Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина: Модели данных и системы управления базами данных

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе
на тему

ВЕБ-ПРИЛОЖЕНИЕ ДЛЯ РАБОТЫ ЗООМАГАЗИНА

БГУИР КП 1-40 04 01 029 ПЗ

Студент Д. С. Шевцова

Руководитель А. В. Давыдчик

Минск 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире бизнес сталкивается с необходимостью оперативно реагировать на вызовы, связанные с развитием технологий и изменениями потребностей клиентов. Наряду с традиционными магазинами для людей, активно развиваются зоомагазины, ориентированные на владельцев домашних животных. Этот рынок динамично растет, что приводит к высокой конкуренции. В условиях постоянных изменений бизнес должен обеспечить доступность информации и удовлетворить требования клиентов. Всё больше потребителей предпочитают онлайн-шоппинг и избегают походов в физические магазины.

Целью данной курсовой работы является создание веб-приложения для зоомагазина, которое будет включать базу данных для хранения и управления информацией о товарах, клиентах, заказах, питомцах и услугах. Разработанное приложение должно упростить процесс администрирования и ускорить доступ к информации, что особенно важно для обеспечения качественного обслуживания клиентов.

Исходя из цели проекта был составлен следующий перечень задач:

1 Определить и обосновать перечень информационных сущностей для выбранной предметной области.

2 Разработать структуру базы данных, которая охватывает все необходимые аспекты работы зоомагазина.

3 Реализовать механизм взаимодействия с сущностями приложения.

4 Создать приложение, использующее разработанную базу данных.

5 Создать графический интерфейс для взаимодействия с приложением, использующим разработанную базу данных.

В ходе разработки программного средства будут использованы принципы проектирования и современные технологии для создания надежной и безопасной базы данных, которая станет важным инструментом для работы зоомагазина. В конечном итоге данная система должна упростить рабочие процессы и минимизировать ошибки, связанные с человеческим фактором, что будет способствовать повышению качества обслуживания посетителей.

1 АРХИТЕКТУРА ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ

1.1 Структура и архитектура вычислительной системы

PostgreSQL – это объектно-реляционная система управления базами данных (ORDBMS), наиболее развитая из открытых СУБД в мире. Имеет открытый исходный код и является альтернативой коммерческим базам данных. С помощью PostgreSQL можно создавать, хранить базы данных и работать с данными с помощью запросов на языке SQL.

Одной из наиболее сильных сторон СУБД PostgreSQL является архитектура. Как и в случаях со многими коммерческими СУБД, PostgreSQL можно применять в среде клиент-сервер – это предоставляет множество преимуществ и пользователям, и разработчикам.

В основе PostgreSQL – серверный процесс базы данных, выполняемый на одном сервере. Доступ из приложений к данным базы PostgreSQL производится с помощью специального процесса базы данных. То есть клиентские программы не могут получать самостоятельный доступ к данным даже в том случае, если они функционируют на том же ПК, на котором осуществляется серверный процесс.пользователям, и разработчикам.

В основе PostgreSQL – серверный процесс базы данных, выполняемый на одном сервере. Доступ из приложений к данным базы PostgreSQL производится с помощью специального процесса базы данных. То есть клиентские программы не могут получать самостоятельный доступ к данным даже в том случае, если они функционируют на том же ПК, на котором осуществляется серверный процесс.

Типичная модель распределенного приложения СУБД PostgreSQL (рисунок 1.1):

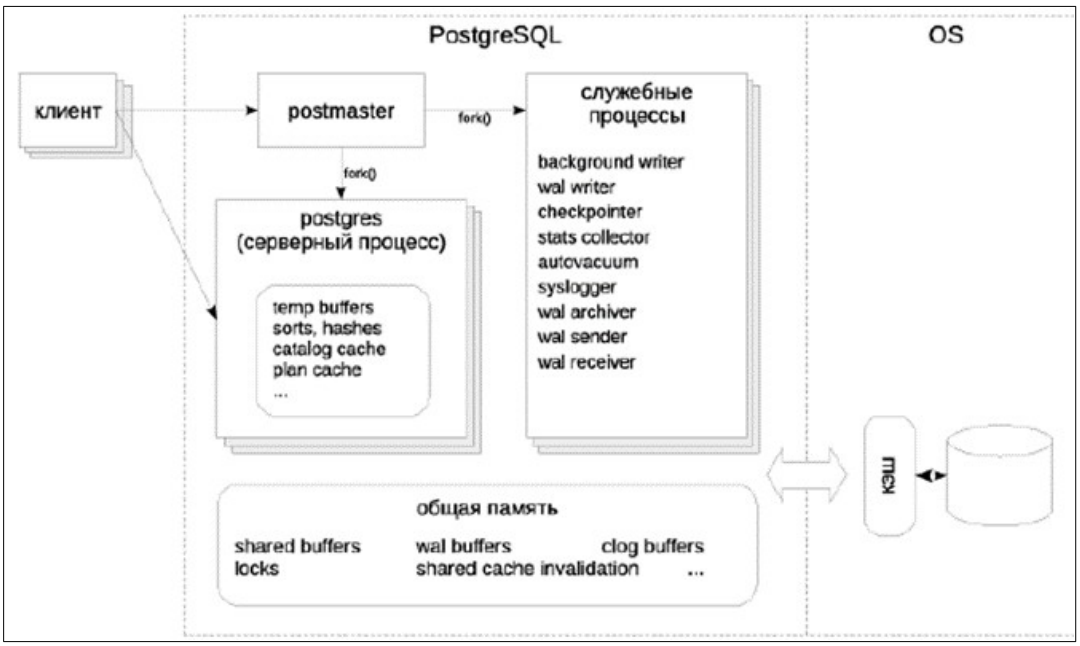

Рисунок 1.1 – Схема СУБД PostgreSQL

СУБД PostgreSQL ориентирована на протокол TCP/IP (локальная сеть либо Интернет), при этом каждый клиент соединён с главным серверным процессом БД (на рисунке 1.1 этот процесс обозначен Postmaster). Именно Postmaster создает новый серверный процесс специально в целях обслуживания запросов на доступ к данным определенного клиента.

Сервер PostgreSQL может обрабатывать несколько одновременных подключений от клиентов. Для этого он запускает новый процесс для каждого соединения. С этого момента клиент и новый серверный процесс обмениваются данными без вмешательства исходного процесса postgres. Таким образом, процесс сервера-супервизора всегда работает, ожидая клиентских подключений, в то время как клиентские и связанные серверные процессы приходят и уходят.

Данные, которыми управляет PostgreSQL, хранятся в базах данных. Один экземпляр PostgreSQL одновременно работает с несколькими базами, которые вместе называются кластером баз данных.

