**Содержание**

Введение

1. Исследование предметной области

1.1 Актуальность систем электронной коммерции

1.2 Требования к системе

1.3 Сравнение с аналогами

1.4 Описание способов оплаты

1.5 Решения по реализации

1.6 Обработка страниц на web-сервере и модель событий ASP.NET

2. Проектирование БД

2.1 Физическая модель БД

2.2 Описание таблиц БД

3. Взаимодействие с БД

3.1 Архитектура ADO.NET

3.2 Компоненты доступа к данным

3.3 Привязка данных

3.4 Пример взаимодействия с БД

4. Разработка компонентов web-приложения

4.1 Общая структура web-приложения

4.2 Компоновка и оформление web-страниц

4.3 Элементы управления проверкой достоверности

4.4 Класс web-страницы с возможностью отслеживания обновления

4.5 Аутентификация и авторизация пользователей

4.6 Описание профилей пользователей

4.7 Основные элементы интерфейса музыкального Интернет-магазина

4.8 Основные элементы интерфейса администратора музыкального Интернет-магазина

5. Производительность и масштабируемость web-приложения

5.1 Способы повышения производительности и масштабируемости

5.2 Применение механизма кэширования

5.3 Настраиваемое разбиение на страницы с помощью ObjectDataSource

6. Развёртывание web-приложения и модели компиляции

6.1 Развёртывание web-приложения

6.2 Модели компиляции

7. Разработка бизнес-плана

7.1 Резюме

7.2 Анализ положения дел в отрасли

7.3 Суть разрабатываемого проекта

7.4 План маркетинга

7.5 Производственный план

7.6 Планирование разработки и определение себестоимости программного обеспечения

7.7 Оценка экономической эффективности инвестиционного проекта

7.8 Анализ рисков и неопределённостей

8. Безопасность и санитарно-гигиенические условия труда на рабочем месте пользователя ПЭВМ

8.1 Рабочее место, задачи его организации и проектирования

8.2 Требования к микроклимату рабочего помещения

8.3 Требования к содержанию в воздухе вредных веществ и пыли

8.4 Расчёт системы вентиляции

8.5 Требования к уровню шума и вибрации

8.6 Нормы по освещённости рабочего места. Расчёт системы освещения

8.7 Требования к электромагнитному излучению

8.8 Требования к обеспечению электробезопасности

8.9 Требования к обеспечению пожарной безопасности

8.10 Общие требования к организации режима труда при работе с ПЭВМ

Заключение

Список источников

**Введение**

Стремительное развитие сети Интернет, применение новейших технологий и коммуникаций в коммерческой деятельности и повседневной жизни обусловило возникновение новых экономических явлений, таких как электронная коммерция. Электронная коммерция – это довольно широкая сфера экономики, включающая в себя многие виды деятельности, такие как Интернет-магазины, Интернет-маркетинг и реклама, различные платёжные системы и многие другие. Сегодня всё больше и больше магазинов, от маленьких узкоспециализированных до крупных торговых сетей стремятся иметь своё представительство в сети Интернет, а многие вообще осуществляют свою деятельность исключительно в сети. Интернет предоставляет новые каналы сбыта продукции, открывает широкие возможности для рекламы и маркетинговых исследований.

Естественно, что для осуществления электронной коммерции, необходимо современное программное обеспечение. Сегодня существует множество технологий (такие как PHP, JSP, ASP, ASP.NET), позволяющих создавать надёжные и эффективные web-приложения, отвечающие самым современным требованиям, удобные в использовании, как со стороны продавца, так и со стороны покупателя.

Целью данного дипломного проекта является создание web-приложения, представляющего собой интернет – магазин по продаже музыкальных CD и DVD дисков, которое позволит продавцу размещать в Интернет информацию о продукции и принимать заказы покупателей. А покупатели соответственно смогут просматривать каталог и заказывать музыкальную продукцию, доставка которой будет осуществляться через систему почтовой связи.

