ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА МАТЕМАТИЧЕСКОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

СОЗДАНИЕ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ УЧЕТА СКЛАДСКОЙ ПРОДУКЦИИ НА ПРЕДПРИЯТИИ „АГРОШИНА 31“

Выпускная квалификационная работа
обучающегося по направлению подготовки 02.03.02
Фундаментальная информатика и информационные технологии
заочной формы обучения, группы 07001350
Кулакова Антона Сергеевича

Научный руководитель
к.т.н., доцент
Бурданова Е.В.

БЕЛГОРОД 2018

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Еще совсем недавно обработка данных в областях управления складской продукции чаще всего осуществлялась вручную. Канцелярия, наборы всякого рода бумаг и калькулятор — все это именно то, что ранее являлось основными инструментами любого бухгалтера. Однако сейчас в 21 веке для управления складским хозяйством, а так же любой другой организацией, необходимо легкое, быстрое и комплексное ведение учёта.

Повсеместно во многих предприятиях по продаже товаров и услуг, как правило, работает только один бухгалтер, который должен делать все записи в ручную, и поэтому ему часто необходимо выполнять большой объем работы. Из-за этого, в частности, существует потребность в ПО, которое поможет сделать работу бухгалтера на предприятии по продаже товаров более автоматизированной.

Автоматизация — это вектор движения научно-технического прогресса, применимый для саморегуляции технических средств и устройств, экономико-математических методик и разных систематик управления, освобождающих человека как сущность от участия в процессах получения, преобразования, переработки и использования энергии, средств или информации, что существенно уменьшает степень его участия, тем самым снижая трудоёмкость выполняемой операции на предприятии.

Использование автоматизированной системы должно решить ряд проблем, которые возникают с рабочим процессом по мере его автоматизации в компании по продаже шин «Агрошина 31». Поэтому, чтоб предоставлять высокое качество услуг в текущий момент, сферы розничной и оптовой торговли, а также, любые другие сферы услуг, нуждается в современных и качественных решениях.

Использование новейших онлайн-технологий, которые повсеместно внедряются во все сферы нашей жизни, без компромиссов, сделают этот процесс максимально легким, быстрым и эффективным, и достаточно простым.

Это облегчит и ускорит работу не только бухгалтера, а так же целого ряда сотрудников — от грузчиков до кладовщиков. Так же, это сильно поможет снизить вероятности обнаружения ошибок при работе с массивами данных. Основные части, которые необходимо автоматизировать на предприятии — это учетная информация о состоянии пришедших и ушедших товаров; расчёт начислений и сведения об оплате. Создание квитанции для оплаты и их заполнение для каждого отдельного клиента. Резервы заказов и получение готового товара. Последний вид работ особенно рутинный и отнимает колоссальное количество времени, конечно, он не самый сложный, но сильно утомительный, поэтому его нужно улучшить путем автоматизации процесса.

Использование автоматизированной системы управления для организации сделает процесс получения и обработки информации более быстрым и легким. Таким образом, разработка текущего проекта оправдана для автоматизации огромного набора процессов, которые в конечном итоге значительно уменьшат затраты времени на каждый цикл выполняемой работы от небольшой прибавки в производительности - до нескольких раз.

Цель выпускной квалификационной работы — создание ПО, способного автоматизировать учёт складской продукции. И в соответствии с поставленной задачей при создании проекта ставились следующие цели:

- анализ предметной области;
- пояснение использования стороннего ПО при разработке;
- проектирование и создание логической схемы базы данных;
- создание базы данных;
- проектирование и разработка структуры web-приложения;

- получение доступа к базе данных посредством PHP;
- реализация запросов к базам SQL;
- создание user-frendly интерфейса;
- возможность редактировать, сортировать, искать нужные данные;
- реализация учёта товаров на складе;
- тестирование автоматизированной системы

В данном отчёте на рассмотрении находится вопрос об автоматизации процессов обработки заказов на предприятии, основными целями данной работы являются проектирование и разработка автоматизированной системы для постоянной работы без возникновения ошибок.

В первой главе отчёта рассматриваются популярные технологии приёма, регистрации, учёта и обработки заказов на предприятии, и обоснование использования современных технологий для решения этих проблем.

Вторая глава отчёта содержит в себе изложение технологии разработки автоматизированной системы для учета складской продукции. Такая система позволит сотрудникам «Агрошина 31» избежать рутинной и ежедневной работы на предприятии по подтверждению и обработке заказов. Так же автоматизация операций позволяет существенно снизить время оформления заказа, а так же составления маршрута доставки товара заказчику. Автоматизация вышеуказанных процессов позволит сохранить информацию в единой базе данных, работа с которой проходит с помощью удобного web-интерфейса. В заключение будут сделаны итоги о проделанной работе.

Поскольку учет складской продукции является большим подспорьем для существования товарооборота на предприятии, а контроль учета заказов — довольно трудоемкая работа, то создание автоматизированных систем, которые будут работать с данными, является весьма важным.

Практическая суть проекта состоит в том, что имеется возможность эксплуатировать разработанную систему, в различных предприятиях для автоматизации складской продукции.

В данном отчете используется 28 изображений, из них:

10 изображений - таблицы баз данных

2 изображения — инфологическая и даталогические модели

16 изображений — компиляция по выполнению работ и разработке

Всего страниц - 95

из них страниц отчета - 65

страниц приложения - 30

ГЛАВА 1 ПОСТАНОВКА ЗАДАЧИ НА РАЗРАБОТКУ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ

1.1. Виды деятельности предприятия «Агрошина 31»

Розничная и оптовая продажа товаров и услуг в настоящее время является одной из основных парадигм развития нашей экономики, и она оказывает весомое влияние на большинство аспектов жизни нашего общества. В текущий момент эта область находится не в самом лучшем состоянии, и это связано с рядом многих причин и факторов: неэффективная система администрирования, высокие затраты, связанные с транспортировкой и доставкой товара клиента, неразвитость системы доставки, периодические задержки поставок.

Розничные и оптовые продажи товаров и услуг - это те вещи, которыми люди в современных реалиях пользуются очень часто. В нашем же случае, это понятие задает набор сделок и комплексных решений связанных с вопросами по эксплуатации, приобретению и обслуживанию шин, камер, колес и тормозных дисков с тормозными колодками — иными словами неотъемлемой частью ходовой системы любого двух или четырех колесного авто-транспорта.

Однако на сегодняшний день, следует отметить, что складское предприятие как и точка оптовой и/или розничной торговли являются технически сложными структурами, которые требуют модернизации процессов и внедрения новых технологий, чтоб обслуживающие компании не только предоставляли новые услуги клиентам, но и значительно снижали время ожидания этих самых услуг. Но

для того, чтобы предоставлять качественные услуги, от управляющих компаний требуется использовать современные методы обслуживания.

«Агрошина 31» занимается организацией и проведением всеми видами ремонта, обслуживания и подготовки товаров с прицелом на клиента ; благоустройством автопарков и модельных рядов; сотрудничеством с официальными дилерами, учётом количества и качества оптовых поставок для авто производителей и многим другим. Работа «Агрошина 31» направлена на постоянное улучшение качества услуг, предоставляемых клиентам.

Компания «Агрошина 31» на текущий момент осуществляет следующие виды деятельности:

- оптовая и розничная продажа товаров и услуг;
- техническая проверка тормозных дисков;
- техническое обслуживание тормозных колодок;
- организация и планирование технического обслуживания комплектующих;
- подготовка для сезонной эксплуатации тормозных систем;
- поддержание управляемых систем, съема, подгонки, и адаптации частей;
- установка внешних устройств для защит дисков;
- техническое обслуживание и ремонт ходовых конструкций;
- составление комплектации под заказ;
- технической обвязкой оборудования;
- внешним дизайном тормозной системы;
- работа с клиентской базой;
- документооборот;

А также компания занимается вспомогательными видами деятельности:

- производство, передача и распределение товаров посредникам;
- эксплуатация мастерских;

- монтаж, ремонт, наладка и техническое обслуживание ходовых систем;
- удаление и обработка б/у комплектующих;

Основные цели, для достижения которых была разработана автоматизированная системы учёта «Агрошина 31»:

- предоставить работникам склада более быстрый и легкий поиск необходимой информации в базах данных;
- обеспечить безопасность для данных, которые будут поступать или уже хранятся;
- отслеживать изменений хранящихся данных;
- обеспечить высокий уровень защиты данных.

Среди большинства задач, которые требуют решения в ходе программной реализации автоматизированной системы, на особом месте находятся защищенность и доступность для информации, которая разрабатывается, используется или уже выходит и распространяется рабочей среде. Чаще всего, это информация о товарах и услугах, предположительные и точные даты поставки товара, своевременная информация о текущем состоянии товара и/или услуги. Качественно решить эти задачи можно только при использовании современных информационных технологий разработки и администрирования.

Автоматизированный учёт и выборка данных позволяет максимально точно, быстро и без ошибок собирать различные массивы данных а так же производить необходимые операции над этими данными. Проще говоря, это позволит сотрудникам выполнять свою работу качественно и своевременно, не отвлекаясь на такие вещи как человеческий фактор.

1.2. Особенности предоставления товаров и услуг.

Автоматизация учёта компании «Агрошина 31» даст весомые преимущества — прежде всего, это увеличение скорости работы поставщиков услуг, сокращение торговых пошлин на логистику и перевозку товара. Следующий фактор трудно выразить в деньгах — это удобство для клиентов в виде заказа интересующей продукции. Третий фактор — более эффективное использование средств в сфере продаж и обслуживания. И, наконец, четвертый фактор — автоматизация повысит качество и производительность труда, и позволит получать информативную, полную и своевременную информацию для учёта данных на предприятии. Если рассматривать прием заказов и учёт, то использование для этого компьютерной техники во много раз упростит такие задачи управления персоналом как например оперативная реакция на возникновение ошибки, или скажем компенсация человеческого фактора. Весь персонал компании, что занимается выполнением работы, будет постоянно в курсе внутренней ситуации и сможет заблаговременно и оперативно отреагировать на динамически меняющеюся обстановку, а диспетчер будет в любое время работы иметь перед собой исчерпывающую информацию о заявках, а сам прием заявок для складского хозяйства значительно разгрузит телефонную линию, экономя драгоценное рабочее время на прием заявок от клиентов и посредников.

Подводя итог, отличительными преимуществами компьютерных технологий при приеме заявок и учета являются:

- повышение лояльности клиентов, за счет компенсации человеческого фактора и повышения предоставления качества услуг;
- повышение эффективности и скорости работы сотрудников фирмы путем внутреннего распределения ресурсов;

- контроль над рабочими нагрузками сотрудников, работающих по заказам;
- уменьшение множества расходов на издержки из-за недобросовестного выполнения работы (отчетность позволяет контролировать и улучшать качество исполняемых работы).

Работники внутренней инфраструктуры предприятия «Агрошина 31» большую часть рабочего времени расходуют на выполнение многочисленных и трудоемких операций по учёту, формированию и обработке исходящей информации. Однако, хоть и выполнение основных процедур по обработке баз данных не требует каких-либо узкоспециализированных знаний, то по мере роста количества посылаемой информации время на обработку таких операций возрастает в геометрической прогрессии.

С введением современной автоматизированной системы, работа сотрудников, которые несут ответственность за ведение учета базы данных будет выполняться быстрее, а количество ошибок уменьшится, например, точность расчёта итоговой стоимости и т.д. Эта система приспособлена для исполнения на технике IBM/PC, а ее уровень соответствует высоким современным требованиям. Так же использование дружественного для неопытного пользователя интерфейса сильно упростит работу с ней. Обслуживание системы будет задачей системного администратора, ответственного за работу автоматизированной системы. Для эксплуатации и поддержки система не требует владения особенными техническими навыками, за исключением общих навыков владения работы с компьютером.

1.3. Подход к проектированию автоматизированной системы

Перед началом проведения основных работ, давайте рассмотрим технические аспекты создания и функционирования автоматизированных систем учёта для складской продукции: общие инструменты используемые при программировании, создании базы данных и средства связи для передачи больших массивов данных.

Общий перечень ПО, используемого для разработки сложных автоматизированных систем или программ, разного рода утилит — это все языки программирования, которые использовались в течение последних 10-15 лет, и все, что существуют сейчас.

Подход к созданию системного программного обеспечения обычно делится на две категории: для местного применения и программные продукты. Программное обеспечение из первой группы используется на предприятиях учета складской продукции, в которых работают программисты, написавшие и внедрившие это ПО, а затем сопровождающие его. Оно не предназначено для продажи, таким образом, данное ПО не обладают необходимыми атрибутами программных продуктов, например, документацией, поддержкой, обновлениями. С этим программным обеспечением хорошо работать имеет возможность только программист, написавший его.

Многие системы, которые в той или иной степени соответствуют требованиям предлагаемых продуктов, используют в основном набор инструментов, которые не только ускоряет процесс разработки, но и позволяют затем сопровождать систему во время её эксплуатации, практически не требуя вмешательства разработчиков. Из систем таких наиболее распространен набор инструментов 1С [Хрусталева Е.Ю. Разработка сложных отчетов в 1С. Предприятии 8. Системы компоновки данных.]. Но те системы, которые

разработаны на основе этой платформы, могут иметь ограниченную функциональность и несовместимость между отдельными модулями. Наиболее распространенные примеры — это средства, которые используют данные для экспорта или импорта.

Проанализировав строение баз данных, которые используются для хранения и обработки информации в системе учета складской продукции, можно выделить два момента: основы построения и месторасположение.

Самой старой и наиболее широко используемой основой для базы данных являются DBF-файлы. Причиной, по которой она распространена, является их широкое применение в 90-х годах прошлого века и в начале 21-го. В то время, они, конечно же, отвечали потребностям разработчиков, так как решались в основном проблемы локального характера, и таблицы баз данных состояли из десятки или сотни тысяч записей. Но для современных сложных автоматизированных систем такой подход не уместен по двум основным причинам: ограниченное число записей в одной таблице и небольшая надежность.

Первая проблема выражается в том, что при наступлении границы количества записей внесение данных по индексу может происходить в произвольном месте таблицы. Это особенно вероятно при совместном доступе к использованию файлов. Вторая причина выражается в несоответствии индекса файлов и основной таблицы данных. Это несоответствие может быть вызвано различными причинами. И главная проблема в том, что его можно заметить только при визуальной проверке информации на несоответствие считанных данных.

Современные базы данных строятся на основе технологий клиент-сервер. Существуют различные системы управления базами данных, которые имеют свои преимущества и недостатки. Каждый из них используется, по меньшей мере, в одной системе для автоматизации. Наиболее распространенной является MySQL.

Следующая по популярности идёт Interbase. Существует также реализации, работающих под управлением Oracle, которая являются достаточно дорогой и громоздкой системой для управления базами данных. Следует также отметить такую тенденцию, что поставщики систем управления базами данных, зачастую предлагают бесплатную версию с некоторыми ограничениями.

Самая простая топология базы данных — это расположение всей информации на одном сервере. Клиенты подключаются удалённо и используют сервер в своих нуждах. К очевидным недостаткам можно отнести необходимость постоянной связи, а иногда и медленную скорость передачи данных. Тем не менее, эти недостатки становится менее значимым, так как связь становится всё более быстрой и менее дорогой.

Другой способ — это распределенная база данных. Она не имеет недостатков предыдущего варианта, но имеет свою собственный — необходимость синхронизации данных. Синхронизация данных осуществляется различными способами. Наиболее распространенный раньше способ — экспорт и импорт. Данные вручную переносились между предприятиями учета складской продукции. Существенное влияние на производительность системы мог повлиять человеческий фактор — все нужно было делать вовремя и точно.

Второй вариант синхронизации — обмен данными между модулями системы, связываясь с удаленными серверами. Этот обмен запускается пользователем, или по заданному расписанию. Он заключается в том, что некоторые процедуры на одном из связанных серверов скачивают или отправляют данные, необходимые для синхронизации.

И, наконец, самый высокий уровень — синхронизация баз данных с помощью репликаций. Эти средства позволяют изменять данные как с одной стороны линии связи, так и с другой. В соответствии с указанным графиком или по команде начинается синхронизации баз данных.

Принципы автоматизированной системы должны быть общими: согласованность, гибкость, устойчивость и эффективность.

- принцип системности позволяет рассматривать систему, как структуру, которая определяется функциональным назначением.
- принцип гибкости означает приспособляемость системы к возможным перестройкам благодаря модульной конструкции всех подсистем и стандартизации их элементов.
- принцип стабильности заключается в том, что система должна выполнять основные функции независимо от воздействия возможных внутренних и внешних факторов. Это означает, что проблемы в некоторых из её частей должны быть легко устранены, а работоспособность системы быстро восстанавливаться.

Различные компании-поставщики предлагают очень разные цены для своих программных продуктов и услуг по внедрению и сопровождению. Цена зависит, конечно, от функциональных возможностей системы, но это всё не так однозначно. И есть большой шанс найти систему дешевле, которая больше подходит для использования, чем предлагают некоторые другие известные компании.

