МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

кафедра Компьютерных технологий

«Допущен к защите»
Заведующий кафедрой _____
_____
(Ф.И.О., ученая степень, звание)
_____ «_____»_____ **20__ г.**
(подпись)

ДИПЛОМНЫЙ ПРОЕКТ

На тему: „Разработка интернет магазина компьютерных комплектующих"

Специальность 5В070400

Выполнил (а) Мирзахметов Э.Э. ВТиПО 12-2
(Фамилия и инициалы) группа

Научный руководитель Рахимбергенов С.Р. доцент к.т.н.
(Фамилия и инициалы, ученая степень, звание)

Консультанты:
по экономической части:
Бекишева А.Н., к.э.н., доцент
(Фамилия и инициалы, ученая степень, звание)
_____ «_____»_____ 20 16 г.
(подпись)

по безопасности жизнедеятельности:
Приходько А.Г. д.х.н. профессор
(Фамилия и инициалы, ученая степень, звание)
_____ «_____»_____ 20____ г.
(подпись)

по применению вычислительной техники:
Рахимбергенов С.Р. доцент к.т.н.
(Фамилия и инициалы, ученая степень, звание)
_____ «31» мая 20 16 г.
(подпись)

Нормоконтролер: _____
(Фамилия и инициалы, ученая степень, звание)
_____ «_____»_____ 20___ г.
(подпись)

Рецензент: _____
(Фамилия и инициалы, ученая степень, звание)
_____ «_____»_____ 20___ г.
(подпись)

Алматы 2016 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

Факультет Аэрокосмических и информационных технологий
Специальность 5В070400 - Вычислительная техника и прогр. обеспечение
Кафедра компьютерных технологий

ЗАДАНИЕ
на выполнение дипломного проекта

Студент Мурзахметов Ержан Элмуратович
(фамилия, имя, отчество)

Тема проекта Разработка интернет-магазина компьютерных технологий (комплектующих)

утверждена приказом ректора № 148 от «19» октября 2015 г.
Срок сдачи законченной работы «___» _____ 20___ г.
Исходные данные к проекту требуемые параметры результатов проектирования (исследования) и исходные данные объекта
Требуется разработать интернет магазин компьютерных комплектующих. Проанализировать рынок магазинов.

Перечень подлежащих разработке дипломного проекта вопросов или краткое содержание дипломного проекта:
1 Определить подходящий язык программирования и СУБД
2. Стандартизировать портирование сайта на разных браузерах
3. Разработка уникального дизайна

Перечень графического материала (с точным указанием обязательных чертежей)

В расчетной части приведена диаграмма построения магазина.

В экономической части приведена диаграмма роста и тенденции роста интернет-магазинов и экономическое обоснование.

В разделе о безопасности и жизнедеятельности приведен план помещения, которое кондиционируется и освещается

Рекомендуемая основная литература

1 Д.Кирсанов, Web-дизайн: „Символ", 1999
2 Уолл, „Использование WWW", 1997
3 Дэниел Этлок „Интернет-магазин и закупочная деят-ть", 200
4 Ресмон „HTML с самого начала", 1995г.

Консультанты по проекту с указанием относящихся к ним разделов

Раздел	Консультант	Сроки	Подпись
Экономика	Бекишева А.У.		
Вычислительная техника	Рахимбергенов С.Р.		
Безопасность жизнед-ти	Приходько Н.Г.		

Г Р А Ф И К
подготовки дипломного проекта

№ п/п	Наименование разделов, перечень разрабатываемых вопросов	Сроки представления руководителю	Примечание
1	Демонстрация выбранной для сайта графической среды	04.02.16г.	
2	Разработка макета сайта	15.03.16г.	
3	Разработка баз данных	22.04.16г.	
4	Демонстрация рабочего сайта - прототипа	29.04.16г.	
5	Готовый продукт (интернет-магазин со всеми необходимыми функциями	04.05.16г.	

Дата выдачи задания «____»_____20___ г.

Заведующий кафедрой _____ _____
(подпись) (Фамилия и инициалы)

Руководитель _____ Гаджиберликов С.Р. _____
(подпись) (Фамилия и инициалы)

Задание принял к исполнению
студент _____ _____
(подпись) (Фамилия и инициалы)

АҢДАТПА

Бұл бітіру жұмысында ғаламтор дүкені компьютерлік бөлшектерін сатуда мамандырылған, қол жетімді өнімдерді көру мүмкіндігін, тапсырыс өнімдерін, кері байланыс функциясын орнатылған. Құрылған жүйенің экономикалық тиімділігі есептелінген. Өміртіршілік қауіпсіздігі, еңбек ету жағдайлары мәселелері қарастырылған, жұмыс орнының жарықтандыру есептеулері келтірілген.

АННОТАЦИЯ

В данной дипломной работе рассматривается Интернет-магазин компьютерных технологий, в котором реализованы следующие возможности: возможность просмотра имеющихся продуктов, заказа продуктов, функция обратной связи для связи с администратором сайта. Произведен расчет экономической эффективности разработанной системы. Рассмотрены вопросы безопасности жизнедеятельности, условия труда.

ABSTRACT

In this thesis work is considered a grocery shop online, in which the following features: the ability to view available products, order products, the feedback function to contact a site administrator. The calculation of economic efficiency of the developed system. We consider health and safety issues, working conditions.

Содержание

Введение…….7
1. Функционирование и развитие электронной коммерции……10
1.1 Место «электронного» бизнеса в национальной экономике……...10
1.2 Преимущества и недостатки использования электронной коммерции..14
1.3 Понятие электронного магазина и его особенности……17
1.4 Сравнительная характеристика Казахстанских интернет-магазинов…...21
2. Проектная часть……30
2.1 Требования и средства проектирования web-документов……30
2.2 Выбор и описание языка программирования при разработке web-страниц……...…33
2.3 Использование базы данных mysql при проектировании…….34
2.4 Архитектура проектирования электронного магазина……50
3. Интерфейс разработки интернет магазина…….69
3.1 Правила построения интерфейса магазина……70
4. Экономическое обоснование проекта…….76
4.1 Маркетинговые исследования магазина ……76
4.2 Расходы по созданию и размещению магазина в сети интернет……..77
5. Мероприятия по охране труда и технике безопасности……..83
5.1 Эргономическое обеспечение…….83
5.2 Рабочее место программиста……...83
5.3 Средства пожаротушения……

Заключение…….87
Список литературы……91
Приложение А

Введение

Использование электронных технологий и Интернета во всех сферах жизни является одной из существенных черт и предпосылок формирования информационного общества и процесса глобализации мировой экономики. Интернет открыл новые возможности и формы экономических связей между участниками движения товаров, ресурсов и денег. В экономике развитых стран получили бурное развитие различные формы электронного бизнеса и, в частности, его важнейшего сегмента - электронной торговли. В Казахстане в этой области делаются только первые шаги. Это обусловлено не только относительной неразвитостью материальной базы и инфраструктуры, необходимых для распространения и применения электронных форм торговли, но и недостаточной разработанностью теоретических основ функционирования сферы электронной торговли.

Миллионы людей ежедневно, не выходя из дому, покупают различные товары в электронных магазинах. В мире, а в частности в Казахстане огромными темпами растет количество пользователей internet и как следствие количество «электронных» покупателей, потенциальных «электронных» покупателей.

Электронные магазины существенно уменьшают издержки производителя, сэкономив на содержании обычного магазина, расширяют рынки сбыта, так же как и расширяет возможность покупателя - покупать любой товар в любое время в любой стране, в любом городе, в любое время суток, в любое время года. Это дает электронным магазинам неоспариваемое преимущество перед обычными магазинами. Этот момент является существенным при переходе производителей с «обычной» торговли на «электронную»[1, с.23].

Что делает предприятие успешным на рынке? Высокое качество продукции, умение донести информацию о продукте до потребителя и

эффективная система сбыта. Предположим, что первые два условия выполнены, остается - наладить успешный процесс продаж. Его составляющие также известны - структурное подразделение, решающее задачи сбыта, квалифицированный персонал в этом подразделении, действующая система материальных и моральных стимулов и технология сбыта.Как показывает практика, именно с технологией сбыта в российских компаниях дело обстоит особенно плохо. Умение продавать считается сегодня чуть ли не искусством, даром свыше. Поэтому удачливых продавцов ценят, в их работу предпочитают не вмешиваться, а основная часть сотрудников отдела продаж сменяется с регулярностью, достойной удивления. Деятельность отдела продаж выглядит как "черный ящик", где на входе - множество контактов с потенциальными клиентами, а на выходе - тот или иной финансовый результат. Во многих компаниях встречаются проблемы сбыта, которые мешают эффективно работать отделу продаж, и не исчезают даже с подбором хороших продавцов. Решить их можно только путем автоматизации процесса продаж. В узком и технологическом смысле, под электронным бизнесом ранее понималось использование информационных технологий (в первую очередь связанных с Интернетом) для организации взаимодействия предприятия с внешней средой, включая поставщиков, потребителей, партнеров и т.д. При таком подходе электронный бизнес выступает, прежде всего, как достаточно сложная прикладная информационная система. Более широкий, или концептуальный, подход рассматривает электронный бизнес как способ предпринимательства, способствующий достижению стратегического успеха в новую информационную эпоху. При таком понимании электронный бизнес отнюдь не сводится к информационным технологиям или активности в Интернете. Он затрагивает все аспекты бизнеса, включая стратегию, процессы, организацию и технологию, и выводит его далеко за сложившиеся границы.

Темой данной дипломной работы является «Разработка интернет - магазина компьютерных комплектующих.

Целью дипломной работы является характеристика и оценка состояния и тенденций развития электронной торговли в Казахстане,

проектирование электронного магазина, оценка эффективности затрат по созданию организаций этой сферы.

Для достижения поставленной цели необходимо решить следующие задачи:

-Дать понятие электронного магазина и рассказать о его особенностях;

-Рассмотреть классификацию электронных магазинов;

-Спроектировать архитектуру электронного магазина;

-Разработать алгоритм работы электронного магазина;

-Разработать интерфейс интернет магазина.

Предмет исследования - являются закономерности, тенденции и показатели развития электронной торговли.

Объектом исследования - электронная торговля Интернет-товаров в Республике Казахстан.

Методологической основой исследования явились положения институциональной теории, теории менеджмента, экономики торговли и интернет-экономики, изложенные в трудах отечественных и зарубежных ученых.

При анализе изучаемых процессов использовался системный подход, современные методы обработки и разработки статистической информации и оценки эффективности электронного магазина, гипотетико-дедуктивный и индуктивный методы научного познания.

1. Функционирование и развитие электронной коммерции

1.1 Место «электронного» бизнеса в национальной экономике

Сеть Интернет, разработанная военными и учеными США на рубеже 60 и 70 гг., стала доступна рядовым пользователям во всем мире лишь в 1995 году. Тем не менее, за этот короткий промежуток времени эта сеть стала популярной не только среди простых пользователей персональных компьютеров, но и среди разного рода коммерческих структур. И это не удивительно – роль Интернета трудно переоценить. Сейчас Интернет является открытой сетью, которую можно использовать как канал для проведения сделок и продажи товаров и услуг.

Использование открытой сети рискованно и где-то даже опасно, так как нелегко защищать конфиденциальную информацию. С другой стороны следствием открытости является дешевизна использования этой сети и ее доступность, а это тоже немаловажный фактор для коммерческих структур, имеющих дело с большим количеством конкурентов. Таким образом, за 10 лет своего существования Интернет вырос из модной игрушки до серьезного бизнес-инструмента.[7,с. 324]

Многие отождествляют понятие электронного бизнеса с совершением покупок через Интернет. Но это разные понятия. Электронный бизнес – это любая сделка, совершенная посредством связанных между собой компьютеров (не обязательно при помощи Интернет), по завершению которой происходит передача прав собственности товаром или права пользования услугой. Электронный бизнес существует уже более 30 лет, первые же розничные продажи через Интернет начались лишь в 1995 г. Таким образом, покупки через Интернет – это лишь часть электронного бизнеса.

В настоящее время в Интернет представлено главным образом два сектора электронного бизнеса:

- сделки между деловыми партнерами;
- сделки между продавцом и конечным потребителем.

В принципе возможны и другие формы электронного бизнеса – индивидуум-государство, предприятие-государство, индивидуум-индивидуум и прочие. Тем не менее, часто рассматриваются лишь две первые модели, так как первая занимает на рынке около 85% денежного оборота, вторая – чуть менее 15%. Причем в эти 15% входит не только совершение покупок через Интернет, но и различные банковские операции, которые пользователи проводят, не выходя из дома, страхование и пр.

Итак, мы четко разделяем понятие электронной торговли и электронного бизнеса. Второе понятие значительно шире и включает в себя:

- продажи товаров;
- финансовый анализ;
- различные платежи;
- поиск работы и рабочей силы;
- поддержка клиентов и партнеров;
- реклама и прочее.

Электронный бизнес имеет значительный потенциал для роста. Наиболее многообещающими областями Интернет-экономики являются:

- рекламная деятельность
- домашние банковские операции
- продажа программного обеспечения и книг
- доставка товаров
- оффшорное программирование
- дистанционное образование.

По сути, в данный момент мы находимся в только самом начале новой эры экономики, которую экономисты называют Интерет-экономика или «экономика цифрового мира».[8,с. 2]

Говоря об электронном бизнесе в Казахстане, можно найти массу причин, которые осложняют развитие этой перспективной отрасли экономики:

- Небольшое количество пользователей сети (не менее 15%, а у нас 6 млн.).
- Скорость и качество каналов связи в Интернет.
- Средний пользователь Интернет отличается от среднестатистического жителя Казахстана.
- Низкая платежеспособность населения (книги, кассеты, мелочевка).
- Явный недостаток профессиональных специалистов в этой области.
- Недоверие покупателей и торговцев к отечественной банковской системе.
- Отсутствие правового регулирования.

О каждой из этой проблем можно весьма долго дискутировать. Однако одной из главных проблем на сегодняшний день видится именно правовой аспект. Именно он является основным сдерживающим фактором Интернет-экономики.

Совершение сделок через Интернет в первую очередь требует правового регулирования передачи и обмена данными. В настоящее время в нашей стране суд не принимает к рассмотрению электронные документы как доказательства доброй воли обоих участников сделки. Следовательно, необходимо определить процедуру подтверждения достоверности или подлинности таких документов. Эта проблема тесно связана с таким понятием как электронная подпись. В США и Германии в законах уже обозначено такое понятие, и юристы этих стран весьма смело таким понятием оперируют.

Правовое регулирование также необходимо в борьбе с компьютерной преступностью. Одной из важнейших проблем электронного бизнеса являются махинации в области электронных платежей. До сих пор такое понятие как «электронные деньги» окончательно не определено в российском законодательстве. В связи с этим электронные платежи еще не так безопасны, как те формы расчетов, к которым мы привыкли.

Технологии развиваются и совершенствуются быстрее, чем закон. Закон более консервативен по своей природе. На сегодняшний день суды не готовы разрешать споры и противоречия между участниками электронных сделок: у них нет специальных знаний и опыта в этой области. Они должны консультироваться с экспертами, а по сему они будут находиться под воздействием частного мнения таких экспертов.[9,с.352]

В Казахстане до сих пор не существует правовых документов, точно и четко регулирующих основные вопросы электронного бизнеса. В таком случае в настоящее время нам остается обратиться к опыту других развитых стран (США, Евросоюз, Израиль, Япония).

Так как электронный бизнес весьма успешен в США и имеет хорошие перспективы для развития, правительство США пытается создать хорошие условия для этой отрасли национальной экономики. В настоящее время действует трехлетний мораторий на налоги для компаний, начинающих деятельность в сфере электронного бизнеса.

В задачи государства так же входит защита несовершеннолетних, использующих Интернет-возможности в приобретении товаров и услуг, защита авторских прав, гарантии в приватности при обмене информацией. В США уже существует ряд законов, стоящих на защите Интернет-пользователей. Многие законы в США приняты уже 4-5 и даже 6 лет назад.

Для Казахстана электронная торговля открывает новые перспективы: улучшение качества казахстанских систем связи, обмен информацией, рост многочисленных новых казахстанских и международных предприятий.

Будучи неотрегулированными, такие вопросы как налогообложение и защита конфиденциальной информации могут порождать конфликты с налоговыми службами или даже приводить к нарушению прав человека, гарантированными Конституцией Республикой Казахстан.[10,с. 42]

Для того чтобы Казахстан мог максимально эффективно использовать достижения электронных средств связи в бизнесе, инвестициях и торговле, казахстанским органам власти необходимо:

- Принять Закон об электронно-цифровой подписи.
- Отменить или смягчить законы, предусматривающие обязательное бумажное оформление для заключения или регистрации сделок.
- Пересмотреть и уточнить порядок налогообложения осуществляемых через Интернет сделок.
- Не допускать и устранять рыночные барьеры, которые подавляют инвестиции в техническую инфраструктуру.
- Избегать введения новых налогов, лицензий, регистраций или любых других действий, которые подрывают уверенность в целесообразности капиталовложений в Интернет и электронную торговлю.

Ведущие отечественные и зарубежные эксперты поддерживают необходимость государственного стимулирования электронной торговли и устранения препятствий на пути ее роста. Казахстан пока еще может стать ключевым игроком мировой системы электронной торговли. Позднее влиться в этот процесс станет значительно труднее.

1.2 Преимущества и недостатки использования электронной коммерции

В этой главе мы рассмотрим преимущества и недостатки электронных магазинов, как для покупателей, так и для его создателей.

Преимущества виртуального магазина перед реальным очевидны. Уменьшается численность персонала за счет сокращения объема взаимодействия с клиентами, аренда дискового пространства и размещение

"электронной витрины" дешевле и проще аренды торговых помещений и размещения товаров на полках, нет нужды в кассовом обслуживании и т.д. Так же виртуальный магазин можно использовать как эффективный способ маркетингового исследования, тем более, что сегодня эта услуга довольно дорога в маркетинговых агентствах. Любой пользователь сети Интернет может быстро заполнить анкету, предлагаемую ему магазином через компьютер. Это позволяет без особых затрат изучить потребности и вкусы потенциальных покупателей и учесть результаты маркетингового обследования в своей работе.

А на чем проигрывает Интернет-магазин? На необходимости иметь хорошие каналы связи и аппаратно-программное обеспечение, да и доля доставки в себестоимости существенно возрастает. Проигрывает и на "синдроме недоверия", поскольку в Интернет-торговле покупатель менее защищен от недобросовестного продавца, да и постоянно присутствующий в Интернете хакерский фактор существенно повышает риск сделки.

Преимущества Интернет-магазинов для потребителей:

1.Экономия времени.

Когда человек работает 6 дней в неделю с 10 до 19 , ему бывает некогда сходить в магазин. Интернет-магазин позволяет сделать покупку, не выходя из офиса в любое время, а выбор и заказ товара займет у него несколько минут, если он точно знает, что хочет купить. Служба доставки интернет-магазина доставит выбранный товар в удобно время и место. Кроме этого выбор и оценка свойств товара происходит в интернет-магазине намного быстрее и удобнее чем в обычном магазине. Представьте, что вы пришли в обычный магазин бытовой техники и хотите сравнить параметры 20-30 радителефонов. Для этого вы должны изучить все ценники: запомнить характеристики, цены и названия моделей. Можно обратиться к продавцу-консультанту, если он не занят, с просьбой рассказать обо всех этих товарах. Но обычно ни один продавец не имеет столько свободного времени для работы с одним клиентом, и нет гарантии, что он владеет всей информацией.

2.Неограниченный ассортимент и информативность.

Кроме того, ассортимент интернет-магазина ничем не ограничен (как, например, ассортимент обычного магазина ограничен площадью торгового павильона). А если предусмотрен поиск по параметрам, то Вы просто можете указать характеристики, которым должен соответствовать товар, и затем уже выбирать из списка моделей, удовлетворяющих вашему запросу. И еще одно очень важное замечание — ни один менеджер по продажам не в состоянии помнить столько информации о товаре, сколько может предоставить Вам интернет-магазин. Кроме того, интернет-магазин в состоянии выдать Вам такую информацию, как покупательский рейтинг, советы и отзывы о товаре, статьи о товаре, которые могут предопределить Ваш выбор.

3. Экономия денег.

Затраты на работу интернет-магазина, включая доставку, существенно ниже чем у обычного. В отличие от обычного магазина интернет-магазин способен обслужить несколько сотен клиентов одновременно. Хотя на практике такого не встретишь. Кроме того, если покупатель проживает в другом городе, он получает возможность не тратиться на междугородние телефонные консультации. Всю информацию он может найти на страницах интернет-магазина.

Недостатки интернет-магазинов для покупателя:

-Нельзя "пощупать", нельзя узнать больше, чем написано (пример: мебель, одежда)

-проблемы гарантии, сопровождения (весьма критично в Казахстане)

-зачастую долгая доставка (до 6 недель), особенно если мы говорим о международности

Так, исследования, проведенные в начале 2004 г. английской фирмой Armor Group, показали, что 35 % дорогих товаров, продающихся через магазины, работающие в Интернет, являются подделками. Другими словами, например, предлагаемый покупателю «Panasonic» может оказаться не самим «Panasonic» этой известной фирмы, а его азиатской подделкой,

произведенной где-нибудь в Юго-Восточной Азии, но украшенной громким товарным знаком. Исследования также показали большую доверчивость покупателей. Например, покупатели легко готовы поверить во всемирную известность несуществующей торговой марки. Так, о знакомстве с никогда не существовавшей торговой маркой «Royal Alliance Insuarance» заявило 80 % опрошенных респондентов. Простота доступа в Интернет, массовость аудитории и потенциальная анонимность рекламодателей делает Интернет идеальной платформой для недобросовестных предпринимателей.[6,с. 645]

Члены Всемирной торговой организации (ВТО) договорились не вводить таможенные сборы на международные электронные платежи в сети Интернет. Покупка через глобальную сеть осуществляется так же, как и по обычному почтовому каталогу, только в электронном виде. Покупатель указывает свои банковские реквизиты и тип покупки. Затем получает товар. И здесь взимаются обычные почтовые сборы, как при обыкновенной пересылке.

1.3 Понятие электронного магазина и его особенности

Начиная с середины 90-х годов, во всем мире наблюдается рост активности в области онлайновой торговли. Вслед за крупными компаниями, производящими компьютерное оборудование в Сеть стали выходить торговцы традиционными товарами. Появилось большое количество книжных магазинов, магазины компакт-дисков и видеокассет, винные магазины. Сейчас практически любые товары можно купить через Сеть.

Электронная коммерция (e-commerce) - это ускорение большинства бизнес-процессов за счет их проведения электронным образом. В этом случае информация передается напрямую к получателю, минуя стадию создания бумажной копии на каждом этапе[2,с. 452].

Термин "электронная коммерция" объединяет в себе множество различных технологий, в числе которых - EDI (Electronic Data Interchange -

электронный обмен данными), электронная почта, Интернет, интранет (обмен информацией внутри компании), экстранет (обмен информацией с внешним миром). Таким образом, электронную коммерцию можно характеризовать как ведение бизнеса через Интернет.

Системы электронной коммерции можно разделить на два класса - системы для организации розничной торговли и системы для взаимодействия с деловыми партнерами (системы бизнес для потребителя и бизнес для бизнеса).

Под определение электронной коммерции подпадают не только системы, ориентированные на Internet, но также и "электронные магазины", использующие иные коммуникационные среды - BBS, VAN и т.д. В то же время процедуры продаж, инициированных информацией из WWW, но использующих для обмена данными факс, телефон и пр., могут быть лишь частично отнесены к классу электронной коммерции. Отметим также, что, несмотря на то, что WWW является технологической базой электронной коммерции, в ряде систем используются и другие коммуникационные возможности. Так, запросы к продавцу для уточнения параметров товара или для оформления заказа могут быть посланы и через электронную почту.

Виртуальный магазин — это реализованное в сети Интернет представительство путем создания Web-сервера для продажи товаров и услуг другим пользователям сети Интернет. Виртуальный магазин называют также Интернет-магазином. К нему полностью подходит определение виртуального предприятия. Иначе говоря, виртуальный магазин — это сообщество территориально разобщенных сотрудников магазина (продавцов, кассиров) и покупателей, которые могут общаться и обмениваться информацией через электронные средства связи при полном (или минимальном) отсутствии прямого личного контакта.[3,с. 64]

Электронная торговля в виртуальном магазине основывается на той же структуре, что и традиционная торговля.

Таблица 1 - Сравнительная характеристика традиционной и электронной торговли

Традиционный магазин	Виртуальный магазин
Торговый зал Ходьба покупателя по торговому залу и осмотр товаров на полках Магазина Личный контакт покупателя с продавцом (консультация) Выбор покупателем товара Заказ товара Выписка продавцом и вручение покупателю счета на оплату Оплата, покупателем счета на товар в кассе магазина наличными деньгами или банковской картой	Виртуальный магазин Просмотр покупателем страниц сервера Консультация у продавца (при необходимости) по компьютерной сети или по телефону Выбор покупателем товара Заказ товара через сервер Пересылка продавцом по компьютерным сетям покупателю счета на оплату Оплата покупателем счета по какой-нибудь системе электронных платежей (банковская карта, электронный чек, цифровые деньги, электронные деньги)

Перевод традиционной торговли в сеть Интернет делает ее более гибкой, так как электронная торговля, оперируя цифровой информацией в компьютерных сетях, облегчает сотрудничество людей.