В качестве ОС для разработки была выбрана Windows XP Professional. В качестве технологии разработки будет использоваться ASP.NET 2.0. Информационная система будет представлять собой реляционную базу данных под управлением СУБД SQL Server 2000.

**1. Исследование предметной области**

**1.1 Актуальность систем электронной коммерции**

Как уже было сказано выше, с развитием сети Интернет, электронная коммерция приобретает всё более и более широкий размах. И соответственно требуются различные программные средства, которые позволяют сделать её максимально эффективной. Ниже приведены некоторые преимущества систем электронной коммерции:

* Информация о товарах и услугах оборачивается быстрее. Фактически, вы получаете дополнительный коммуникационный канал, открытый 24/7 (оборот, вошедший в английский язык в связи с развитием электронной коммерции и обозначающий работу 24 часа в сутки, 7 дней в неделю). Вы более доступны для клиента географически и во времени, к тому же он получает новое средство поиска и работы с информацией.
* Быстрее оборачивается внутренняя информация (документы, служебная переписка, принятие и утверждение решений и пр.). Расширенная возможность для контроля исполнения. Иначе говоря, ускоряются все бизнес-процессы за счет доступности и скорости передачи информации.
* Интернет предлагает новые сервисы для клиентов – например, отслеживание продвижения заказа. Наличие дополнительных сервисов создает конкурентное преимущество и приводит к вам новых посетителей.
* Поскольку Интернет является технологической оболочкой, он позволяет собирать важную информацию о ваших клиентах. Кроме того, он позволяет использовать маркетинговый инструментарий – опросы, рассылки и т.п. оперативно и без дополнительных затрат.
* Системы электронной коммерции помогут сэкономить на персонале и на аренде площадей для торговых залов.

Таким образом, основными преимуществами электронной коммерции является доступность для большого количества клиентов, географически и во времени, широкие возможности рекламы и маркетинга, экономия средств на персонале и торговых площадях.

**1.2 Требования к системе**

Разрабатываемая система должна удовлетворять следующим требованиям:

* Иметь систему регистрации пользователей;
* Иметь систему аутентификации и авторизации пользователей;
* Предоставлять потенциальному покупателю полную информацию о музыкальной продукции (название, исполнитель, год выпуска, звукозаписывающая компания, жанр, год выпуска);
* Наличие расширенной системы поиска. Поиск по категориям (жанр, год выпуска, издатель) и по названию, путем ввода части слова или целиком;
* Предоставлять потенциальному покупателю всю необходимую информацию для совершения покупки в Интернет-магазине;
* Предоставлять потенциальному покупателю возможность задавать вопросы по работе Интернет-магазина;
* Иметь удобную и быструю систему заказа музыкальной продукции;
* Иметь несколько разных способов оплаты продукции;
* Наличие системы скидок для постоянных покупателей;
* Возможность просмотра истории заказов и отслеживание хода выполнения текущих;
* Возможность оставлять отзывы о музыкальной продукции;
* Возможность редактирование покупателями личных данных;
* Иметь систему восстановления пароля;
* Возможность подписки на новости Интернет-магазина;
* Обеспечивать конфиденциальность информации о покупателях и заказах.
* Возможность добавления, удаления и редактирования информации о продукции;
* Возможность добавления, удаления и редактирования информации о музыкальных жанрах, исполнителях, издателях, категориях музыкальной продукции;
* Иметь систему поиска заказов по дате и состоянию выполнения;
* Возможность просмотра заказов и изменение хода их выполнения;
* Возможность просмотра личных данных о покупателе;
* Возможность отвечать на вопросы покупателей;

**1.3 Сравнение с аналогами**

Первые Интернет-магазины появились в России в 1998 году, тогда их количество исчислялось несколькими десятками, но уже к 2001 году их количество превысило 800, а в 2003 уже 4000. Какие-то из них являются Интернет-версиями реальных магазинов, какие-то действуют только в сети, кто-то торгует широким спектром товаров, а кто-то специализируется на конкретном виде товара. Все Интернет магазины предоставляют потенциальному покупателю в целом идентичный набор функций, отличаясь друг от друга только широтой ассортимента и дизайном.