Стоимость систем автоматизации, а также для любой другой системы управления состоит из нескольких компонентов: прикладное программное обеспечение, системные инструменты, а также затраты на ввод в эксплуатацию и техническое обслуживание.

Прикладное программное обеспечение — это именно та программа, которая нужна для решения поставленной задачи, в данном случае — автоматизация учета складской продукции. Стоимость такого ПО зависит от ценовой политики поставщика, она может быть либо фиксированной или возрастать с увеличением

количества рабочих мест пользователей, или увеличиваться с числом клиентских аккаунтов, обрабатываемых в системе.

Практически нет ценовой в зависимости от качества системы: функциональная полноты, надежности, простоты в использовании. Но есть очень сильная зависимость от «бренда». Многие крупные фирмы позволяют себе устанавливать цены на свою продукцию значительно выше, чем другие.

Ещё существует графа расходов, которые часто не указаны в стоимости автоматизации. К ним можно отнести стоимость операционной системы, офис, связь и т.д. Особенно в эти расходы могут входить системы управления базами данных. Например, цена программы может иметь небольшое значение, в то время как эта программа использует базу данных, стоимость которой составляет куда больше. Поэтому всегда нужно учитывать этот аспект затрат при покупке программного обеспечения для автоматизированной системы.

Следующий компонент стоимости — это ввод системы в эксплуатацию. Внедрение автоматизированной системы, связанной с различными настройками, обучением пользователей, загрузкой базы данных, все это требует услуг специалистов, а работа специалистов стоит денег. Время на обучение зависит от простоты системы и квалификации пользователей.

Сложные технические системы нуждаются в поддержке. Наиболее очевидной проблемой является доработка при изменяющихся условиях эксплуатации. Ещё одна проблема — надежность. Такие функции, как архивирование данных, восстановление данных, вероятность отказа в целостности базы данных зависит от уровня разработки программного обеспечения, другими словами, от его качества.

Таким образом, при выборе системы, необходимо проанализировать все составляющие затрат, чтобы оценить время и деньги, необходимые для завершения всех этапов автоматизации. Разработка такой системы позволит

автоматизировать деятельность компании без больших денежных затрат на приобретение и установку программного обеспечения. Система не требует больших ресурсов компьютера, проста в эксплуатации и техническом обслуживании. Это позволит быстро обучить персонал работе в системе без каких-либо дополнительных затрат.

1.4. Требования к автоматизированной системе учёта складской продукции

Так как основной целью является автоматизация обработки информации по приему товаров и учёту услуг, а целью автоматизированной системы — обеспечение возможности работникам управляющих компаний иметь полную и точную информацию, а также повышение эффективности выполнения основных функций склада, составления документов для отчетности, это позволит повысить эффективность, сократится количество ошибок за счет автоматизации процесса обработки информации, а также будет присутствовать эффективный и безопасный доступ к базе данных. Автоматизированная информационная сеть позволяет хранить, обрабатывать и интегрировать информацию по учёту и выполнению услуг организации «Агрошина 31».

Техническое задание:

1. характеристика области применения

Система предназначена к использованию в предприятии учета складской продукции.

2. функциональное назначение

Функциональным назначением системы является автоматизация учёта складской продукции, а так же упрощение приема заказов товара.

3. требования к составу выполняемых функций

- авторизованный вход в личный кабинет;
- отображение информации о доступных товарах и услугах;
- ввод данных;
- расчёт стоимости;
- получение заявок от клиентов;
- изменение статуса выполнения заявки на товары и услуги;
- возможность поиска, фильтрации и сортировки данных в таблицах.

4. Требования к сущностям в базе данных

Обязательными сущностями должны быть «товар», «категории товара» и «корзина».

5. Требования к хранению данных

данные должные храниться в реляционной базе данных под управлением СУБД.

6. Требования к техническим средствам

для СУБД требуется сервер под управлением ОС Windows или Linux.

7. Требования к программным средствам, используемых системой

клиентская часть системы должна работать и быть совместимой с веб-браузерами IE, Chrome, Firefox.

8. Стадии разработки

- анализ структуры автоматизированной системы;
- проектирование базы данных;
- определение связей между сущностями;
- реализация базы данных;
- проектирование веб-приложения;
- реализация взаимодействия веб-приложения с базой данных;
- реализация интерфейса веб-приложения;
- тестирование.

ГЛАВА 2 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА

2.1. Обоснование использование выбранных СУБД и программного обеспечения для разработки системы

Программно-аппаратные средства, процессы и люди, которые хранят, собирают, обрабатывают и доставляют информацию для выполнения некоторых задач, представляет из себя информационную систему.

База данных представляет собой интегрированный набор структурированных и связанных данных, организованных по определенным правилам, которые предусматривают описание, хранение и обработку данных. Базы данных обычно создаются для некоторых предметных областей.

Предметные области отображают часть реального мира, который изучается для того, чтобы создать базы данных для автоматизации процесса управления.

Программы, структурирующие информацию, помещающие её в таблицах данных и управляющие данными называются системами управления базами данных (СУБД).

Набор принципов, с помощью которых организуется и определяется логическая структура хранения базы данных называются моделями данных.

Существуют три основных модели базы данных — иерархические, сетевые и реляционные, которые различаются по тому, как установлены связи между данными.

В реляционных базах данных вся информация находится в таблицах, которые в свою очередь состоят из строк и столбцов и называются записями и полями

соответственно. Эти таблицы получили название реляций, поэтому модель стала называться реляционной.

Записи в таких таблицах не повторяются. Сохранность и непротиворечивость данных в таких таблицах обеспечивает первичный ключ, содержащий набор полей, однозначно определяющих запись.

Для выборки и поиска данных из нескольких связанных таблиц используются значения одного или нескольких совпадающих полей, при этом типы связующего ключа и поля другой таблицы обязательно должны совпадать. Для лучшей ориентации при связи таблиц, рекомендуется называть данные поля одинаковыми именами.

Для поиска информации в базе данных создаются индексы по одному или нескольким полям таблицы. Для автоматической поддержки целостности связанных данных, находящихся в разных таблицах, используются первичные и внешние ключи.

Для эффективной работы пользователей в автоматизированной системе с большим потоком информации, база данных должна отвечать следующим требованиям:

- хранить большой объём достоверной и актуальной информации;
- быть простой в использовании;
- иметь возможность вводить, удалять, сортировать, изменять данные;
- позволять производить поиск информации по таблицам;
- иметь возможность расширения при внесении изменений в систему.

Одной из популярных систем управления базами данных является MySQL. К возможностям MySQL можно отнести:

1. Ввод данных, можно осуществить такими способами, как:

- ручной набор данных в таблицах;
- ручной набор данных в полях формы;

- импорт данных из других источников.

2. Изменение данных, можно осуществить такими способами, как:

- ручной набор данных в таблицах;
- ручной набор данных в полях формы;
- ручной набор в окне браузера, в котором загружена web-страница из БД;

3. Вывода данных, который можно осуществить такими способами, как:

- вывод на экран монитора в табличном виде, полях форм или отчетов;
- экспорт в другие форматы данных;
- вывод на печать, в виде отчетов;

4. Взаимодействие с другими источниками и потребителями информации.

Firebird 2

Это одно из ПО для работы с базами данных. К основным плюсам Firebird 2 можно отнести высокую скорость работы, быстроту обработки данных и оптимальную надежность. Немаловажно и то, что данная СУБД распространяется бесплатно и представляет собой программное обеспечение с открытым кодом. За счет этого Вы можете вносить свои изменения и модифицировать код, что весьма полезно для веб-мастеров.

PHP

Для разработки автоматизированной информационной системы наиболее удобным и простым языком программирования является PHP, выбранный язык программирования обеспечивает относительно высокую скорость работы и позволяет создавать проекты хорошего качества.

HTML

Для подготовки гипертекстовых документов используется язык HTML (Hyper Text Markup Language — язык разметки гипертекстовых документов), который предоставляет широкие возможности по форматированию и структурной

разметке документов, организации связей между различными документами, средства включения графической и мультимедийной информации. Зачастую возникает необходимость применить в процессе создания html-документа сложное форматирование — от абзаца к абзацу менять шрифт, расположение текста, его цвет, формировать различные таблицы данных. Для этого нужно подключить к странице внешний файл, выполненный в стандарте CSS.

CSS

Для формирования дизайна пользовательского интерфейса был выбран CSS. — Cascading Style Sheets (каскадные таблицы стилей), в котором с помощью специального макроязыка можно задать форматирование страницы. Таким образом, файл CSS выполняет роль некоего шаблона, применяемого для форматирования текста, таблиц и иных элементов в документе HTML. Есть возможность подключать один и тот же физический файл CSS к различным web-страницам сайта.

JavaScript

JavaScript является наиболее популярным языком сценариев в интернете, и работает в большинстве браузеров, таких как Intenet Explorer, Firefox, Chrome, Opera, и Safari. Через него можно к любому элементу HTML-кода получить доступ и делать с этим элементом множество манипуляций. Можно загружать данные не перезагружая страницу, выводить сообщения, считывать или устанавливать cookie и выполнять множество других действий.

PhpMyAdmin

Это веб-приложение, которое распространяется с открытым кодом, написанное на языке web-программирования PHP и представляющее собой веб-интерфейс для администрирования СУБД MySQL. Для работы с базой данных нужен браузер, который и будет передавать на сервер все команды. В качестве языка работы с БД используется широко известный SQL.

2.2. Проектирование и разработка базы данных для автоматизированной системы

Входными данными в системе является информация о предоставляемых товарах и услугах, стоимость тарифов на проводимые работы в зависимости от типа, сложности а так же предоставленного оборудования. Результатом работы системы является расчет наличия заказанных товаров и услуг а так же итоговый расчет стоимости.

Цель этого моделирования состоит в том, чтобы обеспечить разработчика концептуальной схемой базы данных информационной системы в виде одной или нескольких локальных моделей, которые довольно легко могут быть отображены в любой системе базы данных.

Очень распространенным средством моделирования данных являются диаграммы «сущность-связь» (ERD). С их помощью определяются важные для предметной области сущности, их атрибуты и связи между ними. ERD непосредственно используются при проектировке реляционных баз данных.

Логическая модель данных является первичным прототипом будущей базы данных. Логическая модель строится в терминах информационных единиц, но без привязки к конкретной СУБД.

В основе процесса проектирования лежит метод нормализации, т. е. декомпозиции отношения, находящегося в предыдущей нормальной форме, на два или более отношений, которые удовлетворяют требованиям следующей нормальной формы (с этим мы столкнемся, когда нам самим придется по мере прохождения материала проводить нормализацию того или иного базового отношения).

Сущности находятся в первой нормальной форме тогда и только тогда, когда схема этого отношения содержит только простые и только однозначные атрибуты, причем обязательно с одной и той же семантикой.

Сущности находится во второй нормальной форме относительного заданного множества функциональных зависимостей тогда и только тогда, когда оно находится в первой нормальной форме и, кроме того, каждый не ключевой атрибут полностью функционально зависит от каждого ключа.

Сущности находятся в третьей нормальной форме относительно заданного множества функциональных зависимостей тогда и только тогда, когда оно находится во второй нормальной форме и каждый не ключевой атрибут полностью функционально зависит только от ключей.

Для базы данных автоматизированной системы учёта складской продукции были разработаны следующие сущности:

Сущность товаров — представляет собой основную информацию о товаре. Имеет связь с сущностями «загруженные изображения», «категории товаров», «корзина», «отзывы», «купленные товары» и «регистрация администраторов» . Состоит из полей:

- ID товара
- название
- цена
- бренд
- ключевые теги для поиска товара
- ключевые теги для поиска товара из его описания
- мини-описание для вывода в неполном виде
- изображение
- полное описание

- мини-характеристики для вывода в неполном виде
- полные характеристики
- дата поступления товара
- новинка
- лидер продаж
- скидка
- видимость (позволяет выводить или скрывать товар)
- количество на складе
- тип товара
- ID бренда товара
- голосование (участвует ли товар в голосовании)
- голоса

  итоговый вид данной сущности в виде таблицы базы данных показан в рисунке (Рис. 2.1):

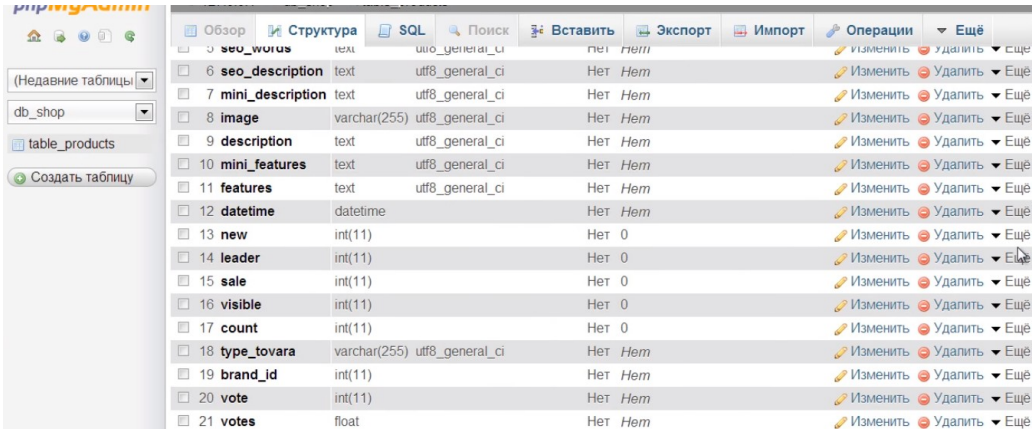

Рис. 2.1 . - итоговый вид сущности товаров в виде таблицы базы данных

сущность категории товара — представляет собой краткую информацию о товаре для выбора по критериям из дополнительного меню. Имеет связи с сущностями «товары» и «регистрация администраторов». Состоит из полей:

- ID категории
- тип
- бренд

итоговый вид данной сущности в виде таблицы базы данных показан в рисунке (Рис. 2.2):

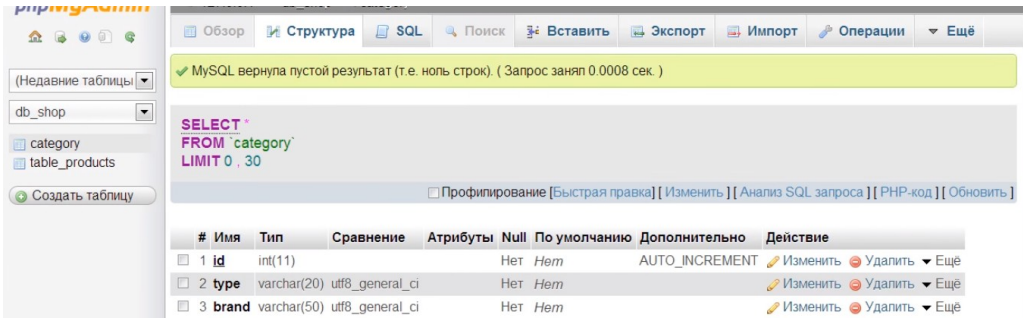

Рис. 2.2 . - итоговый вид сущности категории товара в виде таблицы базы данных

сущность корзины — представляет собой информацию о всех отмеченных товарах пользователем для покупки. Имеет связи с сущностью «Товар». Состоит из полей:

- ID корзины
- ID товаров корзины (для отображения товара из таблицы товаров)
- цена товара (для сложения стоимости товара от его количества)
- количество товара
- дата добавления товара в корзину
- ip пользователя (чтобы отследить уникальность товара для пользователя)

итоговый вид данной сущности в виде таблицы базы данных показан в рисунке (Рис. 2.3):

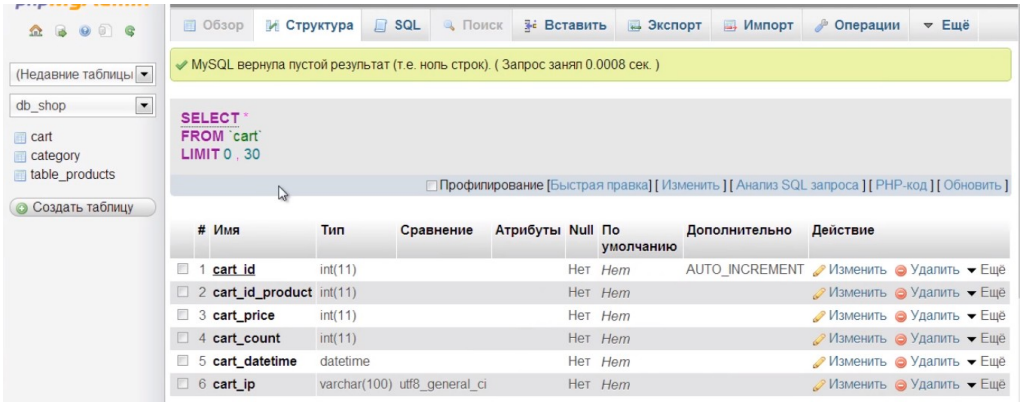

Рис. 2.3. - итоговый вид сущности корзины в виде таблицы базы данных

сущность новостей — представляет собой сущность для вывода новостей компании в web-приложении. Имеет связи с сущностью «регистрация администраторов». Состоит из полей:

- ID новостей
- название новости
- текст новости
- дата новости

итоговый вид данной сущности в виде таблицы базы данных показан в рисунке (Рис. 2.4):

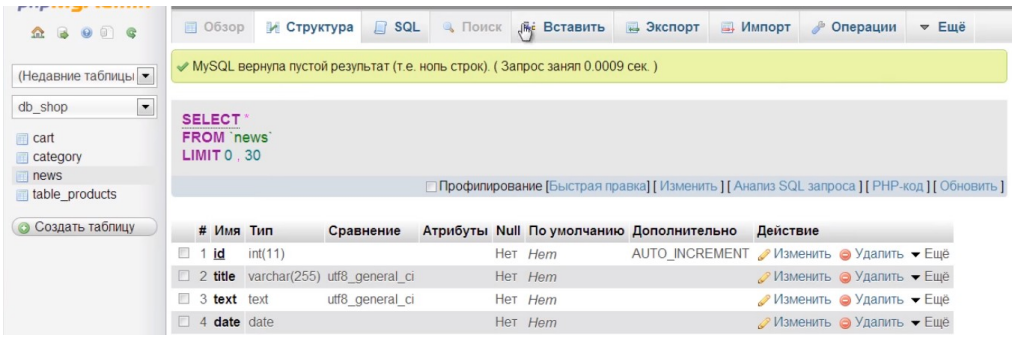

Рис. 2.4. - итоговый вид сущности новостей в виде таблицы базы данных

сущность регистрации администратора — представляет собой сущность хранящую информацию о зарегистрированных администраторах и их правах. Имеет связи с сущностями «товары», «категории товаров», «отзывы», «заказы», «регистрация» и «новости». Состоит из полей:

- ID регистрации администратора
- логин
- пароль
- фамилия, имя, отчество
- должность
- электронная почта
- телефон
- доступ к заказам
- доступ к подтверждению заказов
- доступ к удалению заказов
- доступ к добавлению товаров
- доступ к редактированию товаров
- доступ к удалению товаров
- доступ к модерации отзывов
- доступ к удалению отзывов
- доступ к списку клиентов
- доступ к удалению клиентов
- доступ к добавлению новостей
- доступ к удалению новостей
- доступ к добавлению категорий товаров
- доступ к удалению категорий товаров

- доступ к списку администраторов

итоговый вид данной сущности в виде таблицы базы данных показан в рисунке (Рис. 2.5):

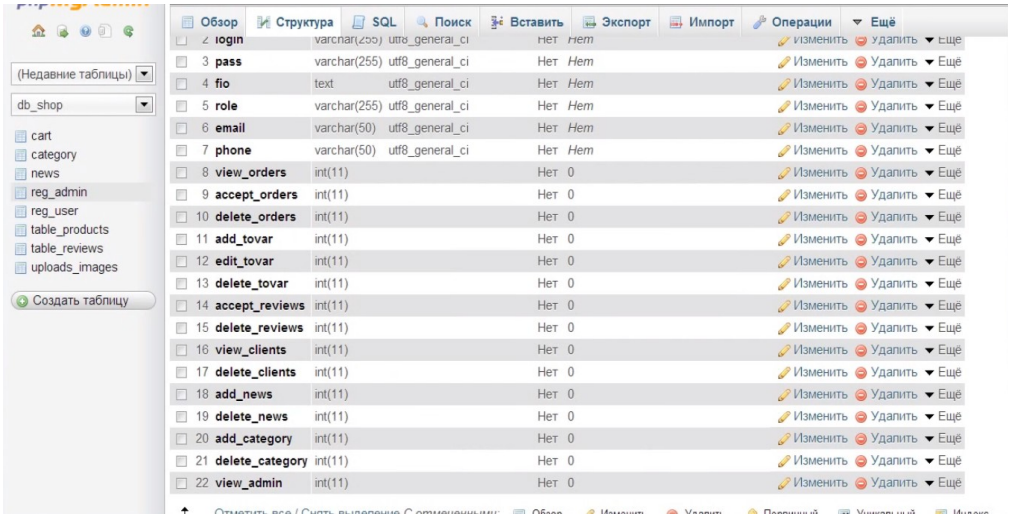

Рис. 2.5. - итоговый вид сущности регистрации администратора в виде таблицы базы данных

сущность регистрации — представляет собой сущность хранящую информацию о зарегистрированных клиентах. Имеет связи с сущностью «регистрация администраторов». Состоит из полей:

- ID регистрации пользователя
- логин
- пароль
- фамилия
- имя
- отчество
- электронная почта
- телефон

- адрес
- дата регистрации
- уникальный идентификатор ip пользователя

итоговый вид данной сущности в виде таблицы базы данных показан в рисунке (Рис. 2.6):

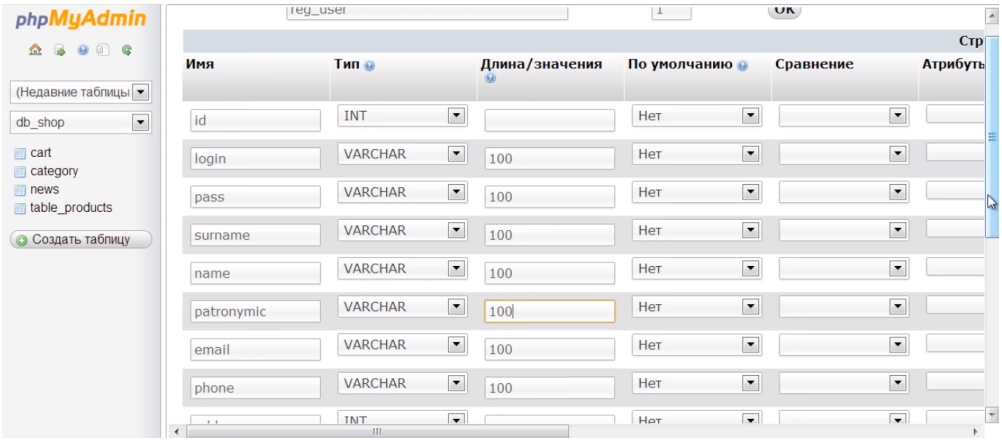

Рис. 2.6. - итоговый вид сущности регистрации в виде таблицы базы данных

сущность загруженных изображений — представляет собой информацию для прикрепления изображений. Имеет связи с сущностью «товары». Состоит из полей:

- ID загруженного изображения
- ID продукта (указывает на продукт к которому прикреплено данное изображение)
- изображение (указывает на сам файл изображения)

итоговый вид данной сущности в виде таблицы базы данных показан в рисунке (Рис. 2.7):

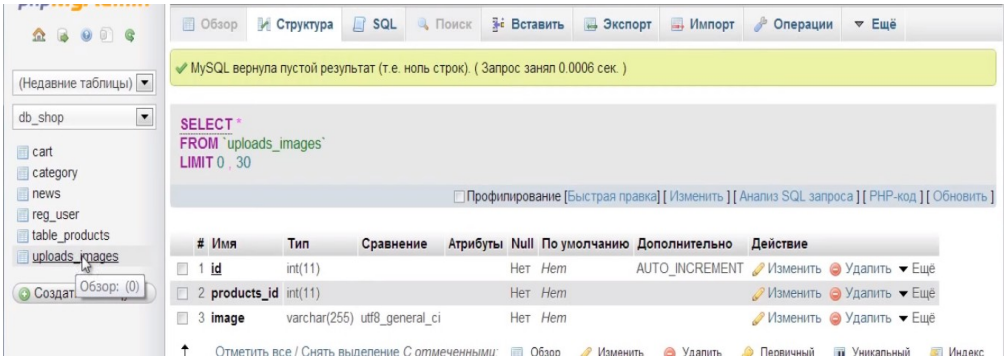

Рис. 2.7. - итоговый вид сущности изображения в виде таблицы базы данных

сущность отзывов — представляет собой отзывы пользователей о том или ином товаре. Имеет связи с сущностями «товары» и «регистрация администраторов». Состоит из полей:

- ID отзыва
- ID товара
- имя отзыва
- хороший отзыв (как и в случае ниже используется для идентификации отзыва — негативный или позитивный)
- плохой отзыв
- комментарий
- дата
- модерация (для выявления одобрения отзыва, чтобы выбрать показывать ли отзыв о товаре или нет)

итоговый вид данной сущности в виде таблицы базы данных показан в рисунке (Рис. 2.8):

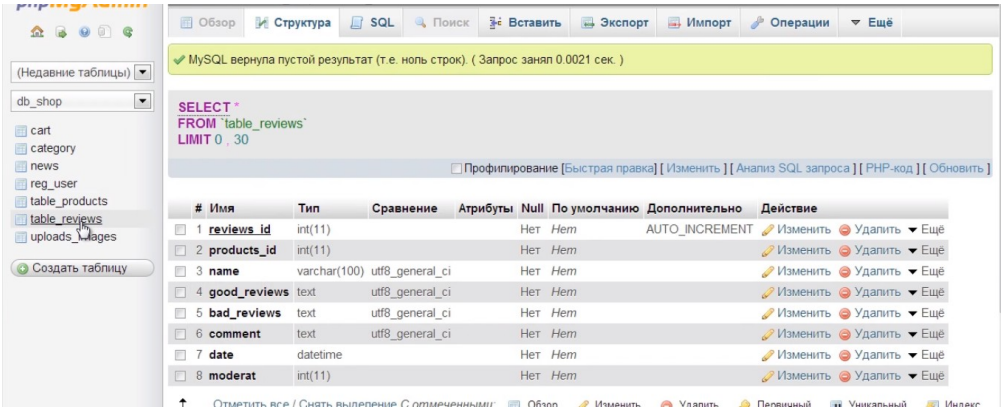

Рис. 2.8. - итоговый вид сущности отзывов в виде таблицы базы данных

сущность заказы — представляет собой комплексную информацию о товаре. Имеет связи с сущностями «товары», «купленные товары» и «регистрация администраторов». Состоит из полей:

- ID заказа
- дата заказа
- подтверждение заказа
- доставка заказа
- оплата заказа
- тип оплаты заказа
- имя, фамилия, отчество
- адрес доставки заказа
- телефон
- примечание к заказу
- электронная почта

итоговый вид данной сущности в виде таблицы базы данных показан в рисунке (Рис. 2.9):

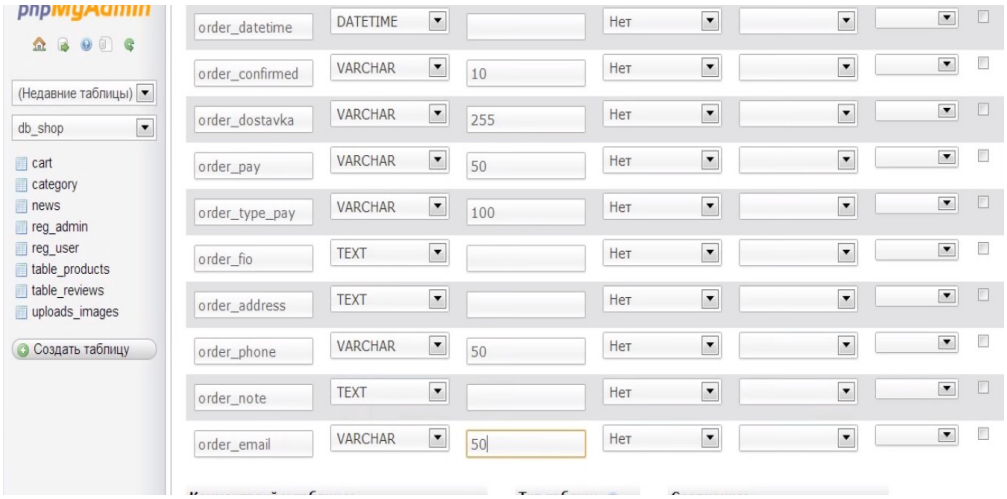

Рис. 2.9. - итоговый вид сущности заказов в виде таблицы базы данных

сущность купленных товаров — содержит в себе информацию о приобретенных товарах. Имеет связи с сущностями «товары» и «заказы». Состоит из полей:

- ID купленного товара
- ID заказа
- ID продукта
- количество купленных товаров

итоговый вид данной сущности в виде таблицы базы данных показан в рисунке (Рис. 2.10):

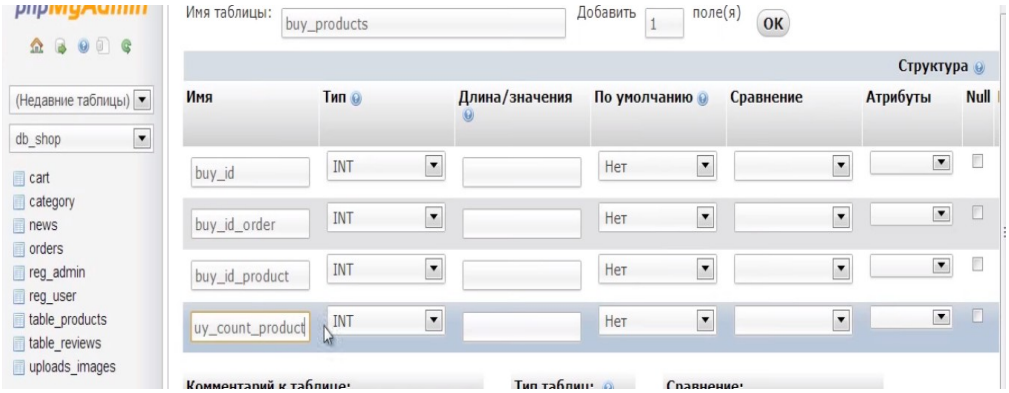

Рис. 2.10. - итоговый вид сущности купленных товаров в виде таблицы базы данных

Отношения между сущностями:

отношение между сущностями «товары» и «загруженные изображения» — один ко многим, так как у одного товара может быть несколько изображений.

Отношение между сущностями «товары» и «категории товаров» — один ко многим, так как один товар может относится к разным категориям.

Отношение между сущностями «товары» и «корзина» — один ко многим, так как в одной корзине может быть несколько товаров.

Отношение между сущностями «товары» и «отзывы» — один ко многим, так как у одного товара может быть несколько отзывов.

Отношение между сущностями «товары» и «купленные товары» — один ко многим, так как купленных товаров может быть несколько.

Отношение между сущностями «товары» и «регистрация администраторов» — один ко многим, так как администратор может добавлять, изменять, удалять множество товаров.

Отношение между сущностями «регистрация администраторов» и «категории товаров» — один ко многим, так как администратор может добавлять, изменять, удалять множество категорий товаров.

Отношение между сущностями «регистрация администраторов» и «отзывы» — один ко многим, так как администратор может модерировать, публиковать и удалять множество отзывов.

Отношение между сущностями «регистрация администраторов» и «заказы» — один ко многим, так как администратор может добавлять, изменять, удалять множество заказов.

Отношение между сущностями «регистрация администраторов» и «регистрация» — один ко многим, так как администратор может добавлять, изменять, удалять множество аккаунтов пользователей.

Отношение между сущностями «регистрация администраторов» и «новости» — один ко многим, так как администратор может добавлять, изменять, удалять множество новостей

Отношение между сущностями «заказы» и «купленные товары» — один ко многим, так как в одном заказе может быть несколько купленных товаров.