Каталог, в котором размещаются все файлы, относящиеся к кластеру, обычно называют словом PGDATA, по имени переменной окружения, указывающей на этот каталог.

При инициализации в PGDATA создаются три одинаковые базы данных (рисунок 1.2):

1 template0 используется, например, для восстановления из логической резервной копии или для создания базы в другой кодировке и никогда не должна меняться;

2 template1 служит шаблоном для всех остальных баз данных, которые может создать пользователь в этом кластере;

3 postgres представляет собой обычную базу данных, которую можно использовать по своему усмотрению.

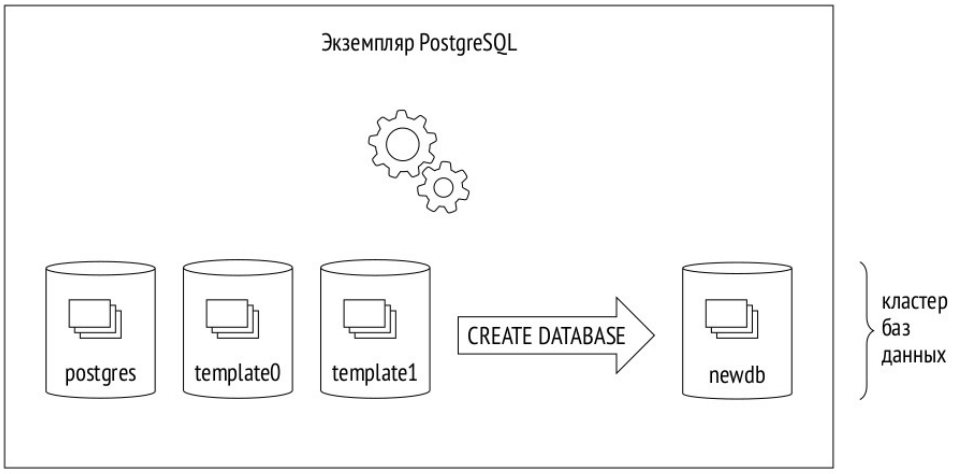

Рисунок 1.2 – Кластер PostgreSQL

Метаинформация обо всех объектах кластера (таких как таблицы, индексы, типы данных или функции) хранится в таблицах, относящихся к системному каталогу. В каждой базе данных имеется собственный набор таблиц (и представлений), описывающих объекты этой конкретной базы. Существует также несколько таблиц системного каталога, общих для всего кластера, которые не принадлежат какой-либо определенной базе данных и доступны в любой из них.

1.2 История, версии и достоинства

Ранние версии системы были основаны на старой программе POSTGRES University, созданной университетом Беркли: так появилось название PostgreSQL. И сейчас СУБД иногда называют «Постгрес». Существуют сокращения PSQL и PgSQL – они тоже обозначают PostgreSQL.

По состоянию на июнь 2024 года PostgreSQL занимает четвертое место в общемировом рейтинге популярных СУБД (рисунок 1.1).

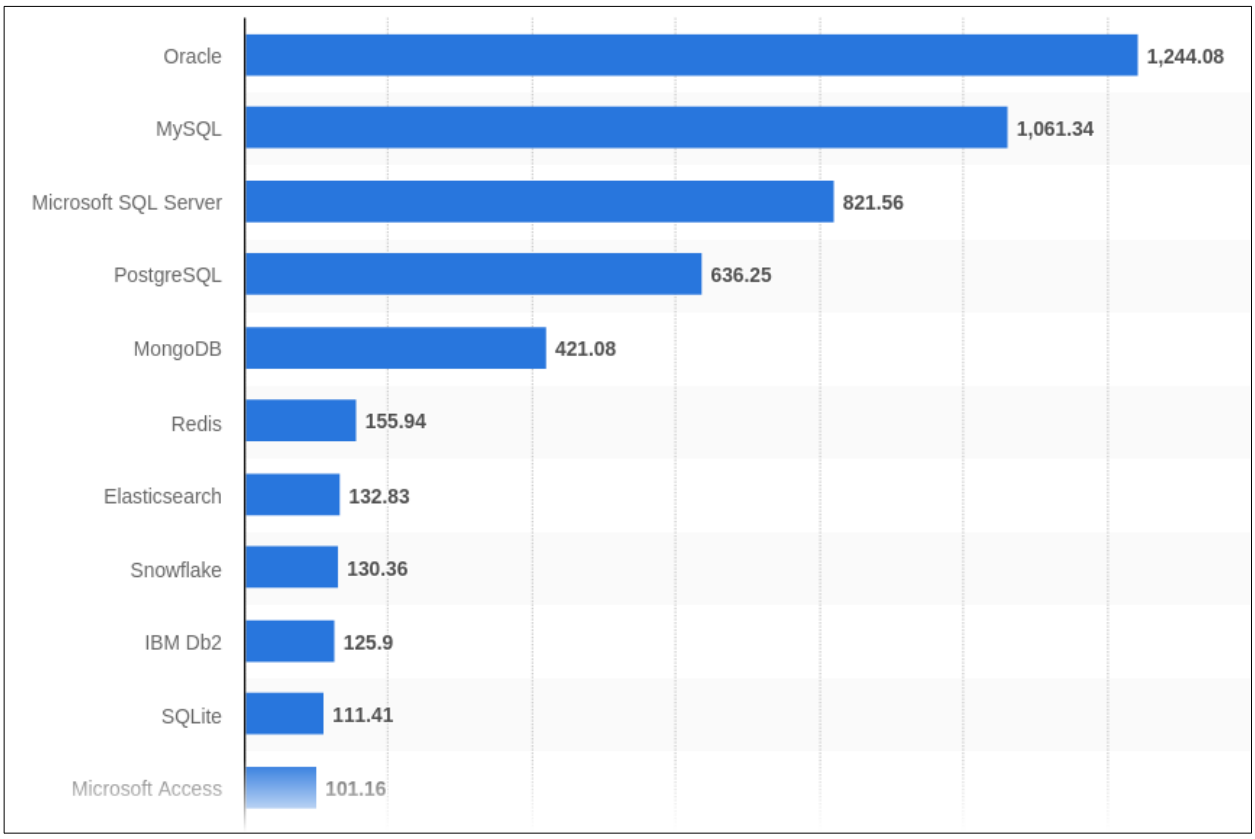

Рисунок 1.1 – Рейтинг популярности СУБД в июне 2024 года

У СУБД PostgreSQL много преимуществ, которые продолжают повышать ее популярность:

1 Любой специалист может бесплатно скачать, установить СУБД и сразу начать работу с базами данных.