Виртуальный магазин имеет доменный адрес. Как любой Web-сервер, виртуальный магазин состоит из целого ряда гипертекстовых страниц, зачастую с мультимедийными элементами.

Классифицировать электронные магазины можно по различным критериям. Наиболее интересной классификацией является классификация по модели бизнеса: Чисто онлайновый магазин[4,с. 12]

Совмещение оффлайнового бизнеса с онлайновым (когда интернет-магазин был создан на основе уже действующей реальной торговой структуры).

По отношению с поставщиками:

-имеют собственный склад (наличие реальных товарных запасов)

-работают по договорам с поставщиками (отсутствие значительных собственных запасов).

Классификация по товарному ассортименту – сотовые телефоны, аксессуары.

Среди методов розничной продажи товаров в Сети можно выделить:

- Интернет-магазины (автоматические магазины)
- Web-витрины
- торговые автоматы.

Вот эти разновидности интерактивных магазинов мы в подробности рассмотрим ниже: Internet-витрина, торговый автомат и автоматический магазин.

Internet-витрина - скорее это рекламный сервер. На витрине выкладывают информацию о товарах, которую постоянно обновляют. Затраты на ее создание и администрирование могут быть довольно низкими, а практическая польза такой витрины очевидна. Но это еще не торговля. Потенциальный покупатель, посетив витрину, должен позвонить на фирму, оплатить товар, договориться о доставке. Поэтому Internet-витрина оправданна в тех случаях, когда покупателя надо познакомить со сложной продукцией, на изучение которой в торговом зале у него уйдет слишком много времени.[5, с 58]

Internet-витрина может быть размещена где угодно - на собственном сервере, на сервере провайдера, на сервере, предоставляющем бесплатные страницы. Для работы с витриной достаточно иметь подключение через телефонную линию и минимум навыков работы с HTML.

Торговый автомат может не только выполнять функции витрины, но и принимать заказы и передавать их менеджеру, то есть оформлять заказы и выписывать счета на оплату без присутствия покупателя. Торговый автомат реально торгует и по соотношению затрат к результату наиболее предпочтителен для пилотных и тестовых проектов с небольшим потоком покупателей. Торговый автомат, так же как и Internet-витрину, можно

разместить и на своем сервере, и на сервере провайдера. Однако его создание и администрирование требует навыков и определенной квалификации.

Автоматический магазин - эффективное и комплексное решение в торговом бизнесе. Он не только выписывает счета, но и отслеживает заказы, принимает электронные платежи и формирует заявки на доставку товаров покупателям. Здесь задача менеджера - контролировать работу системы, сложную в обслуживании.

Поскольку автоматический магазин должен иметь постоянную связь с информационной системой компании, то размещать его лучше либо на корпоративном сервере в локальной сети, либо на удаленном сервере с постоянно действующим каналом связи. Интерактивные магазины могут торговать чем угодно.

1.4 Сравнительная характеристика казахстанских интернет-магазинов

Описание интернет-магазин:

Адрес в интернет: http://emega.kz

Телефон 8 (727) 329-81-65

Ассортимент товаров и услуг

Магазин занимается продажей товаров следующих категорий:

- компьютеров в сборе
- ностители информации
- орг/техника

КАК ОФОРМЛЯЕТСЯ ЗАКАЗ

Если Вы нашли необходимый товар, то можете приступить к формированию заказа. Рядом с описанием товара Вы всегда найдете текст "купить", нажав на который Ваш товар попадает в корзину. В правом верхнем углу окна Вашего браузера отображается состояние Вашей корзины. Нажав на текст "Корзина покупателя", Вы попадаете в корзину, где показаны

все набранные товары. Чтобы приступить к оформлению заказа Вы должны зарегистрироваться, если Вы не сделали этого раньше.

Далее Вам необходимо выбрать способ доставки товара из предлагаемого списка (варианты доставки зависят от места, куда нужно отправить заказ) и способ оплаты, который уже зависит от выбранного Вами способа доставки. Система должна пересчитать стоимость заказа с учетом стоимости доставки, если это не произошло, нажмите кнопку "Пересчитать". Для продолжения оформления заказа - нажмите кнопку "Продолжить".

После этого Вы оказываетесь на странице оформления заказа, внимательно посмотрите все параметры заказа и, если все правильно, нажмите кнопку "Заказать".

На следующей странице Вы сможете посмотреть образцы документов, которые помогут Вам правильно оплатить заказ.

Вы должны обязательно получить от “Болеро” подтверждение по электронной почте о том, что Ваш заказ принят. В отправленном письме будут ссылки для подтверждения Вами заказа или отказе от него. Ваш заказ будет обработан только после его подтверждения, то есть когда Вы нажмете соответствующую ссылку. В случае, если заказ не будет подтвержден в течение 7 дней, он будет автоматически удален.

НЕПОЛНЫЙ ЗАКАЗ

Иногда бывают случаи, когда один из заказанных Вами товаров заканчивается на складе, и магазин не может полностью исполнить Ваш заказ. В этом случае сотрудники магазина связываются с Вами по электронной почте и договариваются, как лучше поступить - отправить сначала часть заказа, а недостающие товары пересылать по мере их появления, дождаться пока не сформируется весь заказ или удалить из заказа отсутствующие товары.

СПОСОБЫ ОПЛАТЫ ТОВАРА

Наложенный платёж

Выбирая доставку почтой и оплату на месте, Вы используете так называемый наложенный платеж. В этом случае Вам, как правило, придется доплатить 10% от стоимости заказа за услуги почты.

Юридическим лицам заказы наложенным платежом не отправляются!

Если Вы проживаете в городе, в котором действует курьерская служба “Болеро”, рекомендую Вам пользоваться именно ею, как более дешевой и быстрой.

НАЛИЧНЫЕ ПРИ ПОЛУЧЕНИИ

Вы оплачиваете заказ курьеру при получении. Все необходимые финансовые документы курьер вручит Вам вместе с заказом.

WEBMONEY

Вы также можете осуществлять безналичные платежи через систему Webmoney Transfer. Для оплаты вашего заказа через систему WebMoney Transfer Вам необходимо после выбора соответствующего способа оплаты и оформления заказа, произвести перевод на кошелек R374736777856. Это кошелек магазина в системе WebMoney Transfer. Для перевода в Вашем WebMoney Keeper необходимо щелкнуть правой кнопкой на кошельке, с которого Вы хотите произвести перевод, а затем выбрать из меню "Передать деньги в кошелек WebMoney". В появившемся диалоговом окне необходимо указать сумму перевода и наш кошелек, а в примечании обязательно указать: Оплата заказа "номер заказа". Перевод необходимо делать без протекции сделки. После того, как Вы произведете перевод, сотрудники в течение 30 минут примем ваш заказ к исполнению.

Налоги

Цены на товары, указанные в Интернет- магазине, уже включают все налоги. В выписываемых платежных документах налоги отдельной строкой не выделяются.

Финансовые документы

Вместе с заказом отправляются необходимые финансовые документы: при безналичных расчетах - оригинал счета и накладная при наличных расчетах - накладная и кассовый чек (или приходный кассовый ордер).

СПОСОБЫ ДОСТАВКИ ТОВАРА

ОГРАНИЧЕНИЯ ДОСТАВКИ

Некоторые товары имеют ограничения по региону продаж и не могут быть доставлены в ряд стран. Это связано с лицензионными соглашениями правообладателей.

ВРЕМЯ ДОСТАВКИ

На странице описания каждого товара Вы найдете сроки формирования заказа, это обычно 1-2 дня, однако редко заказываемые товары могут формироваться в течение большего срока. Время доставки Вашего заказа рассчитывается таким образом:

Общее время доставки = время формирования заказа + время доставки почтовой службой.

Общее время указывается в рабочих днях. Обратим Ваше внимание, что для товаров, которые готовятся к выпуску, срок формирования заказа указывается производителем приблизительно. Это может быть даже несколько месяцев! Время доставки почтовой службой зависит от выбранной Вами службы доставки.

ДОСТАВКА ПОЧТОЙ ПО КАЗАХСТАНУ

Как правило, время доставки составляет 2-3 недели, однако, к сожалению, почта может доставлять заказ и гораздо дольше. Стоимость доставки зависит от веса заказа и действующих почтовых тарифов, дополнительных наценок «Болеро» в цену доставки не включает.

КУРЬЕРСКАЯ ДОСТАВКА

Более удобным и быстрым способом доставки является доставка курьером. «Болеро» использует курьерскую службу компании "Скороход", которая осуществляет доставку товаров более чем в 20 городах Казахстана. Время доставки курьером составляет от 1-го (Алматы) до 10-ти

(Петропавловск) дней, большая часть из которых - это доставка до города, которая осуществляется поездом или самолетом. Стоимость доставки зависит от веса заказа и рассчитывается следующим образом: если вес не превышает 2 кг.

Если вес выше 2 кг, то за каждый последующий кг прибавляется 1000 тг. В некоторых городах Вы можете получить свой заказ в офисе представительства курьерской службы и сэкономить на стоимости доставки курьером по Вашему городу.

ДОСТАВКА НЕПОЛНОГО ЗАКАЗА

Иногда во время формирования заказа, один из товаров заканчивается на складе и поэтому заказ не может быть полностью сформирован. В этом случае, менеджер компании «Болеро» свяжется с Вами и предложит доставить имеющиеся товары, а недостающие после их появления. В этом случае стоимость доставки обоих частей будет равна стоимости доставки полного заказа. Вы можете также отказаться от доставки, как всего заказа, так и недостающей части.

ВОЗВРАТ ЗАКАЗА

Вы можете отказаться от получения и оплаты заказа в случае потери его товарного вида за время транспортировки. В этом случае, если Вы уже оплатили его, магазин вернёт Вам полную сумму, которая была зачислена на счет интернет-магазина.

Интернет-магазин http://www.pcstr.kz

Описание

Адрес в интернет: http://www.pcstr.kz

E-mail: shop@line.kz

МАГАЗИН ЗАНИМАЕТСЯ ПРОДАЖЕЙ ТОВАРОВ СЛЕДУЮЩИХ КАТЕГОРИЙ:

- компьютерных комплектующих
- мониторов

- аксессуаров

ТЕХНОЛОГИЯ ПОКУПКИ ТОВАРА

Как в общем перечне товаров у каждой позиции, так и на странице подробного описания товара есть кнопка "в корзину". Нажав ее, Вы добавляете понравившийся товар в свою корзину. Корзина - просто список товаров, выбранный Вами в магазине. Это почти такая же корзина, в руках с которой ходят посетители большого супермаркета - только виртуальная.

При этом сразу же рассчитывается общая стоимость выбранных товаров и общий вес заказа (важно для пересылки почтой и доставки курьерскими службами). Общее число товаров высвечивается рядом в левом с боку страницы, рядом с логотипом корзины.

Находясь на странице корзины, Вы можете начать процедуру оформления заказа, нажав соответствующую кнопку, или вернуться и продолжить выбирать еще товары. Вы можете удалить любую позицию или даже очистить всю корзину. Можно изменить количество единиц любого из товаров, находящихся в корзине. При этом нажмите на ссылку "пересчитать" для корректировки стоимости и веса.

Для заказа товара вам необходимо пройти несколько этапов:

Идентификация. На данном этапе вы либо регистрируетесь(если вы не зарегистрированы), либо идентифицируетесь под своим именем.

Выбор доставки и оплаты. О способах оплаты и доставки подробнее описано ниже.

Оплата. Здесь вы подтверждаете покупку товара.

Способы оплаты товара

ОПЛАТА КУРЬЕРУ НАЛИЧНЫМИ

Вы просто вручаете деньги курьеру и расписываетесь в получении заказа. Это самый быстрый и удобный для Вас вид оплаты. К сожалению этот способ возможен только, если Вы проживаете в Астане.

ПРЕДОПЛАТА ПО БЕЗНАЛИЧНОМУ РАСЧЕТУ

Предоплата по безналичному расчету для организаций, имеющих расчетный счет в банке возможна, если они находятся в Караганде. При оформлении заказа укажите название Вашей организации. При этом Вы сможете заполнить или распечатать счет.

НАЛОЖЕННЫЙ ПЛАТЕЖ

Вы можете оплатить заказ при получении его на почте. При таком способе оплаты Вам придется заплатить почте дополнительно 15-20% от стоимости заказа. Наложенный платеж является для Вас наиболее затратным. Применим только для жителей Казахстана и только, если стоимость заказа не превышает 10$.

ОПЛАТА ПЛАСТИКОВОЙ КАРТОЙ

Вы можете оплатить заказ пластиковой картой в режиме on-line. Магазин принимает к оплате пластиковые карты платежных систем Visa, Eurocard/MasterCard и STB.

(Процессинг платежей по картам STB осуществляет Банк "Первое ОВК)

После оформления заказа Вы будете перенаправлены на сервер ASSIST (который имеет все необходимые лицензии и сертификаты на проведение платежей через Интернет) для введения данных Вашей кредитной карты. Передача данных происходит в защищенном режиме, при котором полностью исключена возможность перехвата информации о вашей карте. Для защиты информации от несанкционированного доступа на этапе передачи от клиента на сервер системы ASSIST используется протокол SSL 3.0, сертификат сервера (128 bit) выдан компанией Verisign - признанным центром выдачи цифровых сертификатов. Вы можете проверить подлинность сертификата сервера. Обратите внимание, мы не получаем и никогда не

узнаем номер Вашей кредитной карты. Мы его не используем и не храним. Обработка полученных конфиденциальных данных клиента производится в процессинговом центре. Таким образом, никто, даже продавец не может получить персональные и банковские данные клиента, включая информацию о его покупках, сделанных в других магазинах.

Для защиты информации от несанкционированного доступа на всех этапах передачи данных у покупателя есть возможность использовать протокол SET 1.0

где конфиденциальность данных полностью гарантирована.

Обратите внимание! Выполнение заказа возможно только при соблюдении следующих условий:

1. Ф.И.О. заказчика должно совпадать с Ф.И.О владельца карты.

2. При получении заказа необходимо предъявить документ удостоверяющий личность. Наш курьер обязан вписать в бланк заказа название и данные этого документа. При получении заказа в некоторых случаях вас могут попросить предъявить кредитную карту.

ДОСТАВКА КУРЬЕРОМ ПО ГОРОДУ

Доставка курьером по городу доступна только для жителей Астаны. При этом важно указать номер телефона, по которому диспетчер сможет с Вами оперативно связаться и договориться об удобном для Вас время доставки. Вы получите заказанные товары, не выходя из дома. К сожалению, магазин не может гарантировать оперативную доставку заказа, если по указанному телефону с Вами не удастся связаться.

Доставка заказа курьером осуществляется в течении 1-5 дней с момента появления его на складе, т.е. если срок поставки товара указан, например, 10

дней - то не следует ожидать его быстрее чем через заявленные десять дней плюс 1-5 дней.

Если у Вас проблемы с телефоном, обязательно указывайте при регистрации свой e-mail - этот способ диспетчеры также часто используют для связи с клиентами.

Заполняя форму заказа, Вы можете указать удобное для Вас время доставки - в первой (до 15.00), либо во второй половине дня (после 15.00).

Доставка заказа, состоящего из товаров с разными сроками поставки на склад.

Если Вы выбрали несколько товаров с разными сроками поставки на склад, то при выборе вида доставки Вам будет предложено два различных способа: ждать полной комплектации заказа или получать его частями по мере поступления. В первом случае заказ будет послан только после появления на складе товара, имеющего самый длительный срок поставки. Таким образом, если Вы заказали товары со сроком поставки 1-5 дней и 1 месяц, заказ Вам будет послан лишь через 1 месяц.

Если Вы предпочли второй вариант, то при аналогичном заказе, первый товар Вам будет послан сразу, а второй лишь через месяц. При этом следует учесть, что при этом Вы платите за доставку дважды (или по количеству различных сроков поставок выбранных Вами товаров).

2. Проектная часть

2.1 Требования и средства проектирования web-документов

Огромное количество источников по web-дизайну предлагают такое же количество правил и советов по созданию и оформлению web-страниц.

Мы же постараемся выделить основные моменты, вокруг которых, собственно и сконцентрировано внимание этих бесчисленных советов. Это использование графики, цветовое решение, скорость загрузки страницы, использование технологий, содержание, навигация по сайту.

- 5-7 пунктов меню - это предел для хорошего восприятия содержащейся в меню информации.
- что оптимальное использование не более 30-40 кбайт графики на страницу.
- что дизайн первой страницы хоть немного, но должен отличатся от всех остальных страниц.
- что бегущие строки лучше не использовать или использовать, но крайне редко. Потому что от них только глаза болят, но если в этом есть особая необходимость, то не следует нагружать строку большим количеством информации.
- что в web-страницах используются только форматы JPG, JPEG и GIF.
- чтобы пользователь находящийся в середине сервера - мог попасть в любую точку вашего сайта или странички.
- что необходимо разбивать большое количество информации.
- что именовать, созданные странички следует на английском языке.
- что необходимо избавлятся от горизонтальных прокруток экрана.
- что цвет ссылки не должен быть схожим с фоном страницы. Иногда многие в этом делают ошибку. Если, например, фон страницы темно-

красный, а цвет ссылки синий, то стоит посетителю просмотреть ссылку как она становится темно-красной и исчезает.

- что лучше всего делать такие страницы, которые можно изменять. Делайте страницы, которые доступны всем читателям вне зависимости от того, каким браузером, платформой или экраном они решили или вынуждены пользоваться.
- что, на первой странице сайта должна находиться краткая информация о содержании сайта, представленных материалах, авторах сайта. Посетитель, случайно попавший на сайт, не должен гадать, что он может на нем найти.
- крайне желательно, чтобы графика на сайте имела лишь вспомогательное значение (за исключением, конечно, сайтов, посвященных изобразительному искусству и другим подобным тематикам). Многие пользователи отключают отображение графики в своих броузерах для ускорения путешествий по Интернету.
- Не используйте на сайте текст, набранный заглавными буквами.
- что более всего нежелательно - это использование при создании сайта фреймов! (о том, почему их лучше не использовать будет указано ниже).
- Размер каждой страницы сайта не должен превышать предел в 80-100 килобайт вместе с графикой (оптимально - 40-50), за исключением страниц с большим количеством информации - электронных книг, например. В любом случае при большом размере того или иного файла об этом должно быть сказано около ссылки на него.

Web-страницы могут существовать в любом формате, но в качестве стандарта принят Hyper Text Markup Language - язык разметки гипертекстов, предназначенный для создания форматированного текста, насыщенного изображениями, звуком, анимацией, видеоклипами и гипертекстовыми ссылками на другие документы, разбросанные как по всему Web-

пространству, так и находящиеся на этом же сервере или являющиеся составной частью этого же Веб-проекта.

При помощи языка HTML можно создавать Web-страницы в обычном блокноте или Word-е. Но текстовые редакторы, возможно, использовать только тем, кто является профессионалом и очень хорошо знает язык HTML. Также можно работать на Web и без знания языка HTML, поскольку тексты HTML могут создаваться разными специальными редакторами и конвертерами. Писать же непосредственно на HTML достаточно нетрудно. Возможно, это даже легче, чем изучать HTML-редактор или конвертер, которые часто ограничены в своих возможностях, содержат ошибки или проводят плохой HTML код, который не работает на разных платформах.

Все в языке осуществляется при помощи тегов, т.е. команд, которые заключены в скобки такого вида: “< “ и “>”. Например, <title>Энциклопедия WEB 2000 Ver. 1.02</title>.

Язык HTML существует в нескольких вариантах и продолжает развиваться, но конструкции HTML скорее всего будут использоваться и в дальнейшем. Изучая HTML и познавая его глубже, создавая документ в начале изучения HTML и расширяя его насколько это возможно, мы имеем возможность создавать Web-страницы, которые могут быть просмотрены многими браузерами Web, как сейчас, так и в будущем. Это не исключает возможности использования других методов, например, метод расширенных возможностей, который предоставляется Netscape Navigator, Internet Explorer или некоторыми другими программами.

Работа по HTML - это способ усвоить особенности создания документов в стандартизированном языке, используя расширения, только если это действительно необходимо.

HTML был ратифицирован World Wide Web Consortium. Он поддерживается несколькими широко распространенными броузерами, и, возможно, станет основанием почти всего программного обеспечения, которое имеет отношение к Web.

2.2 Выбор и описание языка программирования при разработке web-страниц

PHP (Hypertext Preprocessor - Препроцессор Гипертекста)– это широко используемый язык сценариев общего назначения с открытым исходным кодом.

PHP - язык программирования, специально разработанный для написания web-приложений (скриптов, сценариев), исполняющихся на Web-сервере. Синтаксис языка во многом основывается на синтаксисе C, Java и Perl. Он очень похож на C и на Perl, поэтому для профессионального программиста не составит труда его изучить. С другой стороны, язык PHP проще, чем C, и его может освоить веб-мастер, не знающий пока других языков программирования.

Огромным плюсом PHP, в отличие от, например, JavaScript, является то, что PHP-скрипты выполняются на стороне сервера. PHP не зависит от скорости компьютера пользователя или его браузера, он полностью работает на сервере. Пользователь даже может не знать, получает ли он обычный HTML-файл или результат выполнения скрипта.

Сценарии на языке PHP могут исполняться на сервере в виде отдельных файлов, а могут интегрироваться в html страницы.

PHP способен генерировать и преобразовывать не только HTML документы, но и изображения разных форматов - JPEG, GIF, PNG, файлы PDF и FLASH. PHP способен формировать данные в любом текстовом формате, включая XHTML и XML.

PHP - кроссплатформенная технология. Дистрибутив PHP доступен для большинства операционных систем, включая Linux, многие модификации Unix, Microsoft Windows, Mac OS и многих других. PHP поддерживается на большинстве вебсерверов, таких, как Apache, Microsoft Internet Information Server (IIS), Microsoft Personal Web Server и других.

Для большинства серверов PHP поставляется в 2-х вариантах - в качестве модуля и в качестве CGI препроцессора.

PHP поддерживает работу с ODBC и большое количество баз данных: MySQL, MSQL, Oracle, PostgreSQL, SQLite и др.

Язык программирования PHP, особенно в связке с популярнейшей базой данных MySQL - оптимальный вариант для создания интернет-сайтов различной сложности.

Язык PHP постоянно совершенствуется, и ему наверняка обеспечено долгое доминирование в области языков web -программирования.

2.3 Использование базы данных mysql при проектировании

MYSQL является относительно небольшой и быстрой реляционной СУБД, основанной на традициях Hughes Technologies Mini SQL (mSQL).

Чем хорош MySQL?

Перечислю основные приятные стороны пакета MySQL.

- Многопоточность. Поддержка нескольких одновременных запросов.
- Оптимизация связей с присоединением многих данных за один проход.
- Записи фиксированной и переменной длины.
- ODBC драйвер в комплекте с исходником
- Гибкая система привилегий и паролей.
- До 16 ключей в таблице. Каждый ключ может иметь до 15 полей.
- Поддержка ключевых полей и специальных полей в операторе CREATE.
- Поддержка чисел длинной от 1 до 4 байт (ints, float, double, fixed), строк переменной длины и меток времени.
- Интерфейс с языками C и perl.
- Основанная на потоках, быстрая система памяти.

- Утилита проверки и ремонта таблицы (isamchk).
- Все данные хранятся в формате ISO8859_1.
- Все операции работы со строками не обращают внимания на регистр символов в обрабатываемых строках.
- Псевдонимы применимы как к таблицам, так и к отдельным колонкам в таблице.
- Все поля имеют значение по умолчанию. INSERT можно использовать на любом подмножестве полей.
- Легкость управления таблицей, включая добавление и удаление ключей и полей.

Что такое SQL?

SQL - это сокращение от Structured Query Language (структурированный язык запросов). SQL создан для работы с реляционными базами данных. Он позволяет пользователям взаимодействовать с базами данных (просматривать, искать, добавлять и управлять данными). MySQL соответствует спецификации ANSI 92 SQL.

Установка MySQL.