На основе составленных сущностей спроектирована следующая схема (Рис. 2.11):

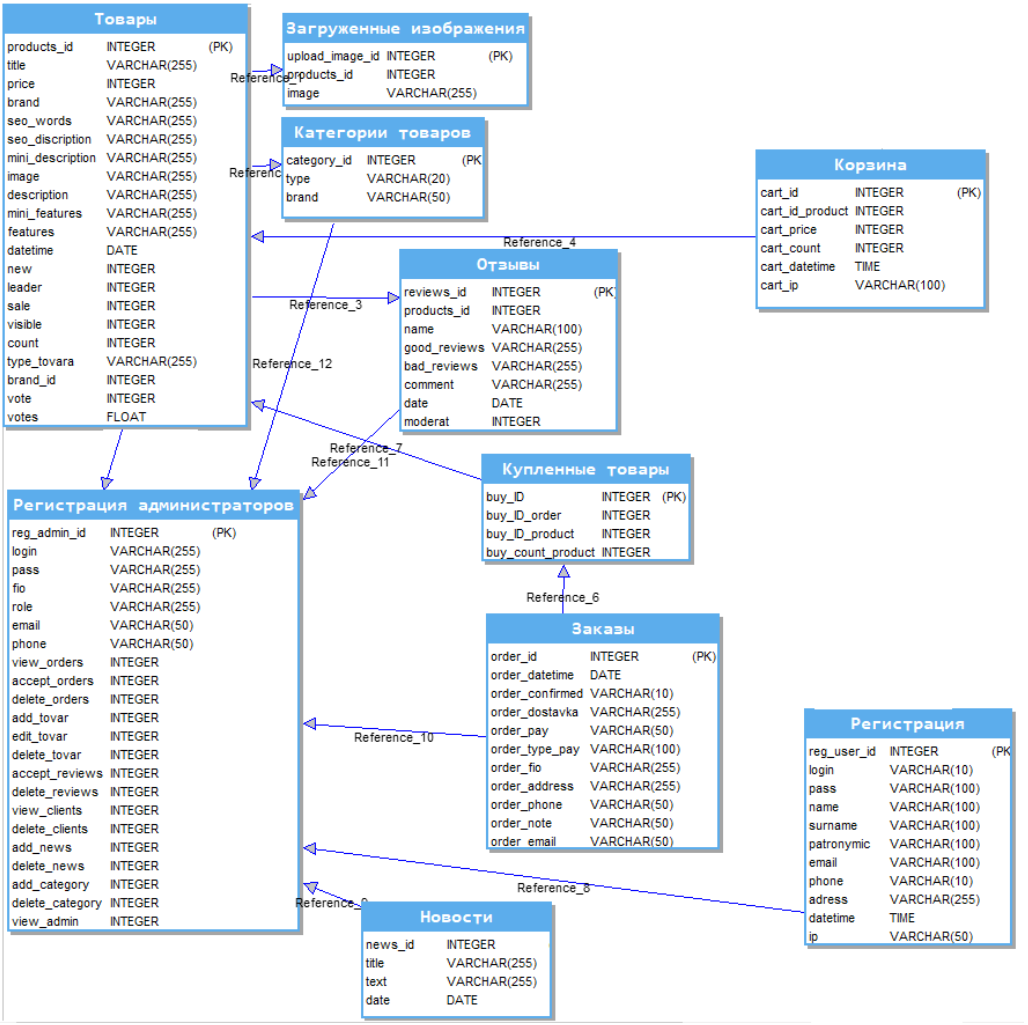

Рис. 2.11. — Инфологическая схема базы данных

2.3. Разработка web-приложения для автоматизированной системы

Главная страница web-приложения представляет собой интернет-ресурс, на котором находится таблица с товарами. Авторизоваться могут как и простые пользователи, так и администратор для управления системой.

Сама страница разделена на несколько отдельных блоков, это сделано потому, что используется SHELL-ON-WALL или же классическая структура. Данная структура подразумевает собой то, что все web-приложение будет разбито на блоки, кроме основных, которые будут общие для всех (или большинства) страниц.

Общими блоками для большинства страниц являются:

- тело страницы (присвоен id block-body)
- низ страницы (присвоен id block-footer)
- голова страницы (присвоен id block-header)
- верхняя часть головы страницы (присвоен id header-top-block)
- правый блок (присвоен id block-right)
- центральный блок с контентом из баз данных (присвоен id block-content)

К сайту была подключена страница стилей [1. Эрик Мейер — "CSS-каскадные таблицы стилей. Подробное руководство (Cascading Style Sheets: The Definitive Guide)"] Style.css см.(Приложение 4)

index.php является начальной страницей.

В начале идет встроенная информация об HTML от W3C:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
```

Далее в документе задается главная часть сайта. Задается параметром <head>

```
<head>
```

Указываем содержимое content, content="text/html означает что весь встроенный контент будет набран текстом на языке PHP.

```
<meta http-equiv="content-type" content="text/html;
```

Чарсет — переменная для выбора кодировки символов. Для локального использования задается windows-1251, для интернет ресурсов — UTF-8

```
charset=windows-1251" />
```

Подключение пустой таблицы стилей reset.css позволяет сбросить настройки таблицы браузера, чтоб он не повредил разметку сайта

```
<link href="css/reset.css" rel="stylesheet" type="text/css" />
```

Здесь мы подключаем основную таблицу стилей Style.css, в ней делается все:от верстки до подключения картинок и точной позиции пикселов//

```
<link href="css/style.css" rel="stylesheet" type="text/css" />
```

Параметр <title> задает текст во вкладке браузера.

```
<title>Автоматизированнная система</title>
```

Переходим к основному устройству index.php

Переменная </head> означает что мы работаем с основной частью сайта

```
</head>
```

Переменная <body> означает что мы переходим к основному телу сайта

```
<body>
<div id="block-body">
```

Командой div id что указана выше мы задали параметр id для блока тела сайта

Через PHP мы подключаем шапку к сайту как отдельный файл.

```
<?php
    include("include/block-header.php");
```

?>

Задаем id для тела сайта — тут будет контент т.е. информация из базы данных

Задаем id для правого блока сайта

При помощи PHP задаем id и подключаем нижний блок

[]Гутманс Э., Баккен С, Ретанс Д. PHP 5. Профессиональное программирование. / Пер. с англ. СПб: Символ- Плюс, 2006. 704 с., ил.

```
<?php
    include("include/block-footer.php");
?>
```

Подключаем нижнюю часть сайта

block-header.php

Для головы сайта задаем id

Тут же задаем id для верхней части головы

На рисунке ниже показано как будет выглядеть разметка с включенными border в style.css (Рис. 2.12):

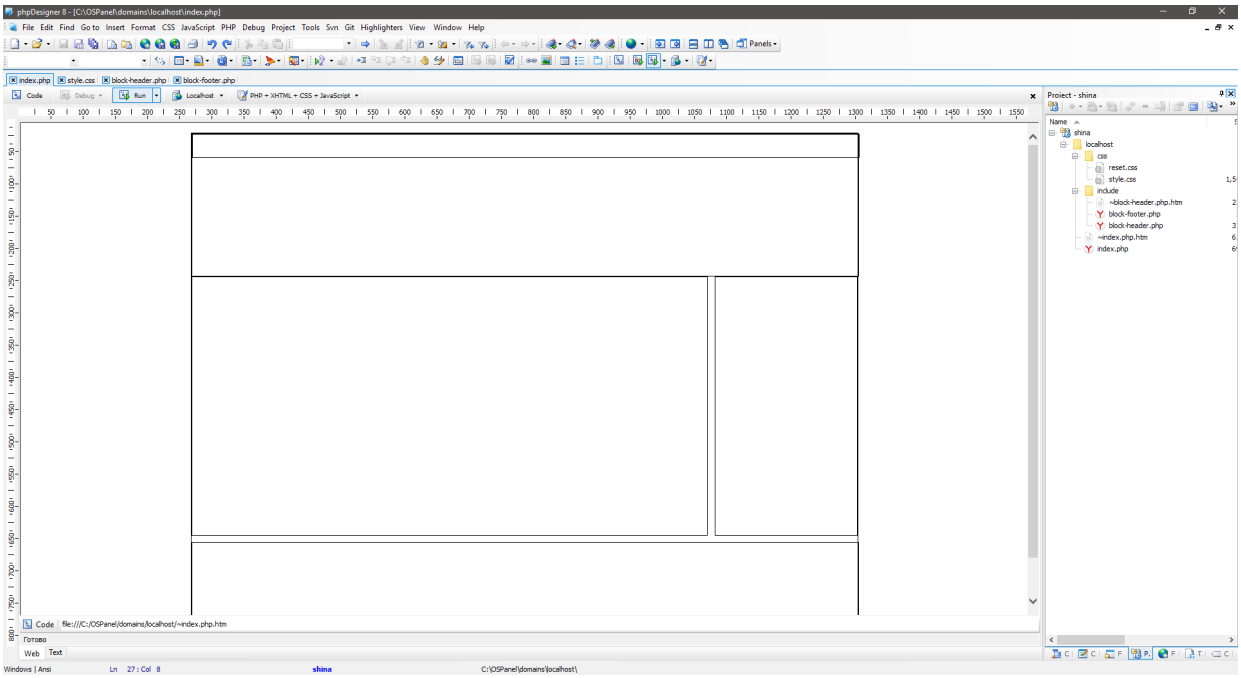

Рис. 2.12. — разметка блоков сайта

Задаем <ul> id для верхнего меню, переменная <li> позволяет нам задать текст для ссылок

```
<ul id="header-top-menu">
<li>Мы находимся: <span>Студенческая 28</span></li>
<li><a href="o-nas.php">О нас</a></li>
<li><a href="contacts.php">Контакты</a></li>
</ul>
```

Задаем ссылки для входа и регистрации, переадресация ведет в файлу с регистрацией - registration.php

```
<p id="reg-auth-title" align="right"><a class="top-auth">Вход</a><a
href="registration.php">Регистрация</a></p>
</div>
```

Задаем id отделительной линии между головой и ее верхней частью

```
<div id="top-line"></div>
```

Как будет выглядеть топ хедер показано на (Рис. 2.13):

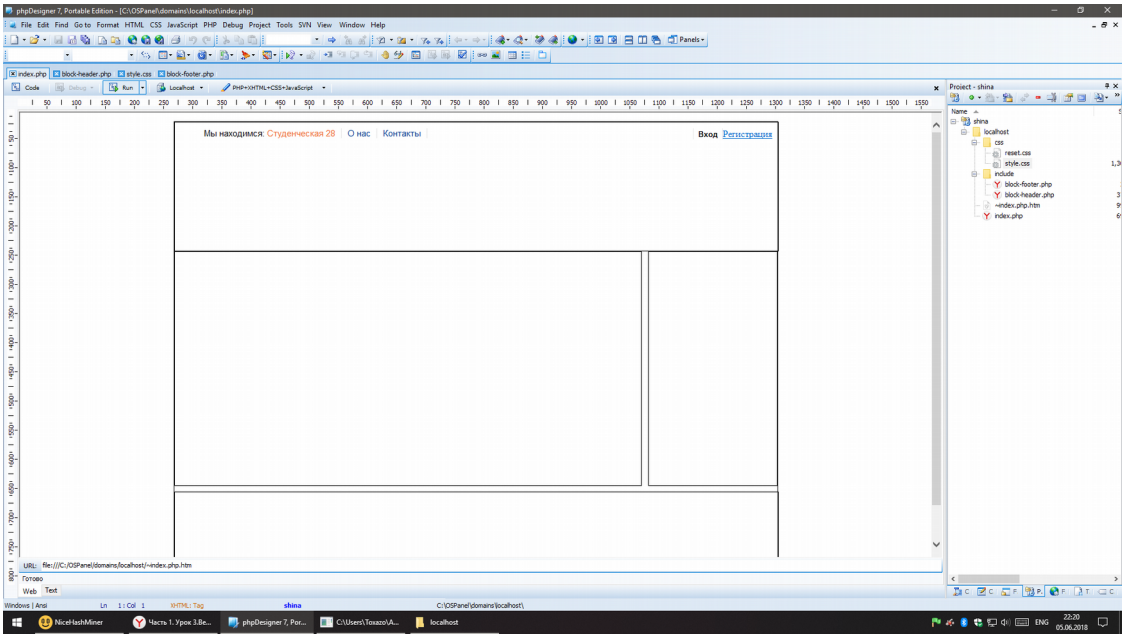

Рис. 2.13. - заполненый топ хедер (border отключен)

Задаем id для логотипа, чтоб выставить его положение в таблице style.css

```
<img id="img-logo" src="/images/logo.png" />
```

Задаем id для информационного блока

```
<div id="personal-info">
```

При помощи </p> вводим текст что будет в шапке сайта

```
<p align="right">Звонок бесплатный</p>
<h3 align="right">8 (800) 358 - 74 - 89</h3>
<p align="right">Доставка осуществляется</p>
<p align="right">Будние дни: с 9:00 до 18:00</p>
<p align="right">Суббота, Воскресенье - выходные</p>
</div>
```

Задаем id для блока с поиском

При нажатии на кнопку будет срабатывать срипт со странички search.php

<form method="GET" action="search.php?q=" >

При помощи placeholder задаем что будет написано внутри блока для поиска

<input type="text" id="input-search" name="q" placeholder="Поиск по тысячам наименований" />

<input type="submit" id="button-search" value="Искать!"/>

</form>

Задаем id для верхнего меню — тут и будет поиск

Корзина пуста

Задаем id для линии что будет отделять строчку поиска от блок - контента

полученный результат видно на (Рис. 2.14):

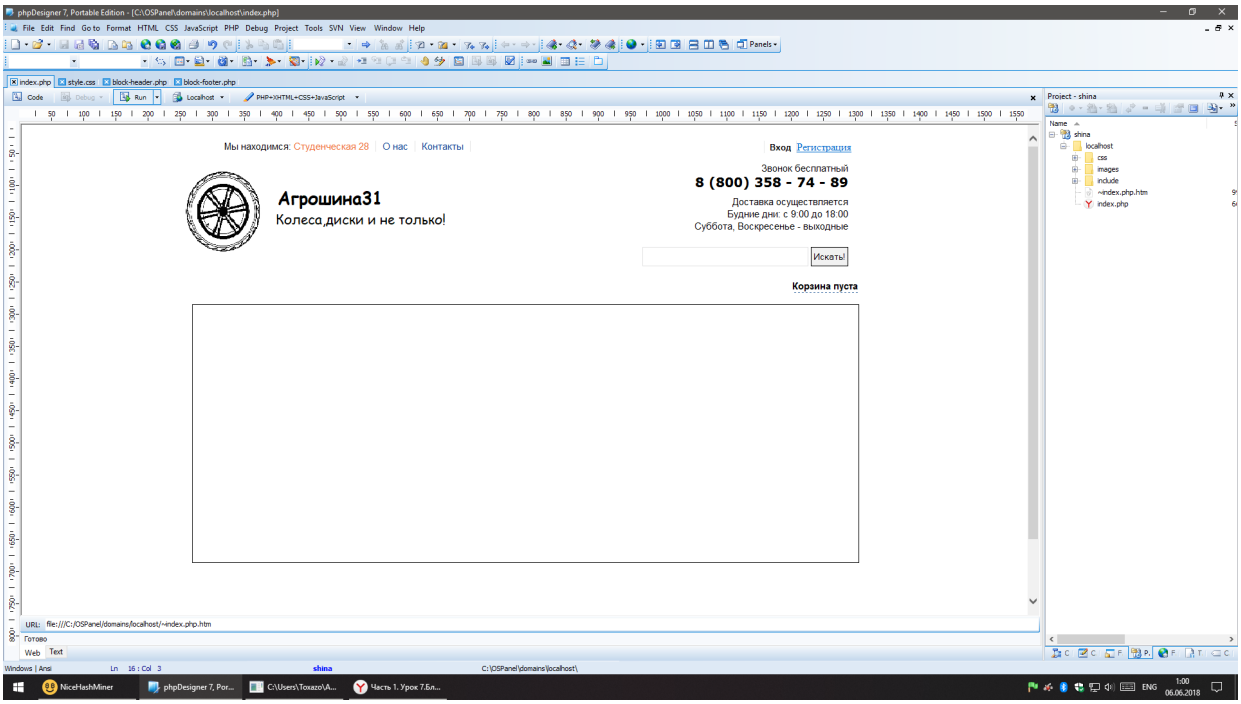

Рис. 2.14. - эмблема и контактные данные

[Робин Никсон «Создаем динамические веб-сайты с помощью PHP, MySQL и JavaScript». Питер, 2013. – 496 с.]

Включаем в часть блок контента сортировку

```
<div id="block-content">
<div id="block-sorting">
```

Задаем id для выползающего списка сортировки

```
<ul id="options-list">
<li>Сортировать:</li>
<li><a id="select-sort"><?php echo $sort_name; ?></a>
<ul id="sorting-list">
```

Задаем виды сортировки с ссылками к id

```
<li><a href="index.php?sort=price-asc">От дешовых к дорогим</a></li>
<li><a href="index.php?sort=price-desc">От дорогих к дешовым</a></li>
<li><a href="index.php?sort=brand">От А до Я</a></li>
```

```
</ul>
</li>
</ul>
</div>
```

Задаем id для блока товаров

```
<ul id="block-tovar-grid">
```

block-footer.php

Задаем id, используем 3 и 4 H-размеры для форматирования текста

```
<div id="block-footer">
<div id="bottom-line"></div>
<div id="footer-phone">
<h4>Есть вопросы? Звоните!</h4>
<h3>8 (800) 358 - 74 - 89</h3>
```

Командой <p> задаем текст для блока ног