2 PostgreSQL подходит для работы в любой операционной системе: Linux, macOS, Windows. Пользователь получает систему «из коробки» – чтобы установить и использовать программу, не нужны дополнительные инструменты.

3 PostgreSQL поддерживает много разных типов и структур данных, в том числе сетевые адреса, данные в текстовом формате JSON и геометрические данные для координат геопозиций. Все эти форматы можно хранить и обрабатывать в СУБД. Также при работе с PostgreSQL можно создавать собственные типы данных, их называют пользовательскими.

4 Размер базы данных в PostgreSQL не ограничен и зависит от того, сколько свободной памяти есть в месте хранения: на сервере, локальном компьютере или в облаке.

5 PostgreSQL реализует принципы ACID. Это четыре требования для надежной работы систем, которые обрабатывают данные в режиме реального

времени. Если все требования выполняются, данные не будут теряться из-за технических ошибок или сбоев в работе оборудования.

6 PostgreSQL поддерживает все современные функции баз данных: оконные функции, вложенные транзакции, триггеры.

7 Хотя большинство операций в PostgreSQL и используют классический стандарт языка SQL, помимо него поддерживается и свой отдельный диалект, позволяющий еще комфортнее писать запросы.

8 Поддерживается репликация «из коробки». Репликация – это сохранение копии базы данных. Копия может находиться на другом сервере.

9 PostgreSQL позволяет быстро без потерь перенести данные из другой СУБД.

10 Возможность одновременного доступа к базе с нескольких устройств. В СУБД реализована клиент-серверная архитектура, когда база данных хранится на сервере, а доступ к ней осуществляется с клиентских компьютеров. Для ситуаций, когда несколько человек одновременно модифицируют базу используется технология MVCC – Multiversion Concurrency Control, многоверсионное управление параллельным доступом.

Благодаря перечисленным выше преимуществам иногда PostgreSQL называют бесплатным аналогом Oracle Database. Обе системы адаптированы под большие проекты и высокую нагрузку. Но есть разница: они по-разному хранят данные, предоставляют разные инструменты и различаются возможностями. Важная особенность PostgreSQL в том, что эта система – feature-rich: так называют проекты с широким функционалом.

1.3 Обоснование выбора вычислительной системы

Для разработки приложения для зоомагазина была выбрана система управления базами данных PostgreSQL, так как она сочетает в себе высокую функциональность, надежность и популярность среди реляционных СУБД. К июню 2024 года PostgreSQL занимает одно из ведущих мест среди мировых СУБД, что подтверждает ее востребованность в разных областях, включая торговлю и онлайн-бизнес.

Одним из главных преимуществ PostgreSQL является свободная лицензия, что позволяет использовать систему без дополнительных затрат. Она поддерживается на всех популярных операционных системах, таких как Linux, macOS и Windows, что делает ее универсальной и доступной для использования в разных условиях. Также PostgreSQL предоставляет все

необходимые инструменты для работы с базами данных прямо «из коробки», не требуя дополнительных программных решений или надстроек.

PostgreSQL выделяется поддержкой различных типов данных и структур, что особенно важно для приложений, работающих с разнообразной информацией, такой как товары, заказы, клиенты и их питомцы. Помимо стандартных типов данных, PostgreSQL поддерживает работу с JSON, геометрическими данными и многими другими типами, что позволяет гибко управлять данными и хранить информацию о товарах и услугах зоомагазина.

Масштабируемость PostgreSQL является важным аспектом для зоомагазина, так как объем данных может увеличиваться по мере роста бизнеса. База данных PostgreSQL может работать с большими объемами данных, ограничиваясь только размерами доступной памяти сервера или облака, что позволяет эффективно масштабировать систему по мере необходимости.

Надежность PostgreSQL обеспечена ее соответствием стандартам ACID, что критически важно для обработки и хранения данных, таких как информация о клиентах и заказах. Это гарантирует целостность и безопасность данных, а также защищает от потерь и сбоев.

Кроме того, PostgreSQL поддерживает многоверсионный контроль параллельного доступа (MVCC), что позволяет нескольким пользователям одновременно работать с базой данных без блокировки. Это особенно важно для зоомагазина, где несколько сотрудников могут одновременно обрабатывать заказы, обновлять информацию о товарах или консультировать клиентов.

Таким образом, PostgreSQL была выбрана для разработки приложения для зоомагазина благодаря своей надежности, функциональности, поддержке различных типов данных и масштабируемости, что делает ее идеальной СУБД для решения задач управления данными в сфере торговли и обслуживания клиентов.

2 ПЛАТФОРМА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1 Выбор операционной системы

Linux – это операционная система на базе UNIX с открытым исходным кодом. Основным компонентом операционной системы Linux является ядро Linux. Он разработан для предоставления недорогого или бесплатного обслуживания операционной системы пользователям персональных систем, включая систему X-window, редактор Emacs, графический интерфейс IP/TCP и т. д .

Минимальный набор системных библиотек был разработан как часть проекта GNU, с улучшениями, разработанными сообществом Linux. Средства сетевого администрирования Linux были разработаны на основе версии 4.3 Berkeley Software Distribution UNIX. Недавние производные от BSD (например, UNIX FreeBSD), в свою очередь, заимствовали код из Linux.

Система Linux поддерживается слабо связанной сетью разработчиков, взаимодействующих через Internet. Небольшое число публично доступных ftp-серверов используется как хранилища информации о дефакто стандартах.

Стандартный предварительно откомпилированный набор пакетов, или дистрибутивов, включает базовую систему Linux, утилиты для инсталляции системы и управления системой, а также готовые к инсталляции пакеты инструментов для UNIX. Ранние дистрибутивы включали диалекты SLS и Slackware. Red Hat и Debian – популярные дистрибутивы, соответственно, основанные на коммерческих и некоммерческих исходных кодах. [10]

Ядро Linux – один из крупнейших проектов с открытым исходным кодом в мире, код которого вносят тысячи разработчиков, и для каждого выпуска вносятся миллионы строк кода. Оно распространяется под лицензией GPLv2, которая требует, чтобы любая модификация ядра, выполненная в программном обеспечении, поставляемом клиентам, была доступна им (клиентам), хотя на практике большинство компаний делают исходный код общедоступным.

Существует множество компаний (часто конкурирующих), которые вносят свой код в ядро Linux, а также представители научных кругов и независимые разработчики. Текущая модель разработки основана на выпуске релизов через фиксированные промежутки времени (обычно 3–4 месяца). Новые функции добавляются в ядро в течение одной или двух недель. После окна слияния еженедельно создается кандидат на выпуск (rc1, rc2 и т. д.).