Процесс установки довольно прост как для пользователей, ранее работавших с подобными программами, так и для тех, кто устанавливает и конфигурирует сервер впервые. Для последних следует сделать небольшие пояснения. Только что скачанный zip архив необходимо разархивировать в отдельную папку. Затем следует запустить файл setup.exe. Появится стандартное окно инсталляции. Здесь вы должны внимательно читать все сообщения программы инсталляции и аккуратно жать кнопку “далее”, до тех пор, пока не придется выбирать директорию, куда следует установить сам MySQL сервер и где будут находиться базы. Здесь следует оговориться, что если вы измените папку, которая стояла по умолчанию, то вам придется впоследствии настраивать отдельные переменные в файле конфигурации сервера, что не является препятствием для “профи”, но серьезная проблема для начинающих пользователей. В отношении домашнего сервера, нет

смысла изменять путь папки. Разве только для удобства можно поставить поближе к Apache. На реальном сервера пути установки выглядят примерно так: /usr/htdocs — основная папка с документами, /usr/php — здесь живет php, /usr/bin/perl — интерпретатор perl, и, наконец, /usr/web/databases/mysql. Все эти пути условные, и в каждом конкретном случае могут отличаться. У меня на компьютере структура папок организована несколько иначе: основные документы /web/htdocs/ (C:\web\htdocs), php — /web/php/ (C:\web\php), perl — /web/bin/perl(C:\web\bin\perl), mysql — /web/mysql/ (C:\web\mysql). Но для простоты и удобства настройки все же поставим MySQL в папку по умолчанию, т.е. c:\mysql. Это позволит нам в дальнейшем сразу начать работу, без каких-либо дополнительных настроек. Больше никаких существенных изменений в процессе установки нам делать не нужно, просто жмем везде кнопку “далее” и ждем, пока программа установки сделает свое “черное” дело:). После установки мы сможем наблюдать примерно следующую иерархию папок:

C:\mysql — корневая директория.

|_bench — контрольные замеры и тест “crash-me”.

|_bin — клиентские программы и сценарии.

|_data — именно здесь будут находиться сами базы данных.

|_docs — различная информация: копирайты, лицензии, краткий мануал и т.д.

|_examples — несколько примеров использования базы (см. мануал).

|_include — файлы заголовков.

|_lib — различные библиотеки.

|_scripts — несколько Perl-скриптов. Подробнее о них в readme’шке.

|_share — файлы сообщений об ошибках.

Все вроде бы все установили. Сразу после установки у нас есть всего один суперпользователь root, обладающий всеми правами администратора, и пароль в виде пустой строки. В принципе, на этом этапе у нас достаточно

технических возможностей для начала наших тестов и разработки серьезных приложений.

Обычно в приложениях, использующих MySQL, присутствуют три стандартные переменные, которые отвечают за доступ к самой базе данных:

$userName = "root"; -имя пользователя, которому разрешен доступ к базе.

$password = ""; пароль, по умолчанию он отсутствует.

$hostName = "localhost"; имя хоста, на котором “живет” база данных.

Интерфейс с языками программирования

Наиболее простой способ работы с MySQL сводится к использованию программы MySQL. Это клиентская часть СУБД MySQL. Можно выполнять команды SQL непосредственно из командной строки системы unix или из интерактивного режима MySQL. Подробнее о клиентских программах.

СУБД MySQL имеет библиотеку C API. Ее можно использовать для запросов к базе данных, вставки данных, создания таблиц и т.п. C API поддерживает все функции MySQL. Подробности в главе "Интерфейс для C (C API)".

Язык perl поддерживается сразу двумя способами:

- Портирован интерфейс с perl из mini-SQL, разработанный Андреасом Коенигом (Andreas Koenig a.koenig@mind.de).
- Есть модуль perl DBD

Подробнее этот вопрос рассмотрен в главе "Интерфейс с perl (MySQL perl API)".

Также доступен 32-битный ODBC драйвер для MySQL. Он позволяет запрашивать и получать данные из других источников с поддержкой ODBC. С подробностями можно ознакомиться на домашней страничке MySQL (увы, только на английском языке).

Работа с базами данных

В системе PHP работа с БД осуществляется в основном путем работы с различными SQL-серверами, причем SQL-сервер в любом случае

рассматривается как удаленный, то есть создается сетевое соединение. Благодаря этому возможно открывать из одного скрипта либо несколько пользовательских сессий, либо работать с различными SQL-серверами. После установки соединения с сервером, выбирается рабочая база данных, после чего можно отправлять и обратывать запросы (так как SQL является клиент-серверной архитектурой, любая работа с данными осуществляется с помощью запросов к SQL-серверу на получение или изменение данных). При выполнении запроса создается некий объект, в котором хранится результат выполнения запроса, после чего можно получать отдельные ряды, путем выполнения специальных функций.

Перед началом эксплуатации электронного магазина требуется:

1. Установить Web Сервер Apache версии не ниже 1.3
2. Установить интерпретатор PHP версии не ниже 4.0
3. Установить сервер БД MYSQL версии не ниже 4.1
4. Установить оболочку управления СУБД phpMyAdmin версии не ниже 2.6

Следующим этапом создается база данных. Для этого требуется произвести следующие действия:

1. Первым делом нам нужно создать отдельную папку для нашего сайта на локальном сервере.
2. Для этого Вам нужно зайти в директорию, в которой хранятся все Ваши сайты.
3. Путь этой директории такой: Desktop\Istore.
4. Далее нам нужно запустить локальный сервер (или перезапустить, если он уже был у Вас запущен). Для этого запускаем WampServer.

Рисунок 1-Запуск локального сервера

1. Папка «adminka »Управление настройками сайта происходит через панель управления, доступ в которую имеет только администратор сайта.

Вход в панель управление осуществляется по ссылке «управление магазином»

В панели управления вы можете:

- Добавлять и удалять категории и товары
- Управлять основными настройками модулей
- Следить за заказами
- Управлять формой оплаты

2. Папка «classes»

Содержит Классы и объекты в PHP

Если класс можно рассматривать как тип данных, то объект — как переменную (по аналогии). Скрипт может одновременно работать с несколькими объектами одного класса, как с несколькими переменными.

Внутри объекта данные и код (члены класса) могут быть либо открыты, либо нет. Открытые данные и члены класса являются доступными для других частей программы, которые не являются частью объекта. А вот закрытые данные и члены класса доступны только внутри этого объекта.

Описание классов в PHP начинаются служебным словом **class**:

class Имя_класса {

// описание членов класса - свойств и методов для их обработки

Для объявления объекта необходимо использовать оператор **new**:

Объект = new Имя_класса;

Данные описываются с помощью служебного слова **var**. Метод описывается так же, как и обыкновенная пользовательская функция. Методу также можно передавать параметры.

Подведем промежуточные итоги: объявление класса должно начинаться с ключевого слова **class** (подобно тому, как объявление функции начинается с ключевого слова **function**). Каждому объявлению свойства, содержащегося в классе, должно предшествовать ключевое слово **var.** Свойства могут относиться к любому типу данных, поддерживаемых в PHP, их можно рассматривать как переменные с небольшими различиями. После объявлений свойств следуют объявления методов, очень похожие на типичные объявления пользовательских функций.

По общепринятым правилам имена классов ООП начинаются с прописной буквы, а все слова в именах методов, кроме первого, начинаются с прописных букв (первое слово начинается со строчной буквы). Разумеется, вы можете использовать любые обозначения, которые сочтете удобными; главное — выберите стандарт и придерживайтесь его.

Пример класса на PHP:

```
<?php
// Создаем новый класс Coor:
class Coor {
// данные (свойства):
var $name;
var $addr;
// методы:
 function Name() {
 echo "<h3>John</h3>";
 }

}
// Создаем объект класса Coor:
```

```
$object = new Coor;
?>
```

Доступ к класам и объектам в PHP

Мы рассмотрели, каким образом описываются классы и создаются объекты. Теперь нам необходимо получить доступ к членам класса, для этого в PHP предназначен оператор ->. Приведем пример:

```
<?php
// Создаем новый класс Coor:
class Coor {
// данные (свойства):
var $name;

// методы:
 function Getname() {
 echo "<h3>John</h3>";
 }
}
// Создаем объект класса Coor:
$object = new Coor;
// Получаем доступ к членам класса:
$object->name = "Alex";
echo $object->name;
// Выводит 'Alex'
// А теперь получим доступ к методу класса (фактически, к функции вн
утри класса):
$object->Getname();
// Выводит 'John' заглавными буквами
?>
```

Чтобы получить доступ к членам класса внутри класса, необходимо использовать указатель **$this**, которы всегда относится к текущему объекту. Модифицированный метод **Getname()**:

```
function Getname() {
echo $this->name;
}
```

Таким же образом, можно написать метод **Setname()**:

```
function Setname($name) {
$this->name = $name;
}
```

Теперь для изменения имени можно использовать метод **Setname()**:

```
$object->Setname("Peter");
$object->Getname();
```

А вот и полный листинг кода:

```
<?php
// Создаем новый класс Coor:
class Coor {
// данные (свойства):
var $name;

// методы:
 function Getname() {
 echo $this->name;
 }

 function Setname($name) {
 $this->name = $name;
 }

}
```

```
// Создаем объект класса Coor:
$object = new Coor;
// Теперь для изменения имени используем метод Setname():
$object->Setname("Nick");
// А для доступа, как и прежде, Getname():
$object->Getname();
// Сценарий выводит 'Nick'
?>
```

Указатель **$this** можно также использовать для доступа к методам, а не только для доступа к данным:

```
function Setname($name) {
$this->name = $name;
$this->Getname();
}
```

Конструкторы

Довольно часто при создании объекта требуется задать значения некоторых свойств. К счастью, разработчики технологии ООП учли это обстоятельство и реализовали его в концепции конструкторов. Конструктор представляет собой метод, который задает значения некоторых свойств (а также может вызывать другие методы). Конструкторы вызываются автоматически при создании новых объектов. Чтобы это стало возможным, имя метода-конструктора должно совпадать с именем класса, в котором он содержится. Пример конструктора:

```
<?
class Webpage {
var $bgcolor;
 function Webpage($color) {
 $this->bgcolor = $color;
 }
```

```
}
// Вызвать конструктор класса Webpage
$page = new Webpage("brown");
?>
```

Раньше создание объекта и инициализация свойств выполнялись раздельно. Конструкторы позволяют выполнить эти действия за один этап.

Интересная подробность: в зависимости от количества передаваемых параметров могут вызываться разные конструкторы. В рассмотренном примере объекты класса Webpage могут создаваться двумя способами. Во-первых, вы можете вызвать конструктор, который просто создает объект, но не инициализирует его свойства:

```
$page = new Webpage;
```

Во-вторых, объект можно создать при помощи конструктора, определенного в классе, — в этом случае вы создаете объект класса Webpage и присваиваете значение его свойству bgcolor:

```
$page = new Webpage("brown");
```

Деструкторы

Эта функция уничтожает содержимое переменной и возвращает занимаемые ею ресурсы системе. С объектами **unset()** работает так же, как и с переменными. Допустим, вы работаете с объектом $Webpage. После завершения работы с этим конкретным объектом вызывается функция:

```
unset($Webpage);
```

Эта команда удаляет из памяти все содержимое $Webpage. Действуя в духе инкапсуляции, можно поместить вызов **unset()** в метод с именем destroy() и затем вызвать его:

```
$Website->destroy();
```

Необходимость в вызове деструкторов возникает лишь при работе с объектами, использующими большой объем ресурсов, поскольку все переменные и объекты автоматически уничтожаются по завершении сценария.

Инициализация объектов

Иногда возникает необходимость выполнить инициализацию объекта - присвоить его свойствам первоначальные значения. Предположим, имя класса Coor и он содержит два свойства:имя человека и город его проживания. Можно написать метод (функцию), который будет выполнять инициализацию объекта, например **Init()**:

```
<?php
// Создаем новый класс Coor:
class Coor {
// данные (свойства):
var $name;
var $city;
// Инициализирующий метод:
 function Init($name) {
 $this->name = $name;
 $this->city = "London";
 }
}
// Создаем объект класса Coor:
$object = new Coor;
// Для инициализации объекта сразу вызываем метод:
$object->Init();
?>
```

Главное не забыть вызвать функцию сразу после создания объекта, либо вызвать какой-нибудь метод между созданием (оператор **new**) объекта и его инициализацией (вызовом **Init**).

Для того, чтобы PHP знал, что определенный метод нужно вызывать автоматически при создании объекта, ему нужно дать имя такое же, как и у класса (**Coor**):

```
function Coor ($name)
```

```
$this->name = $name;
$this->city = "London";
}
```

Метод, инициализирующий объект, называется конструктором. Однако, PHP не имеет деструкторов, поскольку ресурсы освобождаются автоматически при завершении работы скриптов.

Обращение к элементам классов

Обращение к элементам классов осуществляется с помощью оператора **::** "двойное двоеточие". Используя "двойное двоеточие", можно обращаться к методам классов.

При обращении к методам классов, программист должен использовать имена этих классов.

```
<?php
class A {
 function example() {
 echo "Это первоначальная функция A::example().<br>";
 }
}
class B extends A {
 function example() {
 echo "Это переопределенная функция B::example().<br>";
 A::example();
 }
}
// Не нужно создавать объект класса A.
// Выводит следующее:
// Это первоначальная функция A::example().
A::example();

// Создаем объект класса B.
```

```
$b = new B;
// Выводит следующее:
// Это переопределенная функция B::example().
// Это первоначальная функция A::example().
$b->example();
?>
```

В PHP5, используя эту лексему, программист может обращаться к константам, статическим или перегруженным свойствам или методам класса.

3 Папка «config»содержит конфигурационные настройуи сайта

4.Папка «css » В папке css содержатся файлы стилей шаблона. Здесь вы можете создавать сколько угодно файлов стилей css (если это необходимо). Конкретно в рассматриваемом шаблоне они называются:admin.css,login.css, jquery.autocomplete.css

5. папка «mails» автоматически отправляет сообщение message получателю to. Можно специфицировать несколько получателей, разделив запятой адреса в to. С помощью этой функции можно высылать Email с присоединением/attachment и содержимое специальных типов.пример: account.html

Здравствуйте, {firstname} {lastname},

Ваши учетные данные для авторизации:

E-mail: {email}

Пароль: {passwd}

{shop_url}

Этап 2: «База данных».

На этом этапе нам необходимо настроить соединение с базой данных. Но для начала ее еще нужно создать.

Поэтому:

Шаг 1. Заходим в phpmyadmin (http://localhost/tools/phpmyadmin). Это специальный инструмент, который позволит нам работать с базами данных

на локальном сервере (на реальном сервере – хостинге, будет точно такой же инструмент).

Шаг 2. Вводим имя базы данных (чтобы не путаться в дальнейшем, я ввел имя БД – «Storedata») и нажимаем по кнопке «Создать».

После создания БД отобразится окно создания таблиц в текущей БД. В этом окне следует ввести наименование таблицы в поле «Имя» и ввести количество полей в таблице в поле «Поля», затем нажать на кнопку «Пошел».

5. Далее в открывшемся окне задаются названия полей, типы полей, атрибуты длинны полей, являются ли поля ключевыми. После ввода требуемых данных нажимается кнопка «Пошел».

6. Пункты 4 и 5 повторяются до тех пор, пока все таблицы не будут созданы.

Пример создания БД:

Функция

mysql_create_db(имя_БД, [идентификатор_подключения]);

создать новую базу данных на сервере связанном с определенным идентификатором связи.

Подключение и выбор базы данных

```
<html>
<head>
<title<Подключение и выбор базы данных</title>
</head>
<body>
<?
$user = "myname";
$pass = "password";
$db = "sample";
$query = "CREATE TABLE ...";
$link = mysql_connect("localhost",$user,$pass);
if (!link) die("Не могу соединиться с MySQL");
```

```
print("Сервер БД найден <br>");
mysql_select_db($db) or die("Не могу открыть $db".mysql_error());
print("Выбрана база данных "$db);
...
$result = mysql_query($query, $link);
...
mysql_close($link);
?>
</body>
</html>
```

Так выглядит диаграмма классов нашего объекта

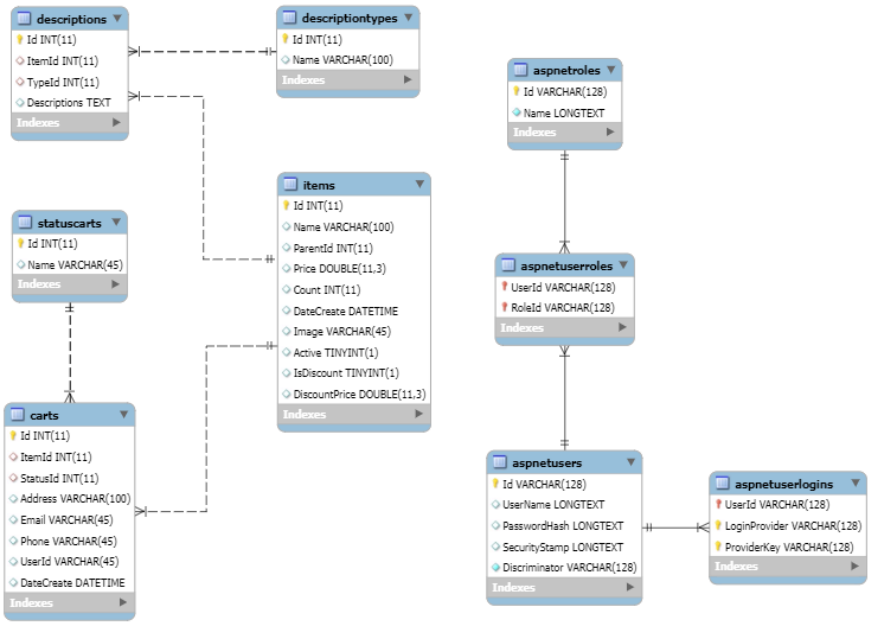

Рисунок 2-Диаграмма классов

Цель инфологического моделирования – обеспечение наиболее естественных для человека способов сбора и представления той информации, которую предполагается хранить в создаваемой базе данных. Поэтому инфологическую модель данных пытаются строить по аналогии с

естественным языком (последний не может быть использован в чистом виде из-за сложности компьютерной обработки текстов и неоднозначности любого естественного языка). Основными конструктивными элементами инфологических моделей являются сущности, связи между ними и их свойства (атрибуты). Информационно-логическая модель БД представлена графически на рисунке.

Рисунок 4 - Физическая модель БД

2.4 Архитектура проектирования электронного магазина.

Архитектура сайта - систематизация информации и навигации по ней с целью помочь посетителям более успешно находить нужные им данные. Хорошо продуманная грамотная архитектура сайта гарантирует, что пользователи потратят меньше времени на поиск нужной информации.

Разработка архитектуры сайта должна вестись с учётом наиболее важной информации с точки зрения продвижения товаров/услуг на интернет-рынке. В процессе создания структуры нового сайта, либо оптимизации структуры уже существующего, необходимо концентрировать внимание потребителей именно на этой информации и управлять посещаемостью сайта потенциальными клиентами именно в наиболее важных разделах сайта в соответствии с позиционированием на рынке, продвигаемых товаров/услуг.

Грамотное распределение приоритетов между разделами и страницами сайта, сделает их основными точками входа на сайт, что позволит потенциальному потребителю быстро найти необходимую ему информацию об искомых товарах/услугах и повысит успешность бизнеса в интернете.[25, с.321]

Архитектура интернет – магазина должна быть проста и интуитивно удобна. И состоит из Клиентской части, Программной части и Администрирования как показано на рисунке 5.

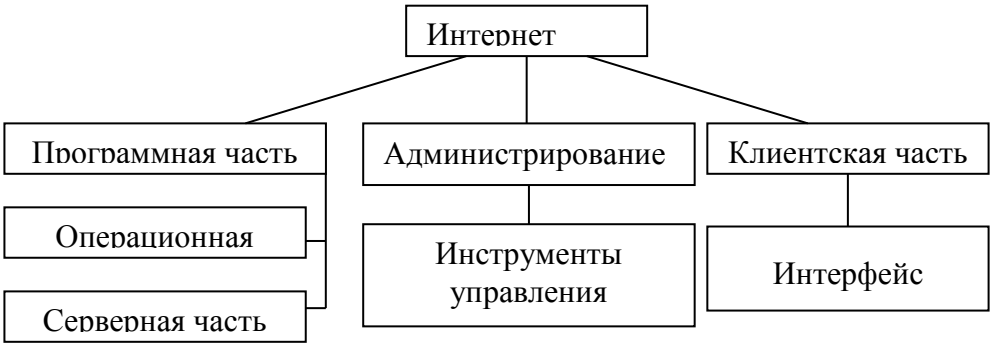

Рисунок 3 - Архитектура Интернет Магазина

Программная часть архитектуры интернет – магазина рассматривается как взаимосвязь операционной и серверной части.

В операционной части рассматривается среда разработки интернет магазина.

Серверная часть содержит в себе размещение интернет магазина на сайте провайдера, поддерживающие технологии, используемые при создании интернет – магазина. См Приложение В

Разработка операционной части.

Предположительно интернет магазин разрабатывается в среде php. Для ответа обоснования выбора было произведено сравнение PHP с другими языками программирования Web-приложений. Это его основные конкуренты — Perl, ASP.NET, ColdFusion и Java.

Введены закрытые (private) и открытые (protected) члены и методы класса, а также абстрактные классы и методы.

Введены интерфейсы, при этом класс может реализовывать произвольный список интерфейсов.

В PHP 5 также присутствуют расширенные возможности для клонирования объектов. Предназначенный для этого конструктор копирования позволяет не просто идентично клонировать объекты, а изменять при этом некоторые их свойства.

Введены постоянные члены и методы класса.

Одним из важнейших нововведений является наличие деструкторов.

Хорошей новостью является введение в PHP 5.0 статических членов класса. В PHP 4.0, если возникала необходимость в использовании статических членов, приходилось пользоваться искусственным приемом, заключающемся в комбинации глобальных переменных со статическими методами. При этом получались члены, которые действовали так же, как статические. Прием этот, мягко говоря, не очень хороший, поскольку привносил с собой все проблемы, связанные с использованием глобальных переменных. Теперь в PHP 5.0 статические методы можно объявлять явно при помощи ключевого слова static.

В PHP 5.0 введен механизм обработки исключений, отсутствующий в PHP 4.0.

Таким образом, в результате подобных нововведений в объектную модель PHP 5.0 стал полностью профессиональным языком программирования, что позволяет использовать его для решения задач любой степени сложности.

В серверной части архитектуры рассматривается работа интернет – магазина в сети Интернет, взаимодействие программного обеспечения магазина и сервисов, предоставляемых собственниками серверов, провайдеров. Так как после изготовления интернет-магазина необходимо будет его разместить в сети интернет и придется столкнуться с выбором мест размещения, иначе говоря выбрать хостинг.

Слово "хостинг" произошло от англ. "hosting". (host - главная машина, хозяин)

Сервисы, предлагающие свои услуги хостинга, дают возможность размещать информацию, интернет сайт на своём сервере. Таким образом, не нужно устанавливать свой собственный сервер в Интернет, что позволяет существенно сэкономить деньги. В зависимости от тарифного плана, предлагается определённый объём места на жёстких дисках сервера, e-mail, возможность работы с CGI, MSQ, и т.д. [29, с.65]

Любой человек может без особых затруднений использовать как платные, так и бесплатные услуги. Как известно, бесплатный сыр бывает только мышеловке и в любом случае придётся платить. В первом (платном) варианте нужно расплачиваться деньгами, а во втором - качеством связи, ограничениями на размер файлов и рекламой, которая будет вывешена в магазине.

Как правило, на платном хостинге - домен будет выглядеть так: hostname.com/yourname, а на бесплатном - предоставляется субдомен и адрес будет выглядеть следующим образом: yourname.hostname.com

В последствии постараемся разместить магазин на платном хостинге в интернете. Цены на именные домены «первого уровня» начинаются от 1-3 доллара в месяц, многое, конечно, зависит от провайдера предоставляющего хостинг.

В настоящее время многие сайты предлагают партнерские программы. Как учат учебники по маркетингу есть масса способов как найти потенциального клиента, как донести до него сведения о предлагаемых услугах. Самый простой способ – реклама в любом виде. Ролики по телевидению и радио, объявления в газетах, баннеры на сайтах, привлекающие массу внимания маркетинговые акции (например, конкурсы или лотереи, временные скидки), заказные статьи в изданиях, спам - это основной перечень наиболее часто используемых способов привлечения клиентов.

Но только реклама все-таки далеко не всегда приводит желаемому результату - заказу. Поэтому часто применяется еще один способ привлечения клиентов – наем агентов, работающих за проценты с продаж. В этом случае компания может не беспокоиться о том, что рекламный бюджет будет потрачен впустую, а также о том, что сотрудники работают недостаточно усердно. Агент получит столько, сколько заказов он обеспечит. Такая схема работы получила довольно широкое распространение и в Интернете. Значительное число хостинг-провайдеров, интернет-магазинов,

интернет-агентств, студий веб-дизайна и других компаний, предоставляющих веб-услуги либо ведущих поиск клиентов в Сети используют т.н. партнерские (агентские) или дилерские (реселлинговые) программы. [30,с. 83]

Особенно большое распространение в Казахстанском интернете партнерские/дилерские программы получили в сфере веб-хостинга, где и конкуренция в настоящее время значительно больше, чем, например, в электронной коммерции. Но и в других веб-услугах партнерские программы тоже постепенно получают все большее распространение. За рубежом, например, партнерские программы также получили очень широкое распространение и у туристических сайтов (особенно в сфере бронирования гостиниц; именно по партнерским программам различной сложности работают, например, такие крупные системы бронирования в Европе как SashaHotels.com или SunScale.com).

