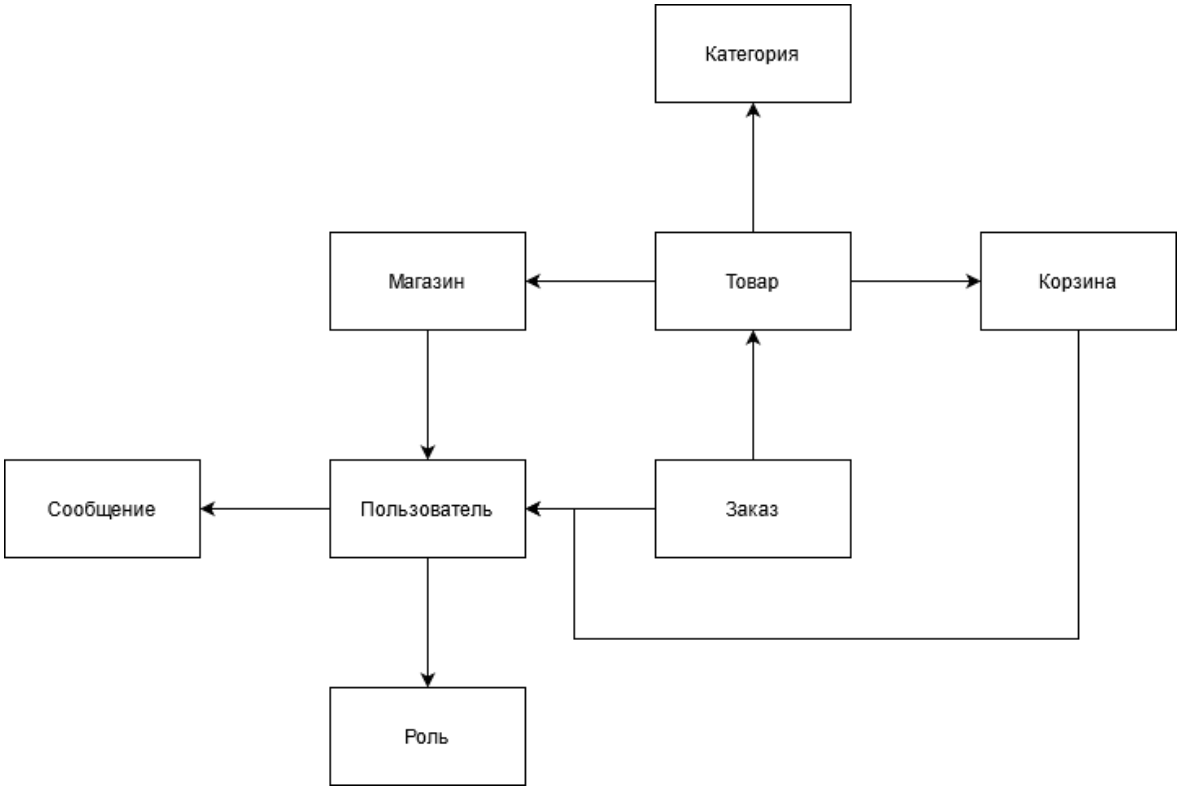

Рисунок 2.2 – Концептуальная модель базы данных

2.3 Логическое проектирование базы данных

На основе анализа предметной области выделены следующие роли:

– пользователь;
– роль пользователя;
– товар;
– категория товара;
– пользовательская корзина;
– заказ;
– сообщение;
– магазин

Сущность «Пользователь» содержит следующую информацию:
– идентификатор;
– имя пользователя;
– адрес электронной почты;
– адрес;
– изображение пароля;
– дата регистрации;
– роль

Сущность «Роль пользователя» содержит следующую информацию:
– идентификатор;
– название роли;

Сущность «Товар» содержит следующую информацию:
– идентификатор;
– магазина;
– фото товара;
– описание;
– название;
– цена;
– категория.

Сущность «Категория товара» содержит следующую информацию:
– идентификатор;
– название категории.

Сущность «Пользовательская корзина» содержит следующую информацию:
– идентификатор;
– список товаров;
– пользователь;

Сущность «Заказ» содержит следующую информацию:
– идентификатор;
– товары;
– пользователь;
– статус;

– время оформления заказа.

Сущность «Сообщение» содержит следующую информацию:

– идентификатор;
– получатель;
– отправитель;
– тело сообщения;
– дата отправки.

Сущность «Магазин» содержит следующую информацию:

– идентификатор;
– название;
– описание;
– рейтинг;
– владелец;
– менеджеры;
– адрес.

Логическая модель базы данных отображена на рисунке 2.3.