Как и любая операционная система, Linux состоит из программного обеспечения, компьютерных программ, документации и аппаратного обеспечения. Основными компонентами операционной системы Linux являются: приложение, оболочка, ядро, оборудование, утилиты. Расположение компонентов можно увидеть на рисунке 2.1.

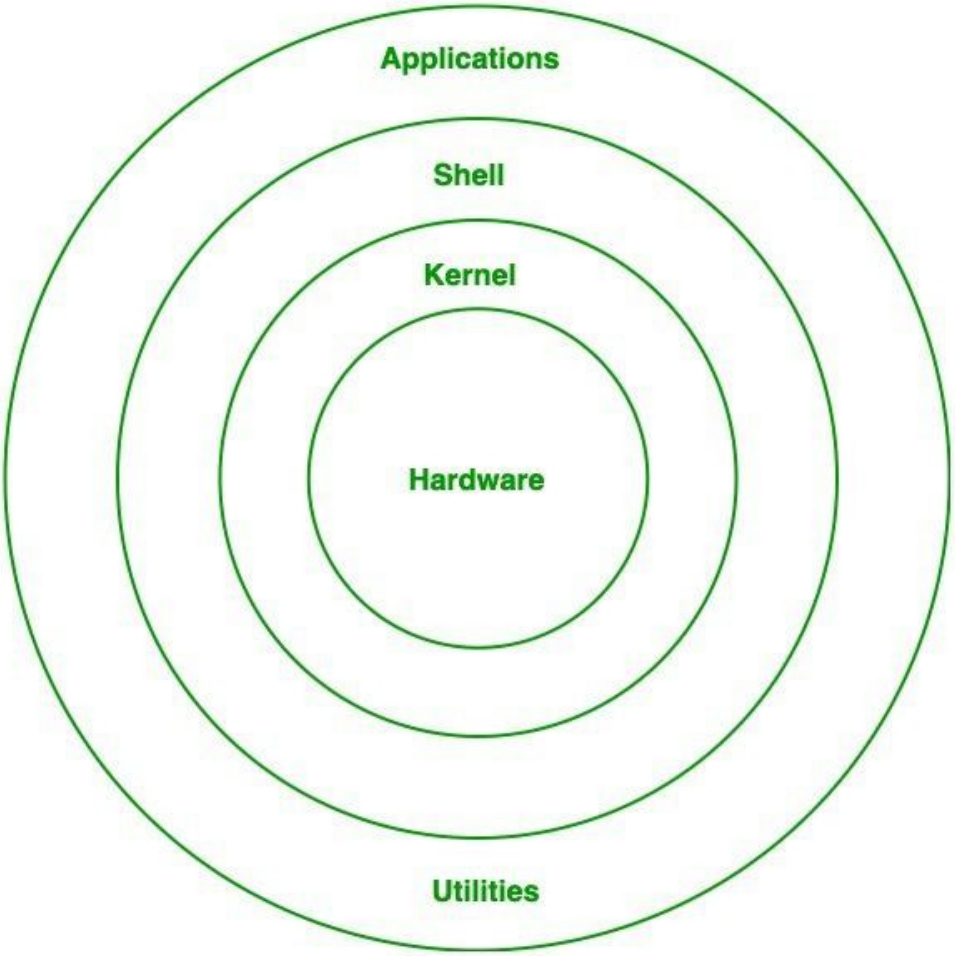

Рисунок 2.1 – Архитектура операционной системы Linux

Ядро (Hardware) – это основной компонент Linux, он контролирует активность других аппаратных компонентов. Он визуализирует общие аппаратные ресурсы и обеспечивает каждый процесс необходимыми виртуальными ресурсами. Это заставляет процесс ждать в очереди готовности и последовательно выполняться, чтобы избежать каких-либо конфликтов.

Существуют различные виды ядер, такие как монолитное, микроядро, экзоядро и гибридное ядро.

Монолитное ядро – это тип ядра операционной системы, в котором все параллельные процессы выполняются одновременно в самом ядре. Все процессы используют одни и те же ресурсы памяти.

В микроядре пользовательские службы и службы ядра выполняются в отдельных адресных пространствах. Пользовательские службы хранятся в адресном пространстве пользователя, а службы ядра – в адресном пространстве ядра.

Exo-kernel предназначен для управления аппаратными ресурсами на уровне приложения. В этой операционной системе используется абстракция высокого уровня, чтобы обеспечить доступ к аппаратным ресурсам ядра.

Это сочетание монолитного ядра и микроядра. Он обладает скоростью и дизайном монолитного ядра, а также модульностью и стабильностью микроядра.

Системные библиотеки (Kernel) – это некоторые предопределенные функции, с помощью которых любые прикладные программы или системные утилиты могут получить доступ к функциям ядра. Эти библиотеки являются основой, на которой может быть построено любое программное обеспечение.

Некоторые из наиболее распространенных системных библиотек: GNU C, libpthread, libdl, libm.

Библиотека GNU C – это библиотека C, которая предоставляет наиболее фундаментальную систему для интерфейса и выполнения программ C. Это обеспечивает множество встроенных функций для выполнения.

Libpthread (POSIX Threads) – библиотека играет важную роль в многопоточности в Linux, она позволяет пользователям создавать и управлять несколькими потоками.

Libdl (динамический компоновщик) – библиотека отвечает за загрузку и связывание файла во время выполнения.

Libm (математическая библиотека) – эта библиотека предоставляет пользователю все виды математических функций и их выполнение.

Некоторые другие системные библиотеки: librt (библиотека реального времени), libcrypt (криптографическая библиотека), libnss (библиотека переключения службы имен), libstdc++ (стандартная библиотека C++).

Оболочка (Shell) – это, в своем роде, интерфейс ядра, который скрывает от пользователя внутреннее выполнение функций ядра. Пользователи могут просто ввести команду и, используя функцию ядра, выполнить конкретную задачу соответствующим образом.

Существует два типа оболочек: оболочка командной строки и графический интерфейс пользователя. Первая выполняет команду,

предоставленную пользователем, указанную в форме команды. Выполняется специальная программа под названием терминал, и результат отображается в самом терминале. Вторая же выполняет процесс, предоставленный пользователем, в графическом виде, а выходные данные отображаются в графическом окне.

На рисунке 2.2 представлено графическое отображение оболочки Linux.