Достаточно бурное развитие партнерских программ и увеличение их количества привело к появлению специализированных интернет-ресурсов по партнерским программам. В первую очередь это форумы по реселингу и партнерским программам, а также узкоспециализированные сайты-каталоги с аналитическими обзорами и партнерскими ссылками на сайты конкурирующих компаний, где посетитель может сделать выбор среди множества предложений по определенной услуге. Но в тоже время сам сайт окупается не за счет размещения рекламы (баннеры и т.д.), а за счет участия в партнерских программах.

Рассмотрим типичную схему работы партнерских программ на примере сайтов по платному хостингу. Тем более, что именно в этой сфере в Казахстанском интернете в настоящее время действует наибольшее количество партнерских программ – это только более двух десятков программ первичных хостинг-провайдеров. Правда, стоит отметить, что партнерские схемы, используемые хостинг-провайдерами, довольно простые, в других сферах могут требоваться гораздо более организационно сложные

схемы партнерства. Но в то же время именно благодаря своей простоте партнерские программы в сфере хостинга получили массовое распространение как у самих провайдеров и так многочисленных владельцев сайтов различного размера и разной тематики.

Обычно при регистрации на сайте провайдера партнер получает уникальный идентификатор партнера, который обязательно включается в код специальной ссылки, которую партнер получает от провайдера для привлечения клиентов. Эти ссылки партнер и должен проставить на своих сайтах. При нажатии на такую ссылку на сайте партнера посетитель не просто перейдет на сайт провайдера, одновременно с этим скрипты на сайте провайдера по идентификатору в партнерской ссылке определят откуда пришел посетитель и в кукисы (coockies) на компьютере данного посетителя будет записан идентификатор партнера. Обычно срок действия таких кукисов составляет от одного месяца до полугода. Если даже спустя несколько дней посетитель вернется на сайт провайдера и заполнит форму заказа хостинга, то скрипты на сайте провайдера определят, от какого партнера пришел заказчик и при оплате клиентом заказа начисляет полагающиеся партнеру проценты за привлечение клиента.

Обычно помимо самих партнерских ссылок провайдеры предоставляют партнерам и уже готовые рекламные материалы для размещения на сайтах – баннеры различных форматов, текстовые блоки рекламы или специальные формы заказа хостинга, содержащие в себе партнерскую ссылку.

Некоторое неудобство использования партнерских ссылок в том, что посетители могут заметить, что переходят не по прямой ссылке на сервер провайдера, а по партнерской. Для партнера же во многих случаях наиболее оптимальным было бы остаться «в тени», чтобы заказчик хостинга вообще не знал, что кто-то получил за его привлечение определенные проценты. Хотя стоимость заказа, что по партнерской программе, что напрямую, для клиентов обычно не отличается, тем не менее, психологически российские пользователи не всегда положительно относятся к партнерам и часто

встречаются отдельные индивидуумы, которые специально удаляют кукисы на компьютере перед заказом хостинга. Поэтому в последнее время получают распространение партнерские программы без специальных партнерских ссылок, когда сервер провайдера определяет партнера не по наличию его идентификатора в адресе, а просто по адресу сайта, с которого пришел посетитель. Такая схема реализуется технически несколько сложнее, зато она обеспечивает абсолютную анонимность партнера (если, конечно, потенциальный заказчик услуг хостинга перед заказом специально не просмотрит на своем компьютере кукисы). Так, без партнерских ссылок в настоящее время работает, например, партнерская программа хостинг-провайдера WebRider.ru, специальную ссылку можно не использовать также и в партнерской программе ATLEX'a (www.atlex.ru). Но у такого подхода есть и свой минус – если партнерские ссылки можно обычно проставлять где угодно, то без них – только на тех сайтах, которые зарегистрированы в партнерской программе у самого провайдера (чтобы потом можно было идентифицировать переход посетителя с определенного сайта как от партнера).

Самый существенный момент – это те проценты, которые получают партнеры за привлечение клиентов. Единой «таксы» в казахстанском интернете нет. В зависимости от провайдера партнер может получать от 5 до 50% от суммы заказа, средняя величина поощрения в казахстанском интернете в настоящее время составляет около 20%. Чаще всего партнеры получают именно проценты с продаж, но отдельные хостинг-провайдеры практикуют и фиксированную величину дохода в 5-20 долларов США в зависимости от величины заказа (по такой схеме функционируют, например, партнерские программы хостеров 350mb.ru, Majordomo.ru или Mastak.ru, а партнеры «Агавы» (www.agava.ru) могут выбрать как работу за проценты, так и за фиксированную плату за привлечение каждого клиента).

Некоторые провайдеры применяют более сложные схемы исчисления доходов по партнерским программам – за счет введения партнерства разного

уровня. Процентные ставки для партнеров, привлекших разное количество клиентов, могут отличаться - в таком случае наиболее активные партнеры (привлекшие несколько десятков клиентов) работают по более выгодным условиям, чем их менее удачные коллеги. Например, у той же «Агавы» существует три уровня исчисления доходов в партнерской программе хостинга – 20% (до 5 клиентов), 25% (5-15 клиентов) и 30% (более привлеченных 15 клиентов). Другие провайдеры могут поощрять привлечение других партнеров и способствовать появлению реферальных сетей, в которых партнер «первого уровня» может получать некоторые проценты от прибыли, привлеченных им партнеров (т.е. партнеров «второго уровня») – такую схему практикуют, например, XP-Hosting.com и SpaceWeb (www.sweb.ru). Первый выплачивает до 20%, а второй – 10% суммарного ежемесячного заработка привлеченных партнеров.

Еще один существенный момент – то, как партнер может получить заработанные в рамках партнерской программы средства. Обычно хостинг-провайдеры предлагают два способа оплаты работы партнеров. Во-первых, как правило, существует возможность оплатить собственный хостинг у этого провайдера из заработанных средств (в этом случае партнера уже нужно рассматривать скорее как реселлера). Во-вторых, возможность вывода средств – например, на банковский счет или почтовым переводом, но чаще всего вывести средства, заработанные по партнерской программе можно только через системы интернет-платежей Webmoney (www.webmoney.ru), e-Gold (www.e-gold.com) или Яндекс.Деньги (money.yandex.ru). Именно так рассчитываются с партнерами, например, ValueHost и «Агава». Правда, т.к. провайдеры все-таки не заинтересованы в выводе средств партнерами, то они могут обставлять выплаты множеством различных условий. Например, у того же ValueHost'a перевод средств на WebMoney обставлен таким образом, что партнер неизбежно первые несколько раз оформит заявку на перевод средств неправильно, да и рассматриваются такие заявки в течение довольно долгого времени – так решение о переводе средств может приниматься в течение

двух недель, а сам перевод может занимать вообще до 35 банковских (именно банковских, т.е. без учета выходных) дней. При этом оплата счетов хостинга на самом ValueHost'e со средств, заработанных по партнерской программе, осуществляется буквально за пару секунд.

Отдельные провайдеры могут вообще работать по старинке – т.е. к ним можно просто привести клиента за руку в офис. Такой вариант теоретически вполне возможен, например, в партнерских программах провайдера Highway.Ru. Соответственно и заработанные средства по партнерской программе у Highway.Ru тоже можно получать прямо в его офисе. Правда, Highway.Ru скорее ориентирован на работу по планам реселинга, чем партнерским программам.

Так же необходимо рассмотреть проблемы платного хостинга.

Как известно, реклама - двигатель прогресса, а Интернет - самый что ни есть продукт этого самого прогресса. Рекламными баннерами обвешаны чуть ли все информационные сервера интернета. При этом, естественно, каждый баннер несет информацию о предложении гораздо выгодном, чем у конкурентов, рекламирующихся на том же сайте. Правда, очень часто после нажатия на баннер оказывается почему-то, что не все так прекрасно: рекламе свойственно приукрашивать предлагаемую услугу. В этом отношении реклама в Сети ничем не отличается от любых маркетинговых акций в офлайне. Будь то реклама какого-то сайта, либо сугубо коммерческой услуги, например, хостинга. Реклама последнего, как одной из наиболее специфических услуг в Сети, занимает довольно большую долю всей интернет-рекламы и уступает, наверное, только рекламе новостных сайтов.

Многие хостинг-провайдеры привлекая клиентов, далеко не всегда говорят полную «правду» о предлагаемой услуге. Расписывая свои услуги, компании часто стремятся отодвинуть на задний план некоторые нюансы, которые могут быть довольно существенными для пользователя, но которые могут привести к уходу клиента к конкуренту. Причем у последнего далеко

не всегда предлагаемые условия лучше, а часто просто гораздо лучше проведена пиар-кампания.

Как правило, «приукрашивание» не носит характер намеренного введения клиента в заблуждение, а выражается в «замалчивании» некоторых аспектов, которые чаще всего приводятся мелким шрифтом в виде примечаний к прейскуранту в нижней части страницы либо в виде приложения к договору. С одной стороны, клиент не сможет обвинить провайдера в сокрытии какой либо информации о предлагаемой услуге, а с другой стороны всегда существует довольно большая вероятность того, что человек, не очень хорошо разбирающийся в нюансах ценовой политики хостинг-провайдеров, обнаружит уже только после оплаты, что купил не совсем то, на что рассчитывал.

Причем ведь набор маркетинговых акций, «приукрашивающих» предоставляемые услуги, достаточно стандартен. И начинается он с того, что хостинг-провайдеры, как и операторы сотовой связи (да и не только они), не очень любят указывать цены с учетом всех налогов (с налогом на добавленную стоимость и налогом с продаж). И их можно понять - без НДС цены на услуги выглядят гораздо привлекательнее.

Стоит отметить, что считается правилом хорошего тона сообщать о том, включены налоги в указанные цены или нет, если не в верхней части страницы перед прайс-листом, то хотя бы первым пунктом в примечаниях. Но есть целый ряд провайдеров на сайтах которых вообще нет ни слова об этом. Так что клиент вполне может обнаружить увеличение стоимости примерно на четверть уже только при получении счета.

Еще один рекламный трюк, которым провайдеры привлекают клиентов, это объявление неограниченности какой-либо услуги: например, трафика, количества почтовых адресов электронной почты или поддоменов. Неограниченного трафика на самом деле не бывает в принципе: просто посещаемость большинства сайтов такова, что укладывается в те минимальные объемы, которые и может предоставить провайдер в рамках

определенного тарифного плана. Но где этот предел известно только самому провайдеру: на сервере лимит трафика может быть вообще нигде не указан, либо если и указан, то не на той странице, где находится прейскурант на услуги. Пользователь узнает о том, что превысил определенный лимит (особенно, если это касается зарубежного трафика) только уже по факту такого превышения, когда провайдер попросит перейти на другой тарифный план.

Неограниченное количество адресов электронной почты тоже, по большому счету, фикция: точнее, виртуальных e-mail-адресов может быть действительно сколько угодно, но реально все письма будут приходит (точнее «собираться») в один почтовый ящик. А неограниченного количества почтовых ящиков быть не может в принципе: их количество в любом случае ограничено объемом каждого ящика, совокупный размер которых чаще всего не может превышать общую дисковую квоту в рамках тарифного плана. [34, с.532]

То же самое в полной мере часто относится и к неограниченному количеству доменов (или поддоменов) на один сервер: скорее всего, это означает не то, что на одном хостинг-аккаунте в пределах дисковой квоты можно будет держать неограниченное количество разных сайтов со своим самостоятельным адресом, а то, что для одного сайта просто может быть любое количество адресов-синонимов («зеркал»). Далеко не каждый непрофессиональный пользователь Сети понимает разницу между просто почтовым адресом и почтовым ящиком (или отдельным доменом и отдельным сервером), и этим с удовольствием могут воспользоваться хостинг-провайдеры. Тем более, что, когда пользователь обнаружит это несоответствие, то ему скорее всего придется «раскошелиться» на еще один заказ хостинга. А провайдеру, как говорится, это только на руку.

Отдельная история - поддержка CGI-скриптов. Под этим у разных провайдеров подразумеваются совершенно разные вещи. Если хостинг-провайдер заявляет о поддержке CGI-скриптов, то необходимо

удостовериться, что в данный сервис входит не только возможность использования определенного стандартного набора уже установленных скриптов (обычно этот набор включает форум, гостевую книгу, чат, счетчики и т.д. - обычный набор сервисов и у серверов бесплатного хостинга), но и возможность использования собственных скриптов. Может быть существенным и то, поддерживается ли CGI в любой директории на сервере или только в специальной папке cgi-bin.

С другой стороны провайдер не всегда может быть заинтересован в очень подробном описании предоставляемых услуг. Ведь, как самое трудное - это привлечь нового клиента, а его удержание - уже другое дело. Данное правило наиболее ярко у многих провайдеров проявляется в отношении регистрации доменов. Например, очень часто при первичной покупке хостинга на какой-то крупный период времени (обычно не менее полугода или года) провайдеры регистрируют для клиента «бесплатно» и домен. Но такая «скидка» очень часто носит только разовый характер и только при первом заказе - через год перерегистрация домена будет, скорее всего, предоставляться уже за отдельную плату.

Выбор места для своего сайта всегда очень ответственное дело – от того, где и как будет располагаться сайт, зависит очень многое: и его посещаемость, и общий интерес пользователей, и, наконец, отдача (экономическая или просто психологическая в зависимости от тематической направленности ресурса и целей своего автора). С этой проблемой так или иначе сталкиваются любые создатели сайтов – как начинающие юзеры, так и «матерые» веб-мастера, программисты и IT-менеджеры. Конечно, для начинающего пользователя Интернета наиболее оптимальным местом для размещения своего персонального сайта являются различные сервера бесплатного хостинга. Но по мере развития такого интернет-проекта, его роста и перехода из любительской категории в профессиональную, сервера бесплатного хостинга перестают удовлетворять потребностям таких сайтов (точнее их владельцев). И дело не только в том, что на сайте вроде Narod.ru

или Boom.ru постоянно что-то «глючит» или раздражают прикрепленные к сайту баннеры. Хотя и этого достаточно для того, чтобы перевести более или менее удачный интернет-проект на более качественный сервер хостинга. По большому счету к сервису бесплатного хостинга не может быть претензий, на то он и бесплатный хостинг, что ничего не гарантирует: ни отсутствие долгих промежутков времени, в течение которых сайт может быть просто недоступен, ни отсутствие рекламных баннеров, к которым владелец сайта не имеет никакого отношения. Сервера бесплатного хостера не гарантируют постоянного предоставления услуги и в любой момент могут изменить ее условия, очень часто это выражается либо введения обязательной рекламы на страницах сайтов пользователей, либо вообще к переходу со временем на коммерческую основу. Наибольший резонанс в Сети в отношении серверов бесплатного хостинга, как правило, вызывает постепенная коммерциализация предоставлемых услуг, которая происходит на протяжении последних 2-3-х лет как на Западе, так и в России. Если изначально большинство free-хостеров предоставляло абсолютно бесплатные услуги, то мере дальнейшего роста каждого проекта своего рода степень «халявности» предоставляемого сервиса обычно уменьшается.

Кроме того, большинство серверов бесплатного хостинга предоставляет довольно ограниченный набор сервисов, в которые, как правило, не входит поддержка различных скриптов (вроде CGI или PHP) и баз данных, необходимых для функционирования солидных интернет-проектов. Такие сервисы предоставляют преимущественно коммерческие хостинг-провайдеры. Хотя в последнее время появился и ряд бесплатных хостеров с поддержкой CGI/PHP/MySQL (например, Hut.ru, Hoha.ru, Webservis.ru), но в любом случае они не поддерживают собственный домен второго уровня. В конце концов, в определенный момент развития интернет-проекта становится понятно, что наличие в адресе сайта чего-то типа «chat.ru» или «narod.ru» просто несолидно.

Если для корпоративного пользователя, особенно крупного, кроме цена важна не только стоимость предоставляемых услуг и их набор, но и в первую очередь обеспечение бесперебойной работы сервера и вопросов безопасности (особенно в отношении физического хостинга), то для частного и массового пользователя на первом месте все-таки стоимость. Большинство потенциальных клиентов хостинг-компаний среди физических лиц – владельцы персональных сайтов на серверах бесплатного хостинга. Поэтому вряд ли владелец сайта, который недавно хостился бесплатно, готов платить в год существенно больше 100 долларов США.

Второй основной вопрос, который интересует пользователя при выборе хостера, это объем дискового пространства. Как показывает практический опыт, в 60-80% случаев для самого сайта «с лихвой» хватает 20-30 Мб, а для почты еще мегабайт десять. Но наиболее продвинутым пользователям (с использованием различных баз данных, большого количества графических материалов, для удаленного хранения файлов, создания индексируемой базы сайта для осуществления опции поиска и т.д.) может понадобиться и существенно больше места - вплоть до ста и более мега байт. Особенно в том случае, если хостинг нужен одновременно для нескольких сайтов. Поэтому объективно, размер дискового пространства, который устраивает массового пользователя, начинается примерно с 30-50 Мб. К тому же чисто психологически вчерашний пользователь бесплатного хостинга скорее ухватится за тот план, который по сравнению с другим, при равной стоимости предлагает большее дисковое пространство, а показатели надежности и безопасности, к которым в основном и аппелируют крупные хостинг-провайдеры, скорее всего при выборе провайдера окажутся отодвинутыми на второй план.

После цены и количества мегабайт потенциального пользователя интересует и общий набор дополнительных сервисов – в первую очередь это поддержка адреса электронной почты, баз данных и скриптов. Но поддержка адреса электронной почты фактически априори входит в любой тарифный

план у любого провайдера, поэтому эта услуга фактически уже не оказывает особого влияния на выбор пользователя (может иметь только значение количество поддерживаемых адресов). Другое дело – базы данных и скрипты. Конечно, они нужны далеко не каждому частному пользователю, но тем не менее необходимость в них может возникнуть даже у того, кто никакого понятия не имеет о веб-программировании, а просто захочет установить, например, готовый скрипт форума, а то и вообще воспользоваться готовым порталом вроде PHP-Nuke.

Фактически эти три критерия (цена/объем дискового пространства/поддержка скриптов) и есть тот базис, который предопределяет выбор пользователя в пользу той или иной компании, предоставляющей услуги хостинга. Остальные «навороты» уже играют более подчиненную роль. Хотя, например, неограниченный трафик тоже избавит пользователя от лишней головной боли. Ну и различные прочие «бонусы» (как например, скидки при заказе хостинга на год, бесплатный домен, партнерские программы и прочие маркетинговые акции) тоже нельзя не учитывать.

Развитие рынка веб-хостинга в Казахстане и общее снижение цен на нем привело к тому, что если раньше позволить себе содержание собственного сайта на коммерческом хостинге с соответствующим набором услуг и атрибутов (например, собственного домена второго уровня) могли преимущественно только организации, либо люди с уровнем доходов выше среднего, то в настоящее время в Казахстанском интернете можно встретить уже довольно большое количество персональных профессиональных сайтов, хостящихся отнюдь не на сервере бесплатного хостинга.

Администрирование содержит инструменты управления интернет – магазином и включает в себя как общие настройки магазина, так и специальные настройки (см.рис.2).

В администрировании будут содержаться основные настройки интернет-магазина:

- общие настройки магазина: название магазина, адрес, телефон, e-mail адрес магазина и т.д;

- настройки формы регистрации клиента в интернет-магазине;

- общие настройки доставки и упаковки товара;

- настройки склада;

- настройки логов, файлов, куда будет записываться служебная информация;

- настройки формата вывода товара в интернет-магазине. Вы можете настроить формат вывода товара по своему желанию;

- всевозможные настройки каталога т.е. добавление, удаление, редактирование товара и категорий, работа с производителями, excel импорт/экспорт товаров и т.д.;

- настройки различных модулей доставки, оплаты, модули скидок и т.д. Здесь Вы можете устанавливать новые модули, удалять существующие модули, настраивать способы оплаты и доставки заказов интернет-магазина;

- управление оформленными заказами, управление зарегистрированными клиентами;

- добавление, удаление, изменений курсов валют;

- статистические отчёты о работе интернет-магазина;

- важные инструменты для работы интернет-магазина. Такие как резервное копирование базы данных, незавершённые заказы, поисковые запросы и т.д.

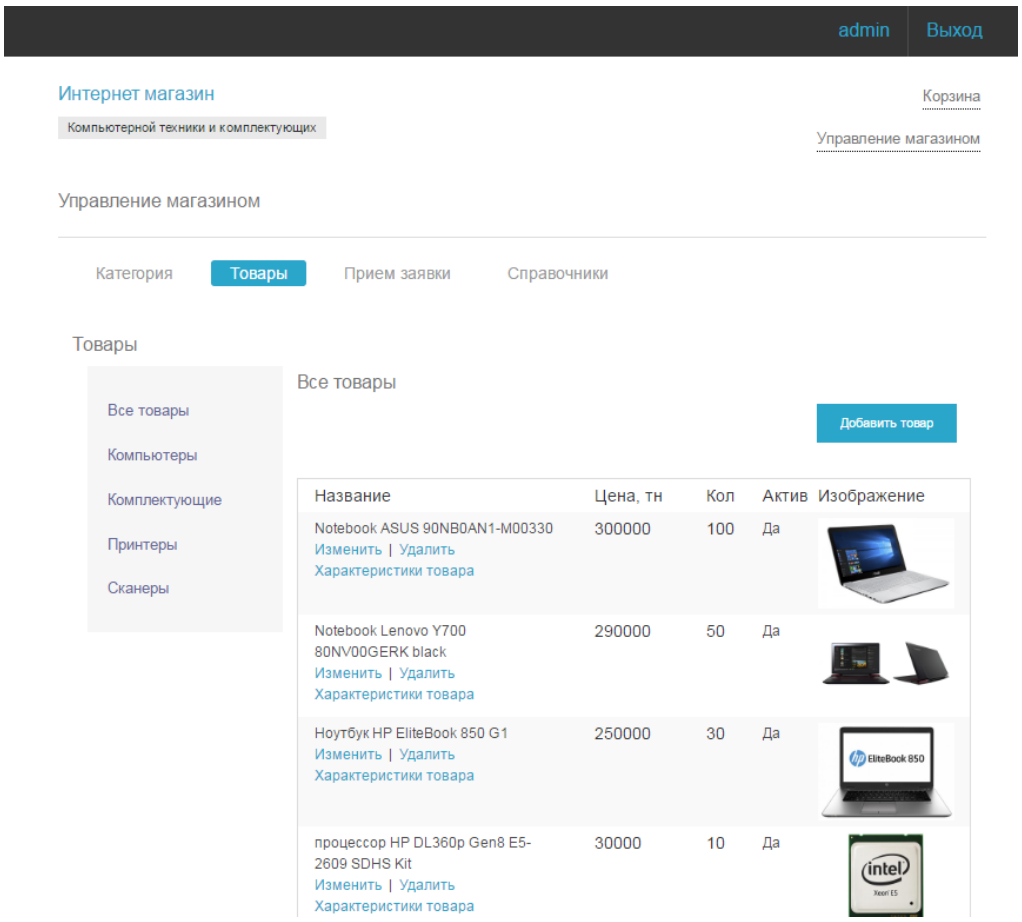

Рисунок 4 - Панель управления интернет-магазином

В клиентской части архитектуры разрабатывается максимально удобная и доступная работа потенциального клиента на страницах интернет – магазина. Разработка интерфейса, доступные и понятные диалоговые окна, удобные системы оплаты и доставки товаров. Немаловажным фактором является обратная связь, позволяющая высказать клиенту свое мнение о том или ином товаре/услуге, о качестве обслуживания и магазина в целом.

Проанализировав работу уже работающих интернет – магазинов, был сделан вывод о том, что обязательно будет реализовано в проекте.

Витрина магазина будет оформлена так, чтобы покупатель без труда мог находить интересующий его товар и иметь возможность полечить о нём исчерпывающую информацию (описание в виде текста плюс несколько фотографий).

Товары будут разделены по группам, обеспечится возможность поиска товаров по части названия и описания. Для каждого товара будет предусмотрено краткое и полное описание, плюс несколько фотографий.

Для наглядности будут добавлены специальные разделы, содержащие товары, сгруппированные по маркетинговым признакам. Допустим:

- «Новинки» (товары, недавно поступившие в продажу);

- «Специальные предложения» (товары, на которые по каким-либо причинам снижены цены);

- «Товары дня» (самые модные товары);

- «Лидеры продаж» (наиболее покупаемые товары).

О том или ином товаре зарегистрированный пользователь сможет оставить отзыв.

При оформлении заказа покупатель вносит контактную информацию: логин, пароль, адрес доставки, телефон и т.д. После регистрации покупателю будет отправляется по электронной почте письмо с сохраненными данными.