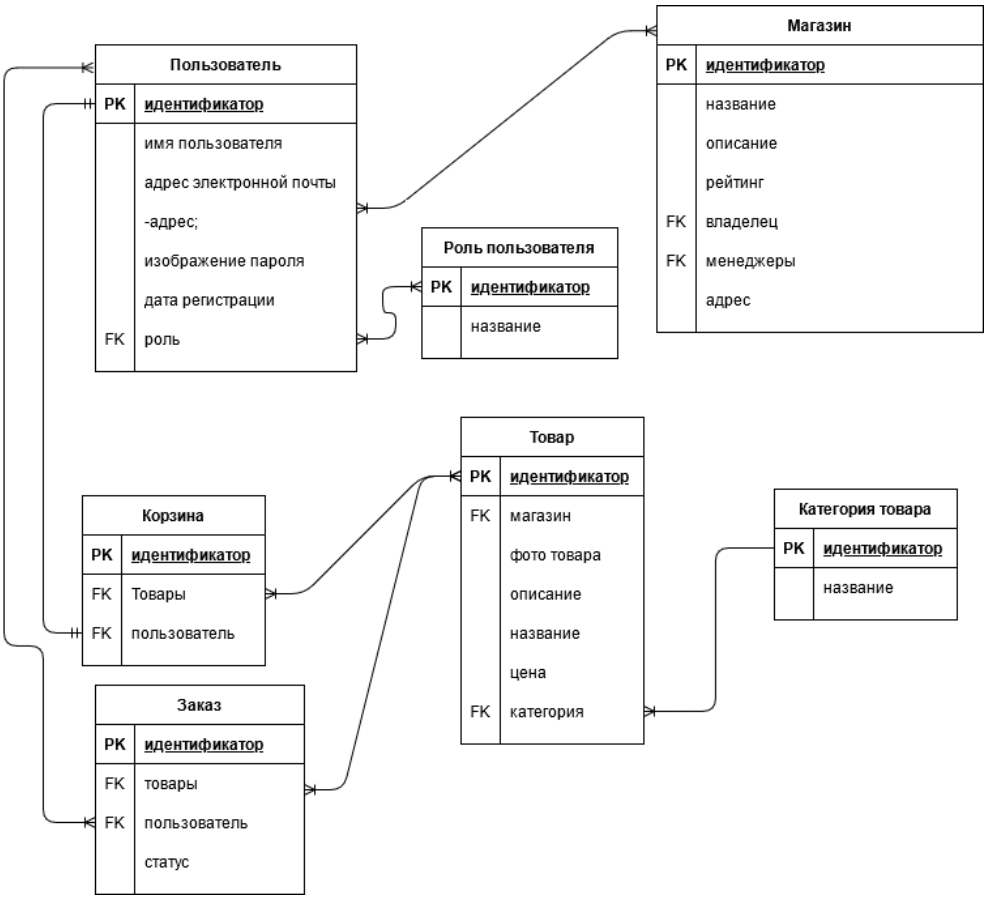

Рисунок 2.3 – Логическая схема базы данных

2.4 Физическое проектирование базы данных

На основе анализа предметной области, результатов концептуального и логического проектирования были созданы следующие таблицы:

- Categories;
- GoodUserBasket;
- Goods;
- Managers;
- Messages;
- Order;
- Stores;
- UserBaskets;
- Users;
- AspNetRoleClaims;
- AspNetRoles;
- AspNetUserClaims;
- AspNetUserLogins;
- AspNetRoles;
- AspNetTokens;
- AspNetUsers;
- Images.

Было принято решение о разделении большой базы данных на три части.

Первая часть, схема которой приведена на рисунке 2.4, расположена на самом нижнем слое приложения – слое доступа к данным. Эта база данных содержит в себе данные, относящиеся явно к предметной области. Структура таблиц отображена в таблицах 2.1 – 2.10.

Вторая часть расположена на втором слое приложения – слое бизнес-логики. В этой базе данных всего одна таблица, содержащая в себе изображения, необходимые для работы приложения. Структура этой таблицы описана в таблице 2.11.

Третья часть находится на слое представления, в ней хранится исчерпывающая информация о пользователе веб-приложения, за исключением информации, которая относится к предметной области. Структура таблиц этой базы данных показана в таблицах 2.12 – 2.18, физическая схема показана на рисунке 2.5.