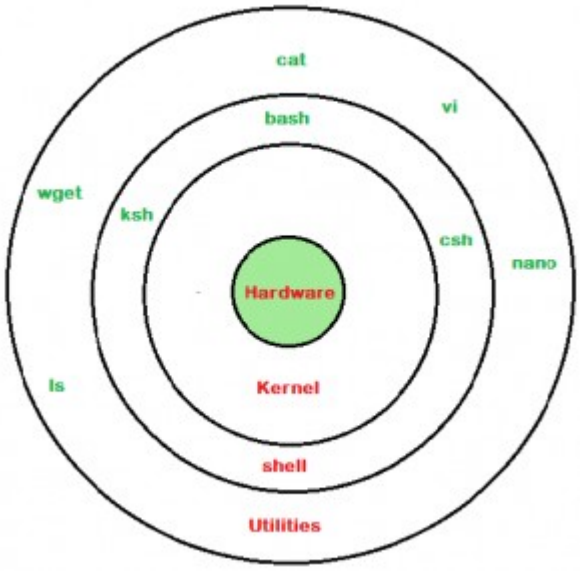

Рисунок 2.2 – Оболочка Linux

Аппаратный уровень Linux (Applications) – это самый нижний уровень операционной системы. Он играет жизненно важную роль в управлении всеми аппаратными компонентами. Он включает в себя драйверы устройств, функции ядра, управление памятью, управление процессором и операции ввода-вывода. Этот уровень обобщает высокую сложность, предоставляя интерфейс для программного обеспечения и гарантируя правильную функциональность всех компонентов.

Системные утилиты (Utilities) – это инструменты командной строки, которые выполняют различные задачи, предоставляемые пользователем, чтобы улучшить управление и администрирование системы. Эти утилиты позволяют пользователю выполнять различные задачи, такие как управление файлами, мониторинг системы, настройка сети, управление пользователями и т. д.

Самым мощным и универсальным компонентом операционной системы Linux является ядро, с помощью которого можно управлять функционалом всей операционной системы. Ядро предоставляет огромный

набор функций, к которым пользователь может легко получить доступ в интерактивном режиме с помощью оболочки. Для правильной работы операционной системы необходимо подходящее оборудование. Все компоненты операционной системы делают ее проще, быстрее, стабильнее и надежнее. [11]

У операционной системы Linux существует множество плюсов для решения типичных задач программирования. Однако стоит выделить основные.

Если сравнивать с проприетарными ОС, то главный плюс GNU/Linux, как и десятка других свободных ОС (например, Free/Net/OpenBSD, OpenIndiana), это то, что они являются свободным ПО. Это означает что, пользователь может без ограничений запускать и использовать эти ОС для любых целей, изучать и модифицировать их работу. А также помогать окружающим, распространяя копии ОС и их модификации.

Широкая поддержка аппаратного обеспечения. Много драйверов для устройств, особенно на домашних системах (где дешевые не серверные компоненты). Есть вероятность что какое-либо оборудование не будет поддерживаться в системе типа BSD или OpenIndiana.

Многие дистрибутивы GNU/Linux могут работать на старых компьютерах гораздо лучше, чем системы типа Windows или macOS. Они зачастую могут вообще отказаться на них работать.

Активная поддержка пользователей. За десятилетия существования у GNU/Linux образовался круг пользователей и разработчиков, которые смогут оперативно помочь с задачами или проблемами, возникающими при работе у неопытных пользователей.

Плюсом персонально для разработчиков является возможность переделать ОС под ваши задачи. Можно доработать как всю систему, так и отдельные ее компоненты, найти и исправлять недочеты или нанять разработчиков для этих задач. С несвободным программным обеспечением, есть только надежда, что компания владелец ПО соизволит выполнить ваше желание, еще и за вменяемый срок. А также возможность делиться своими наработками и исправлениями. [12]

2.2 Выбор платформы для написания программы

Python является одним из наиболее популярных и универсальных языков программирования, и его идеальность для программирования проявляется во многих аспектах, таких как синтаксис. Python прост и

лаконичен, что делает его легким для изучения и использования. Это способствует написанию чистого и читаемого кода, что упрощает поддержку и совместную работу. Python имеет огромное количество стандартных библиотек, покрывающих практически все аспекты разработки, начиная от обработки данных и веб-разработки, и заканчивая машинным обучением и искусственным интеллектом. Это позволяет разработчикам быстро создавать функциональные приложения без необходимости писать код с нуля. Также он является языком с динамической типизацией, что означает, что разработчики могут создавать и изменять переменные без необходимости объявления типа заранее. Это упрощает процесс разработки и делает код более гибким. И нельзя забыть огромное и активное сообщество разработчиков, которые создают и поддерживают библиотеки, фреймворки и инструменты для разработки. Это обеспечивает доступ к большому объему ресурсов, документации и поддержки, что помогает разработчикам быстро решать проблемы и улучшать свои навыки.

Для написания кода будет использоваться среда разработки PyCharm.

PyCharm – это программное обеспечение, которое представляет собой интегрированную среду разработки для языка программирования Python.

Использование IDE PyCharm позволяет значительно ускорить разработку программного обеспечения, минимизировать ошибки и облегчить сотрудничество между программистами. Благодаря широкому спектру возможностей, эта интегрированная среда разработки помогает создавать качественное ПО в короткие сроки.

К основным возможностям PyCharm относятся:

1 Интеллектуальное автодополнение кода. PyCharm анализирует код и предлагает варианты автодополнения, которые соответствуют контексту. Это значительно ускоряет написание кода и уменьшает количество ошибок.

2 Проверка кода. PyCharm проверяет код на наличие ошибок и потенциальных проблем, выделяя их прямо в редакторе. Это помогает писать более качественный и надежный код.

3 Отладка. PyCharm включает в себя мощный отладчик, который позволяет пошагово выполнять код, устанавливать точки останова и исследовать значения переменных.

4 Рефакторинг. PyCharm предлагает различные инструменты рефакторинга, которые помогут легко изменить структуру вашего кода, не нарушая его функциональность.

5 Интеграция с системами контроля версий. PyCharm интегрируется с популярными системами контроля версий, такими как Git, Mercurial и SVN, что упрощает управление вашим кодом.

6 Поддержка различных фреймворков. PyCharm предоставляет поддержку популярных Python-фреймворков, таких как Django, Flask и Pyramid.

7 Множество плагинов. PyCharm имеет богатую экосистему плагинов, которые расширяют его функциональность.

3 ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРОГРАММНОГО ПРОДУКТА

3.1 Обоснование необходимости разработки

Современные зоомагазины сталкиваются с рядом вызовов, связанных с эффективным управлением товарами, заказами и обслуживанием клиентов. В условиях роста конкуренции и изменяющихся потребностей потребителей, необходима система, которая обеспечит удобство и оперативность в обслуживании, а также повысит общую эффективность работы магазина.

С каждым годом все больше людей предпочитают совершать покупки через интернет, включая покупку товаров для своих питомцев. Веб-приложение для зоомагазина позволяет клиентам быстро и удобно выбирать товары, делать заказы и оплачивать их без необходимости посещать физический магазин. Это особенно важно для людей, у которых нет времени на походы в магазин, а также для тех, кто ищет редкие или специализированные товары, которые могут быть недоступны в обычных зоомагазинах.