В электронном магазине будут предусмотрены и информационные разделы:

- с данными о магазине (сфера деятельности, адрес, контактные телефоны и т.д.);

- с информацией по доставке товара;

- с информацией по скидкам;

- новости магазина;

- статьи (системы управления новостями и статьями предоставляют возможность использовать интернет-магазин как настоящий информационный портал);

- прочая полезная информация.

Реализуется рассылка новостей. Посетитель имеет возможность подписаться (и отписаться) на новости интернет-магазина. После подписки покупателю периодически высылается информация о новинках магазина.

Будет так же налажена обратной связи администратора с клиентами, что будет способствовать увеличению посещаемости интернет-магазина.

Обратной, невидимой покупателю, стороной интернет-магазина является система управления. Вход в систему администрирования осуществляется только после ввод администратором логина и пароля (логин и пароль администратор может менять). Администратор будет иметь возможность полностью управлять содержимым интернет-магазина:

- добавлять или удалять товары, описания и фотографии к ним, изменять их стоимость, условия доставки товаров и уровень скидок;

- редактировать разделы магазина (новости, статьи, вопросы и ответы, отзывы и вопросы к товарам и пр.);

- редактировать специальные разделы магазина (новинки, специальные предложения, товары дня, лидеры продаж);

- редактировать контактную информацию интернет-магазина;

- редактировать содержание заголовков и текстов писем, отправляемых покупателю при регистрации и покупке товара;

- составлять и рассылать письма с новостями магазина подписчикам;

- просматривать историю заказов и статистику покупателей;

- изменять курс валюты на витрине магазина.

Заходя на сайт интернет – магазина открывается главная страница, где просится зарегистрироваться либо ввести персональные данные зарегистрированным пользователям.

Переход к диску возможен по категории, по производителю, по поиску названия, либо просто нажав на картинку, соответствующей обложке диска.

После выбора товара зарегистрированному пользователю предложат выбрать способ оплаты и способ доставки товара. Проверив все данные, необходимо будет подтвердить заказ.

Информация о сделанном заказе просматривается менеджером магазина и передается на выполнение.

Со схемой алгоритма работы интернет – магазина можно будет ознакомиться на рисунке 6

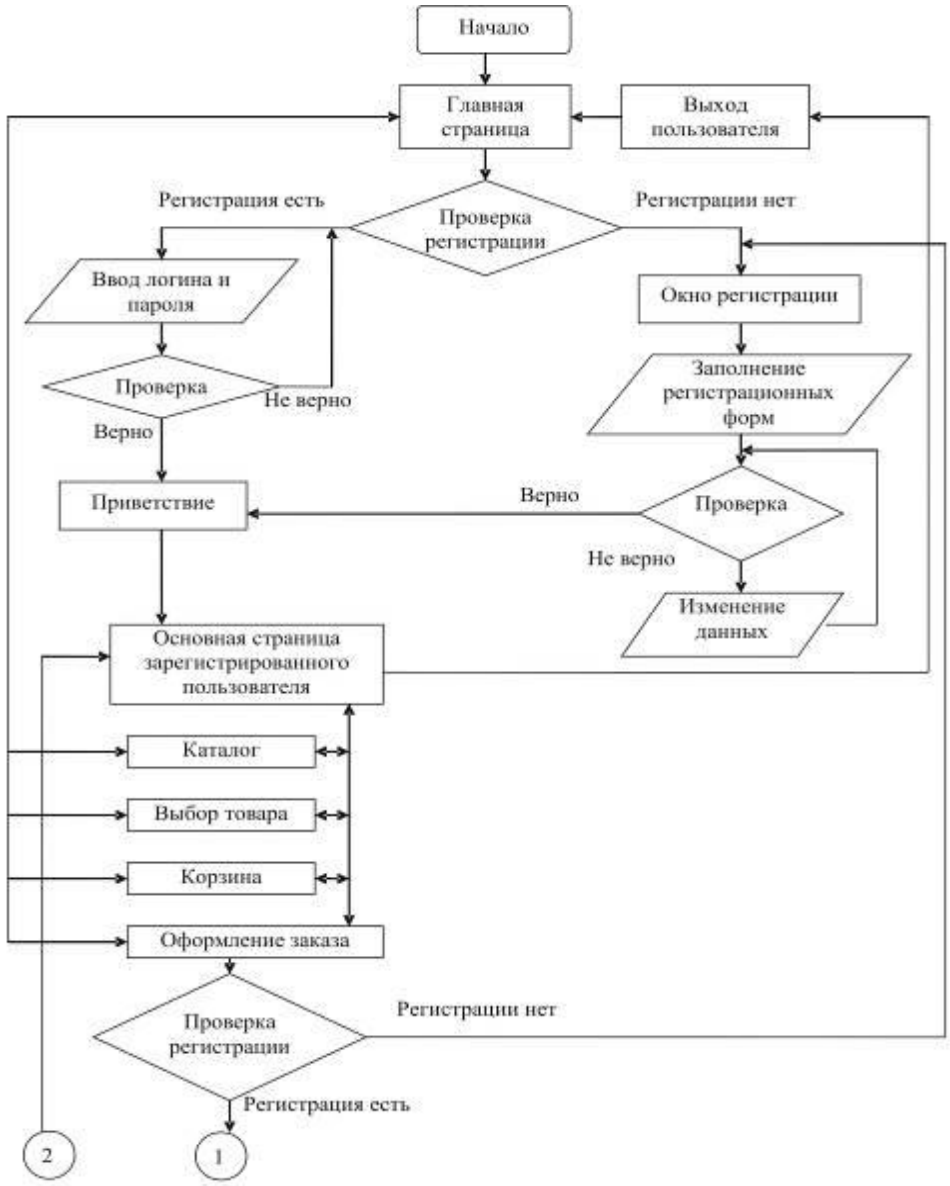

Рисунок 5 – Алгоритм работы интернет магазина

3. Интерфейс разработки интернет магазина

С ростом интернета широкое распространение получили веб-интерфейсы позволяющие взаимодействовать с различными программами через браузер (например, управление своим заказом в интернет-магазине).

Веб-интерфейсы удобны тем, что дают возможность вести совместную работу сотрудникам, не находящимся в одном офисе (например, веб-интерфейсы часто используются для заполнения различных баз данных или публикации материалов в интернет – СМИ).

Интерфейс - в широком смысле - определенная стандартами граница между взаимодействующими независимыми объектами. Интерфейс задает параметры, процедуры и характеристики взаимодействия объектов.

Интерфейс пользователя - элементы и компоненты программы, которые способны оказывать влияние на взаимодействие пользователя с программным обеспечением. В том числе:

- средства отображения информации, отображаемая информация, форматы и коды;

- командные режимы, язык пользователь-интерфейс;

- устройства и технологии ввода данных;

- диалоги, взаимодействие и транзакции между пользователем и компьютером;

- обратная связь с пользователем;

- поддержка принятия решений в конкретной предметной области;

- порядок использования программы и документация на нее.

3.1 Правила построения интерфейса магазина

Есть несколько простых правил, позволяющих интерфейсу магазина быть понятым клиенту.

1. Чем проще, тем лучше. Это вовсе не значит, что должен быть только текст и контактная информация, но на сайте не должно быть бесполезной информации, шрифт должен легко читаться. Графические элементы должны быть были чёткими, выразительными и быстро загружаться. На сайте магазина недопустимо использовать анимацию и звук, которые долго загружаются и отвлекают внимание покупателя.

Человеческий глаз просматривает страницы сайта сверху вниз. Наибольшее внимание сосредотачивается на верхней левой части страницы. Поэтому в верхней части страницы сайта, как правило, размещается наиболее важная информация: название фирмы, логотип, само название сайта и т.д.

Чем проще выполнена верхняя часть страницы, тем легче запомнить название сайта и саму фирму.

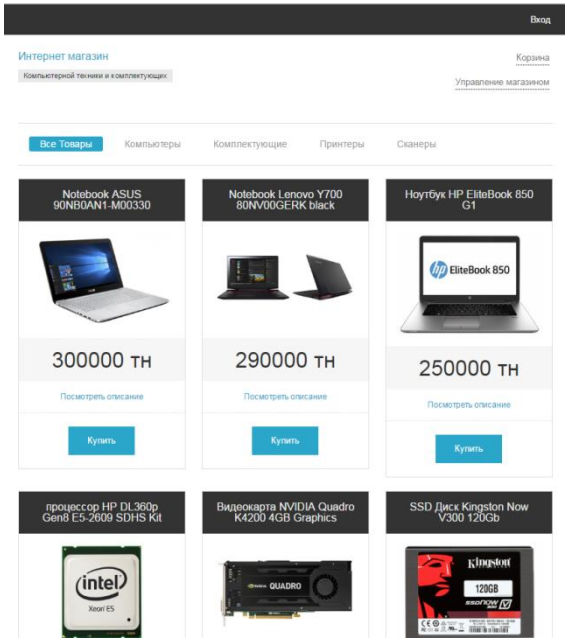

Рисунок 6 – Главная страница сайта

Типовой ошибкой многих сайтов является нагромождение в верхней части страницы сайта авангардной и сложной композиции из многих рисунков, среди которых едва заметно располагается название фирмы и название сайта. В результате человеческий глаз, сканируя данную часть страницы сайта, не успевает распознать и запомнить нужную информацию. Происходит перенасыщение.

2. Покупатель должен без труда находить интересующий его товар и иметь возможность полечить о нём исчерпывающую информацию (описание в виде текста плюс несколько фотографий). На витрине желательно разместить данные о деятельности фирмы-продавца (чем занимается

компания, какие регионы обслуживает, адрес, контактную информацию и т. д.). (см.Рисунок 9)

3. Товары должны быть распределены по группам. Необходимо обеспечить возможность поиска товаров по части названия и описания. Для каждого товара обычно предусмотрены: краткое и полное описание, плюс несколько фотографий.

4. Расчёт стоимости и вывод цен должен осуществляться в тенге.

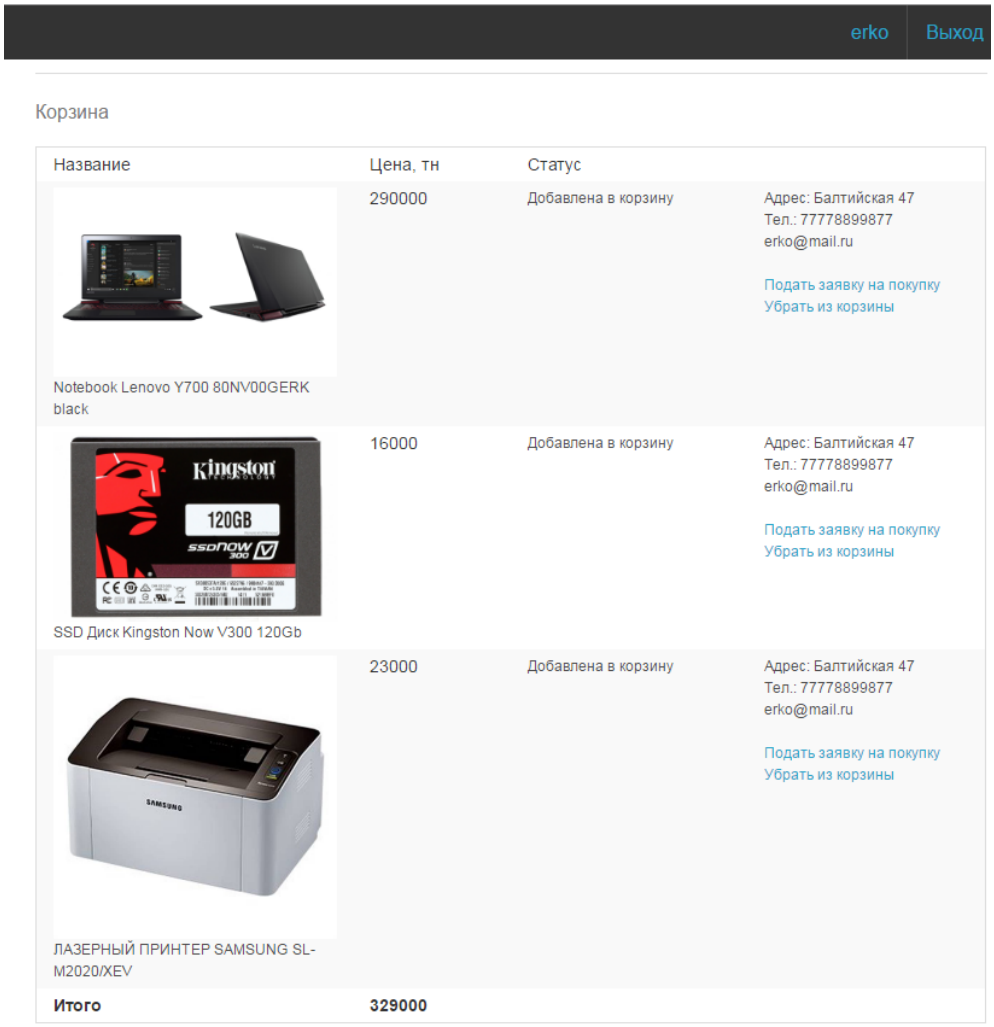

Рисунок 7 - Корзина клиента

5. В электронном магазине могут быть и информационные разделы:

- с данными о магазине (сфера деятельности, адрес, контактные телефоны и т.д.);

- с информацией по доставке товара;

- с информацией по скидкам;

- новости магазина;

- статьи (системы управления новостями и статьями предоставляют возможность использовать интернет-магазин как настоящий информационный портал);

- прочая полезная информация.

6. Рассылка новостей. Посетитель имеет возможность подписаться (и отписаться) на новости интернет-магазина. После подписки покупателю периодически высылается информация о новинках магазина.

7. Увеличению посещаемости интернет-магазина и количества постоянных клиентов способствует также наличие обратной связи администратора с клиентами. Важно, чтобы администраторы как можно быстрее и максимально полно отвечали на письма.

8. Обратной, невидимой покупателю, стороной интернет-магазина является система управления. Вход в систему администрирования осуществляется только после ввод администратором логина и пароля (логин и пароль администратор может менять). Администратор имеет возможность полностью управлять содержимым интернет-магазина:

- добавлять или удалять товары, описания и фотографии к ним, изменять их стоимость, условия доставки товаров и уровень скидок;

- редактировать разделы магазина (новости, статьи, вопросы и ответы, отзывы и вопросы к товарам и пр.);

- редактировать специальные разделы магазина (новинки, специальные предложения, товары дня, лидеры продаж);

- редактировать контактную информацию интернет-магазина;

- редактировать содержание заголовков и текстов писем, отправляемых покупателю при регистрации и покупке товара;

- составлять и рассылать письма с новостями магазина подписчикам;

- просматривать историю заказов и статистику покупателей;

- изменять курс валюты на витрине магазина.

Если заказчик интернет-магазина собирается работать ещё и с оптовыми клиентами, необходимо предусмотреть работу сайта как с розничными, так и с оптовыми ценами на товары.

9. Загрузка товарных предложений должна осуществляться из файла Microsoft Excel. Файл, в котором описаны все группы, подгруппы каталога, а также информация о товарах, создается на локальном компьютере (ограничений к количеству групп, категорий и самих товаров быть не должно). Затем файл загружается в интернет-магазин. После загрузки прайс-листа товары автоматически появляются на витрине. Таким образом, можно оперативно изменять данные о ценах и наличии товаров.

10. Аккуратная работа с цветом. Правильно примененный цвет может, например, передавать тонкие различия между однородными элементами. Неправильно примененный цвет может мешать работать с программой.

Особенно это относится к красному цвету. Так уж получилось, что для всех людей красный цвет ассоциируется с некой опасностью. Большое количество красного цвета в каком-либо месте на экране привлекает внимание, заставляет пользователя настораживаться, думая что что-то не так.

Дорожные знаки красного цвета либо запрещают, либо предупреждают об опасности. Поэтому, если кнопка на экране окрашена красным, независимо от того, что на ней написано пользователь будет стараться избегать нажатия на нее.

В малых количествах красный цвет может исправно служить в качестве ненавязчивого указания наличия каких-либо проблем. Например, если получившееся в результате расчета число превышает норму.

Красный цвет может также использоваться в парах с другими. Существуют две метафоры – «термометр» когда красному противостоит синий, и «светофор» - зеленый. Обе они должны использоваться только, если это уместно.

Руководствуясь данными принципами разработки интерфейса, было решено сделать ставку на простоту и информативность, что бы пользователь, попадая на сайт, должен получать четкую информацию о товаре, новинках, предстоящих релизах. Так же о том, как он сможет оплатить заказ, каковы условия и сроки доставки и т.д.

В интернет – магазине должен быть реализован удобный и быстрый поиск необходимого пользователю товара, так как не все имеют неограниченный доступ в интернет, и многие оплачивают его по часам. Да и утомительный просмотр каталогов мало кому по душе.

Все товары, предлагаемые на сайте, должны быть в наличии. Так как вряд ли кого-то порадует сообщение о том, что заказанный компакт-диск, он сможет получить только через месяц. Будет предусмотрена система заказа товара, отсутствующего на складе.

Будет сделана простая и предельно удобной навигация, с тем чтобы пользователь оперативно получал ответы, на любые вопросы по тематике сайта. Не найдя нужной информации, посетитель может просто уйти, так и не получив четкого представления о вашем ресурсе.

Цветовая гамма будет ориентированна на бордовые тона, удачно гармонирующие с черным цветом текста. Цветные обложки DVD или компакт- дисков будут иметь небольшой размер и будут удачно считаться с общей цветовой гаммой страницы. [35, с.45]

4. Экономическое обоснование проекта

4.1 Маркетинговые исследования магазина

Главными преимуществами интернет-магазина являются самообслуживание и специализация.

Многие магазины в крупных городах работают по принципу «супер-маркетов», где реализована позиция выставления товара к покупателю в наиболее выгодном свете, есть возможность беспрепятственно взять товар с полки, рассмотреть его.

Классификация спроса мобильных телефонов может проводиться и по способности товаров удовлетворять потребности определенных групп потребителей: ученики-студенты, молодые люди до 40 лет, взрослое поколение. Особенностью исследования таких товарных рынков является учет взаимосвязей различных форм и средств удовлетворения определенной потребности.

Ученики-студенты, это, как правило, лица с невысоким материальным положением, могут позволить себе купить 1-3 единицы товара за одно посещение магазина. Они всегда в курсе новинок, имеют потребность ко всему новому, модному, креативному. Практически всегда знают, чего хотят. Обращают внимание на упаковку товара, на расположение, на цену. Не любят прибегать к помощи продавца, поэтому магазин самообслуживания, каким и является магазин для них оптимальный вариант.

Молодые люди до сорока лет, это обеспеченные клиенты покупающие, как правило, 2-5 единиц товара. Интересуются как новинками, так и известными вещами. Любят прибегать к помощи продавцов-консультантов. Охотно оставляют заказы на отсутствующие в данный момент позиции. Так же предпочитают походить – рассмотреть товар индивидуально.[36, с.94]

Для удобства постоянных покупателей и привлечения потенциальных клиентов мы решили создать дополнительную услугу: покупка диска с доставкой.

Удобство для постоянных покупателей заключается в том, что узнать о новый либо об ожидаемых поступлениях, они смогут, просто зайдя на сайт магазина в Интернете. Там же они смогут при желании отложить для себя товар, для последующей покупки с помощью курьера либо самостоятельно приехав, либо оставить заявку на временно отсутствующий товар.

Очень удобен тот факт, что в интернет-магазине знакомится с товаром и оставлять заказ будет возможно круглосуточно.

Так же к преимуществам данной дополнительной услуги можно отнести: значительное снижение затрат, связанных с обменом информацией за счет использования более дешевых средств коммуникаций и значительно увеличивается оперативность получения информации.

Электронный магазин сможет так же приносить дополнительный доход при продаже продукции через сеть интернет покупателям, находящимся в других городах области. Заинтересовав клиента, относительно невысокими ценами, широким ассортиментом надеясь выйти на общеказахстанский уровень.[37, с. 65]

В собственности магазина имеется вся необходимая оргтехника и дополнительное помещение, необходимый товарный запас, поэтому введение новой услуги поднимет имидж магазина, снизит затраты на единицу продаваемой продукции, обеспечит подъем продаж за счет привлечения новых клиентов и обеспечит удобство постоянным и потенциальным покупателям.

4.2 Расходы по созданию и размещению магазина в сети интернет

Так как за основу берется бесплатная версия программного продукта OsCommerce, в затратную часть создания интернет – магазина относятся такие расходы как: расходы по электроэнергии, расходы по размещению

магазина в сети интернет (хостинг), заработная плата программисту и курьеру и прочие всевозможные расходы на канцелярские товары и расходные материалы для компьютера. Такие расходы как аренда помещения, амортизация компьютера и оргтехники и прочие расходы относятся к основному магазину

Таблица 3 -Расчет электроэнергии для девятичасового рабочего дня.

Наименование	кол-во	кВт/час	кВт в сутки (примерно)	кВт в месяц
Компьютер	1	0,17	1,53	45,9
Освещение	3	0,36	9,72	291,6
Сплит	1	0,7	6,3	189
ИТОГО:		1,23		526,5

Для предприятий 1 кВт / ч= 5,25

В месяц 5,25*526,5=2764,125

В сети интернет магазин планируется разместить на ресурсах провайдера города Астана, что обеспечит удобное обслуживание и рекламную ссылку на магазин с главной страницы сайта провайдера http://kaztelekom.kz/

Заработная плата программисту составляет 90000 тг.

Таблица 4– Расчет ежемесячных затрат на содержание интернет – магазина.

Наименование	Сумма, тг
Зарплата программиста	90000
Зарплата курьера	50000
Транспортные расходы курьера	4000
Электроэнергия	2764,125
Хостинг	10000
Интернет	15000
Прочие расходы	12000

Итого:	183764,125

$R_{Пост}$ = 183764,125– постоянные ежемесячные расходы.

Так как помещение и оборудование уже имеется в наличии, ассчитаем годовую сумму амортизационных отчислений.

Годовая сумма амортизационных отчислений рассчитывается по формуле:

$$A = \frac{Ф * N_A}{100\%},$$

где Ф– первоначальная стоимость основных фондов по видам, тг

N_A– норма амортизации по видам основных фондов, в %.

Годовую сумму амортизационных отчислений отразим в таблице 4.3.

Таблица 5– Расчет годовой суммы амортизационных отчислений.

Элементы основных фондов.	Кол-во	Стоимость, тг	Сумма тг	Норма амортизации, %	Амортизационные отчисления, тг
Компьютер	1	90000	90000	20%	18000
Сплит система	1	45000	45000	20%	9000
Помещение	13,6м2	6000	81600	3%	2448
ИТОГО:					29448

Таким образом, годовая сумма амортизационных отчислений составляет 29448 тг.

Исходя из того, что трудоёмкость создания информационной системы составляет 10 дней, рассчитываем амортизацию оборудования за этот период по формуле:

$$А_{факт} = \frac{A_{год} * T_{факт}}{365},$$

Рассчитаем сумму амортизационных отчислений для перечисленной группы оборудования с учетом числа календарных дней на разработку программного обеспечения (интернет – магазина) по формуле:

$$А = \frac{29448 * 10}{365} = 806{,}8 \text{ тг}$$

Заработная плата программиста составляет 90000 тг. Соответственно, затраты на заработную плату включаемые в себестоимость программы с учетом работы над программой в течение 12 дней составят:

$$ЗП_{пр} = \frac{ЗП_{мес} * T_{факт}}{Д},$$

где $ЗП_{пр}$ – заработная плата в месяц программиста, тг;

$Т_{факт}$ – число календарных дней на разработку интернет – магазина;

Д – число дней в периоде (месяц).

$$ЗП_{пр} = \frac{90000 * 10}{22} = 40909 \text{ тг.}$$

Таблица 6– Расчет ежемесячных материальных затрат.

Наименование	Сумма, тг /мес.
Электроэнергия	2764,125
Хостинг	10000
Интернет	15000
Прочие расходы	12000
Итого:	39764,125

$З_м$= 39764,125 тг в месяц

Следовательно, затраты на период разработки программного продукта рассчитаем по формуле:

$$З_{пр} = \frac{З_м * Т_{факт}}{Д},$$

где $З_м$ – ежемесячные затраты, тг

$Т_{факт}$ – число календарных дней на разработку интернет – магазина;

Д – число дней в периоде (месяц).

$$З_{пр} = \frac{3976,125*10}{22} = 18074,6,$$

Рассчитаем себестоимость программного продукта по формуле:

$С_{ст}$ – себестоимость разработки программы

$$С_{ст} = З_{пр} + ЗП_{пр} + ЕСН + А$$

$С_{ст}$ = 18074,6+806,8 = 18881,4 тг.

Данная себестоимость является приблизительной, так как в ней не учтены некоторые детали, которые существенно не повлияют на итог.

$С_{ст} \approx$ 24100 тг.