Таким образом, главным является качественная реализация функций Интернет-магазина, максимально ориентированная на удобство потенциального покупателя.

Основные требования покупателей, их ожидания от Web-сайта Интернет-магазина:

"легко найти" – понятный интерфейс и удобная навигация;

"хорошо предложить" – система перекрестных ссылок и предоставление дополнительной информации, объем которой не превышает разумных пределов;

"быстро купить" – чем меньше кликов мышкой нужно сделать для совершения покупки, тем лучше.

"способы оплаты" – предоставление различных способов оплаты.

"время доставки" – чем меньше время доставки товара, тем лучше.

В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении.

Во многих Интернет-магазинах отсутствует такой способ оплаты, как наложенный платёж. Для многих покупателя важно сначала подержать товар в руках, а только потом заплатить за него, особенно если это первый заказ в Интернет-магазине. Скорее всего, покупатель выберет тот магазин, в котором будет присутствовать этот способ оплаты.

Ещё один важный аспект это скорость доставки. Хотя непосредственно реализация сайта никак не влияет на этот фактор, мы можем создать систему отслеживания продвижения заказа. При этом пользователь сможет узнать, на каком этапе выполнения находится его заказ, принят он к исполнению или уже отправлен на почту.

Таким образом, чтобы превзойти аналогичные Интернет-магазины, и исходя из требований покупателей, предъявляемых к Web-сайту Интернет-магазина, основной упор в разработке будет сделан на удобство пользовательского интерфейса, предоставление обширной информации о продукции, а также наличие удобных для пользователя способов оплаты.

**1.4 Описание способов оплаты**

Для успешной работы в сфере электронной коммерции необходимо предусмотреть различные способы оплаты продукции, которые подойдут максимальному числу потенциальных покупателей. В разрабатываемом Интернет-магазине планируется применять как традиционные способы оплаты (банковский перевод и наложенный платёж), так и новые, характерные для Интернет (web-money, yandex-деньги).

При осуществлении платежей через Интернет всё большую популярность приобретают такие электронные платёжные системы как web-money и yandex-деньги. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. У Интернет-магазина будут свои кошельки в этих платёжных системах. Пользователь должен лишь, используя web-сервис соответствующей платёжной системы, перевести необходимую сумму на счёт Интернет-магазина, указав номер своего заказа. О том, как это сделать, подробно описано на сайте платёжной системы. Безопасность денежного перевода также гарантируется средствами платёжной системы. Плюсом этого способа оплаты является то, что перевод средств происходит практически мгновенно, это означает, что заказ быстрее будет принят к исполнению и соответственно быстрее будет доставлен покупателю.

Однако не все потенциальные покупатели имеют виртуальные Интернет-кошельки, поэтому необходимо предусмотреть и другие способы оплаты.

Оплата через банковский перевод является ещё одним способом заказа продукции с предоплатой. Однако он не всегда удобен для пользователя. Как минимум сначала необходимо идти в банк для оплаты продукции, а потом почту, чтобы забрать товар.

Более удобным для пользователя является способ оплаты – наложены платёж. Покупатель оплачивает товар непосредственно при получении его на почте, при этом он может проверить его целостность.

Также возможна доставка товара курьером. При этом оплата происходит при получении товара. Это наиболее быстрый способ доставки, но также и наиболее дорогой. Однако доставка курьером может осуществляться на ограниченной территории, как правило, в том городе, где расположен склад Интернет-магазина.

**1.5 Решения по реализации**

**1.5.1 Выбор технологии для разработки**

Сегодня существует много различных технологий, позволяющих создавать web-приложения разной степени сложности. К ним относятся ASP, ASP.NET, PHP, JSP и многие другие. Наиболее популярной и распространённой технологией на сегодняшний день является PHP, во многом благодаря своей бесплатности. Однако с появлением технологии ASP.NET у PHP появился серьёзный конкурент. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении.