В зоомагазинах часто обновляется ассортимент товаров, и управление ими вручную становится сложной задачей. Веб-приложение позволит автоматизировать процессы учета товарных запасов, обновления данных о наличии товаров и их цен. Это улучшит процессы инвентаризации и предотвратит ошибки, связанные с некорректной информацией о товаре.

Веб-приложение будет доступно с любого устройства, что позволяет владельцам и сотрудникам зоомагазина иметь доступ к информации в любое время и из любого места. Это повышает гибкость работы магазина и позволяет эффективно управлять бизнесом, независимо от физического местоположения. В целях повышения удобства и доступности для пользователей будет разработан интуитивно понятный интерфейс, который позволит быстро находить нужные товары, оформлять заказы и следить за статусом доставки. Также будет предусмотрена система фильтров и поиска для удобства работы с большим ассортиментом товаров. Разработка данного веб-приложения необходима для обеспечения конкурентоспособности зоомагазина в условиях быстро развивающегося рынка и изменения потребительских предпочтений. Это позволит не только улучшить взаимодействие с клиентами, но и повысить эффективность работы бизнеса в целом.

3.2 Технологии программирования, используемые для решения поставленных задач

В качестве языка программирования для написания программы используется Python.

Python – высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно-ориентированным в том плане, что всё является объектами.

Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации. Сам же язык известен как интерпретируемый и используется в том числе для написания скриптов[. Недостатками языка являются зачастую более низкая скорость работы и более высокое потребление памяти написанных на нём программ по сравнению с аналогичным кодом, написанным на компилируемых языках, таких как C или C++.

Python предоставляет удобные инструменты для отладки и разработки, что облегчает создание и отладку кода, работающего с потоками и процессами.

Графический интерфейс приложения будет реализован с использованием фреймворка FastAPI. FastAPI – это современный и высокопроизводительный веб-фреймворк для Python, который позволяет быстро создавать API. Он использует асинхронные операции, поддерживает современные стандарты Python и автоматически генерирует документацию (Swagger UI и ReDoc). FastAPI является открытым и бесплатным, что делает его подходящим для проектов любого масштаба. Среди его ключевых преимуществ – высокая производительность благодаря использованию Uvicorn и Starlette, интуитивный синтаксис с поддержкой аннотаций типов, встроенные средства безопасности и модульность. Это упрощает разработку, позволяет легко масштабировать приложения и предоставляет готовые инструменты для защиты от атак CSRF, SQL-инъекций и других угроз.

FastAPI использует гибкую архитектуру, которая адаптируется к стандартам MVC. Работа с данными осуществляется через ORM, например SQLAlchemy или Tortoise ORM. Модели данных, описывающие структуру таблиц базы данных, размещаются в файле models.py. Для обработки HTTP-запросов используются маршруты, описываемые в файле routes.py с

помощью декораторов вроде @app.get() или @app.post(). Они взаимодействуют с моделями и возвращают клиенту соответствующие ответы. Шаблоны HTML рендерятся с использованием движка Jinja2, файлы располагаются в папке templates, а статические файлы – в папке static. Рендеринг осуществляется через модуль fastapi.templating.

FastAPI позволяет разрабатывать приложения быстрее благодаря автоматической генерации документации, поддержке асинхронных операций и встроенной проверке данных через Pydantic. Он интегрируется с различными библиотеками и поддерживает модульную структуру, что упрощает масштабирование и распределение нагрузки. Пример структуры проекта включает файлы main.py для запуска приложения, models.py для описания моделей данных, routes.py для маршрутов, schemas.py для валидации данных, а также каталоги templates и static для шаблонов и статических файлов. Такой подход позволяет эффективно решать задачи разработки, сохраняя гибкость и удобство использования.

4 ПРОЕКТИРОВАНИЕ ФУНКЦИОНАЛЬНЫХ ВОЗМОЖНОСТЕЙ ПРОГРАММЫ

4.1 Подключение к базе данных

4.2 Регистрация и авторизация пользователей

4.3 Управление пользователями

4.4 Взаимодействие с сущностями приложения

4.5 Общее описание системы

4.6 Руководство пользователя

5 ПРОЕКТИРОВАНИЕ РАЗРАБАТЫВАЕМОЙ БАЗЫ ДАННЫХ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Данная база данных предназначена для системы управления зоомагазином. Она организована таким образом, чтобы поддерживать ключевые бизнес-процессы: управление товарами, категориями, заказами, корзинами, пользователями, ролями и купонами. В базе данных определено 10 сущностей, которые связаны между собой через различные типы отношений.

5.1 Разработка информационной модели

Процесс разработки информационной модели базы данных для программного обеспечения зоомагазина включал несколько этапов, направленных на создание оптимальной структуры данных, обеспечивающей эффективное хранение и обработку информации.

При проектировании базы данных для зоомагазина был проведен анализ существующих онлайн-магазинов, чтобы понять основные требования к системе и эффективные модели данных. Подобные сайты используют базы данных, которые включают сущности, такие как товары, пользователи, заказы, корзины, а также систему скидок и акций. Все эти элементы связаны через реляционные таблицы, где товары классифицируются по категориям, пользователи могут создавать заказы и управлять корзинами, а купоны предоставляют скидки на покупки.

На следующем этапе было определено, какие данные необходимо хранить в системе. Были выделены ключевые сущности, такие как товары, фирмы, категории товаров, типы животных, корзина, заказы, пользователи и их действия, роли и купоны. Каждая сущность была тщательно изучена, чтобы определить ее характеристики и взаимодействия с другими объектами.

На этапе разработки модели было уделено особое внимание обеспечению корректной структуры данных, минимизации избыточности и исключению аномалий. Каждая сущность была нормализована с целью обеспечения целостности данных и предотвращения дублирования информации. Например, для предотвращения аномалий в данных были использованы ключи и ограничения целостности, такие как уникальные индексы, внешние ключи и обязательные поля. Это обеспечивало правильное связывание данных и минимизацию ошибок при их изменении или удалении.

Конкретная структура таблиц была выбрана с учетом специфики системы и потребностей зоомагазина. Например, связь "многие ко многим" между заказами и товарами, реализуемая через вспомогательную таблицу "OrderGood", была выбрана для того, чтобы точно отслеживать, какие товары были заказаны в рамках конкретного заказа. Такая структура позволяет эффективно работать с большими объемами данных, а также легко расширять функциональность системы. Использование вспомогательных таблиц для реализации сложных связей между сущностями, таких как "CartGood" или "CouponUser", позволяет минимизировать дублирование информации и улучшает гибкость системы при добавлении новых функций.