```
<p>
Прием заказов:<br />
Будние дни: с 9:00 до 16:00<br />
Суббота, Воскресенье: с 10:00 до 14:00
</p>
</div>
</div>
```

на рисунке (Рис. 2.15) будет показано меню сортировки и часть футера

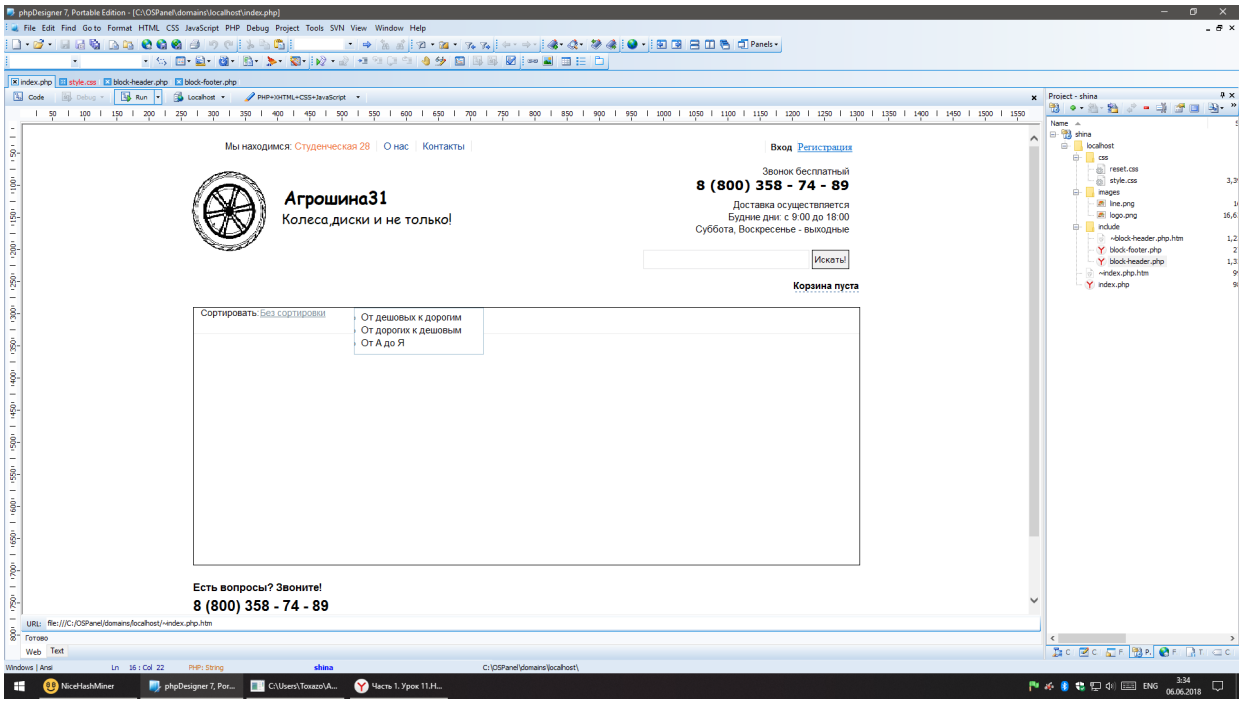

Рис. 2.15. - новое меню сортировки товара, футер внизу

Используем msql и подключаем базу данных, убираем бордер и делаем java скрипт для того чтоб спрятать кнопку сортировки

connect_db.php

Тут же создаем файл php для основного соединения с базой данных. Информация из таблиц будет выводится в блоке контента

```
<?php
```

Далее указываем данные для соединения с базой данных.

Имя хоста

```
$db_host     ='localhost';
```

Логин пользоватея

```
$db_user     ='admin';
```

Парол пользователя

```
$db_pass     ='123456';
```

Имя подключаемой базы данных

```
$db_database  ='db_shop';
```

Переменной $link задается связь с базой, если имя хоста, логин и пароль введены верно то будет установленно соединение

```
$link = mysql_connect($db_host,$db_user,$db_pass);
```

А если один или несколько параметров заданы не верно то OR DIE вернет нам код с ошибкой

```
mysql_select_db($db_database,$link) or die("Нет соединения с БД".mysql_error());
```

Задаем кодировку windows 1251

```
mysql_query("SET names cp1251");
?>
```

index.php

Подключаем наш файл db-connect.php с конфигурацией базы данных

[2. Томсон Лаура и Веллинг Люк. Разработка Web-приложений на PHP и MySQL: Пер. с англ./Лаура Томсон, Люк Веллинг. — 2-е изд., испр. — СПб: ООО «ДиаСофтЮП», 2003. — 672 с.]

```
<?php
  include("include/db-connect.php");
```

Далее переменной $sorting задаем сортировку

```
='sort';
if(isset($_GET['sort'])) {
    $sorting = $_GET['sort'];
      }
```

Если уже отсортировано, то будет выводится текуший аттрибут сортировки

```
    switch($sorting)
      {
```

Сортировка по цене

```
case'price-asc':
$sorting = 'price ASC';
$sort_name = 'От дешевых к дорогим';
break;
```

Команда break означает конец данного типа сортировки

```
case'price-desc':
$sorting = 'price DESC';
$sort_name = 'От дорогих к дешевым';
break;
```

Сортировка по имени от а до я, обращается к таблице базы данных товара и проверяет его производителя. Сортировка происходит в алфавитном порядке потому, что в таблице базы данных все производители по умолчанию идут в алфавитном порядке

```
case'brand':
$sorting = 'brand';
$sort_name = 'От А до Я';
break;
```

Задаем отсутствие какой — либо сортировки

```
default:
$sorting = 'products_id DESC';
$sort_name = 'Нет сортировки';
break;
?>
```

Задаем <ul> id для позиционирования выбора сортировки в style.css

```
<div id="block-content">
<div id="block-sorting">
```

```
<ul id="options-list">
```

Задаем имя

```
<li>Сортировать:</li>
<li><a id="select-sort"><?php echo $sort_name; ?></a>
<ul id="sorting-list">
```

Задаем имя для всех видов сортирвок

```
<li><a href="index.php?sort=price-asc">От дешовых к дорогим</a></li>
<li><a href="index.php?sort=price-desc">От дорогих к дешовым</a></li>
<li><a href="index.php?sort=brand">От А до Я</a></li>
</ul>
</li>
</ul>
</div>
```

на рисунке (Рис. 2.16) показано как мы добавили в главный блок товары из базы данных, так же убираем центральный бордер

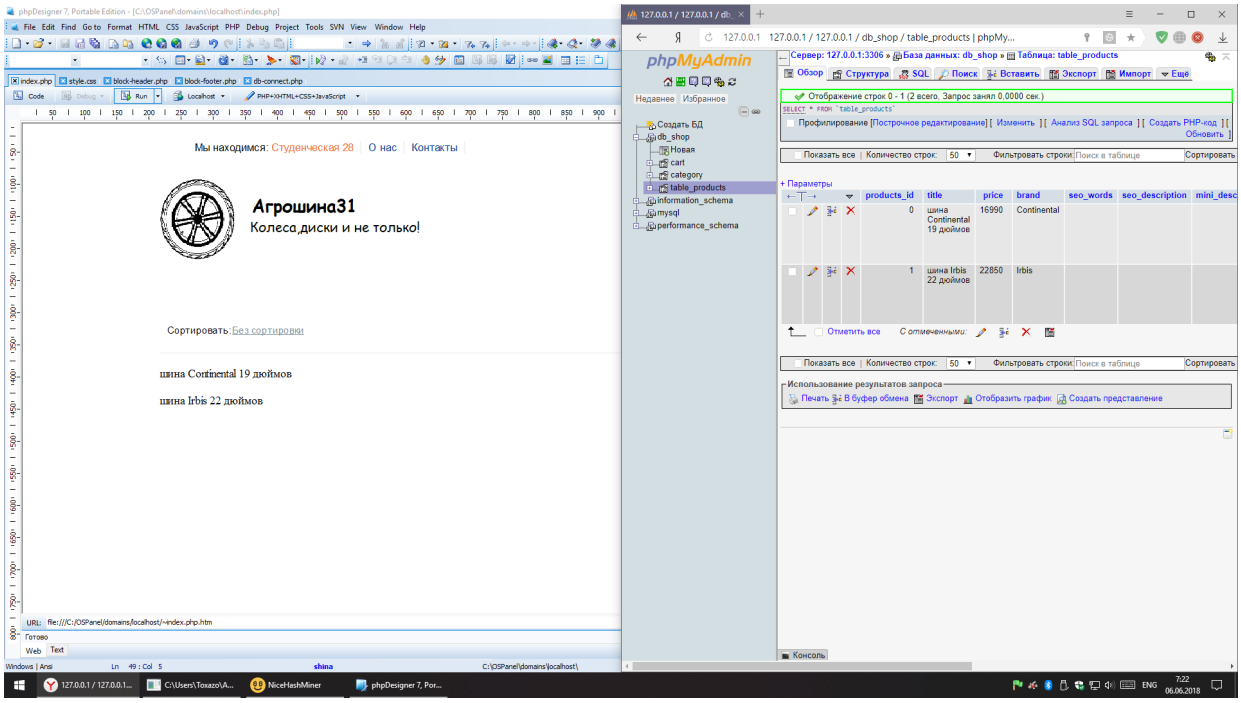

Рис. 2.16 — успешно подключенная база данных.

Index.php

Подключаем модуль для переключения страниц написанного на PHP

```
<?php
  $num = 2;
```

Переменная $count берет из базы данных товар и если в поле таблицы visible указано значение 1 то товар будет показан в блок контенте, если значение visible равно 0 то товар отображаться не будет

```
  $count = mysql_query("SELECT COUNT(*) FROM table_products WHERE visible = '1'",$link);
  $temp = mysql_fetch_array($count)
  If  ($temp[0] > 0)
  {
  $tempcount = $temp[0];
```

Задается формула расчета нумерации страничек в адресной строке, чтоб ни кто не перешел на Index например, у всех страничка будет называться по уникальному ip, что из базы данных reg_user + формула//

[Съерра К. И Бейтс Б. Изучаем java script и java – сходства и преемственность 2012]

```
$total = (($tempcount - 1) / $num) + 1;
$total = intval($total);
if(empty($page) or $page < 0) $page = 1;
 if($page > $total) $page = $total;
$start = $page * $num - $num;
$qury_start_num = "LIMIT $start, $num";
}
$result = mysql_query("SELECT * FROM table_products Where visible='1' ORDER BY $sorting",$link);
if (mysql_num_rows($result) > 0)
{
 $row = mysql_fetch_array($result);
 do
 {
  echo '
```

Если в таблице товаров больше чем 0 то они выведутся командой echo

Задаем классы для корзины

```
<li>
<p class="style-title-grid" ><a href="" >'.$row["title"].'</a></p>
<a class="add-cart-style-grid" ></a>
```

Если наводим на стоимость то она меняет цвет, курсор, и обретает подчеркивание/

```
<p class="style-price-grid" ><strong>'.$row["price"].'</strong> руб.</p>
```

```
<div class="mini-features" >
'.$row["mini_features"].'
</div>
</li>
';
}
while($row = mysql_fetch_array($result));
}
?>
</ul>
</div>
```

Делаем страничку на кнопку регистрации

registration.php

```
<?php
```

Подключаем скрипт для работы с БД

```
include("include/db-connect.php");
?>
```

Снова информация HTML от WC3

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
```

Главная часть

```
<head>
```

Задаем тип контента — текс и кодировку

```
<meta http-equiv="content-type" content="text/html; charset=windows-
1251" />
```

Подключаем стили

```
<link href="css/reset.css" rel="stylesheet" type="text/css" />
<link href="css/style.css" rel="stylesheet" type="text/css" />
```

Указываем название вкладки

```
<title>Регистрация</title>
</head>
<body>
```

Задаем id для тела регистрации

```
<div id="block-body">
```

При помощи PHP подключаем шапку

```
<?php
include("include/block-header.php");
?>
</div>
```

Начинаем наполнение основного блока странички регистрации

```
<div id="block-content">
```

Задаем поля на регистрации, размер форматирование H-2

```
<h2 id="h2-title">Регистрация</h2>
<form method="post"> id="form-reg" action="/reg/handler_reg.php"
<p id="reg_message"></p>
<div id="block-basket-form-registration">
<ul id="form-registration">
```

Задаем поля для регистрации

Задаем поле для ввода логина

```
<li>
<label>Логин</label>
<span>*</span>
```

```
        <input type="text" name="reg_login" id="reg_login"  />
        </li>
```
Задаем поле пароля
```
        <li>
        <label>Пароль</label>
        <span>*</span>
        <input type="text" name="reg_pass" id="reg_pass"  />
        </li>
```
Задаем поле для ввода фамилии
```
        <li>
        <label>Фамилия</label>
        <span>*</span>
        <input type="text" name="reg_surname" id="reg_surname"  />
        </li>
```
Задаем поле для ввода имени
```
        <li>
        <label>Имя</label>
        <span>*</span>
        <input type="text" name="reg_name" id="reg_name"  />
        </li>
```
Задаем поле для ввода Отчества
```
        <li>
        <label>Отчество</label>
        <span>*</span>
        <input type="text" name="reg_patronymic" id="reg_patronymic"  />
        </li>
```
Задаем поле для ввода электронной почты

```
<li>
<label>E-mail</label>
<span>*</span>
<input type="text" name="reg_email" id="reg_email"  />
</li>
```

Задаем поле для ввода мобильного телефона

```
<li>
<label>Мобильный телефон</label>
<span>*</span>
<input type="text" name="reg_phone" id="reg_phone"  />
```

Задаем поле для для ввода адреса доставки товаров

```
<li>
<label>Адрес доставки</label>
<span>*</span>
<input type="text" name="reg_address" id="reg_address"  />
</li>
</ul>
</div>
<p  align="right"><input  type="submit"  name="reg_submit"  id="form_submit"
value="Регистрация" /></p>
```

Задаем кнопку регистрации

```
</form>
</div>
```

На PHP подключаем низ сайта

```
<?php
        include("include/block-footer.php");
?>
```

```
</div>
</body>
</html>
```

на рисунке (Рис. 2.17) показано как выглядит регистрационный блок

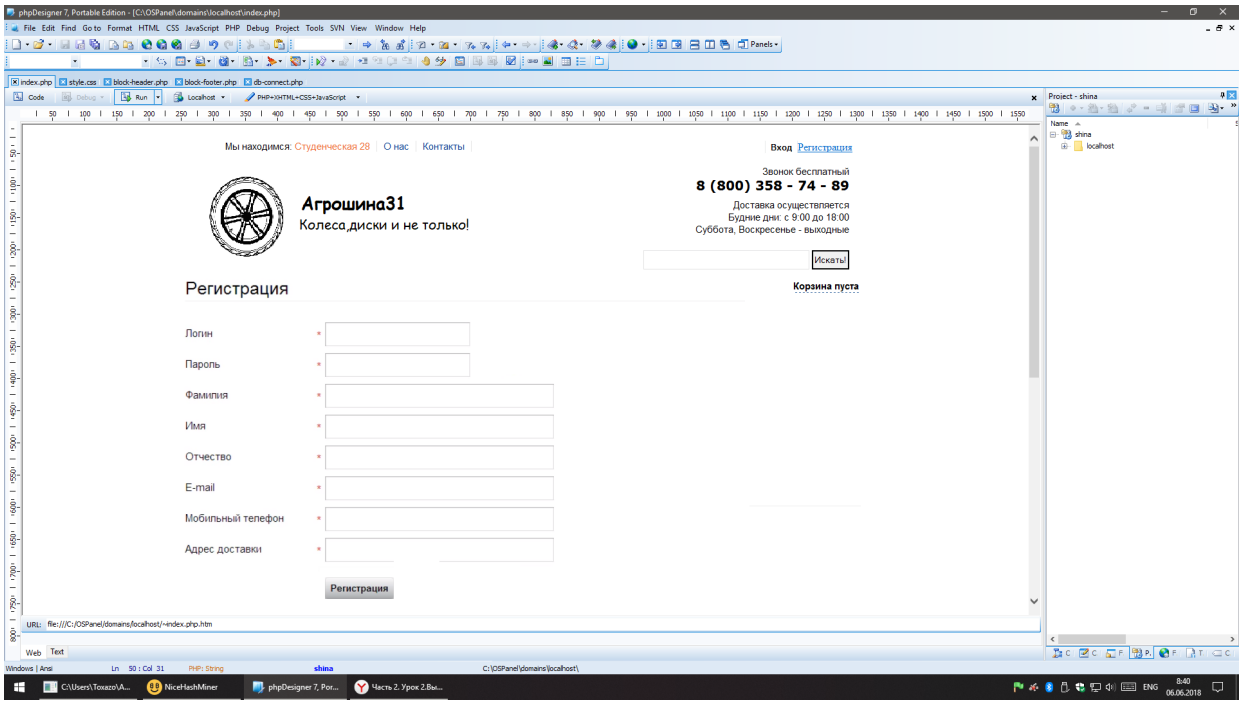

Рис. 2.17. - Готовый макет регистрации

Задаем стиль при наведении на корзину

```
{
   $row = mysql_fetch_array($result);
   do
   {
    echo '
    <li>
    <p class="style-title-grid" ><a href="" >'.$row["title"].'</a></p>
    <a class="add-cart-style-grid" ></a>
```

При наведении на кнопку добавить в корзину — цена подчеркивается

```
    <p class="style-price-grid" ><strong>'.$row["price"].'</strong> руб.</p>
```

```
<div class="mini-features" >
'.$row["mini_features"].'
</div>
</li>
';
}
```

Задаем результат из таблицы БД

```
while($row = mysql_fetch_array($result));
}
?>
</ul>
</div>
```

На PHP подключаем блок футер

```
<?php
include("include/block-footer.php");
?>
</div>
</body>
```

block-header.php

Заполняем блок с поиском по товарами, задаем id блок поиска

```
<div id="block-search">
```

Внутри поля для ввода по умолчанию бдет вписан текст

```
<form method="GET" action="search.php?q=" >
<input type="text" id="input-search" name="q" placeholder="Поиск по тысячам
наименований" />
```

Делаем кнопку искать

<input type="submit" id="button-search" value="Искать!"/>

</form>

Задаем id для блока с корзиной

Корзина пуста

Задаем id для линии что будет отделять поисковой блок сверху от шапки

В конце — концов мы подводим web-приложение к своему финальному виду!