Исходя из нормального уровня рентабельности 20% мы можем определить цену разработанной нами программы:

$$Ц = С_{ст} + \frac{С_{сп} * R}{100\%},$$

где $С_{ст}$ – себестоимость разработки программы;

R – планируемый уровень рентабельности.

$$Ц = 24100 \frac{24100 * 20}{100} = 28920 \text{ тг.}$$

Затраты на внедрение программного продукта составят 28920 тг.

Предполагается, что самоокупаемость интернет – магазина при ежемесячных затратах в 183764,125 тг произойдет уже при увеличении товарооборота на 15% и прибыли на 6,5%.

$П = (Тоб_н + Тоб_н * Тоб_{\%}) * П_{\%}$,

где П – предполагаемая прибыль;

$Тоб_н$ – базовый вариант товарооборота;

$Тоб_{\%}$- предполагаемый процент прироста товарооборота;

$П_{\%}$ - предполагаемый прирост прибыли.

$П_1 = (200000+200000*15/100)*6,5/100 = 149500$ тг.

Так как продажи программного обеспечения в магазине составляют примерно 20% от общего товарооборота, соответственно самоокупаемость магазина произойдет при приросте товарооборота на 50% и приросте прибыли на 25%.

$П_3 = (40000+40000*50/100)*25/100 = 150000$ тг.

Поэтому для более быстрой окупаемости интернет магазина было решено к программному обеспечению ввести дополнительные товары.

Поэтому полная окупаемость интернет – магазина с учетом затрат на внедрение произойдет за 2 месяца, после достижения прироста товарооборота магазина на 20 % и прибыли на 8%.

$П_3 = (200000+200000*20/100)*8/100 = 192000$ тг.

Таким образом, произведенные расчеты подтверждают предположение о том, что получение прибыли интернет – магазином ожидается с третьего месяца после начала его деятельности.

5. Мероприятия по охране труда и технике безопасности

5.1 Эргономическое обеспечение

Кратко опишем эргономические требования к рабочему месту инженера.

Персональный компьютер спроектирован с учётом эргономических требований:

- регулируемое положение дисплея;
- регулирование цвета и яркости фона и символов;
- перемещаемая клавиатура с регулируемым наклоном;
- возможность рационального размещения устройств;
- кнопки включения расположены в зоне досягаемости.

Эти свойства позволяют достигнуть индивидуальной приспособленности аппаратуры.

То есть, всё это способствует экономии движений оператора или программиста, уменьшает утомляемость при работе с компьютером. При планировке рабочего места необходимо учитывать зоны досягаемости рук оператора, которые устанавливаются на основе антропометрических данных человека. Важную роль играет построение физически обоснованного режима труда и отдыха, то есть рациональное их чередование.

Рекомендуется, чтобы продолжительность работы за видеотерминалом не превышала 50% рабочего времени.

5.2 Рабочее место программиста.

Рассмотрим помещение, в котором ведется разработка:

- тип помещения: офис;
- размеры рабочего помещения: длина 4 м, ширина 3 м, высота 3 м;
- остекление помещения - двойное(одно окно размером 2000x2000

мм);

- искусственное освещение - светильники: 2 светильника, в каждом по 2 люминесцентные лампы (ПВЛМ-1×40);
- внутренняя отделка стен - светлая;
- помещение по зрительным условиям работы относится к V разряду, т. к. наименьший объект различения от 1 до 5 мм;
- вид работы - разработка интернет-магазина светотехники;
- количество рабочих мест - 2;
- категория работ - легкая.

План помещения представлен на рисунке 7.

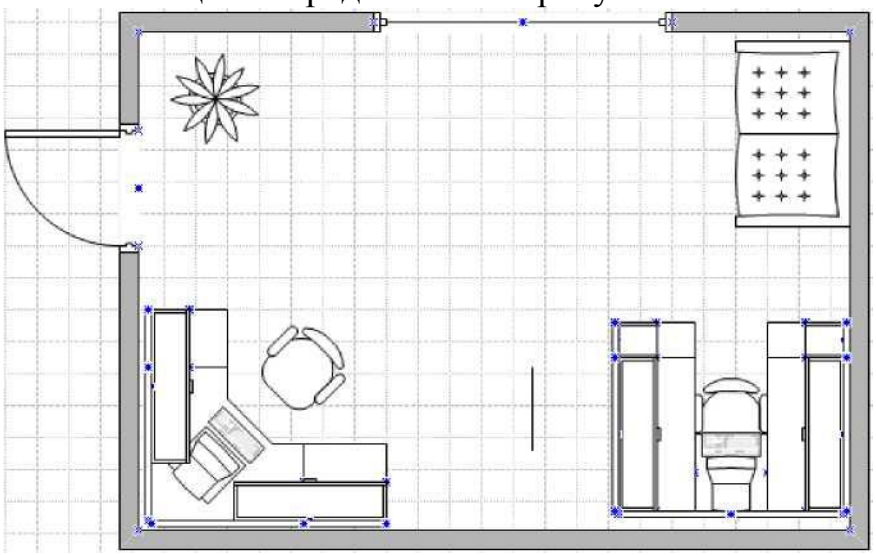

Рисунок 7 – План помещения

5.3 Средства пожаротушения

Рассматриваемое помещение по взрывопожарной и пожарной опасности соответствует категории В (пожароопасное помещение) в соответствии со СНиП РК 3.02-04-2009 Административные и бытовые здания в разделе Противопожарные требования, так как в нем имеются горючие материалы(мебель, бумага, покрытие пола, шторы), которые горят при взаимодействии с кислородом воздуха.

Любой процесс горения сводится к взаимодействию

$$C + O_2 = CO_2$$

Количество горючих веществ составляет $G_{ГВ}$ = 130 кг, а тепловой эффект реакции составляет $Q_{тепл.углерода}$ = 34,07 МДж/кг.

Пожарная нагрузка определяется по формуле

$$Q = Q_{тепл.углерода} \times G_{ГВ} \quad (5.10)$$

$$Q = 130 \times 34{,}07 = 4511 \text{ МДж}$$

Следовательно, пожарная нагрузка Q составляет 4511 МДж.

Удельная пожарная нагрузка определяется в соответствии санитарно эпидемиологическими требованиями

$$q = Q / S , \quad (5.11)$$

где $S = 12м^2$, площадь помещения

$$q = \frac{4511}{12} = 375 \text{ МДж/м}^2$$

Удельная пожарная нагрузка q составляет 375 $МДж/м^2$

Проверяется условие

$$Q < 0{,}64 \times q \times H^2$$

Оно не выполняется, следовательно, помещение соответствует категории В2. В качестве средств пожаротушения применяется ручной углекислотный огнетушитель ОУ – 2. Исправность огнетушителя периодически проверяется.

Характеристика воздействия выявленных опасных и вредных производственных факторов (ОВПФ) на организм человека и окружающую среду представлена в таблице 18.

Таблица 18 - Опасные и вредные производственные факторы

№	Фактор	Воздействие фактора на человека
1	2	3
1	Недостаточное освещение	Быстрая утомляемость глаз, ослабление внимания
2	Высокий уровень шума	Вызывает раздражение заболевания органов слуха
3	Опасность поражения электрическим током	Тепловые ожоги, поражение внутренних органов, судороги
4	Неправильная рабочая поза	Утомляемость, возможность искривления позвоночника

Работники организации имеют разный режим труда и отдыха. При круглосуточном сменном режиме труда перерывы для приема пищи и кратковременного отдыха не регламентированы и входят в рабочее время.

Режим труда и отдыха программистов и операторов ЭВМ должен зависеть от характера выполняемой работы: при вводе данных,

редактировании программ, чтении информации с экрана непрерывная продолжительность работы не должна превышать 4 часов при 8-часовом рабочем дне, через каждый час работы необходимо делать перерыв на 5-10 мин, а через 2 часа - на 15 мин.

Количество обрабатываемых символов (знаков) не должно превышать 30 тысяч за 4 часа работы. В ночные часы работники предприятия не должны выполнять работы или задания, требующие сложных решений или ответственных действий.

С целью снижения или устранения нервно-психического, зрительного и мышечного напряжения необходимо проводить сеансы психофизиологической разгрузки и снятия усталости во время регламентированных перерывов, и после окончания рабочего дня.

Для создания оптимальной световой среды, то есть рациональной организации естественного и искусственного освещения помещения и рабочих мест предусматриваются меры для ограничения слепящего действия светопроемов, имеющих высокую яркость и прямых солнечных лучей для обеспечения благоприятного распределения светового потока в помещении и исключения на рабочих поверхностях ярких и темных пятен, засветки экранов посторонним светом.

Это в известной мере достигается путем соответствующей ориентации светопроемов, правильного использования и размещения рабочих мест и солнцезащитных средств.

Для работы с вычислительной техникой рекомендуются помещения с односторонним боковым естественным освещением с северной, северо-восточной ориентацией светопроемов. Рабочие столы следует располагать подальше от окон и таким образом, чтобы оконные проемы находились сбоку от работающих, а естественный свет падал с левой или с правой стороны.

Заключение

В условиях рыночной экономики в Казахстане появился новый вид бизнеса – интернет бизнес. Он коренным образом отличается от обычного несетевого – он развивается так же стремительно, как и сама среда интернет позволяет работать на виртуальном рынке, что в значительной мере снижает издержки предприятий малого бизнеса; позволяет существенно расширить круг потенциальных клиентов. Одним из выгодных для Казахстана свойств этого бизнеса является то, что он чрезвычайно привлекателен для инвесторов, как западных так и казахстанских. А это в данный момент для Казахстана очень важно: экономике сегодня необходимы денежные вливания для развития малого бизнеса, формирования конкурентной среды для крупного бизнеса, который является основой экономики любого развитого государства.

В ходе написания дипломной работы бал разработан электронный магазин компьютерных комплектующих, занимающийся продажей техники.

При разработке архитектуры интернет-магазина, для удобства было обозначено несколько частей: администраторская, клиентская и программная.

Администраторская часть содержит инструменты управления интернет – магазином и включает в себя как общие настройки магазина, так и специальные настройки.

В клиентской части архитектуры разрабатывается максимально удобная и доступная работа потенциального клиента на страницах интернет – магазина. Разработка интерфейса, доступные и понятные диалоговые окна, удобные системы оплаты и доставки товаров.

Программная часть архитектуры интернет – магазина рассматривается как взаимосвязь операционной и серверной части.

Серверная часть содержит в себе размещение интернет магазина на сайте провайдера, поддерживающие технологии, используемые при создании интернет – магазина.

В операционной части рассматривается среда разработки интернет магазина.

Для создания интернет-магазина выбор пал на php и mysql . Это мощная среда для разработки, совместимая со всеми операционными системами и браузерами, не требующая высоких аппаратных средств компьютера, довольно проста в освоении и продолжает развиваться и совершенствоваться. Также он поддерживается подавляющим большинством платных хостингов, что является несомненным плюсом.[38]

Для создания и первоначального тестирования интернет – магазина будет использоваться локальный сервер. Он значительно упростит систему отладки работы интернет – магазина.

На начальном этапе в проекте интернет – магазина будут реализованы такие способы оплаты, как оплата переводом и оплата курьеру наличными. Доставка курьером будет нацелена на работающее население города, имеющего возможность выхода в интернет.

В данном дипломе были определены экономические затраты на создании электронного магазина компьютерных комплектующих по продаже мобильных телефонов. А также рассчитана экономическая эффективность создания данного программного продукта.

На основе произведенных расчетов можно сделать следующий вывод о том, что разрабатывать этот продукт для решения задач рассматриваемого предприятия выгодно.

Совсем не обязательно привлекать сторонние предприятия, которые занимаются созданием программных продуктов, т.к. эти предприятия, во-первых, за свои услуги потребуют высокой оплаты; во-вторых, программный продукт может не решать многих существенных задач интернет-магазина, и руководителю и персоналу все равно придется

дорабатывать программный продукт в соответствии с целями и задачами своего предприятия, что соответственно вызовет дополнительные затраты.

Проанализировав работу электронного магазина, можно прийти к заключении о экономической целесообразности и окупаемости дополнительной услуги – электронного магазина. Так как создавая данную услугу, потенциальным клиентам предоставляется возможность ознакомления с компанией, магазином и предоставляемым ассортиментом товаров. Существование возможности ознакомление с новинками и ожидаемыми поступлениям так же поднимет рейтинг магазина создаст расширение круга постоянных покупателей. Очень удобен тот факт, что в интернет-магазине знакомится с товаром и оставлять заказ будет возможно круглосуточно.

Электронный магазин сможет так же приносить дополнительный доход при продаже продукции через сеть интернет покупателям, находящимся в других городах области. Заинтересовав клиента, относительно невысокими ценами, широким ассортиментом надеясь выйти на общероссийский уровень.

Основные надежды и основания для усилий в области Internet сегодня - постоянный рост объемов продаж через Internet, затрат на рекламу в Сети, все большая доступность Internet и рост его возможностей. Конкретнее - стремительный рост количества Internet-проектов, все новые технологические и технические возможности (баннерныые сети, Internet - магазины, базы данных в Internet, и др.), рост конкуренции провайдеров и как следствие, падение цен на услуги и т.д. Все больше известных фирм активно используют Internet. Этот рынок растет, и, в отличие от других, растет постоянно и очень быстро.[39, с. 34]

В настоящее время сетевые торговцы вынуждены расширять ассортимент товаров, повышать качество предлагаемых услуг, совершенствовать системы оплаты и доставки товаров. Большинство интернет-магазинов предпринимают серьезные шаги по обеспечению реального наличия на складе тех товаров, которые представлены в Web-

витрине, и оптимизации доставки путем использования собственных хранилищ-накопителей в различных регионах.

Список использованной литературы

1. Юрасов А.В. Электронная коммерция: Учебное пособие – М.: Дело, 2003. – 480 с
2. В.В.Гуров «Интернет для бизнеса», М.: ООО «Электронинформ», 1997 г.
3. «Информационные Технологии: Теория и практика рекламы в России» Крылов, «Центр», 1996
4. Климеко С.В., Уразметов В. “Internet. Среда обитания информационного общества”, Протвино, ИВФЭ, 1996.
5. Бизнес план инвестиционного проекта: отечественный и зарубежный опыт. Современная практика и документация. – М.: Финансы и статистика, 1997. – 418с.
6. Технологии электронных коммуникаций том 43 “Международная компьютерная сеть Internet”, Москва, СП “Эко-Трендз” 1995.
7. Кантарь И.Л. “Автоматизированные рабочие места управленческого аппарата”. - М.: 1990.
8. Воронов К. Проблемы оценки инвестиционных проектов, осуществляемых на действующем предприятии// ЭКО. –1996. - №1.
9. Балабанов И.Т. - «Торговля через виртуальный магазин» /«Электронная коммерция»/ 2004г. С.195-197
10. Субботин С. Интернет-коммерция – «за» и «против».// «Рынок ценных бумаг», 2000, № 7.
11. Решетников К.С. “Internet бизнес ”.-М.: Изд-во «Мир», 2001г.-с. 68
12. Холзнер Стивен. Perl: специальный справочник :Пер. с анг. – СПб.: Питер, 2000. – 496с.: ил.
13. Хейл, Бернард Ван. JDBC: Java и базы данных :Пер. с англ. М.,1999.-320с.
14. LAN/Журнал сетевых решений, № 2, 3, 4, 1997.

15. Кристиансен Т., Торкингтон Н. Perl: Библиотека программиста :Пер. с англ.- СПб.: Издательство «Питер», 2000. – 736с.: ил.
16. Симкин Стив, Бартлет Нейл, Лесли Алекс. Программирование на Java. Путеводитель :Пер. с англ. – К. НИПФ «ДиаСофт Лтд», 1996. 736 с.
17. Международный компьютерный еженедельник “Computer World” Россия, № 4, 5, 6, 1997.
18. Хафкемейер Х. "Интернет. Путешествие по всемирной компьютерной сети", 1999г.
19. Корпоративные системы. (Газета для корпоративных пользователей информационных технологий, 2марта № 5 2001г.
20. Грошев С.В. Современный самоучитель профессиональной работы на компьютере, 1998г.
21. Джейсон Мейнджер. Java: основы программирования :Пер. с англ. - К.: Издательская группа BHV,1997.-320с.
22. Эферган М. Java: справочник. – СПб.: Питер, 1998. -448с.: ил.
23. Колесник А.П. Компьютерные системы в управлении финансами. - М.: “Финансы и статистика”, 1996.
24. Курс ЦИТ «Internet-технологии в проектах с пластиковыми карточками», Завалеев, «Центр», 1998.
25. Румянцев «Сам себе WEB – программист», Москва 2001
26. Бизнес план: методические материалы. – М.: Финансы и статистика, 1996. – 80с.
27. Левин "Internet для "чайников", Москва 1996
28. Денисова Л. Анализ инвестиционных проектов// ЭКО. – 1999. - №10.
29. Рассохин Лебедев "World Wide Web - глобальная информационная паутина в сети Internet", Москва 1997.
30. Перри "Секреты World Wide Web", Москва 1996.
31. Хеслоп "HTML с самого начала", СПб: Санкт-Петербург, 1995
32. Kevin Werbach. Перевод: Станислав Малишев. - Краткое пособие по HTML, Москва 1998.

33. Д. Кирсанов , Web-дизайн: «Символ», 1999

34. Журнал «Мир Интернет», №2(29), 2001 г.

35. Соломатин Е. Internet-трейдинг: тенденции, концепции, технологии.// Computerworld, 2000, № 8.

36. Уолл "Использование WWW", Москва 1997

37. Дэниел Эймор «Internet-магазины и закупочная деятельность»/«Электронный бизнес. Эволюция и/или революция» изд. «Вильямс» 2001 г. С. 291-302.

38. Анна Гласман “Маркетинговые принципы построения виртуальных страницInternet” СПб: ДуксНет, 1999г.