В результате, данная структура таблиц соответствует задачам системы и позволяет эффективно управлять данными о товарах, заказах, пользователях и купонах, обеспечивая при этом минимизацию избыточности и исключение аномалий. Спроектированная информационная модель позволяет системе работать быстро и слаженно, поддерживая высокую степень масштабируемости и надежности.

5.2 ER-диаграмма базы данных

ER-диаграмма базы данных (диаграмма «сущность-связь») представляет собой визуальное отображение структуры базы данных, где схематично показаны основные сущности, их ключевые атрибуты, а также связи, которые образуют логику взаимодействия между ними. Такие диаграммы служат основой для проектирования и понимания того, как будет организована и функционировать база данных в рамках системы. Важной составляющей ER-диаграммы является наличие различных типов связей между сущностями, таких как «один к одному», «один ко многим» и «многие ко многим», что позволяет адекватно отразить взаимоотношения в реальной предметной области.

С помощью ER-диаграммы можно наглядно понять, как данные будут храниться и обрабатываться в базе, а также как они будут взаимодействовать друг с другом. Это представление позволяет избежать избыточности данных, уменьшить возможность возникновения аномалий при обработке информации и гарантирует, что база данных будет нормализована, что в свою очередь способствует ее эффективному функционированию и высокой производительности. Разработка ER-диаграммы является неотъемлемой частью процесса проектирования, поскольку она помогает определить

структуру и логику базы данных до того, как она будет реализована в техническом плане.

На рисунке 5.1 представлена ER-диаграмма базы данных, отражающая ключевые сущности, их атрибуты и взаимосвязи, которые обеспечивают полноценное функционирование системы.

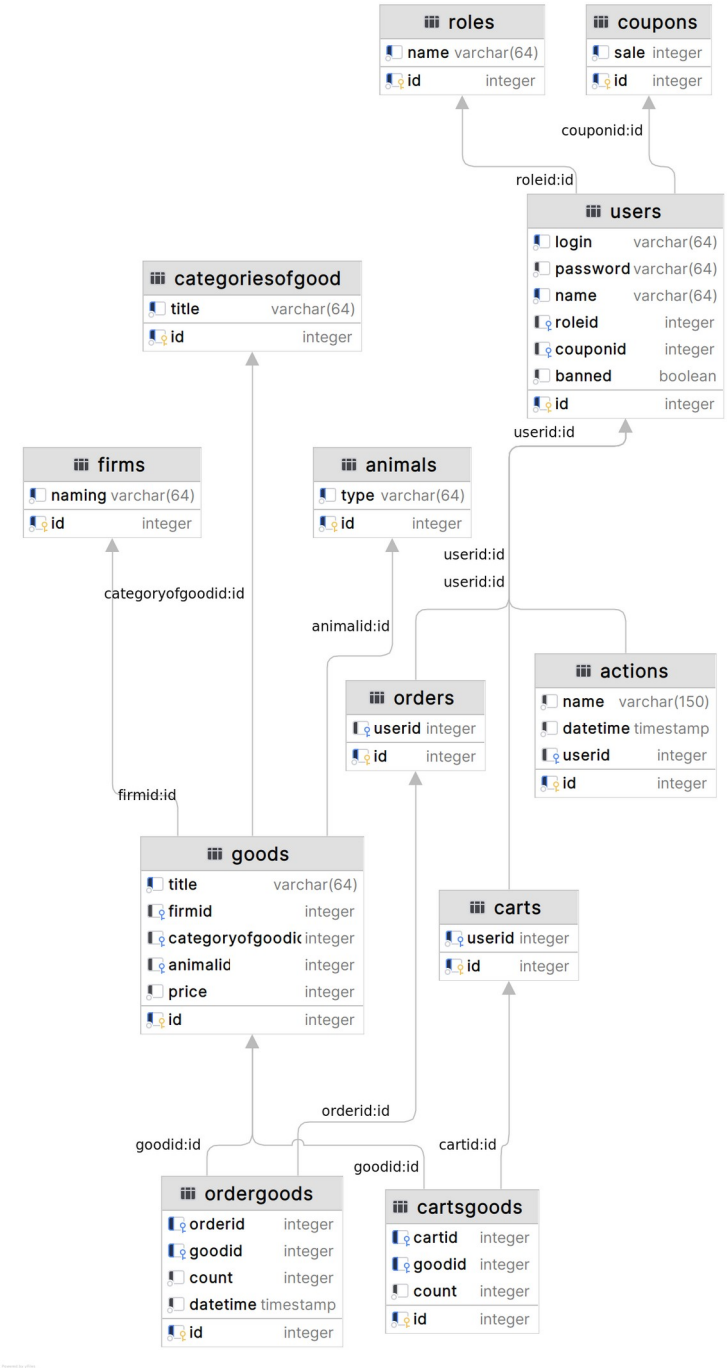

Рисунок 5.1 – ER-диаграмма базы данных

5.3 Оптимизация и целостность разработанной базы данных

Оптимизация структуры базы данных является ключевым этапом в проектировании информационной системы, направленным на повышение производительности, обеспечение целостности данных и минимизацию избыточности. Одним из способов оптимизации является использование триггеров, которые представляют собой специальные процедуры, автоматически выполняющиеся в ответ на изменения в данных. Триггеры могут использоваться для поддержания бизнес-логики, такой как проверка наличия определенных данных перед выполнением операции, или для автоматического выполнения дополнительных действий, например, создание записей в других таблицах.

Например, можно реализовать триггеры для предотвращения удаления или изменения критических данных, если они нарушают бизнес-правила, такие как отсутствие администратора в системе или попытка удаления данных, которые должны быть сохранены для целостности системы. Триггеры также могут использоваться для автоматического создания связанных записей в других таблицах, когда новые данные добавляются в базу, что исключает необходимость вручную контролировать такие зависимости.

Для оптимизации быстродействия базы данных важную роль играют индексы. Индексы создаются на столбцах, которые используются для поиска и фильтрации данных, таких как идентификаторы, внешние ключи или часто используемые атрибуты, например, названия товаров или электронные адреса пользователей. Индексы значительно ускоряют операции выборки, сортировки и соединения таблиц, однако важно, чтобы индексы использовались разумно, так как их избыточное применение может привести к ухудшению производительности при вставке, обновлении или удалении данных.

Еще одной важной частью оптимизации является нормализация базы данных. Нормализация представляет собой процесс реорганизации данных в таблицах с целью уменьшения избыточности и предотвращения аномалий, таких как дублирование данных или трудности при обновлении информации. Нормализация обеспечивает целостность данных и упрощает управление ими. Однако в некоторых случаях, например, при необходимости ускорения чтения данных, может быть разумным применение денормализации, когда часть данных избыточно сохраняется в таблицах для ускорения запросов.