Необходимо выбрать технологию для разработки Музыкального Интернет-магазина, которая позволит наилучшим образом реализовать все функции системы, обеспечить высокое быстродействие при одновременной работе большого числа пользователей, а также обеспечить безопасность данных от несанкционированного доступа. Также стоит отметить, что на создание web-приложения отводится короткий промежуток времени, поэтому необходимо наличие мощной среды программирования, позволяющей максимально ускорить процесс разработки и отладки.

Ниже приведён небольшой сравнительный анализ технологий ASP.NET и PHP. Его целью не является доказать какая технология лучше, а просто обосновать выбор для разработки конкретного web-приложения, а именно Музыкального Интернет-магазина.

ASP.NET – новая технология, разработанная компанией Microsoft и являющаяся частью среды.NET Framework. Она позволяет динамически создавать документы на web-сервере, когда они запрашиваются по протоколу HTTP.

PHP – это открытая и бесплатная технология. PHP – это скриптовый язык, созданный для динамического вывода HTML.

Для реализации функций разрабатываемого web-приложения подходят обе технологии, однако, используя ASP.NET, это сделать намного проще. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении.

Технология PHP является кроссплатформенной, а ASP.NET на сегодняшний момент реализована только для ОС Windows, что является минусом и автоматически привязывает нас к одной ОС.

Одним из важнейших факторов, влияющим на выбор той или иной технологии, является быстродействие, особенно при одновременной работе с множеством пользователей. Утверждать, что какая-то из технологий всегда выигрывает в быстродействии нельзя, необходимо рассматривать каждый конкретный случай отдельно. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. Скорость работы обеспечивается тем, что все РНР-приложения работают в едином адресном пространстве, тогда как ASP.NET за счет сложной модели классов многократно проверяет и перепроверяет данные, удерживая каждое приложение в отдельном адресном пространстве. Первый подход более быстр, но менее надежен, второй – более надежен, но за это приходится платить. Отдельно стоит упомянуть об обеспечении быстродействия при взаимодействии с СУБД. PHP ориентирован на работу с MySQL, а ASP.NET – с MS SQL Server. Это значит, что наилучшее быстродействие будет достигаться при использовании соответствующей СУБД. При этом необходимо отметить, что MS SQL Server по функциям и производительности превосходит MySQL.

Также при выборе технологии необходимо учитывать стоимость разработки и поддержки web-приложения. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении.

Таким образом, сравнив обе технологии, было принято решение использовать для разработки технологию ASP.NET. Главными аргументами в пользу этой технологии являются большие функциональные возможности, наличие высокоуровневых средств организации пользовательского интерфейса и работы с базой данных. Отдельно стоит отметить наличие мощной среды программирования – Visual Studio, которая позволяет быстро реализовать все функции Интернет-магазина в условиях ограниченного времени, отведённого на разработку.

**1.5.2 Основные особенности технологии ASP.NET**

Итак, в качестве технологии разработки была выбрана ASP.NET. Ниже более подробно описываются основные возможности этой технологии.

Технология ASP.NET написана с чистого листа. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. При этом сама результирующая страница не содержит ни кодов ASP.NET, ни элементов управления и может быть просмотрена в любом из браузеров, отредактирована в любом редакторе HTML.

Особенности ASP.NET:

– ASP.NET является составной частью среды.NET Framework, поэтому на стороне сервера мы имеем доступ ко всей богатейшей библиотеке классов.NET.

– ASP.NET предлагает полностью объектно-ориентированную модель программирования, включающую управляемую событиями, основанную на элементах управления архитектуру, которая обеспечивает инкапсуляцию и.

– ASP.NET позволяет писать код на любом языке, поддерживаемом.NET (включаю Visual Basic, C#, J# и множество других языков с компиляторами от).

– ASP.NET ориентирован на достижение высоких показателей производительности. Страницы и компоненты ASP.NET компилируются по требованию, а не интерпретируются при каждом использовании. Гибкий механизм кэширования также позволяет.