39. Галкин С. “Бизнес в Интернет”.- М.: Изд-во "Центр", 2000г

Приложение А

Программный код интернет- магазина компьютерных комплектующих

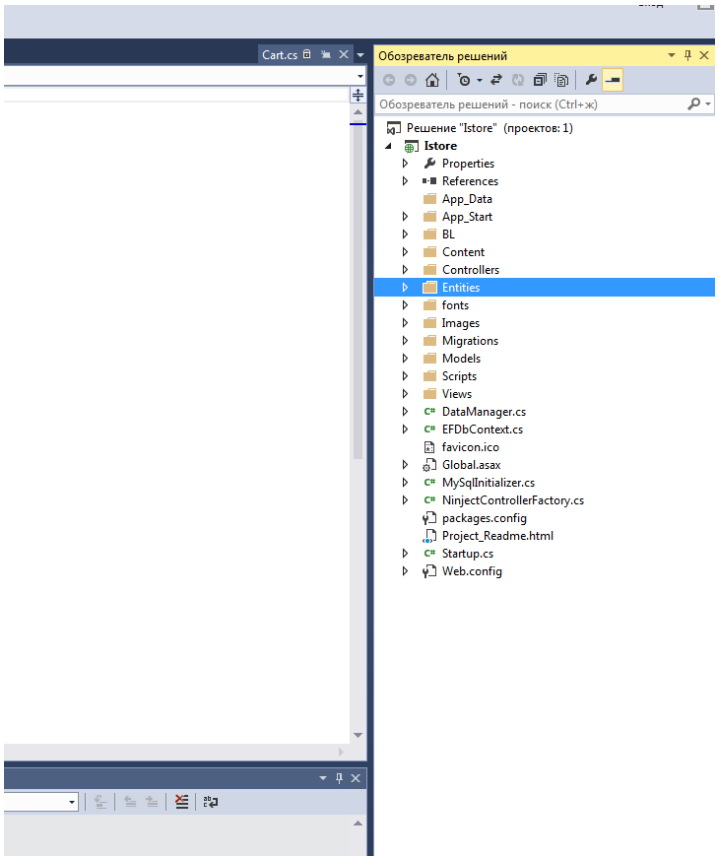

Рисунок 8-Обозреватель решений

```
#region Сборка mscorlib.dll, v4.0.0.0
// C:\Program Files (x86)\Reference
Assemblies\Microsoft\Framework\.NETFramework\v4.5\mscorlib.dll
#endregion

using System;
using System.Runtime.InteropServices;

namespace System.Reflection
{
    // Сводка:
    //     Приказывает компилятору использовать определенный номер версии для ресурса
    //     версии файла Win32. Не требуется, чтобы версия файла Win32 совпадала с номером
    //     версии сборки.
    [AttributeUsage(AttributeTargets.Assembly, Inherited = false)]
    [ComVisible(true)]
    public sealed class AssemblyFileVersionAttribute : Attribute
    {
        // Сводка:
        //     Инициализирует новый экземпляр класса
System.Reflection.AssemblyFileVersionAttribute,
        //     определяющий версию файла.
```

```
        //
        // Параметры:
        //   version:
        //     Версия файла.
        //
        // Исключения:
        //   System.ArgumentNullException:
        //     Параметр version имеет значение null.
        public AssemblyFileVersionAttribute(string version);

        // Сводка:
        //     Возвращает имя ресурса версии файла Win32.
        //
        // Возвращает:
        //     Строка, содержащая имя ресурса версии файла.
        public string Version { get; }
    }
}
using System.Web;
using System.Web.Optimization;

namespace Istore
{
    public class BundleConfig
    {
        // For more information on bundling, visit
http://go.microsoft.com/fwlink/?LinkId=301862
        public static void RegisterBundles(BundleCollection bundles)
        {
            bundles.Add(new ScriptBundle("~/bundles/jquery").Include(
                        "~/Scripts/jquery-{version}.js"));

            bundles.Add(new ScriptBundle("~/bundles/jqueryval").Include(
                        "~/Scripts/jquery.validate*"));

            // Use the development version of Modernizr to develop with and learn from.
Then, when you're
            // ready for production, use the build tool at http://modernizr.com to pick
only the tests you need.
            bundles.Add(new ScriptBundle("~/bundles/modernizr").Include(
                        "~/Scripts/modernizr-*"));

            bundles.Add(new ScriptBundle("~/bundles/bootstrap").Include(
                      "~/Scripts/bootstrap.js",
                      "~/Scripts/respond.js"));

            bundles.Add(new StyleBundle("~/Content/css").Include(
                      "~/Content/bootstrap.css",
                      "~/Content/site.css"));
        }
    }
}
using System.Web;
using System.Web.Mvc;

namespace Istore
{
    public class FilterConfig
    {
        public static void RegisterGlobalFilters(GlobalFilterCollection filters)
        {
            filters.Add(new HandleErrorAttribute());
        }
    }
```

```
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace Istore
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Home", action = "Index", id =
UrlParameter.Optional }
            );
        }
    }
}
using Microsoft.AspNet.Identity;
using Microsoft.Owin;
using Microsoft.Owin.Security.Cookies;
using Owin;

namespace Istore
{
    public partial class Startup
    {
        // For more information on configuring authentication, please visit
http://go.microsoft.com/fwlink/?LinkId=301864
        public void ConfigureAuth(IAppBuilder app)
        {
            // Enable the application to use a cookie to store information for the signed
in user
            app.UseCookieAuthentication(new CookieAuthenticationOptions
            {
                AuthenticationType = DefaultAuthenticationTypes.ApplicationCookie,
                LoginPath = new PathString("/Account/Login")
            });
            // Use a cookie to temporarily store information about a user logging in with
a third party login provider
            app.UseExternalSignInCookie(DefaultAuthenticationTypes.ExternalCookie);

            // Uncomment the following lines to enable logging in with third party login
providers
            //app.UseMicrosoftAccountAuthentication(
            //    clientId: "",
            //    clientSecret: "");

            //app.UseTwitterAuthentication(
            //   consumerKey: "",
            //   consumerSecret: "");

            //app.UseFacebookAuthentication(
            //   appId: "",
            //   appSecret: "");

            //app.UseGoogleAuthentication();
```

```
        }
    }
}using Istore.BL.Interfaces;
using Istore.Entities;
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Web;

namespace Istore.BL.Implementations
{
    // Данный класс предназначен для реализации логики Добавление/Изменение/Удаление
данных в Корзину
    // Реализует интерфейс ICarts

    public class EFCarts: ICarts
    {

        private EFDbContext context;

        public EFCarts(EFDbContext context)
        {
            this.context = context;
        }

        // Список товаров в корзине
        public IEnumerable<Entities.Cart> GetCarts()
        {
            return context.Carts;
        }

        // Корзина по Id
        public Cart GetCartById(int id)
        {
            return context.Carts.FirstOrDefault(x => x.Id == id);
        }

        // Добавление в Корзину
        public void AddToCart(int statusId, string address, string email, int itemId,
string phone, string userId)
        {
            Cart cart = new Cart
            {
                StatusId = statusId,
                Address = address,
                Email = email,
                ItemId = itemId,
                Phone = phone,
                UserId = userId,
                DateCreate = DateTime.Now,
            };
            SaveCart(cart);
        }

        // Сохранение данных
        public void SaveCart(Cart cart)
        {
            if (cart.Id == 0)
                context.Carts.Add(cart);
            else
                context.Entry(cart).State = EntityState.Modified;
            context.SaveChanges();
        }
```

```
        // Изменение данных в корзине
        public void UpdateCart(int id, int statusId, string address, string email, int
itemId, string phone)
        {
            Cart c = context.Carts.FirstOrDefault(x => x.Id == id);
            c.StatusId = statusId;
            c.Address = address;
            c.Email = email;

            c.ItemId = itemId;
            c.Phone = phone;

            SaveCart(c);
        }

        // Удаление данных корзины
        public void DeleteCart(Cart cart)
        {
            context.Carts.Remove(cart);
            context.SaveChanges();
        }


        public void SetStatus(int id, int statusId)
        {
            Cart c = context.Carts.FirstOrDefault(x => x.Id == id);
            c.StatusId = statusId;
            SaveCart(c);
        }
    }
}using Istore.BL.Interfaces;
using Istore.Entities;
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Web;

namespace Istore.BL.Implementations
{
    // Данный класс предназначен для реализации логики Добавление/Изменение/Удаление
данных Характеристики (свойства) товара
    // Реализует интерфейс IDescriptions
    public class EFDescriptions: IDescriptions
    {
        private EFDbContext context;

        public EFDescriptions(EFDbContext context)
        {
            this.context = context;
        }


        // Список характеристик товара
        public IEnumerable<Description> GetDescriptions()
        {
            return context.Descriptions;
        }

        // Характеристики товара
        public Description GetDescriptionById(int id)
        {
            return context.Descriptions.FirstOrDefault(x => x.Id == id);
```

```
        }

        // Добавление характеристики товара
        public void AddDescription(int itemId, int typeId, string descriptions)
        {
            Description description = new Description
            {
                ItemId = itemId,
                TypeId = typeId,
                Descriptions = descriptions
            };
            SaveDescription(description);
        }

        // Сохранение товара
        public void SaveDescription(Description description)
        {
            if (description.Id == 0)
                context.Descriptions.Add(description);
            else
                context.Entry(description).State = EntityState.Modified;
            context.SaveChanges();
        }

        // Изменение данных
        public void UpdateDescription(int id, int itemId, int typeId, string
descriptions)
        {
            Description d = context.Descriptions.FirstOrDefault(x => x.Id == id);
            d.ItemId = itemId;
            d.TypeId = typeId;
            d.Descriptions = descriptions;
            SaveDescription(d);
        }

        // Удаление данных
        public void DeleteDescription(Entities.Description description)
        {
            context.Descriptions.Remove(description);
            context.SaveChanges();
        }
    }
}using Istore.BL.Interfaces;
using Istore.Entities;
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Web;

namespace Istore.BL.Implementations
{
    // Данный класс предназначен для реализации логики Добавление/Изменение/Удаление
данных Справочника Типов характеристик товара
    // Реализует интерфейс IDescriptionTypes
    public class EFDescriptionTypes: IDescriptionTypes
    {

        private EFDbContext context;

        public EFDescriptionTypes(EFDbContext context)
        {
            this.context = context;
        }
```

```
        // Список типов характеристик товара
        public IEnumerable<DescriptionType> GetDescriptionTypes()
        {
            return context.DescriptionTypes;
        }

        // Тип характеристик товара
        public DescriptionType GetDescriptionTypeById(int id)
        {
            return context.DescriptionTypes.FirstOrDefault(x => x.Id == id);
        }

        // Добавление типа характеристик
        public void AddDescriptionType(string name)
        {
            DescriptionType dtype = new DescriptionType
            {
                Name = name
            };
            SaveDescriptionType(dtype);
        }

        // Сохранение данных
        public void SaveDescriptionType(DescriptionType descriptionType)
        {
            if (descriptionType.Id == 0)
                context.DescriptionTypes.Add(descriptionType);
            else
                context.Entry(descriptionType).State = EntityState.Modified;
            context.SaveChanges();
        }

        // Изменение данных
        public void UpdateDescriptionType(int id, string name)
        {
            DescriptionType dtype = context.DescriptionTypes.FirstOrDefault(x => x.Id ==
id);
            dtype.Name = name;
            SaveDescriptionType(dtype);
        }

        // Удаление данных
        public void DeleteDescriptionType(DescriptionType descriptionType)
        {
            context.DescriptionTypes.Remove(descriptionType);
            context.SaveChanges();
        }
    }
}using Istore.BL.Interfaces;
using Istore.Entities;
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Web;

namespace Istore.BL.Implementations
{

    // Данный класс предназначен для реализации логики Добавление/Изменение/Удаление
Товара
    // Реализует интерфейс IItems
    public class EFItems: IItems
```

```
    {

        private EFDbContext context;

        public EFItems(EFDbContext context)
        {
            this.context = context;
        }

        // Список товаров
        public IEnumerable<Item> GetItems()
        {
            return context.Items;
        }

        // Товар по ID
        public Item GetItemById(int id)
        {
            return context.Items.FirstOrDefault(x => x.Id == id);
        }

        // Добавление товара
        public void AddItem(string name, int parentId, double price, int count, string
image, bool isDiscount, double discountPrice, bool active)
        {
            Item item = new Item
            {
                Name = name,
                ParentId = parentId,
                Price = price,
                Count = count,
                Image = image,
                IsDiscount = isDiscount,
                DiscountPrice = discountPrice,
                Active = active,
                DateCreate = DateTime.Now
            };
            SaveItem(item);
        }

        // Сохранение данных
        public void SaveItem(Entities.Item item)
        {
            if (item.Id == 0)
                context.Items.Add(item);
            else
                context.Entry(item).State = EntityState.Modified;
            context.SaveChanges();
        }

        // Изменение данных товара
        public void UpdateItem(int id, string name, int parentId, double price, int
count, string image, bool isDiscount, double discountPrice, bool active)
        {
            Item i = context.Items.FirstOrDefault(x => x.Id == id);
            i.Name = name;
            i.ParentId = parentId;
            i.Price = price;

            i.Count = count;
            i.Image = image;
            i.IsDiscount = isDiscount;
            i.DiscountPrice = discountPrice;
            i.Active = active;
```

```
            SaveItem(i);
        }

        // Удаление данных
        public void DeleteItem(Entities.Item item)
        {

            IEnumerable<Description> descriptions = context.Descriptions.Where(x =>
x.ItemId == item.Id);
            foreach (Description d in descriptions)
            {
                context.Descriptions.Remove(d);
            };

            IEnumerable<Item> subItems = this.GetItems().Where(x => x.ParentId ==
item.Id);
            if (subItems.Count() > 0)
            {
                foreach (Item i in subItems)
                {
                    this.DeleteItem(i);
                }
            };

            context.Items.Remove(item);
            context.SaveChanges();
        }

        // Добавление раздела
        public void AddMainItem(string name)
        {
            Item item = new Item
            {
                Name = name,
                Active = true,
                DateCreate = DateTime.Now
            };
            SaveItem(item);
        }


        public void UpdateMainItem(int id, string name, bool active)
        {
            Item i = context.Items.FirstOrDefault(x => x.Id == id);
            i.Name = name;
            i.Active = active;
            SaveItem(i);
        }

        // Список товаров по категориям
        public FullItemList GetItemListById(int id = 0)
        {
            IEnumerable<Item> mainItemList = this.GetItems().Where(x => x.ParentId == 0);
// все категории товара
            IEnumerable<Item> subItemList = this.GetItems().Where(x => (x.ParentId > 0)
&& (x.Active = true));

            if (id > 0)
            {
                subItemList = subItemList.Where(x => x.ParentId == id);
            };

            FullItemList resultList = new FullItemList {
                MainItemList = mainItemList,
```

```
                SubItemList = subItemList
            };
            return resultList;
        }
    }
}using Istore.BL.Interfaces;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Istore.BL.Implementations
{

    // Данный класс реализует загрузку данных из хранимых процедур БД,
    // реализует логику интерфейса IMyStoredProcedures
    public class EFMyStoredProcedures: IMyStoredProcedures
    {

        private EFDbContext context;
        public EFMyStoredProcedures(EFDbContext context)
        {
            this.context = context;
        }

        public List<T> ExecuteStoredProcedure<T>(string storedProcedureName, params
object[] parameters)
        {
            string storedProcedureCommand = "CALL " + storedProcedureName + "(";
            if (parameters != null)
            {
                List<object> pParameters = parameters.ToList();
                storedProcedureCommand = AddParametersToCommand(storedProcedureCommand,
pParameters);
            };
            storedProcedureCommand += ");";

            return context.Database.SqlQuery<T>(storedProcedureCommand).ToList<T>();
        }

        private static string AddParametersToCommand(string storedProcedureCommand,
List<object> pParameters)
        {
            for (int i = 0; i < pParameters.Count(); i++)
            {
                storedProcedureCommand = AddParameterToCommand(storedProcedureCommand,
pParameters, i);
            }
            return storedProcedureCommand;
        }

        private static string AddParameterToCommand(string storedProcedureCommand,
List<object> pParameters, int i)
        {
            if (pParameters[i].GetType() == typeof(string))
            {
                storedProcedureCommand += "'";
            }

            storedProcedureCommand += (pParameters[i].ToString());

            if (pParameters[i].GetType() == typeof(string))
            {
                storedProcedureCommand += "'";
```

```
            }

            if (i < pParameters.Count - 1)
            {
                storedProcedureCommand += ",";
            }

            return storedProcedureCommand;
        }
    }
}using Istore.BL.Interfaces;
using Istore.Entities;
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Web;

namespace Istore.BL.Implementations
{
    // Данный класс предназначен для реализации логики Добавление/Изменение/Удаление
данных Справочника Статус состояния товара в корзине
    // Реализует интерфейс IStatusCarts
    public class EFStatusCarts: IStatusCarts
    {

        private EFDbContext context;

        public EFStatusCarts (EFDbContext context)
        {
            this.context = context;
        }

        // Список статусов состоянии товара в корзине
        public IEnumerable<StatusCart> GetStatusCarts()
        {
            return context.StatusCarts;
        }

        // Статус состояния товара в корзине по Id
        public StatusCart GetStatusCartById(int id)
        {
            return context.StatusCarts.FirstOrDefault(x => x.Id == id);
        }

        // Добавление нового статуса состояния товара в корзине
        public void AddStatusCart(string name)
        {
            StatusCart sc = new StatusCart
            {
                Name = name
            };
            SaveStatusCart(sc);
        }

        // Сохранить данные
        public void SaveStatusCart(StatusCart statusCart)
        {
            if (statusCart.Id == 0)
                context.StatusCarts.Add(statusCart);
            else
                context.Entry(statusCart).State = EntityState.Modified;
            context.SaveChanges();
        }
```

```
        // Изменить данные
        public void UpdateStatusCart(int id, string name)
        {
            StatusCart sc = context.StatusCarts.FirstOrDefault(x => x.Id == id);
            sc.Name = name;
            SaveStatusCart(sc);
        }

        // Удаление данных
        public void DeleteStatusCart(StatusCart statusCart)
        {
            context.StatusCarts.Remove(statusCart);
            context.SaveChanges();
        }
    }
}using Istore.Entities;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Istore.BL.Interfaces
{
    public interface ICarts
    {
        // Список содержимое корзины
        IEnumerable<Cart> GetCarts();

        // Содержимое Корзины по Id
        Cart GetCartById(int id);

        // Добавить в Корзину
        void AddToCart(int statusId, string address, string email, int itemId, string
phone, string userId );

        // Сохранить данные Корзины
        void SaveCart(Cart cart);

        // Изменить данные Корзины
        void UpdateCart(int id, int statusId, string address, string email, int itemId,
string phone);

        // Удалить данные корзины
        void DeleteCart(Cart cart);

        void SetStatus(int id, int statusId);
    }
}
using Istore.Entities;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Istore.BL.Interfaces
{

    public interface IDescriptions
    {
        // Характеристики товара
        IEnumerable<Description> GetDescriptions();
```

```
        // Характеристики по Id
        Description GetDescriptionById(int id);

        // Добавить Характеристику
        void AddDescription(int itemId, int typeId, string descriptions);

        // Сохранить данные Характеристики
        void SaveDescription(Description description);

        // Изменить данные Характеристики
        void UpdateDescription(int id, int itemId, int typeId, string descriptions);

        // Удалить Характеристики
        void DeleteDescription(Description description);

    }
}
using Istore.Entities;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Istore.BL.Interfaces
{


    public interface IDescriptionTypes
    {
        IEnumerable<DescriptionType> GetDescriptionTypes();

        // Характеристики по Id
        DescriptionType GetDescriptionTypeById(int id);

        // Добавить Характеристику
        void AddDescriptionType(string name);

        // Сохранить данные Характеристики
        void SaveDescriptionType(DescriptionType descriptionType);

        // Изменить данные Характеристики
        void UpdateDescriptionType(int id, string name);

        // Удалить Характеристики
        void DeleteDescriptionType(DescriptionType descriptionType);
    }
}
using Istore.Entities;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Istore.BL.Interfaces
{
    public interface IItems
    {

        // Список товаров
        IEnumerable<Item> GetItems();

        // Список товаров по выбранной категории
```

```
        FullItemList GetItemListById(int id = 0);

        // Товар по Id
        Item GetItemById(int id);

        // Добавить Товар
        void AddItem(string name, int parentId, double price, int count, string image,
bool isDiscount, double discountPrice, bool active);

        // Сохранить данные Товар
        void SaveItem(Item item);

        // Изменить данные Товар
        void UpdateItem(int id, string name, int parentId, double price, int count,
string image, bool isDiscount, double discountPrice, bool active);

        // Удалить Товар
        void DeleteItem(Item item);

        void AddMainItem(string name);
        void UpdateMainItem(int id, string name, bool active);



    }
}using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Istore.BL.Interfaces
{
    public interface IMyStoredProcedures
    {
        // Загрузка данных с помощью хранимой процедуры
        List<T> ExecuteStoredProcedure<T>(string storedProcedureName, params object[]
parameters);
    }
}
using Istore.Entities;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Istore.BL.Interfaces
{
    public interface IStatusCarts
    {
        IEnumerable<StatusCart> GetStatusCarts();

        // Статус корзины по Id
        StatusCart GetStatusCartById(int id);

        // Добавить Статус корзины
        void AddStatusCart(string name);

        // Сохранить Статус корзины
        void SaveStatusCart(StatusCart statusCart);

        // Изменить Статус корзины
        void UpdateStatusCart(int id, string name);
```

```
        // Удалить Статус корзины
        void DeleteStatusCart(StatusCart statusCart);
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Claims;
using System.Threading.Tasks;
using System.Web;
using System.Web.Mvc;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.EntityFramework;
using Microsoft.Owin.Security;
using Istore.Models;

namespace Istore.Controllers
{
    [Authorize]
    public class AccountController : Controller
    {
        public AccountController()
            : this(new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(new
ApplicationDbContext())))
        {
        }

        public AccountController(UserManager<ApplicationUser> userManager)
        {
            UserManager = userManager;
        }

        public UserManager<ApplicationUser> UserManager { get; private set; }

        //
        // GET: /Account/Login
        [AllowAnonymous]
        public ActionResult Login(string returnUrl)
        {
            ViewBag.ReturnUrl = returnUrl;
            return View();
        }

        //
        // POST: /Account/Login
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)
        {
            if (ModelState.IsValid)
            {
                var user = await UserManager.FindAsync(model.UserName, model.Password);
                if (user != null)
                {
                    await SignInAsync(user, model.RememberMe);
                    return RedirectToLocal(returnUrl);
                }
                else
                {
                    ModelState.AddModelError("", "Invalid username or password.");
                }
            }
```

```
            // If we got this far, something failed, redisplay form
            return View(model);
        }

        //
        // GET: /Account/Register
        [AllowAnonymous]
        public ActionResult Register()
        {
            return View();
        }

        //
        // POST: /Account/Register
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> Register(RegisterViewModel model)
        {
            if (ModelState.IsValid)
            {
                var user = new ApplicationUser() { UserName = model.UserName };
                var result = await UserManager.CreateAsync(user, model.Password);
                if (result.Succeeded)
                {
                    await SignInAsync(user, isPersistent: false);
                    return RedirectToAction("Index", "Home");
                }
                else
                {
                    AddErrors(result);
                }
            }

            // If we got this far, something failed, redisplay form
            return View(model);
        }

        //
        // POST: /Account/Disassociate
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> Disassociate(string loginProvider, string
providerKey)
        {
            ManageMessageId? message = null;
            IdentityResult result = await
UserManager.RemoveLoginAsync(User.Identity.GetUserId(), new UserLoginInfo(loginProvider,
providerKey));
            if (result.Succeeded)
            {
                message = ManageMessageId.RemoveLoginSuccess;
            }
            else
            {
                message = ManageMessageId.Error;
            }
            return RedirectToAction("Manage", new { Message = message });
        }

        //
        // GET: /Account/Manage
        public ActionResult Manage(ManageMessageId? message)
```

```
        {
            ViewBag.StatusMessage =
                message == ManageMessageId.ChangePasswordSuccess ? "Your password has
been changed."
                : message == ManageMessageId.SetPasswordSuccess ? "Your password has been
set."
                : message == ManageMessageId.RemoveLoginSuccess ? "The external login was
removed."
                : message == ManageMessageId.Error ? "An error has occurred."
                : "";
            ViewBag.HasLocalPassword = HasPassword();
            ViewBag.ReturnUrl = Url.Action("Manage");
            return View();
        }

        //
        // POST: /Account/Manage
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> Manage(ManageUserViewModel model)
        {
            bool hasPassword = HasPassword();
            ViewBag.HasLocalPassword = hasPassword;
            ViewBag.ReturnUrl = Url.Action("Manage");
            if (hasPassword)
            {
                if (ModelState.IsValid)
                {
                    IdentityResult result = await
UserManager.ChangePasswordAsync(User.Identity.GetUserId(), model.OldPassword,
model.NewPassword);
                    if (result.Succeeded)
                    {
                        return RedirectToAction("Manage", new { Message =
ManageMessageId.ChangePasswordSuccess });
                    }
                    else
                    {
                        AddErrors(result);
                    }
                }
            }
            else
            {
                // User does not have a password so remove any validation errors caused
by a missing OldPassword field
                ModelState state = ModelState["OldPassword"];
                if (state != null)
                {
                    state.Errors.Clear();
                }

                if (ModelState.IsValid)
                {
                    IdentityResult result = await
UserManager.AddPasswordAsync(User.Identity.GetUserId(), model.NewPassword);
                    if (result.Succeeded)
                    {
                        return RedirectToAction("Manage", new { Message =
ManageMessageId.SetPasswordSuccess });
                    }
                    else
                    {
                        AddErrors(result);
```

```
                }
            }
        }

            // If we got this far, something failed, redisplay form
            return View(model);
        }

        //
        // POST: /Account/ExternalLogin
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public ActionResult ExternalLogin(string provider, string returnUrl)
        {
            // Request a redirect to the external login provider
            return new ChallengeResult(provider, Url.Action("ExternalLoginCallback",
"Account", new { ReturnUrl = returnUrl }));
        }

        //
        // GET: /Account/ExternalLoginCallback
        [AllowAnonymous]
        public async Task<ActionResult> ExternalLoginCallback(string returnUrl)
        {
            var loginInfo = await AuthenticationManager.GetExternalLoginInfoAsync();
            if (loginInfo == null)
            {
                return RedirectToAction("Login");
            }

            // Sign in the user with this external login provider if the user already has
a login
            var user = await UserManager.FindAsync(loginInfo.Login);
            if (user != null)
            {
                await SignInAsync(user, isPersistent: false);
                return RedirectToLocal(returnUrl);
            }
            else
            {
                // If the user does not have an account, then prompt the user to create
an account
                ViewBag.ReturnUrl = returnUrl;
                ViewBag.LoginProvider = loginInfo.Login.LoginProvider;
                return View("ExternalLoginConfirmation", new
ExternalLoginConfirmationViewModel { UserName = loginInfo.DefaultUserName });
            }
        }

        //
        // POST: /Account/LinkLogin
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult LinkLogin(string provider)
        {
            // Request a redirect to the external login provider to link a login for the
current user
            return new ChallengeResult(provider, Url.Action("LinkLoginCallback",
"Account"), User.Identity.GetUserId());
        }

        //
        // GET: /Account/LinkLoginCallback
```

```
        public async Task<ActionResult> LinkLoginCallback()
        {
            var loginInfo = await
AuthenticationManager.GetExternalLoginInfoAsync(XsrfKey, User.Identity.GetUserId());
            if (loginInfo == null)
            {
                return RedirectToAction("Manage", new { Message = ManageMessageId.Error
});
            }
            var result = await UserManager.AddLoginAsync(User.Identity.GetUserId(),
loginInfo.Login);
            if (result.Succeeded)
            {
                return RedirectToAction("Manage");
            }
            return RedirectToAction("Manage", new { Message = ManageMessageId.Error });
        }

        //
        // POST: /Account/ExternalLoginConfirmation
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult>
ExternalLoginConfirmation(ExternalLoginConfirmationViewModel model, string returnUrl)
        {
            if (User.Identity.IsAuthenticated)
            {
                return RedirectToAction("Manage");
            }

            if (ModelState.IsValid)
            {
                // Get the information about the user from the external login provider
                var info = await AuthenticationManager.GetExternalLoginInfoAsync();
                if (info == null)
                {
                    return View("ExternalLoginFailure");
                }
                var user = new ApplicationUser() { UserName = model.UserName };
                var result = await UserManager.CreateAsync(user);
                if (result.Succeeded)
                {
                    result = await UserManager.AddLoginAsync(user.Id, info.Login);
                    if (result.Succeeded)
                    {
                        await SignInAsync(user, isPersistent: false);
                        return RedirectToLocal(returnUrl);
                    }
                }
                AddErrors(result);
            }

            ViewBag.ReturnUrl = returnUrl;
            return View(model);
        }

        //
        // POST: /Account/LogOff
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult LogOff()
        {
            AuthenticationManager.SignOut();
```

```
            return RedirectToAction("Index", "Home");
        }

        //
        // GET: /Account/ExternalLoginFailure
        [AllowAnonymous]
        public ActionResult ExternalLoginFailure()
        {
            return View();
        }

        [ChildActionOnly]
        public ActionResult RemoveAccountList()
        {