Кроме того, для улучшения производительности базы данных могут быть использованы асинхронные операции и оптимизация запросов. Асинхронные запросы позволяют не блокировать выполнение других операций, что повышает скорость отклика системы. Оптимизация запросов включает использование более эффективных способов выборки данных, минимизацию количества соединений между таблицами и правильное использование агрегатных функций.

В итоге, оптимизация структуры базы данных направлена на обеспечение ее высокой производительности, минимизацию затрат на хранение данных и предотвращение возможных ошибок при их обработке. Эффективное использование триггеров, индексов, нормализации и других методов помогает добиться этих целей, создавая основу для надежной и быстродействующей системы.

5.4 Описание базы данных

На основе диаграммы базы данных, можно описать проектирование разработанной базы данных, включая сущности, связи и функциональное назначение каждой таблицы.

Сущность User представляет пользователей системы и включает такие атрибуты, как идентификатор пользователя, имя, адрес электронной почты, пароль и идентификатор роли. Связь с сущностью Role реализована по схеме "многие-к-одному", где каждому пользователю соответствует одна роль, например, администратор или клиент. Также сущность User связана с сущностями Cart ("один-к-одному", так как у одного пользователя может быть только одна корзина) и Order ("один-ко-многим", так как пользователь может иметь несколько заказов). Кроме того, связь с CouponUser ("многие-ко-многим") позволяет отслеживать использование купонов конкретными пользователями.

Сущность Role содержит перечень ролей с их уникальными идентификаторами и названиями. Она связана с User, чтобы каждому пользователю назначалась одна роль. Это связь типа "один-ко-многим", поскольку одна роль может быть назначена нескольким пользователям.

Сущность Cart представляет корзину покупателя и связана с User по схеме "один-к-одному", так как один пользователь может иметь только одну корзину. Также сущность Cart связана с сущностью CartGood по схеме "один-ко-многим", что позволяет добавлять в корзину несколько товаров.

Сущность CartGood связывает корзины и товары через их идентификаторы. Это промежуточная сущность, реализующая связь "многие-ко-многим" между Cart и Good. В ней хранятся данные о количестве конкретного товара в корзине.

Сущность Good представляет товары, доступные в магазине. Она связана с GoodsType ("многие-к-одному"), что позволяет классифицировать товары по типу, и с CategoryOfGood ("многие-к-одному"), что позволяет группировать товары по категориям. Также сущность Good связана с CartGood и OrderGood, что обеспечивает возможность добавления товаров в корзину и оформления заказов.

Сущность GoodsType определяет типы товаров, такие как "игрушки", "еда" или "аксессуары". Она связана с Good по схеме "один-ко-многим", так как один тип может включать множество товаров.

Сущность CategoryOfGood отвечает за категории товаров, такие как "для собак" или "для кошек". Связь с Good также реализована по схеме "один-ко-многим", что позволяет одной категории включать множество товаров.

Сущность Animal используется для описания животных, для которых предназначены товары. Она связана с Good по схеме "один-ко-многим", что позволяет определять, для какого животного предназначен каждый товар.

Сущность Order хранит информацию о заказах, сделанных пользователями. Она связана с User по схеме "многие-к-одному", так как один пользователь может иметь несколько заказов. Также она связана с OrderGood по схеме "один-ко-многим", чтобы включать информацию о заказанных товарах.

Сущность OrderGood связывает заказы и товары, реализуя связь "многие-ко-многим" между Order и Good. В ней хранятся данные о количестве каждого товара в заказе.

Сущность Coupon представляет скидочные купоны. Она связана с CouponUser по схеме "один-ко-многим", так как один купон может быть использован несколькими пользователями. Также она может быть связана с Order, чтобы фиксировать использование купонов в заказах.

Сущность CouponUser является промежуточной таблицей, реализующей связь "многие-ко-многим" между Coupon и User. Она фиксирует, какие купоны были использованы конкретным пользователем.

Сущность Logging хранит данные о действиях пользователей в системе. Она связана с User по схеме "многие-к-одному", так как один пользователь может выполнить множество действий, фиксируемых в логах.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной курсовой работы было проведено всестороннее исследование и разработка веб-приложения для зоомагазина. Рассмотрены ключевые аспекты работы таких приложений, начиная с их роли в оптимизации процессов управления товарными запасами, заказами и взаимодействием с клиентами, и заканчивая возможностями использования современных технологий для улучшения качества обслуживания. Особое внимание уделено анализу программных систем и технологий, используемых для создания эффективных веб-приложений, таких как базы данных, серверные технологии и интерфейсы взаимодействия с пользователем.

В работе был также проведен анализ потребностей зоомагазинов и их клиентов, выявлены основные задачи, решаемые с помощью веб-приложения, и предложены оптимальные решения для их реализации. В рамках разработки веб-приложения использовались такие технологии, как HTML, CSS, JavaScript для фронтенда, а также выбор базы данных и серверной части для хранения информации о товарах, заказах и клиентах.

Результатом работы стало создание полноценного веб-приложения, которое обеспечит удобство и доступность для клиентов, а также улучшит процесс управления магазином для владельцев и сотрудников. Приложение предлагает интуитивно понятный интерфейс, функции для поиска товаров, оформления заказов и взаимодействия с клиентами, что способствует улучшению качества обслуживания и повышению конкурентоспособности зоомагазина.

Таким образом, результатом выполнения курсовой работы стало веб-приложение для зоомагазина, которое успешно решает задачи автоматизации процессов торговли, упрощает взаимодействие с клиентами и позволяет бизнесу эффективно управлять ассортиментом и заказами. Это решение также способствует улучшению пользовательского опыта и повышению операционной эффективности бизнеса.

СПИСОК ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ

[1] Архитектура Zen: сколько поколений продержится главная технология AMD [Электронный ресурс]. – Режим доступа: https://club.dns-shop.ru/blog/t-100-protsessoryi/61416-arhitektura-zen-skolko-pokolenii-proderjitsya-glavnaya-tehnologi/ – Дата доступа: 01.10.2023.

[2] Поколения процессоров AMD Ryzen [Электронный ресурс]. – Режим доступа: https://te4h.ru/pokoleniya-protsessorov-amd-ryzen – Дата доступа: 01.10.2023.

ПРИЛОЖЕНИЕ А
(обязательное)
Листинг программного кода

```
To be continued
```

ПРИЛОЖЕНИЕ Б
(обязательное)
Конечная схема базы данных

ПРИЛОЖЕНИЕ В
(обязательное)
Ведомость курсового проекта