На рисунке (Рис. 2.18) показан апплет для добавления товара в базу напрямую, а на рисунке (Рис. 2.19) показан результат.

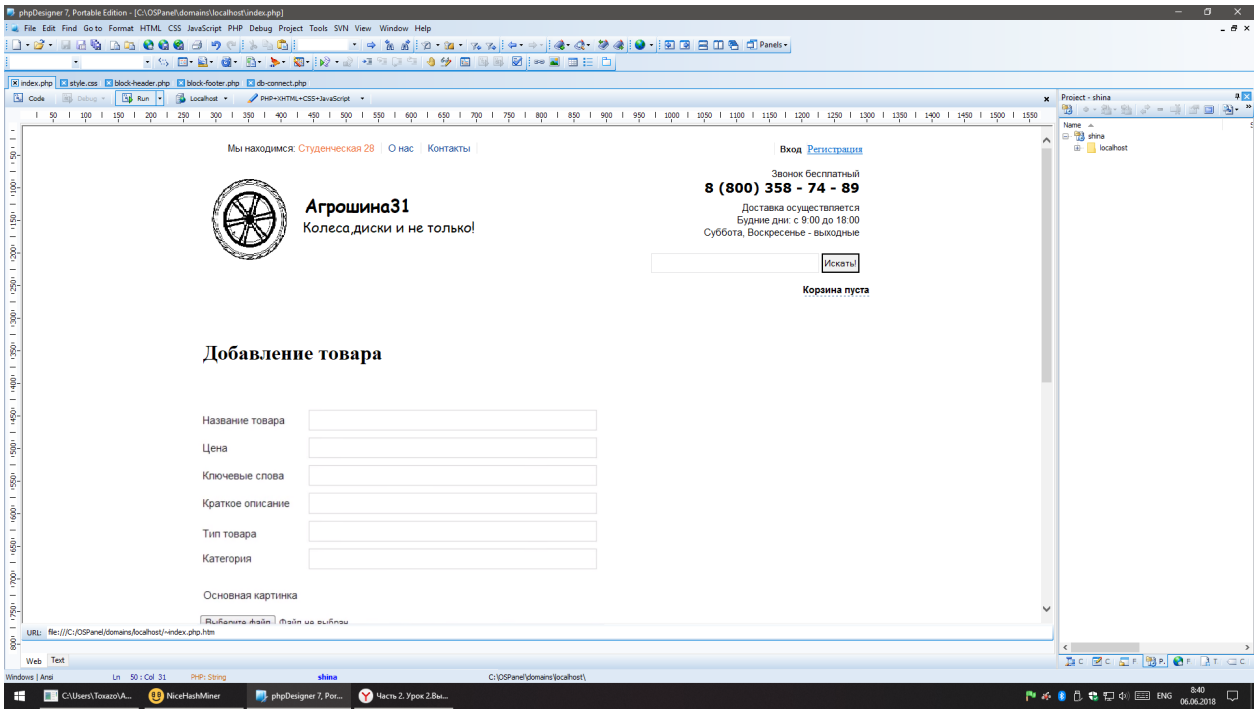

Рис. 2.18. - добавление товара в таблицу товары

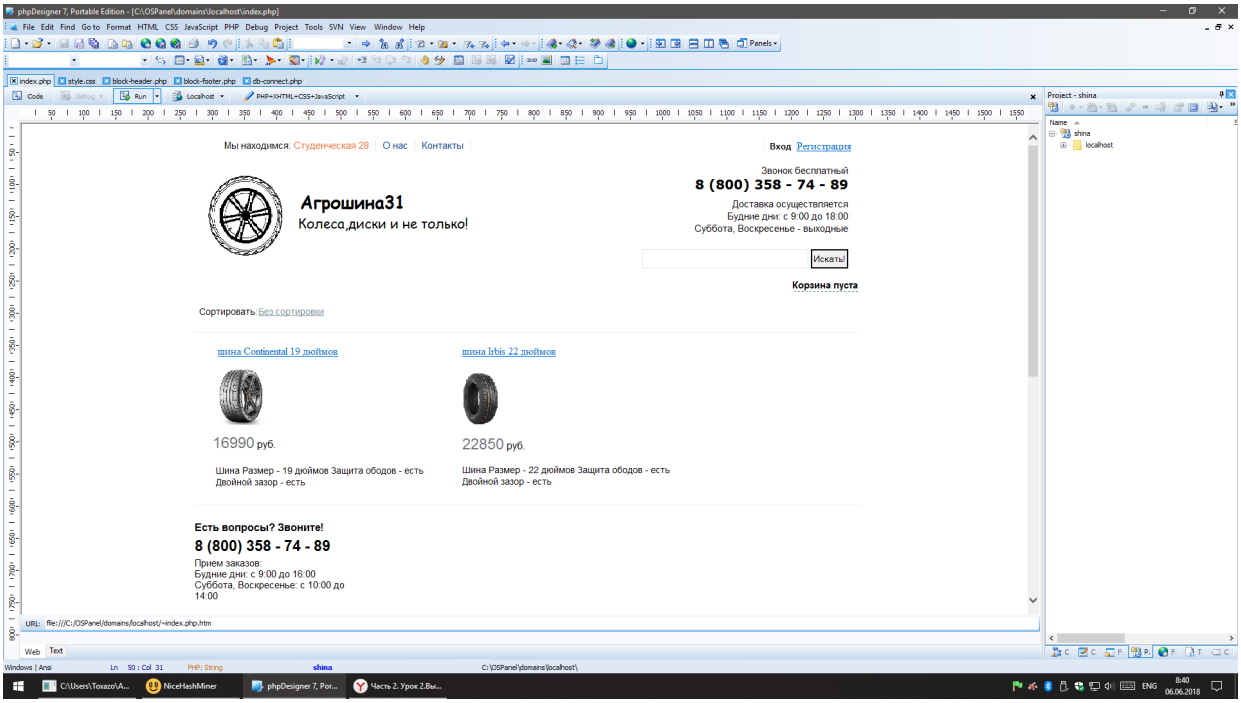

Рис. 2.19. - итоговый вид без верстки

На основе спроектированных страниц получилась следующая схема (Рис. 2. 20):

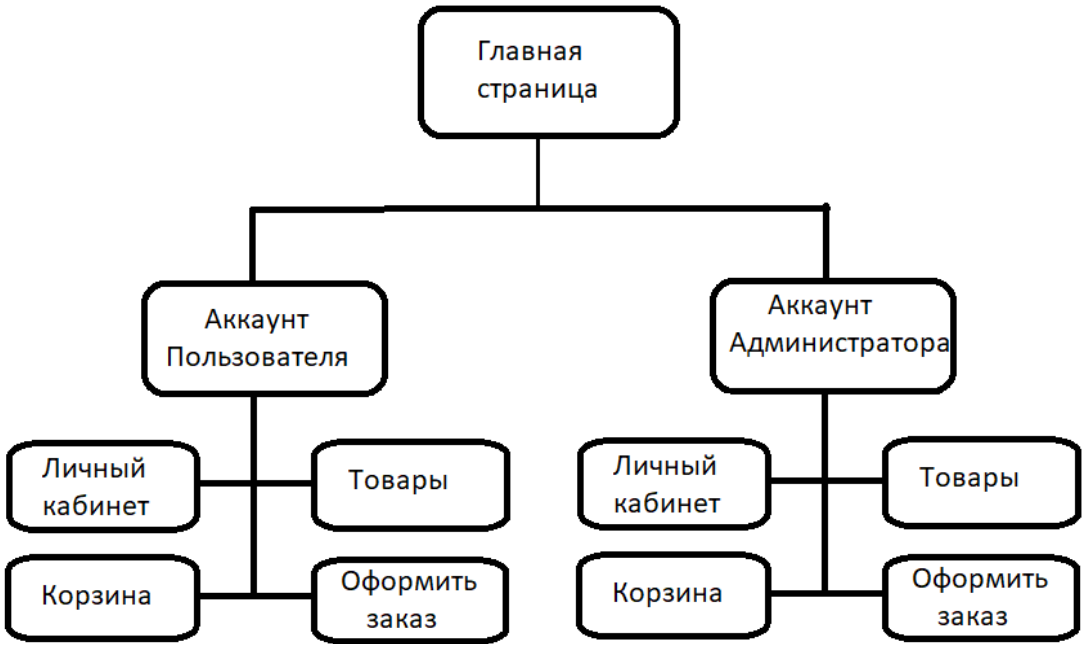

Рис. 2. 20. - даталогическая схема

ГЛАВА 3 ТЕСТИРОВАНИЕ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ

3.1. Программа тестирования

Тестовые сценарии:

1. Авторизация

Необходимо: Ввести данные для авторизации.

Входные данные: логин и пароль.

Ожидаемый результат: Успешная авторизация и переход в личный кабинет.

2. Поиск по товарам

Необходимо: Найти товар по названию.

Входные данные: название.

Ожидаемый результат: В списке товаров останутся только товары с заданным в поисковом запросе названием

3. Сортировка по товарам

Необходимо: Отсортировать список товаров по типу.

Ожидаемый результат: Порядок товаров в таблице изменится в алфавитном порядке по типу.

4. Добавление товара

Необходимо: добавить новый товар в базу данных.

Входные данные: любые данные для теста товара.

Ожидаемый результат: Данные о товаре внесены в базу данных.

5. Поиск по производному

Необходимо: Найти товар по производному значению.

Ожидаемый результат: В списке товаров останется только товар с заданным в поисковом запросе значением.

6. Сортировка товаров по наличию на складе

Необходимо: Отсортировать список товаров по наличию на складе.

Ожидаемый результат: В таблице будет показываться только тот товар, который на текущий момент находится на складе.

3.2. Результаты тестирования

1. На рисунке (Рис. 3. 1) указан интерфейс входа в личный кабинет.

На рисунке (Рис. 3. 2) показано что авторизация прошла успешно.

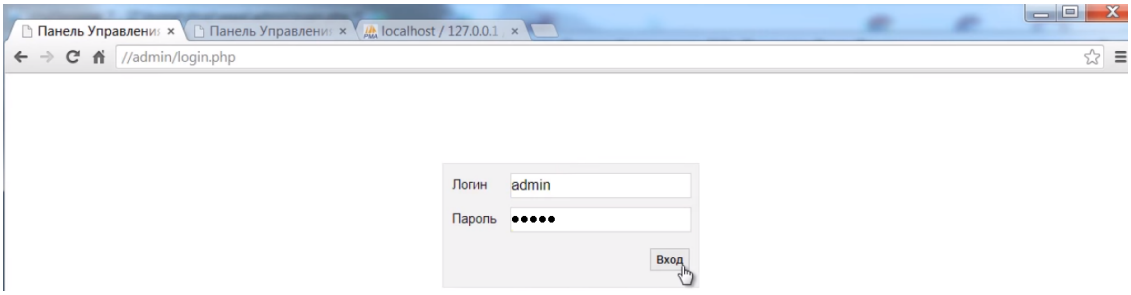

Рис. 3. 1. - страница login.php

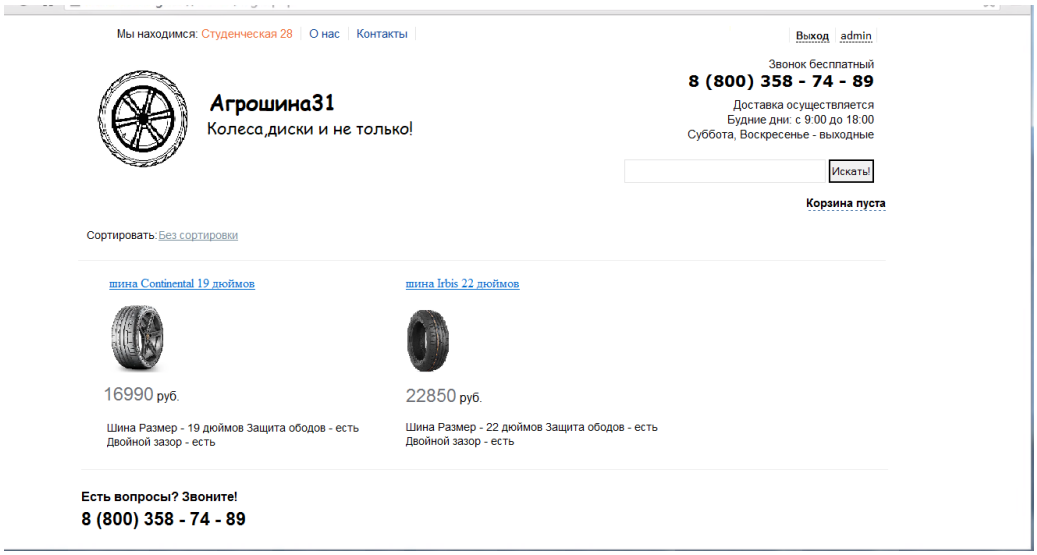

Рис. 3. 2. - страница index.php авторизация прошла успешно!

2. На рисунке (Рис. 3. 3) показан отсортированный товар.

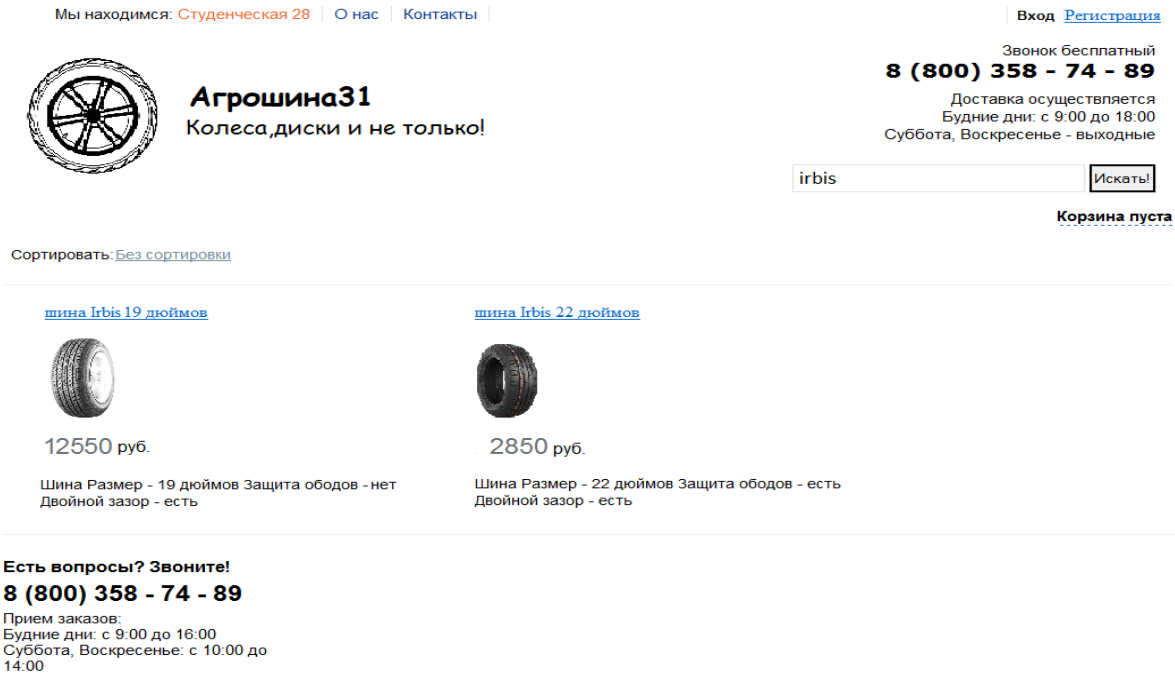

Рис. 3. 3. - показаны найденные совпадения

3. На рисунке (Рис. 3. 4) товар отсортирован по типу.