Таблица 2.1 – Поля таблицы GoodUserBasket

Ключевое поле, тип ключа	**Наименование поля**	**Тип хранимых данных**	**Может принимать значение NULL**
Внешний	BasketId	TEXT	нет
Внешний	SelectedGoodId	TEXT	да

Таблица 2.2 – Поля таблицы Categories

Ключевое поле, тип ключа	**Наименование поля**	**Тип хранимых данных**	**Может принимать значение NULL**
Первичный	Id	INT	нет
–	Name	TEXT	нет
–	PicturePath	TEXT	да
–	Description	TEXT	нет

Таблица 2.3 – Поля таблицы Goods

Ключевое поле, тип ключа	**Наименование поля**	**Тип хранимых данных**	**Может принимать значение NULL**
Первичный	Id	INT	нет
Внешний	StoreId	INT	нет
–	MainPhotoName	TEXT	да
–	Description	TEXT	да
–	Name	TEXT	нет
–	Price	INT	нет
Внешний	CategoryId	INT	нет
–	Quantity	INT	нет

Таблица 2.4 – Поля таблицы Managers

Ключевое поле, тип ключа	**Наименование поля**	**Тип хранимых данных**	**Может принимать значение NULL**
Первичный	Id	INT	нет
Внешний	StoreId	INT	нет
Внешний	UserId	INT	нет

Таблица 2.5 – Поля таблицы Messages

Ключевое поле, тип ключа	**Наименование поля**	**Тип хранимых данных**	**Может принимать значение NULL**
Первичный	Id	INT	нет
Внешний	SenderId	INT	нет
Внешний	RecipientId	INT	нет
–	MessageBody	TEXT	нет
–	MessageTime	TEXT	нет

Таблица 2.6 – Поля таблицы Order

Ключевое поле, тип ключа	**Наименование поля**	**Тип хранимых данных**	**Может принимать значение NULL**
Первичный	Id	INT	нет
Внешний	UserId	INT	нет
–	Notes	TEXT	нет
–	State	TEXT	нет
–	OrderDate	TEXT	нет

Таблица 2.7 – Поля таблицы OrderGood

Ключевое поле, тип ключа	**Наименование поля**	**Тип хранимых данных**	**Может принимать значение NULL**
Внешний	GoodId	INT	нет
Внешний	OrderId	INT	нет

Таблица 2.8 – Поля таблицы Stores

Ключевое поле, тип ключа	**Наименование поля**	**Тип хранимых данных**	**Может принимать значение NULL**
Первичный	Id	INT	нет
–	Name	TEXT	нет
–	Description	TEXT	нет
Внешний	OwnerId	INT	нет
–	Address	TEXT	нет
	Raiting	INT	нет

Таблица 2.9 – Поля таблицы Users

Ключевое поле, тип ключа	**Наименование поля**	**Тип хранимых данных**	**Может принимать значение NULL**
Первичный	Id	INT	нет
–	Name	TEXT	нет
–	Birthday	TEXT	нет
–	RegistrationDay	TEXT	нет
–	Address	TEXT	нет
–	PhotoPath	TEXT	да

Таблица 2.10 – Поля таблицы UserBaskets

Ключевое поле, тип ключа	Наименование поля	Тип хранимых данных	Может принимать значение NULL
Первичный	Id	INT	нет
Внешний	UserId	INT	нет