            var linkedAccounts = UserManager.GetLogins(User.Identity.GetUserId());
            ViewBag.ShowRemoveButton = HasPassword() || linkedAccounts.Count > 1;
            return (ActionResult)PartialView("_RemoveAccountPartial", linkedAccounts);
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing && UserManager != null)
            {
                UserManager.Dispose();
                UserManager = null;
            }
            base.Dispose(disposing);
        }

        #region Helpers
        // Used for XSRF protection when adding external logins
        private const string XsrfKey = "XsrfId";

        private IAuthenticationManager AuthenticationManager
        {
            get
            {
                return HttpContext.GetOwinContext().Authentication;
            }
        }

        private async Task SignInAsync(ApplicationUser user, bool isPersistent)
        {
            AuthenticationManager.SignOut(DefaultAuthenticationTypes.ExternalCookie);
            var identity = await UserManager.CreateIdentityAsync(user,
DefaultAuthenticationTypes.ApplicationCookie);
            AuthenticationManager.SignIn(new AuthenticationProperties() { IsPersistent =
isPersistent }, identity);
        }

        private void AddErrors(IdentityResult result)
        {
            foreach (var error in result.Errors)
            {
                ModelState.AddModelError("", error);
            }
        }

        private bool HasPassword()
        {
            var user = UserManager.FindById(User.Identity.GetUserId());
            if (user != null)
            {
                return user.PasswordHash != null;
```

```
            }
            return false;
        }

        public enum ManageMessageId
        {
            ChangePasswordSuccess,
            SetPasswordSuccess,
            RemoveLoginSuccess,
            Error
        }

        private ActionResult RedirectToLocal(string returnUrl)
        {
            if (Url.IsLocalUrl(returnUrl))
            {
                return Redirect(returnUrl);
            }
            else
            {
                return RedirectToAction("Index", "Home");
            }
        }

        private class ChallengeResult : HttpUnauthorizedResult
        {
            public ChallengeResult(string provider, string redirectUri)
                : this(provider, redirectUri, null)
            {
            }

            public ChallengeResult(string provider, string redirectUri, string userId)
            {
                LoginProvider = provider;
                RedirectUri = redirectUri;
                UserId = userId;
            }

            public string LoginProvider { get; set; }
            public string RedirectUri { get; set; }
            public string UserId { get; set; }

            public override void ExecuteResult(ControllerContext context)
            {
                var properties = new AuthenticationProperties() { RedirectUri =
RedirectUri };
                if (UserId != null)
                {
                    properties.Dictionary[XsrfKey] = UserId;
                }
                context.HttpContext.GetOwinContext().Authentication.Challenge(properties,
LoginProvider);
            }
        }
        #endregion
    }
}using Istore.Entities;
using Istore.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
```

```
namespace Istore.Controllers
{
    public class CartController : Controller
    {

        private DataManager dataManager;

        public CartController(DataManager dataManager)
        {
            this.dataManager = dataManager;
        }

        //Контроллер - Корзина
        // GET: /Cart/
        [Authorize]
        public ActionResult Index()
        {
            IEnumerable<Item> mainItemList = dataManager.items.GetItems().Where(x =>
x.ParentId == 0); // все категории товара
            ViewBag.mainItemList = mainItemList.ToList();

            IEnumerable<FullCartList> cartList =
dataManager.myStoredProcedures.ExecuteStoredProcedure<FullCartList>("sp_cart_list",
User.Identity.Name);

            ViewBag.CartList = cartList; // Список товаров в корзине
            return View();
        }

        #region Добавление товара в корзину
        [Authorize]
        public ActionResult AddToCart(int id = 0)
        {
            if (id == 0)
            {
                return HttpNotFound();
            };

            IEnumerable<Item> mainItemList = dataManager.items.GetItems().Where(x =>
x.ParentId == 0); // все категории товара

            ViewBag.mainItemList = mainItemList.ToList();
            Item subItem = dataManager.items.GetItemById(id);
            ViewBag.SubItem = subItem;

            CartModel model = new CartModel();
            model.ItemId = id;
            return View(model);
        }

        [HttpPost]
        [Authorize]
        public ActionResult AddToCart(int id, CartModel model)
        {

            IEnumerable<Item> mainItemList = dataManager.items.GetItems().Where(x =>
x.ParentId == 0); // все категории товара
            ViewBag.mainItemList = mainItemList.ToList();
            Item subItem = dataManager.items.GetItemById(id);
            ViewBag.SubItem = subItem;

            if (ModelState.IsValid)
            {
                int statusId = 1; // Статус Добавление в корзину
```

```
                dataManager.carts.AddToCart(statusId, model.Address, model.Email,
model.ItemId, model.Phone, User.Identity.Name);
                return RedirectToAction("index", "Home");
            }
            return View(model);
        }

        #endregion

        #region Подать заявку на покупку
        [Authorize]
        public ActionResult ToRequest(int id)
        {
            Cart model = dataManager.carts.GetCartById(id);
            return View(model);
        }

        [HttpPost]
        [Authorize]
        public ActionResult ToRequest( Cart model)
        {
            if (ModelState.IsValid)
            {
                int statusId = 2; // Статус Подать заявку
                dataManager.carts.SetStatus(model.Id, statusId);
                return Json(new { status = true }, JsonRequestBehavior.AllowGet);
            };
            return View(model);
        }
        #endregion


        #region Удаление товара из корзины
        [Authorize]
        public ActionResult DeleteItemInCart()
        {
            return View();
        }

        [HttpPost]
        [Authorize]
        public ActionResult DeleteItemInCart(int id)
        {
            Cart model = dataManager.carts.GetCartById(id);
            if (ModelState.IsValid)
            {
                dataManager.carts.DeleteCart(model);
                return Json(new { status = true }, JsonRequestBehavior.AllowGet);
            }
            return View();
        }
        #endregion


        #region Прием Заявки
        [Authorize]
        public ActionResult RequestView(int statusId=2)
        {
            IEnumerable<FullCartList> cartList =
dataManager.myStoredProcedures.ExecuteStoredProcedure<FullCartList>("sp_cart_list_with_st
atus", statusId);
            ViewBag.CartList = cartList; // Список товаров в корзине
            ViewBag.MenuId = 3;
            return View();
```

```
        }
        #endregion


        #region Принять заявку
        [Authorize]
        public ActionResult RequestFin(int id)
        {
            Cart model = dataManager.carts.GetCartById(id);
            return View(model);
        }

        [HttpPost]
        [Authorize]
        public ActionResult RequestFin(Cart model)
        {
            if (ModelState.IsValid)
            {
                int statusId = 3; // Статус Подать заявку
                dataManager.carts.SetStatus(model.Id, statusId);
                return Json(new { status = true }, JsonRequestBehavior.AllowGet);
            };
            return View(model);
        }
        #endregion
    }
}using Istore.Entities;
using Istore.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Istore.Controllers
{
    public class DescriptionController : Controller
    {

        private DataManager dataManager;

        public DescriptionController(DataManager dataManager)
        {
            this.dataManager = dataManager;
        }

        //
        // GET: /Description/

        // характеристики товара
        [Authorize]
        public ActionResult Index(int id = 0)
        {
            if (id == 0)
            {
                return HttpNotFound();
            }
            Item subItem = dataManager.items.GetItemById(id);

            ViewBag.SubItemName = subItem.Name;
            ViewBag.Id = id;
```

```
            IEnumerable<FullDescriptionList> descriptionList =
dataManager.myStoredProcedures.ExecuteStoredProcedure<FullDescriptionList>("sp_descriptio
n_list", id);
            ViewBag.DescriptionList = descriptionList; // Вывод характеристики товара
            return View();
        }


        // Для Dropdownlist - Список описании товара
        protected SelectList getDescriptionTypeList()
        {
            IEnumerable<DescriptionType> model =
dataManager.descriptionTypes.GetDescriptionTypes();;
            IEnumerable<SelectListItem> items = model.Select(x => new SelectListItem {
Value = x.Id.ToString(), Text = x.Name });
            return new SelectList(items, "Value", "Text");
        }

        #region Добавление характеристики товара
        [Authorize]
        public ActionResult AddDescription(int id)
        {
            if (id == 0)
            {
                return HttpNotFound();
            };

            DescriptionModel model = new DescriptionModel();
            model.ItemId = id; // товар
            model.typeList = getDescriptionTypeList(); // список описании
            return View(model);
        }

        [HttpPost]
        [Authorize]
        public ActionResult AddDescription(DescriptionModel model)
        {
            if (ModelState.IsValid)
            {
                dataManager.descriptions.AddDescription(model.ItemId, model.TypeId,
model.Descriptions);
                return Json(new { status = true }, JsonRequestBehavior.AllowGet);
            }
            return View(model);
        }
        #endregion



        #region Редактирование характеристики товара
        [Authorize]
        public ActionResult UpdateDescription(int id)
        {
            Description model = dataManager.descriptions.GetDescriptionById(id);
            ViewBag.typeList = getDescriptionTypeList(); // список описании
            //model.ItemId = id; // товар
            return View(model);
        }

        [HttpPost]
        [Authorize]
        public ActionResult UpdateDescription(Description model)
        {
```

```
            ViewBag.typeList = getDescriptionTypeList(); // список описании
            if (ModelState.IsValid)
            {
                dataManager.descriptions.UpdateDescription(model.Id, model.ItemId,
model.TypeId, model.Descriptions);
                return Json(new { status = true }, JsonRequestBehavior.AllowGet);
            };
            return View(model);
        }
        #endregion


        #region Удаление данных товара
        [Authorize]
        public ActionResult DeleteDescription()
        {
            return View();
        }

        [Authorize]
        [HttpPost]
        public ActionResult DeleteDescription(int id)
        {
            Description model = dataManager.descriptions.GetDescriptionById(id);
            if (ModelState.IsValid)
            {
                dataManager.descriptions.DeleteDescription(model);
                return Json(new { status = true }, JsonRequestBehavior.AllowGet);
            }
            return View();
        }


        #endregion
    }
}using Istore.Entities;
using Istore.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Istore.Controllers
{
    [Authorize]
    public class DictionaryController : Controller
    {

        private DataManager dataManager;

        public DictionaryController(DataManager dataManager)
        {
            this.dataManager = dataManager;
        }

        //
        // GET: /Dictionary/

        // Страница Справочники

        public ActionResult Index()
        {
            ViewBag.MenuId = 4;
```

```
            return View();
        }

        // Страница Справочники. Описание товаров

        public ActionResult DescriptionTypes()
        {
            IEnumerable<DescriptionType> descTypeList =
dataManager.descriptionTypes.GetDescriptionTypes();
            ViewBag.descTypeList = descTypeList.ToList();
            return View();
        }


        #region Добавление типов описании товаров
        public ActionResult AddDescriptionTypes()
        {
            DescriptionTypeModel model = new DescriptionTypeModel();
            return View(model);
        }

        [HttpPost]
        public ActionResult AddDescriptionTypes(DescriptionTypeModel model)
        {
            if (ModelState.IsValid)
            {
                dataManager.descriptionTypes.AddDescriptionType(model.Name);
                return Json(new { status = true }, JsonRequestBehavior.AllowGet);
            }
            return View(model);
        }
        #endregion

        #region Изменение типов описании товаров
        public ActionResult EditDescriptionTypes(int id = 0)
        {
            if (id == 0)
            {
                return HttpNotFound();
            }
            DescriptionType model =
dataManager.descriptionTypes.GetDescriptionTypeById(id);
            return View(model);
        }

        [HttpPost]
        public ActionResult EditDescriptionTypes(DescriptionType model)
        {
            if (ModelState.IsValid)
            {
                dataManager.descriptionTypes.UpdateDescriptionType(model.Id, model.Name);
                return Json(new { status = true }, JsonRequestBehavior.AllowGet);
            }
            return View(model);
        }
        #endregion

        #region Удаление тип описании товаров
        public ActionResult DeleteDescriptionTypes()
        {
            return View();
        }

        [HttpPost]
```

```
        public ActionResult DeleteDescriptionTypes(int id)
        {
            DescriptionType model =
dataManager.descriptionTypes.GetDescriptionTypeById(id);
            if (ModelState.IsValid)
            {
                dataManager.descriptionTypes.DeleteDescriptionType(model);
                return Json(new { status = true }, JsonRequestBehavior.AllowGet);
            }
            return View();
        }


        #endregion


    }
}using Istore.Entities;
using Istore.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Istore.Controllers
{
    [Authorize]
    public class ItemController : Controller
    {

        private DataManager dataManager;

        public ItemController(DataManager dataManager)
        {
            this.dataManager = dataManager;
        }

        //
        // GET: /Item/
        public ActionResult Index()
        {

            return View();
        }

        #region Форма Добавление раздела товара
        public ActionResult AddMainItem()
        {
            MainItemModel model = new MainItemModel();
            return View(model);
        }

        [HttpPost]
        public ActionResult AddMainItem(MainItemModel model)
        {
            if (ModelState.IsValid)
            {
                dataManager.items.AddMainItem(model.Name);
                return Json(new { status = true }, JsonRequestBehavior.AllowGet);
            }
            return View(model);
        }
        #endregion
```

```
        #region Форма Изменение раздела товара
        public ActionResult EditMainItem(int id = 0)
        {
            if (id == 0)
            {
                return HttpNotFound();
            }
            Item model = dataManager.items.GetItemById(id);
            return View(model);
        }

        [HttpPost]
        public ActionResult EditMainItem(Item model)
        {
            if (ModelState.IsValid)
            {
                dataManager.items.UpdateMainItem(model.Id, model.Name, model.Active);
                return Json(new { status = true }, JsonRequestBehavior.AllowGet);
            }
            return View(model);
        }
        #endregion

        #region Удаление раздела
        public ActionResult DeleteItem()
        {
            return View();
        }

        [HttpPost]
        public ActionResult DeleteItem(int id)
        {
            Item model = dataManager.items.GetItemById(id);
            if (ModelState.IsValid)
            {
                dataManager.items.DeleteItem(model);
                return Json(new { status = true }, JsonRequestBehavior.AllowGet);
            }
            return View();
        }


        #endregion


    }
}using Istore.Entities;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Istore.Controllers
{

    // Управление товаром
    [Authorize]
    public class ManageController : Controller
    {
        //
        // GET: /Manage/
```

```
        // Страница упр. магазином

        private DataManager dataManager;

        public ManageController(DataManager dataManager)
        {
            this.dataManager = dataManager;
        }

        public ActionResult Index()
        {
            IEnumerable<Item> itemList = dataManager.items.GetItems().Where(x =>
x.ParentId == 0);
            ViewBag.itemList = itemList.ToList();
            ViewBag.MenuId = 0;
            return View();
        }
       }
}using Istore.Entities;
using Istore.Models;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Istore.Controllers
{
    // Контроллер: Товары и Комплектующие

    [Authorize]
    public class SubItemController : Controller
    {

        private DataManager dataManager;

        public SubItemController(DataManager dataManager)
        {
            this.dataManager = dataManager;
        }

        //
        // GET: /SubItem/
        // Список товаров по категориям
        public ActionResult Index(int id = 0)
        {
            IEnumerable<Item> mainItemList = dataManager.items.GetItems().Where(x =>
x.ParentId == 0); // все категории товара
            IEnumerable<Item> subItemList = dataManager.items.GetItems().Where(x =>
x.ParentId > 0);

            ViewBag.MainItemName = "Все товары"; // категория по умолчанию
            ViewBag.Id = id;
            if (id > 0)
            {
                subItemList = subItemList.Where(x=> x.ParentId == id);
                ViewBag.MainItemName = mainItemList.Where(x => x.Id == id).First().Name;
// Наименование категории

            };


            ViewBag.mainItemList = mainItemList.ToList();
```

```
            ViewBag.subItemList = subItemList.ToList();
            ViewBag.MenuId = 1;
            return View();
        }

        // Для Dropdownlist - Категория товара
        protected SelectList getItemList()
        {
            IEnumerable<Item> model = dataManager.items.GetItems().Where(x=>x.ParentId ==
0);
            IEnumerable<SelectListItem> items = model.Select(x => new SelectListItem {
Value = x.Id.ToString(), Text = x.Name });
            return new SelectList(items, "Value", "Text");
        }

        // Сохранение файла
        protected string GetSaveFileName(HttpPostedFileBase file)
        {
            string pic = "";
            if (file != null)
            {
                pic = System.IO.Path.GetFileName(file.FileName);
                string path = System.IO.Path.Combine(
                                        Server.MapPath("~/images/items/"), pic);
                // загрузка файла
                file.SaveAs(path);

                using (MemoryStream ms = new MemoryStream())
                {
                    file.InputStream.CopyTo(ms);
                    byte[] array = ms.GetBuffer();
                }
            };
            return pic;
        }

        #region Добавление товара
        public ActionResult AddSubItem(int id)
        {
            SubItemModel model = new SubItemModel();
            model.itemList = getItemList();
            model.ParentId = id;
            model.Active = true;
            ViewBag.Id = id;
            return View(model);
        }

        [HttpPost]
        public ActionResult AddSubItem(SubItemModel model, HttpPostedFileBase file)
        {
            model.itemList = getItemList();
            ViewBag.Id = model.ParentId;
            if (ModelState.IsValid)
            {
                string sImageName = GetSaveFileName(file);

                dataManager.items.AddItem(model.Name, model.ParentId, model.Price,
model.Count, sImageName, false, model.DiscountPrice, model.Active);
                return RedirectToAction("index", "SubItem", new { id = model.ParentId });
            }
            return View(model);
        }
        #endregion
```

```
        #region Редактирование данных товара
        public ActionResult UpdateSubItem(int id)
        {

            Item model = dataManager.items.GetItemById(id);
            ViewBag.itemList = getItemList();
            return View(model);
        }

        [HttpPost]
        public ActionResult UpdateSubItem(Item model, HttpPostedFileBase file)
        {
            ViewBag.itemList = getItemList();
            if (ModelState.IsValid)
            {

                string sImageName = GetSaveFileName(file);
                if (sImageName == "")
                {
                    sImageName = model.Image;
                }
                dataManager.items.UpdateItem(model.Id, model.Name, model.ParentId,
model.Price, model.Count, sImageName, model.IsDiscount, model.DiscountPrice,
model.Active);
                return RedirectToAction("index", "SubItem", new { id = model.ParentId });
            };
            return View(model);
        }
        #endregion


        #region Удаление данных товара
        public ActionResult DeleteSubItem()
        {
            return View();
        }

        [HttpPost]
        public ActionResult DeleteSubItem(int id)
        {
            Item model = dataManager.items.GetItemById(id);
            if (ModelState.IsValid)
            {
                dataManager.items.DeleteItem(model);
                return Json(new { status = true }, JsonRequestBehavior.AllowGet);
            }
            return View();
        }
        #endregion

    }
}using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Istore.Entities
{
    // Корзина
    public class Cart
    {
        public int Id { get; set; }

        // Ссылка на Статус Корзины
```

```
        public int StatusId { get; set; }

        // Ссылка на Пользователь
        public string UserId { get; set; }

        // Дата создание
        public DateTime? DateCreate { get; set; }

        // Адрес доставки
        public string Address { get; set; }

        // Электронная почта
        public string Email { get; set; }

        // Ссылка на товар
        public int ItemId { get; set; }

        // Контакты
        public string Phone { get; set; }
    }
}using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Istore.Entities
{
    // Характеристики товара
    public class Description
    {
        public int Id { get; set; }

        // Ссылка на товар
        public int ItemId { get; set; }

        // Ссылка на тип характеристики
        public int TypeId { get; set; }

        // Описание товара
        public string Descriptions { get; set; }


    }
}using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Istore.Entities
{
    // Справочник: Типы характеристик товаров
    public class DescriptionType
    {
        public int Id { get; set; }

        // Название
        public string Name { get; set; }
    }
}using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Istore.Entities
```

```
{
    // Список товаров в корзине
    public class FullCartList
    {
        public int CartId { get; set; } // Id
        public string ItemName { get; set; } // Наименование товара

        public string Image { get; set; } // Изображение

        public double Price { get; set; }
        public DateTime DateCreate { get; set; } // Дата создание

        public int StatusId { get; set; } // Статус

        public string Address {get; set;} //Адрес

        public string Phone {get;set;} // Контакты
        public string Email {get; set;} // Эл. почта


     }
}using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Istore.Entities
{
    // Используется для вывода данных характеристики товара
    public class FullDescriptionList
    {
        public int Id { get; set; }

        public string DescriptionTypeName { get; set; } // название типа характеристики

        public int TypeId { get; set; } // тип характеристики

        public int ItemId { get; set; } // Id товара

        public string ItemName { get; set; }  // Наименование товара

        public string Descriptions { get; set; } // Описание характеристики товара


    }
}using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Istore.Entities
{
    // Список товаров по категориям
    public class FullItemList
    {
        public IEnumerable<Item> MainItemList { get; set; } // все категории товара
        public IEnumerable<Item> SubItemList { get; set; } // товары
    }
}using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Istore.Entities
```

```
{
    // Товары (комплектующие)
    public class Item
    {
        // Идентификатор
        public int Id { get; set; }

        // Наименование товара
        public string Name { get; set; }

        // Раздел
        public int ParentId { get; set; }

        // Цена
        public double Price { get; set; }

        // Количество товаров
        public int Count { get; set; }

        // Название изображении товара
        public string Image { get; set; }

        // Акционный товар?
        public bool IsDiscount { get; set; }

        // Цена по акции
        public double DiscountPrice { get; set; }

        // Активность
        public bool Active { get; set; }

        // Дата создание
        public DateTime? DateCreate { get; set; }
    }
}using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Istore.Entities
{
    // Статус Корзины
    public class StatusCart
    {
        public int Id { get; set; }

        // Название
        public string Name { get; set; }
    }
}namespace Istore.Migrations
{
    using System;
    using System.Data.Entity;
    using System.Data.Entity.Migrations;
    using System.Linq;

    internal sealed class Configuration :
DbMigrationsConfiguration<Istore.Models.ApplicationDbContext>
    {
        public Configuration()
        {
            AutomaticMigrationsEnabled = false;

            // register mysql code generator
```

```
            SetSqlGenerator("MySql.Data.MySqlClient", new
MySql.Data.Entity.MySqlMigrationSqlGenerator());

            SetHistoryContextFactory("MySql.Data.MySqlClient", (conn, schema) => new
MySqlHistoryContext(conn, schema));

        }

        protected override void Seed(Istore.Models.ApplicationDbContext context)
        {
            //  This method will be called after migrating to the latest version.

            //  You can use the DbSet<T>.AddOrUpdate() helper extension method
            //  to avoid creating duplicate seed data. E.g.
            //
            //    context.People.AddOrUpdate(
            //      p => p.FullName,
            //      new Person { FullName = "Andrew Peters" },
            //      new Person { FullName = "Brice Lambson" },
            //      new Person { FullName = "Rowan Miller" }
            //    );
            //
        }
    }
}
using System;
using System.Collections.Generic;
using System.Data.Common;
using System.Data.Entity;
using System.Data.Entity.Migrations.History;
using System.Linq;
using System.Web;

namespace Istore.Migrations
{
    public class MySqlHistoryContext : HistoryContext
    {
        public MySqlHistoryContext(DbConnection connection, string defaultSchema)
            : base(connection, defaultSchema)
        {

        }

        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);
            modelBuilder.Entity<HistoryRow>().Property(h =>
h.MigrationId).HasMaxLength(100).IsRequired();
            modelBuilder.Entity<HistoryRow>().Property(h =>
h.ContextKey).HasMaxLength(200).IsRequired();
        }
    }

}using System.ComponentModel.DataAnnotations;

namespace Istore.Models
{
    public class ExternalLoginConfirmationViewModel
    {
        [Required]
        [Display(Name = "User name")]
        public string UserName { get; set; }
    }
```

```
    public class ManageUserViewModel
    {
        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Current password")]
        public string OldPassword { get; set; }

        [Required]
        [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters
long.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "New password")]
        public string NewPassword { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Confirm new password")]
        [Compare("NewPassword", ErrorMessage = "The new password and confirmation
password do not match.")]
        public string ConfirmPassword { get; set; }
    }

    public class LoginViewModel
    {
        [Required]
        [Display(Name = "User name")]
        public string UserName { get; set; }

        [Required]
        [DataType(DataType.Password)]
        [Display(Name = "Password")]
        public string Password { get; set; }

        [Display(Name = "Remember me?")]
        public bool RememberMe { get; set; }
    }

    public class RegisterViewModel
    {
        [Required]
        [Display(Name = "User name")]
        public string UserName { get; set; }

        [Required]
        [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters
long.", MinimumLength = 6)]
        [DataType(DataType.Password)]
        [Display(Name = "Password")]
        public string Password { get; set; }

        [DataType(DataType.Password)]
        [Display(Name = "Confirm password")]
        [Compare("Password", ErrorMessage = "The password and confirmation password do
not match.")]
        public string ConfirmPassword { get; set; }
    }
}
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace Istore.Models
{
```

```
    // Корзина
    public class CartModel
    {
        public int ItemId { get; set; } // Товар

        [Display(Name = "Адрес доставки")]
        [Required(ErrorMessage = "Укажите Адрес доставки")]
        public string Address { get; set; } //Адрес доставки

        [Display(Name = "Email")]
        [Required(ErrorMessage = "Укажите Email")]
        public string Email { get; set; } //Email

        [Display(Name = "Контактный Телефон")]
        [Required(ErrorMessage = "Укажите Контактный Телефон")]
        public string Phone { get; set; } //Контактный Телефон
    }
}using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace Istore.Models
{
    // Модель Характеристики товара
    public class DescriptionModel
    {

        public int ItemId { get; set; } // Товар

        [Display(Name = "Характеристика товара")]
        [Required(ErrorMessage = "Укажите Характеристику товара")]
        public string Descriptions { get; set; } //Характеристика товара

        [Display(Name = "Тип")]
        [Required(ErrorMessage = "Укажите Тип описании")]
        public int TypeId { get; set; } // тип описании (характеристики) товара
        public System.Web.Mvc.SelectList typeList;



    }
}using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace Istore.Models
{
    public class DescriptionTypeModel
    {
        [Display(Name = "Название Описании")]
        [Required(ErrorMessage = "Название Описании")]
        public string Name { get; set; } // Название Описании
    }
}using Microsoft.AspNet.Identity.EntityFramework;
using Istore.Models;
using System.Data.Entity;

namespace Istore.Models
{
    public class ApplicationUser : IdentityUser
```

```
    {
    }

    public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
    {
        public ApplicationDbContext()
            : base("LocalMySqlServer")
        {
            Database.SetInitializer(new MySqlInitializer());
        }
    }
}using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace Istore.Models
{
    public class MainItemModel
    {
        [Display(Name = "Название категории")]
        [Required(ErrorMessage = "Название категории")]
        public string Name { get; set; } // Название категории


    }
}using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace Istore.Models
{
    public class SubItemModel
    {
        [Display(Name = "Наименование товара")]
        [Required(ErrorMessage = "Укажите Наименование товара")]
        public string Name { get; set; } // Наименование товара

        [Display(Name = "Категория")]
        [Required(ErrorMessage = "Укажите категорию товара")]
        public int ParentId { get; set; } // категория
        public System.Web.Mvc.SelectList itemList;

        [Display(Name = "Количество товара")]
        [Required(ErrorMessage = "Укажите Количество товара")]
        public int Count { get; set; }

        [Display(Name = "Цена товара")]
        [Required(ErrorMessage = "Укажите цену товара")]
        public double Price { get; set; }

        [Display(Name = "Активный")]
        public bool Active { get; set; }

        [Display(Name = "Акционный товар")]
        public bool IsDiscount { get; set; }

        [Display(Name = "Цена по акции")]
        public double DiscountPrice { get; set; }
```

```
}
   }
```