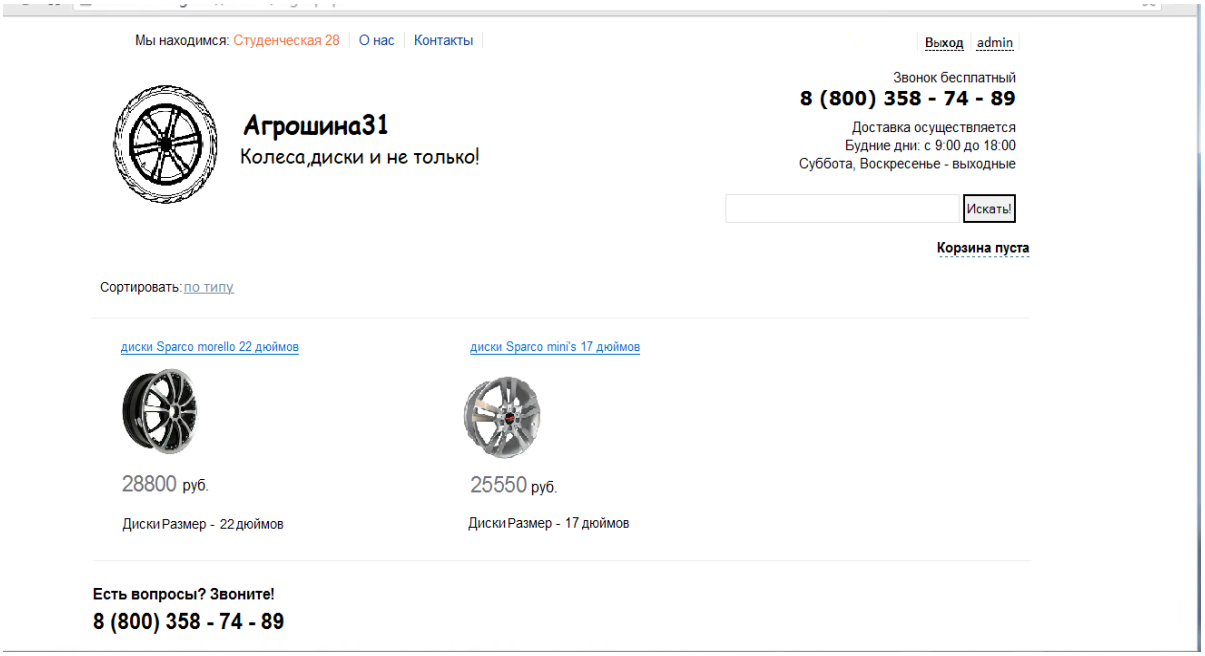

Рис. 3. 4. - товар успешно отсортирован по типу

4. На рисунке (Рис. 3. 5) показано добавление товара.

На рисунке (Рис. 3. 6) показан результат.

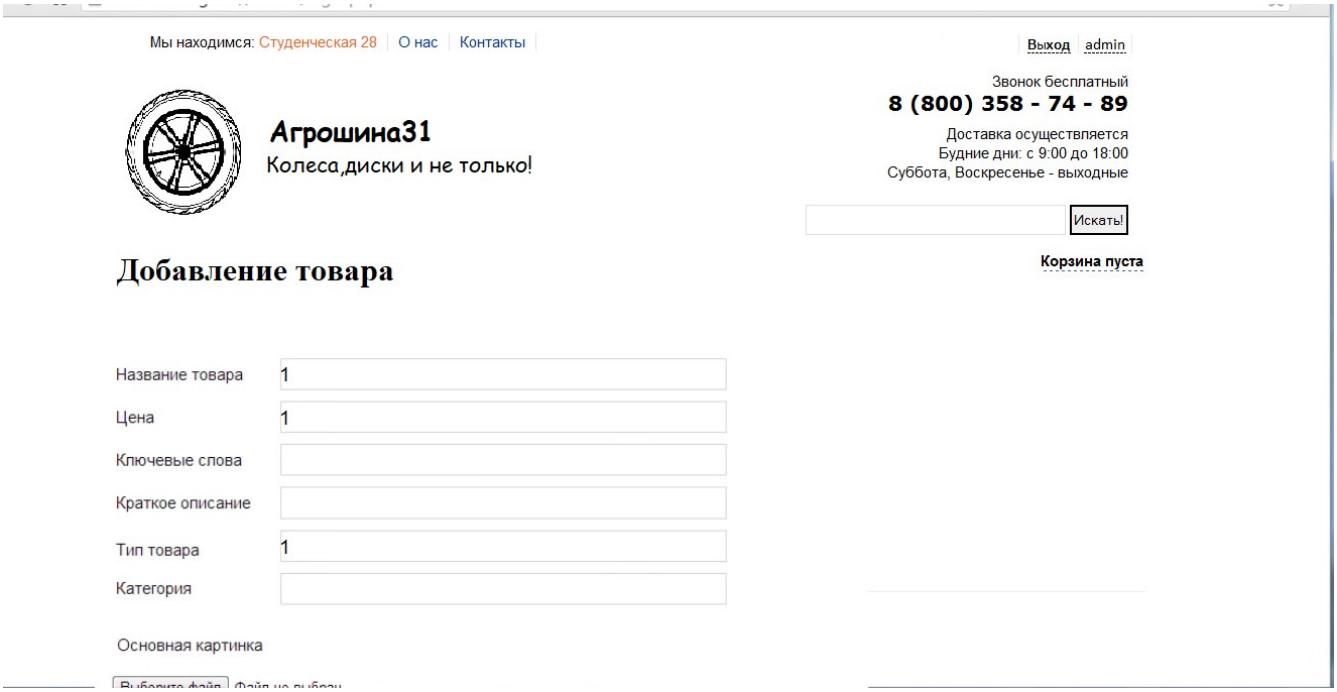

Рис. 3. 5. - добавление товара в таблицу базы данных

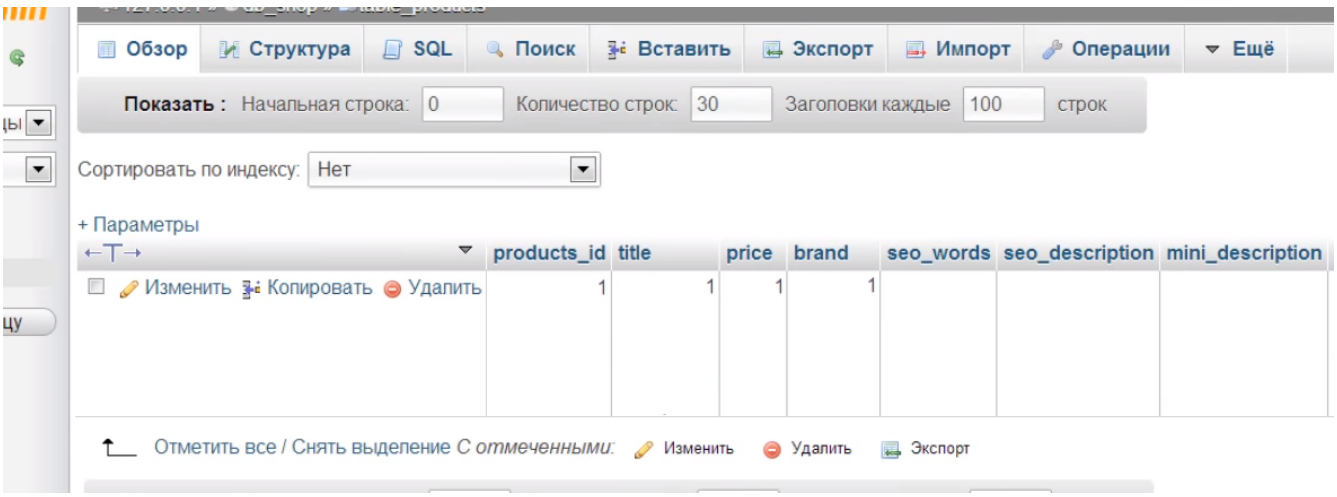

Рис. 3. 6. - товар успешно добавлен в таблицу базы данных

5. На рисунке (Рис. 3. 7) показано изменение товара.

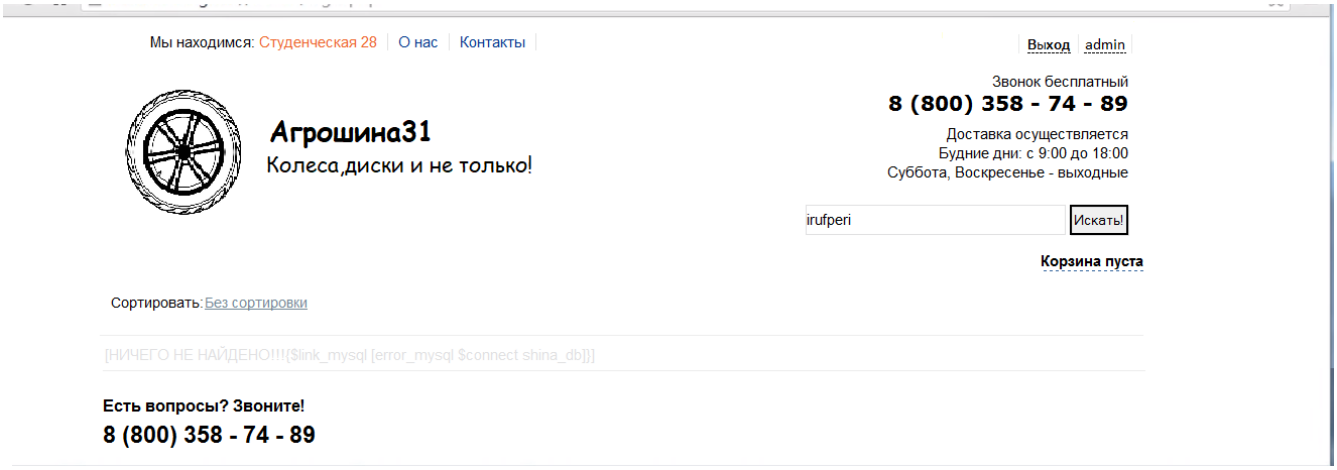

Рис. 3. 7 . - производного значения в БД не обнаружено, сообщение об ошибке

6. На рисунке (Рис. 3.8) показана сортировка

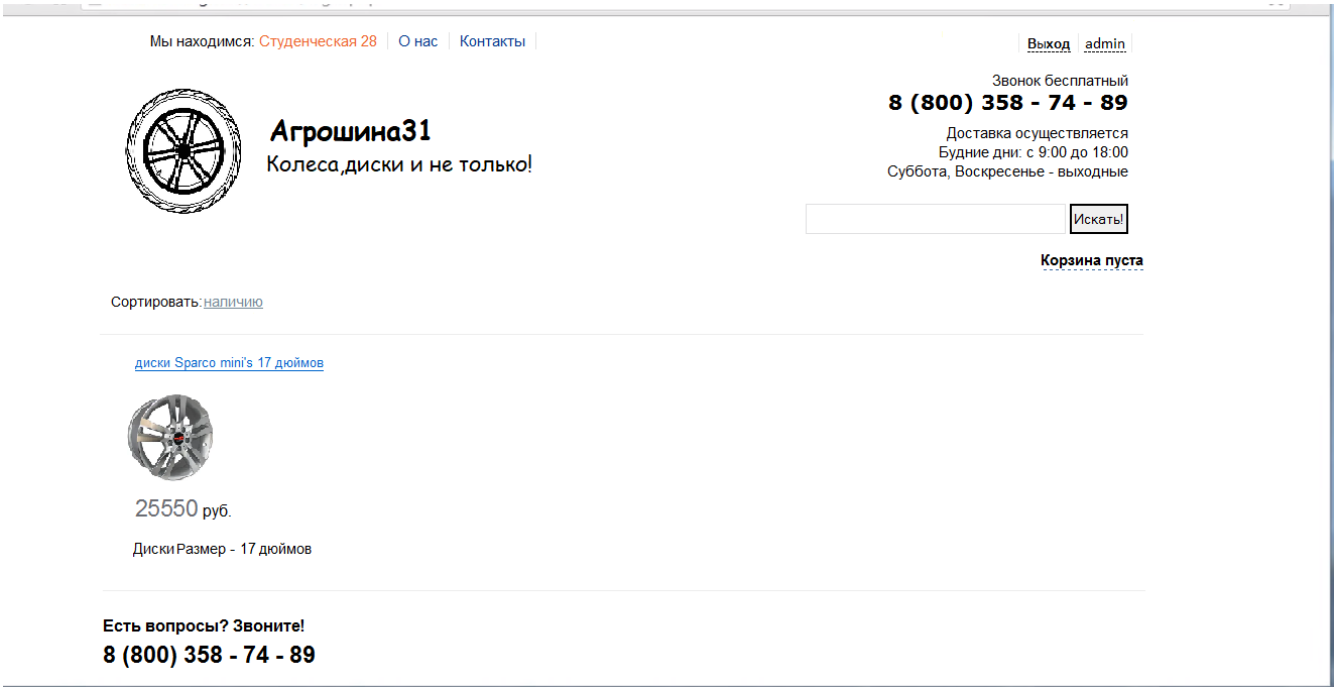

Рис. 3.8 . - по наличию найден только один товар, т.к. у остальный значение count=0

ЗАКЛЮЧЕНИЕ

В ходе работы над ВКР была изучена, спроектирована и разработана автоматизированная система для организации «Агрошина 31». Предложенное решение позволяет обеспечить бесперебойное функционирование и администрирование автоматизированной системы с высоким уровнем защищенности. При разработке были использованы новейшие технологические решения в сфере PHP для web-приложений и MySQLi в сфере таблиц баз данных.

В данной работе была разработана информационная система для автоматизации рутинного процесса. Данный процесс на предприятии наиболее сложен в связи с большим количеством клиентов, разнообразием поддерживаемых услуг, что накладывает свои требования к разрабатываемой системе.

В первой главе был выполнен комплекс работ, направленных на обоснование необходимости автоматизации: определена сущность задачи, описаны основные свойства системы.

Во второй главе дана характеристика информационной архитектуры разрабатываемой системы, построена модель задачи, описана структура таблиц и баз данных, структура веб-приложения и технология реализации системы.

В третьей главе была составлена программа тестирования и проведено само тестирование автоматизированной системы.

Таким образом удалось реализовать автоматизированную систему, которая удовлетворяет всем заданным требованиям заказчика. А благодаря гибкости и поддержке новейших технологий данная система может быть использована и интегрирована в ряде компаний которым необходим смежный комплекс для решения задач.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Эрик Мейер — "CSS-каскадные таблицы стилей. Подробное руководство (Cascading Style Sheets: The Definitive Guide)"

2. Томсон Лаура и Веллинг Люк. Разработка Web-приложений на PHP и MySQL: Пер. с англ./Лаура Томсон, Люк Веллинг. — 2-е изд., испр. — СПб: ООО «ДиаСофтЮП», 2003. — 672 с.

3. Робин Никсон «Создаем динамические веб-сайты с помощью PHP, MySQL и JavaScript». Питер, 2013. – 496 с.

4 Александр Мазуркевич. MB PHP: настольная книга программиста /Александр Мазуркевич, Дмитрий Еловой. — Мн.: Новое знание, 2003. — 480 с.: ил

5. Зольников Д.С. PHP 5. Полная версия – М.: НТ Пресс, 2007. – 352 с.

6. Гутманс Э., Баккен С, Ретанс Д. PHP 5. Профессиональное программирование. / Пер. с англ. СПб: Символ- Плюс, 2006. 704 с., ил.

7. Кузнецов Максим, Симдянов Игорь. Самоучитель MySQL 5. — Спб.: «БХВ-Петербург», 2006. — 560 с.

8. Мишель Е. Дэвис и Джон А. Филипс. Изучаем PHP и MySQL, 2008

9. Тим Конверс, Джойс Парк и Кларк Морган. PHP 5 и MySQL. Библия пользователя, 2006

10. Николай Прохоренок. HTML, JavaScript, PHP и MySQL. Джентельменский набор Web-мастера, 2010

11.Съерра К. И Бейтс Б. Изучаем java script и java – сходства и преемственность 2012

12. Хрусталева Е.Ю. Разработка сложных отчетов в 1С. Предприятии 8. Системы компоновки данных.

13. Полякова, Л.Н. Основы SQL: Курс лекций. Учебное пособие М.: ИНТУИТ.РУ, 2004. - 228 с.

14. Гарнаев, А.Ю. WEB - программирование на Java и Java Script. СПб.: БХВ - Петербург, 2003. - 339 с.