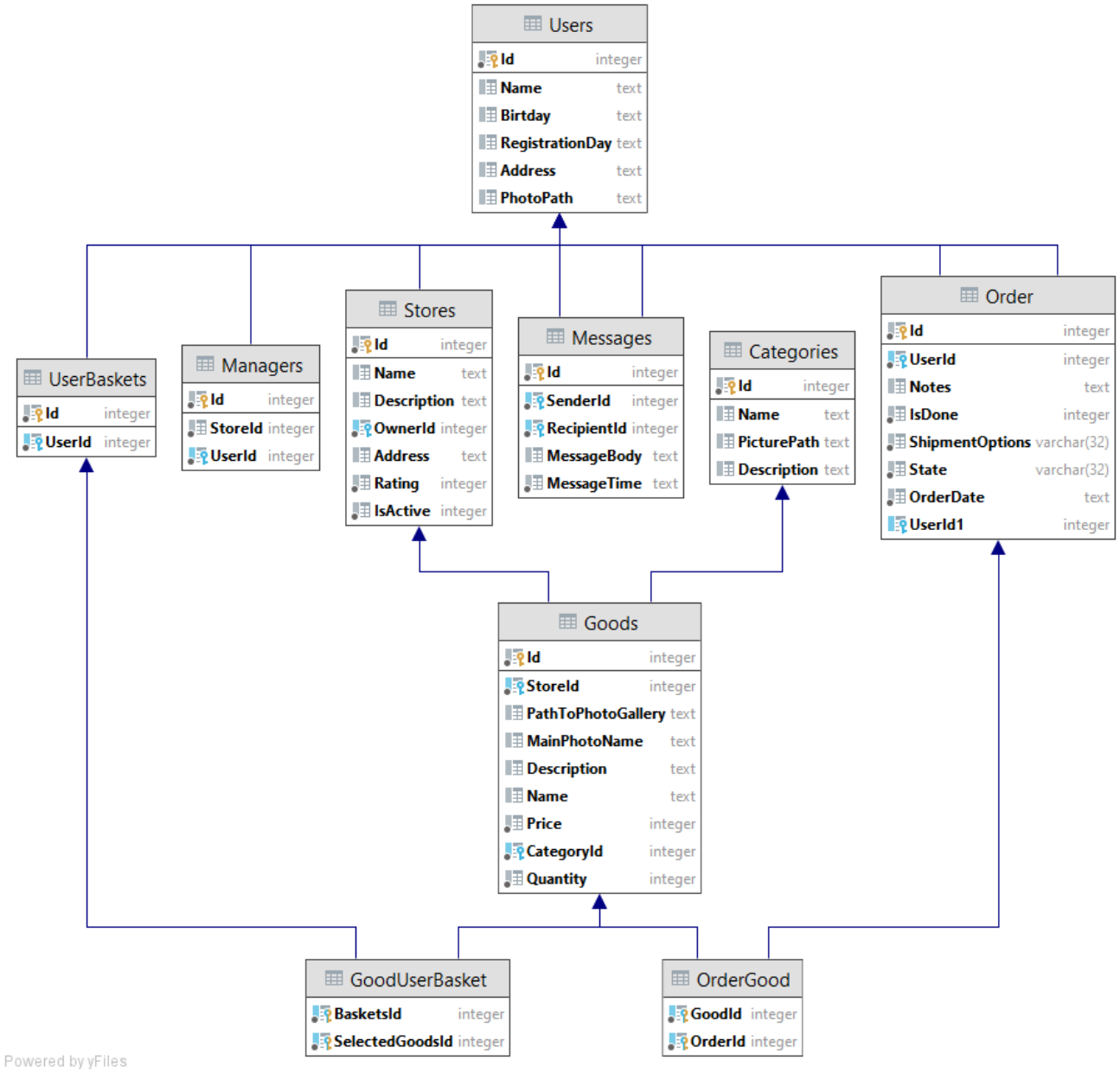

Рисунок 2.4 – Физическая схема базы данных

Таблица 2.11 – Поля таблицы Images

Ключевое поле, тип ключа	**Наименование поля**	**Тип хранимых данных**	**Может принимать значение NULL**
Первичный	Id	TEXT	нет
--	Base64String	TEXT	нет

Таблица 2.12 – Поля таблицы Images

Ключевое поле, тип ключа	**Наименование поля**	**Тип хранимых данных**	**Может принимать значение NULL**
Первичный	Id	TEXT	нет
--	Base64String	TEXT	нет

Таблица 2.13 – Поля таблицы Images

Ключевое поле, тип ключа	**Наименование поля**	**Тип хранимых данных**	**Может принимать значение NULL**
Первичный	Id	TEXT	нет
--	Base64String	TEXT	нет

Таблица 2.14 – Поля таблицы Images

Ключевое поле, тип ключа	**Наименование поля**	**Тип хранимых данных**	**Может принимать значение NULL**
Первичный	Id	TEXT	нет
--	Base64String	TEXT	нет

Таблица 2.15 – Поля таблицы Images

Ключевое поле, тип ключа	**Наименование поля**	**Тип хранимых данных**	**Может принимать значение NULL**
Первичный	Id	TEXT	нет
--	Base64String	TEXT	нет

Таблица 2.16 – Поля таблицы Images

Ключевое поле, тип ключа	**Наименование поля**	**Тип хранимых данных**	**Может принимать значение NULL**

Первичный	Id	TEXT	нет
--	Base64String	TEXT	нет

Таблица 2.17 – Поля таблицы Images

Ключевое поле, тип ключа	**Наименование поля**	**Тип хранимых данных**	**Может принимать значение NULL**
Первичный	Id	TEXT	нет
--	Base64String	TEXT	нет

Таблица 2.18 – Поля таблицы Images

Ключевое поле, тип ключа	**Наименование поля**	**Тип хранимых данных**	**Может принимать значение NULL**
Первичный	Id	TEXT	нет
--	Base64String	TEXT	нет

3 Реализация приложения

Заключение

Список литературы

Приложение А

Приложение Б

Приложение В