– ASP.NET имеет обширную коллекцию элементов управления, позволяющую создавать удобный и эффективный пользовательский интерфейс.

Отдельно стоит отметить простоту и удобство взаимодействия с базами данных и отображение информации на.

– ASP.NET является также платформой для построения Web-служб, являющихся многократно используемыми модулями кода, которые вызываются другими приложениями в рамках различных платформ и.

– Приложения ASP.NET легко развёртываются и конфигурируются. В простейшем случаев потребуется просто скопировать все файлы в виртуальный каталог на.

– Средства разработки web-приложений ASP.NET, такие как Microsoft Visual Studio 2005, позволяют сделать разработку и отладку приложения более быстрым и.

**1.5.3 Выбор ОС и web-сервера**

Как было описано выше, для разработки была выбрана технология ASP.NET. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. Она является мощной и надёжной ОС, отличается относительной лёгкостью администрирования и имеет интегрированный web-сервер.

В качестве Web-сервера используется IIS (Internet Information Services – Информационные службы Internet). На сегодняшний день он является единственным Web-сервером, поддерживающим ASP.NET. Но это не значит, что он мало распространен или неэффективен в использовании. Как раз наоборот, IIS входит в состав всех серверных ОС Windows, а также в Windows Professional, таким образом, для того, чтобы развернуть Web-приложение, достаточно иметь одну из указанных ОС Windows. По сути IIS является службой Windows, отвечающей за обработку запросов, получаемых на определённых портах. В пункте 1.6 будет подробнее описан процесс взаимодействия Web-сервера с приложением ASP.NET.

**1.5.4 Выбор языка программирования для разработки**

Так как для разработки web-приложения была выбрана технология ASP.NET, то в качестве языка программирования мы можем выбрать один из трёх языков – С#, Visual Basic.NET, J# или даже использовать несколько.

В качестве языка программирования был выбран C#. Это достаточно новый язык программирования, который характеризуется следующими двумя преимуществами:

– Он спроектирован и разработан специально для применения с платформой.NET Framework. Таким образом, он позволяет наиболее полно и эффективно использовать все широчайшие возможности, предоставляемые этой средой.

– Это язык, основан на современной объектно-ориентированной методологии проектирования. При его создании был использован весь более чем двадцатилетний опыт разработки подобных языков.

– С# вобрал в себя все лучшее из таких популярных языков как С++, Visual Basic, Java и Object Pascal. С# обеспечивает быструю разработку, в то же время позволяет писать эффективный код. Он имеет прекрасную поддержку компонентов, надежен и устойчив благодаря использованию "сборщика мусора", обработки исключений, безопасности типов.

Основные особенности нового языка:

– автоматическая "сборка мусора" – освобождение неиспользуемой памяти;

– возможность манипулировать указателями и иметь непосредственный доступ к;

– поддержка свойств и (аналогично VB);

– поддержка атрибутов;

– встроенная поддержка основных типов (строка, массив и);

– множественное наследование возможно только от интерфейсов (как в Java);

– поддержка С API, Windows API и COM+ на уровне языка;

– контроль типов;

– автоматическая инициализация переменных;

Таким образом, для разработки был выбран язык программирования C#, который был специально разработан для платформы.NET Framework полностью в соответствие с объектно-ориентированными принципами и позволяет наиболее полно и эффективно использовать все возможности платформы.

**1.5.5 Выбор информационной системы**

Разрабатываемое web-приложение будет хранить значительные объёмы структурированной информации, к которой необходимо обеспечить эффективный множественный доступ. Исходя из этого, в качестве информационной системы была выбрана реляционная база данных.

Для управления базой данных необходимо выбрать СУБД, которая будет максимально отвечать требованиям, предъявляемым к разрабатываемому web-приложению.

Довольно часто для малых и средних проектов используется СУБД MySQL, однако она первоначально была разработана как простая бесплатная база данных, в которой в угоду простоте, дешевизне и быстродействию были исключены многие функции, например, поддержка целостности и хранимые процедуры. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении.