15. Кузнецов, М.В. PHP и MySQL для начинающих - Кудиц-образ, 2005

ПРИЛОЖЕНИЕ

Приложение 1

```
*==========================================================================
==================*/
/* DBMS: Firebird 2*/
/* Created on : 18.06.2018 18:48:00                          */
/
*==========================================================================
==================*/
/
*==========================================================================
==================*/
/*                          TABLES                          */
/
*==========================================================================
==================*/
CREATE TABLE table_products (
  PRODUCTS_ID           INTEGER NOT NULL,
  TITLE                 VARCHAR(255),
  PRICE                 INTEGER,
```

BRAND VARCHAR(255),

SEO_WORDS VARCHAR(255),

SEO_DISCRIPTION VARCHAR(255),

MINI_DESCRIPTION VARCHAR(255),

IMAGE VARCHAR(255),

DESCRIPTION VARCHAR(255),

MINI_FEATURES VARCHAR(255),

FEATURES VARCHAR(255),

DATETIME DATE,

NEW INTEGER,

LEADER INTEGER,

SALE INTEGER,

VISIBLE INTEGER,

"COUNT" INTEGER,

TYPE_TOVARA VARCHAR(255),

BRAND_ID INTEGER,

VOTE INTEGER,

VOTES FLOAT,

UPLOAD_IMAGE_ID INTEGER,

CATEGORY_ID INTEGER,

```
 REVIEWS_ID              INTEGER,
CONSTRAINT PK_table_products PRIMARY KEY (PRODUCTS_ID)
);
CREATE TABLE category (
 CATEGORY_ID             INTEGER NOT NULL,
 TYPE                    VARCHAR(20),
 BRAND                   VARCHAR(50),
 REG_ADMIN_ID            INTEGER,
CONSTRAINT PK_category PRIMARY KEY (CATEGORY_ID)
);
CREATE TABLE cart (
 CART_ID                 INTEGER NOT NULL,
 CART_ID_PRODUCT         INTEGER,
 CART_PRICE              INTEGER,
 CART_COUNT              INTEGER,
 CART_DATETIME           TIME,
 CART_IP                 VARCHAR(100),
 PRODUCTS_ID             INTEGER,
CONSTRAINT PK_cart PRIMARY KEY (CART_ID)
);
```

```
CREATE TABLE news (
  NEWS_ID               INTEGER NOT NULL,
  TITLE                 VARCHAR(255),
  TEXT                  VARCHAR(255),
  "DATE"                DATE,
  REG_ADMIN_ID          INTEGER,
CONSTRAINT PK_news PRIMARY KEY (NEWS_ID)
);
CREATE TABLE reg_user (
  REG_USER_ID           INTEGER NOT NULL,
  LOGIN                 VARCHAR(10),
  PASS                  VARCHAR(100),
  NAME                  VARCHAR(100),
  SURNAME               VARCHAR(100),
  PATRONYMIC            VARCHAR(100),
  EMAIL                 VARCHAR(100),
  PHONE                 VARCHAR(10),
  ADRESS                VARCHAR(255),
  DATETIME              TIME,
```

```
 IP                  VARCHAR(50),
 REG_ADMIN_ID        INTEGER,
CONSTRAINT PK_reg_user PRIMARY KEY (REG_USER_ID)
);
CREATE TABLE upload_images (
 UPLOAD_IMAGE_ID     INTEGER NOT NULL,
 PRODUCTS_ID         INTEGER,
 IMAGE               VARCHAR(255),
CONSTRAINT PK_upload_images PRIMARY KEY (UPLOAD_IMAGE_ID)
);
CREATE TABLE table_reviews (
 REVIEWS_ID          INTEGER NOT NULL,
 PRODUCTS_ID         INTEGER,
 NAME                VARCHAR(100),
 GOOD_REVIEWS        VARCHAR(255),
 BAD_REVIEWS         VARCHAR(255),
 COMMENT             VARCHAR(255),
 "DATE"              DATE,
 MODERAT             INTEGER,
 REG_ADMIN_ID        INTEGER,
```

```
CONSTRAINT PK_table_reviews PRIMARY KEY (REVIEWS_ID)
);
CREATE TABLE reg_admin (
 REG_ADMIN_ID           INTEGER NOT NULL,
 LOGIN                  VARCHAR(255),
 PASS                   VARCHAR(255),
 FIO                    VARCHAR(255),
 ROLE                   VARCHAR(255),
 EMAIL                  VARCHAR(50),
 PHONE                  VARCHAR(50),
 VIEW_ORDERS            INTEGER,
 ACCEPT_ORDERS          INTEGER,
 DELETE_ORDERS          INTEGER,
 ADD_TOVAR              INTEGER,
 EDIT_TOVAR             INTEGER,
 DELETE_TOVAR           INTEGER,
 ACCEPT_REVIEWS         INTEGER,
 DELETE_REVIEWS         INTEGER,
 VIEW_CLIENTS           INTEGER,
 DELETE_CLIENTS         INTEGER,
```

```
 ADD_NEWS               INTEGER,
 DELETE_NEWS            INTEGER,
 ADD_CATEGORY           INTEGER,
 DELETE_CATEGORY        INTEGER,
 VIEW_ADMIN             INTEGER,
 PRODUCTS_ID            INTEGER,
CONSTRAINT PK_reg_admin PRIMARY KEY (REG_ADMIN_ID)
);
CREATE TABLE orders (
 ORDER_ID               INTEGER NOT NULL,
 ORDER_DATETIME         DATE,
 ORDER_CONFIRMED        VARCHAR(10),
 ORDER_DOSTAVKA         VARCHAR(255),
 ORDER_PAY              VARCHAR(50),
 ORDER_TYPE_PAY         VARCHAR(100),
 ORDER_FIO              VARCHAR(255),
 ORDER_ADDRESS          VARCHAR(255),
 ORDER_PHONE            VARCHAR(50),
 ORDER_NOTE             VARCHAR(50),
 ORDER_EMAIL            VARCHAR(50),
```

```
 BUY_ID                INTEGER,
 REG_ADMIN_ID          INTEGER,
CONSTRAINT PK_orders PRIMARY KEY (ORDER_ID)
);
CREATE TABLE buy_products (
 BUY_ID                INTEGER NOT NULL,
 BUY_ID_ORDER          INTEGER,
 BUY_ID_PRODUCT        INTEGER,
 BUY_COUNT_PRODUCT     INTEGER,
 PRODUCTS_ID           INTEGER,
CONSTRAINT PK_buy_products PRIMARY KEY (BUY_ID)
);
/
*==========================================================================*/
/*                    FOREIGN KEYS                    */
/
*==========================================================================*/
ALTER TABLE table_products
  ADD CONSTRAINT FK_REFERENCE_3
```

```
      FOREIGN KEY (REVIEWS_ID)
         REFERENCES table_reviews (REVIEWS_ID)
;
ALTER TABLE table_products
   ADD CONSTRAINT FK_REFERENCE_1
      FOREIGN KEY (UPLOAD_IMAGE_ID)
         REFERENCES upload_images (UPLOAD_IMAGE_ID)
;
ALTER TABLE table_products
   ADD CONSTRAINT FK_REFERENCE_2
      FOREIGN KEY (CATEGORY_ID)
         REFERENCES category (CATEGORY_ID)
;
ALTER TABLE category
   ADD CONSTRAINT FK_REFERENCE_12
      FOREIGN KEY (REG_ADMIN_ID)
         REFERENCES reg_admin (REG_ADMIN_ID)
;
ALTER TABLE cart
   ADD CONSTRAINT FK_REFERENCE_4
```

```
      FOREIGN KEY (PRODUCTS_ID)
         REFERENCES table_products (PRODUCTS_ID)
;
ALTER TABLE news
   ADD CONSTRAINT FK_REFERENCE_9
      FOREIGN KEY (REG_ADMIN_ID)
         REFERENCES reg_admin (REG_ADMIN_ID)
;
ALTER TABLE reg_user
   ADD CONSTRAINT FK_REFERENCE_8
      FOREIGN KEY (REG_ADMIN_ID)
         REFERENCES reg_admin (REG_ADMIN_ID)
;
ALTER TABLE table_reviews
   ADD CONSTRAINT FK_REFERENCE_11
      FOREIGN KEY (REG_ADMIN_ID)
         REFERENCES reg_admin (REG_ADMIN_ID)
;
ALTER TABLE reg_admin
   ADD CONSTRAINT FK_REFERENCE_13
```

```
      FOREIGN KEY (PRODUCTS_ID)
         REFERENCES table_products (PRODUCTS_ID)
;
ALTER TABLE orders
   ADD CONSTRAINT FK_REFERENCE_6
      FOREIGN KEY (BUY_ID)
         REFERENCES buy_products (BUY_ID)
;
ALTER TABLE orders
   ADD CONSTRAINT FK_REFERENCE_10
      FOREIGN KEY (REG_ADMIN_ID)
         REFERENCES reg_admin (REG_ADMIN_ID)
;
ALTER TABLE buy_products
   ADD CONSTRAINT FK_REFERENCE_7
      FOREIGN KEY (PRODUCTS_ID)
         REFERENCES table_products (PRODUCTS_ID)
;
```

Приложение 2

index.php

```
<?php
  include("include/db-connect.php");
$sorting='sort';
if(isset($_GET['sort'])) {
      $sorting = $_GET['sort'];
        }
      switch($sorting)
        {
       case'price-asc':
       $sorting = 'price ASC';
       $sort_name = 'От дешевых к дорогим';
       break;
       case'price-desc':
       $sorting = 'price DESC';
       $sort_name = 'От дорогих к дешевым';
       break;
       case'brand':
```

```
        $sorting = 'brand';
        $sort_name = 'От А до Я';
        break;
        default:
        $sorting = 'products_id DESC';
        $sort_name = 'Нет сортировки';
        break;
}
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
        <meta http-equiv="content-type" content="text/html; charset=windows-1251" />
   <link href="css/reset.css" rel="stylesheet" type="text/css" />
   <link href="css/style.css" rel="stylesheet" type="text/css" />
        <title>Автоматизированнная система</title>
</head>
<body>
<div id="block-body">
<?php
```

```
        include("include/block-header.php");
?>
<div id="block-content">
<div id="block-sorting">
<ul id="options-list">
<li>Сортировать:</li>
<li><a id="select-sort"><?php echo $sort_name; ?></a>
<ul id="sorting-list">
<li><a href="index.php?sort=price-asc">От дешовых к дорогим</a></li>
<li><a href="index.php?sort=price-desc">От дорогих к дешовым</a></li>
<li><a href="index.php?sort=brand">От А до Я</a></li>
</ul>
</li>
</ul>
</div>

<ul id="block-tovar-grid">
<?php
  $num = 2;
  $count = mysql_query("SELECT COUNT(*) FROM table_products WHERE visible
= '1'",$link);
```

```
  $temp = mysql_fetch_array($count);
  If  ($temp[0] > 0)
  {
  $tempcount = $temp[0];
  $total = (($tempcount - 1) / $num) + 1;
  $total = intval($total);
  if(empty($page) or $page < 0) $page = 1;
    if($page > $total) $page = $total;
  $start = $page * $num - $num;
  $qury_start_num = "LIMIT $start, $num";
  }
$result = mysql_query("SELECT * FROM table_products Where visible='1' ORDER
BY $sorting",$link);
if (mysql_num_rows($result) > 0)
{
   $row = mysql_fetch_array($result);
   do
   {
    echo '
    <li>
    <p class="style-title-grid" ><a href="" >'.$row["title"].'</a></p>
```

```
    <a class="add-cart-style-grid" ></a>
    <p class="style-price-grid" ><strong>'.$row["price"].'</strong> руб.</p>
    <div class="mini-features" >
    '.$row["mini_features"].'
    </div>
    </li>
    ';
   }
      while($row = mysql_fetch_array($result));
}
?>
</ul>
</div>
<?php
        include("include/block-footer.php");
?>
</div>
</body>
</html>
```

Приложение 3

connect_db.php

```
<?php
$db_host      ='localhost';
$db_user      ='admin';
$db_pass      ='123456';
$db_database  ='db_shop';
$link = mysql_connect($db_host,$db_user,$db_pass);
mysql_select_db($db_database,$link) or die("Нет соединения с БД".mysql_error());
mysql_query("SET names cp1251");
?>
```

Приложение 4

style.php

```
#block-body{
    width: 1035px;
    height: auto;
    margin: 0 auto;
}
#block-header{
    width: 1035px;
    height: 220px;
}
#block-content{
    width: 1035px;
    height: 1000px;
}
#header-top-block{
    width: 1035px;
    height: 35px;
```

```
}
#block-footer{
    width: 1035px;
    height: 160px;
    margin-top: 10px;
}
#header-top-menu li{
    font: 15px sans-serif;
    float: left;
    margin-top: 10px;
    padding-left: 10px;
    padding-right: 10px;
    border-right: 1px solid #E3E3E3;
}
#header-top-menu a{
    font: 15px sans-serif;
    color: #00378C;
    text-decoration: none;
}
#header-top-menu a:hover{
```

```
    color: #EF662B;
    border-bottom: 1px dashed #EF662B;
}
#header-top-menu span{
    color: #EF662B;
    border-bottom: 1px dashed #EF662B;
}
#reg-auth-title{
    margin-top: 10px;
    margin-right: 10px;
    float: right;
}
#reg-auth-title a.top-auth{
   font: bold 13px sans-serif;
   text-decoration: none;
   padding-left: 8px;
   padding-right: 8px;
   padding-top: 4px;
   padding-bottom: 4px;
   border: 1px solid #E9EAEA
```

```
}
#top-line{
    background: url(/images/line.png);
    width: 1035px;
    height: 5px;
}
#img-logo{
    margin-top: 12px;
    position: absolute;
}
#personal-info{
    width: 300px;
    margin-left: 730px;
    margin-top: 7px;
}
#personal-info p{
    font: 15px sans-serif;
    margin-top: 2px;
    margin-bottom: 0px;
    margin-right: 10px;
```

```
}
#personal-info h3{
    font: bold 20px verdana;
    margin-top: 0px;
    margin-bottom: 10px;
    margin-right: 10px;
}
#block-search{
    width: 345px;
    height: auto;
    margin-left: 700px;
    margin-top: 23px;
}
#block-search input#input-search{
    border: 1px solid black;
    width: 250px;
    height: 26px;
    border: 1px solid #E1E1E2;
    font: 14px sans-serif;
    padding-left: 5px;
```

```
}
#button-search{
    width: 58px;
    height: 30px;
    border: 1px solid black;
}
#block-basket{
    margin-top: 0px;
    padding-top: 7px;
}
#block-basket a{
   font: bold 15px sans-serif;
   text-decoration: none;
   margin-left: 40px;
   border-bottom: 1px dashed #34659D;
   color: black;
}
#block-basket a:hover{
   border-bottom: none;
}
```

```
#nav-line{
    background: url(/images/line.png);
    width: 1035px;
    height: 5px;
}
#footer-phone{
    width: 270px;
    float: left;
    margin-top: 10px;
}
#footer-phone h4{
    font: bold 16px sans-serif;
    margin: 0;
}
#footer-phone h3{
    font: bold 23px sans-serif;
    margin: 5px 0;
}
#footer-phone p{
    font: 15px sans-serif;
```

```
    margin: 5px 0;
}
#bottom-line{
    background: url(/images/line.png);
    width: 1035px;
    height: 5px;
}
#block-sorting{
    height: 40px;
    border-bottom: 1px solid #EEEEEE;
}
#options-list > li{
    float: left;
    margin-left: 1px;
    font: 14px sans-serif;
}
#options-list{
    margin-left: 10px;
}
#select-sort{
```

```
    color: #748996;
    text-decoration: underline;
    cursor: pointer;
}
#sorting-list{
    border: 1px solid #A3C0D2;
    position: absolute;
    width: 200px;
    height: 66px;
    background-color: white;
    padding-top: 5px;
    display: none;
}
#sorting-list li{
    margin-top: 3px;
    margin-left: 10px;
}
#sorting-list a{
    font: 15px sans-serif;
    text-decoration: none;
```

```
    color: black;
}
#sorting-list a:hover{
    text-decoration: underline;
}
#block-tovar-grid > li{
    width: 380px;
    height: 140px;
    float: left;
}
#style-title-grid{
    margin-top: 0;
    margin-bottom: 0;
}
#style-title-grid a{
    font: bold 15px sans-serif;
    color: #0A83AB;
    text-decoration: underline;
}
#style-title-grid a:hover{
```

```
    color: #E26666;
}
.style-price-grid{
    font: 16px sans-serif;
    margin-top: 7px;
}
.style-price-grid strong{
    font: 23px sans-serif;
    color: #6B6D73;
}
.add-cart-style-grid{
    width: 48px;
    height: 25px;
    display: block;
    cursor: pointer;
    float: left;
    margin-top: 0;
    margin-right: 10px;
    border: 1px solid black;
}
```

```
.mini-features{
    font: 14px/18px sans-serif;
    margin-top: 0;
}
.h2-title{
    font: 25px sans-serif;
    border-bottom: 1px solid #EEEEEE;
    padding-bottom: 3px;
}
#block-form-registration{
    height: 450px;
}
#form-registration{
    margin-top: 30px;
}
#form-registration li{
    margin-top: 10px;
}
#form-registration > li >input{
    height: 30px;
```

```
    width: 320px;
    font: 17px sans-serif;
    margin-left: 200px;
    padding-left: 5px;
    color: black;
    border-left: 1px solid #ABABAB;
    border-top: 1px solid #ABABAB;

    border-right: 1px solid #DDDDDD;
    border-bottom: 1px solid #DDDDDD;
}
#form-registration > li > label{
    position: absolute;
    font: 15px sans-serif;
}
```