Таким образом, для управления базой данных Музыкального Интернет-магазина потребуется более производительная СУБД с большим набором функций, например, Oracle 9i Database или MS SQL Server 2000. Ниже приведены некоторые преимущества каждой из них.

Преимущества SQL Server 2000:

– SQL Server 2000 дешевле Oracle 9i Database в 0 раза.

– SQL Server 2000 удерживает высшие результаты TPC-C по производительности и.

– SQL Server 2000, в общем, считается более легким для установки, использования и.

Преимущества Oracle 9i Database:

– Oracle 9i Database поддерживается на всех известных платформах, а не только на Windows-платформах.

– PL/SQL более мощный язык, чем T-SQL.

– Большее количество тонких настроек конфигурации может быть сделано через параметры запуска.

Оба продукта могут быть использованы для построения стабильных и эффективных систем, однако, в нашем случаё более подходящим является MS SQL Server 2000. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. К тому же использование SQL Server обойдётся в 2–3 раза дешевле, чем аналогичный вариант на Oracle.

Также необходимо отметить, что часть среды.NET Framework – ADO.NET, которая обеспечивает взаимодействие с базами данных, во многом ориентирована на взаимодействие именно с SQL Server. Это позволяет достичь наилучшего быстродействия при обращении к базе данных, что особенно важно для web-приложений. Кроме того, в библиотеке классов.NET есть всё необходимое для подключения и взаимодействия с базами данных SQL Server.

Исходя из вышеперечисленного, в качестве СУБД был выбран Microsoft SQL Server 2000. Это эффективный и надёжный сервер, отвечающий самым высоким требованиям к производительности, масштабируемости и отказоустойчивости, имеет широкую поддержку языка XML и стандартов Интернета. Он обладает эффективными и удобными средствами администрирования.

**1.5.6 Выбор среды разработки**

В качестве среды разработки была выбрана Microsoft Visual Studio 2005.

Это профессиональный инструмент разработки приложений любой степени сложности, в том числе и приложений ASP.NET. Она позволяет сделать процесс разработки и отладки быстрым и эффективным. Ниже приведены некоторые преимущества Visual Studio:

– Реализация принципа WYSIWYG ("Что видишь, то и получаешь"), позволяющего максимально упростить проектирование пользовательского интерфейса и.

– Встроенный Web-сервер, наподобие IIS для.

– Многоязыковая разработка. Разные web-страницы в одном приложении могут быть реализованы на.

– Большой набор отладочных средств, позволяющих увеличить скорость и.

– Механизм IntelliSense, отлавливающий ошибки и предлагающий решения в.

Исходя из требований к разрабатываемому web-приложения, было принято вести разработку на ОС Windows Professional, на платформе.NET Framework. В качестве технологии для разработки используется ASP.NET, а языком программирования выбран C#. В качестве СУБД выбран SQL Server 2000, а для доступа к базе данных используется технология ADO.NET. Для разработки используется Visual Studio 2005.

**1.6 Обработка страниц на Web-сервере и модель событий ASP.NET**

В общем случае процесс взаимодействия пользователя с web-приложением происходит следующим образом. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. После соответствующей обработки ресурс возвращается IIS, а затем и клиенту. Общая схема обработки запросов представлена на рисунке 1.

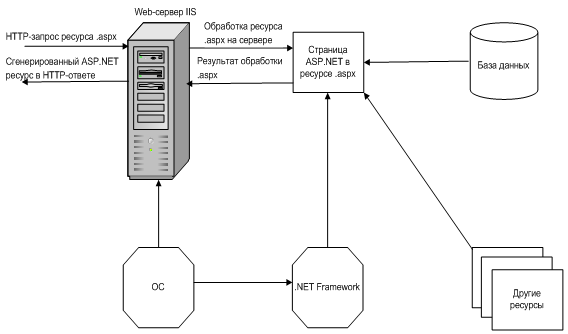


Рис. 1. "Обработка запросов с помощью IIS и ASP.NET"

После того как web-сервер передал запрос исполняющей среде ASP.NET, вся последующая обработка происходит уже там. Ниже кратко описана модель событий ASP.NET:

1. Когда страница запрашивается в первый раз, ASP.NET создаёт объекты страницы и элементов управления, выполняет код инициализации, страница преобразуется в HTML и возвращается клиенту. После чего объекты страницы удаляются из.
2. В определённый момент пользователь выполняет действие, инициирующее обратную отсылку – например, щёлкает по кнопке, расположенной на странице. В этот момент страница со всеми данными формы отправляется на.
3. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении.
4. Затем ASP.NET проверяет, какая именно операция инициировала обратную отсылку, и запускает определённые события, например, событие нажатия кнопки – Button. Click. При этом выполняются определённая операция, например запрос к базе данных или считывание определённой информации из файла. После этого изменяются объекты элементов управления так, чтобы они отображали новую информацию.
5. Изменённая страница преобразуется в HTML и возвращается клиенту. После чего все объекты страницы снова удаляются из памяти. Если происходит ещё одна обратная отсылка, то.

Как было сказано, выше после отправки страницы клиенту вся информация о ней удаляется из памяти web-сервера, что позволяет экономить ресурсы сервера, это означает, что информация будет теряться после каждой обратной отсылки. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. Эта информация используется, чтобы при повторном создании объёктов страницы на web-сервере, привести их в состояние, в котором они были перед последней отправкой страницы клиенту.

На рисунке 2 представлен жизненный цикл страницы ASP.NET.

Инициализация структуры страницы

Браузер выдаёт запрос

Инициализация кода пользователя

Очистка

Браузер получает ответ

Проверка достоверности



Рис. 2. "Жизненный цикл страницы ASP.NET"

В данном пункте был обобщённо рассмотрен процесс взаимодействия пользователя с web-приложением посредством web-сервера и среды ASP.NET, а также приведён порядок обработки страниц средой ASP.NET.

**2. Проектирование БД**

**2.1 Физическая модель БД**

Проектирование таблиц базы данных и связей между ними осуществлялось с помощью встроенного редактора SQL Server 2000. На рисунке 3 представлена физическая модель базы данных web-приложения "Музыкальный Интернет-магазин".

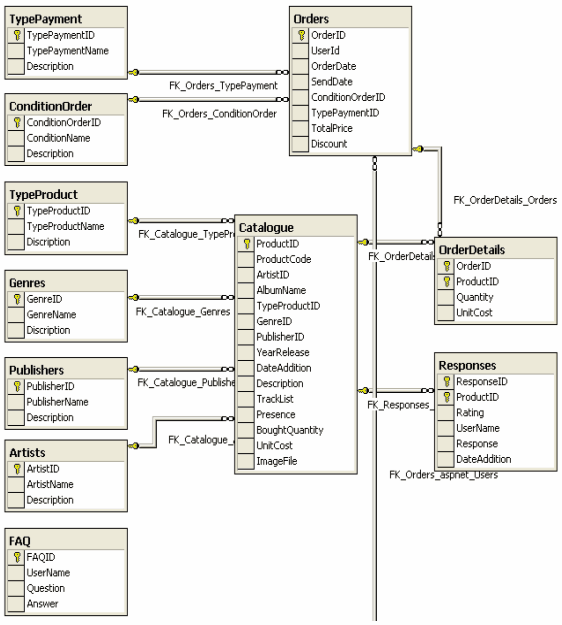


Рис. 3 лист 1 "Физическая модель базы данных"

Скрипт для создания базы данных, а также примеры хранимых процедур, представлены в приложении 4.

Таблицы, представленные на рис. 3 лист 2, специально выделены в отдельную группу. Они необходимы для работы интерфейсов Membership API, Roles API и Profile API, предназначенных для обеспечения аутентификации и авторизации пользователей, а также управление профилями пользователей. Подробнее об этих интерфейсах и их использовании при разработке приложения написано в пунктах 4.5.1, 4.5.3 и 4.6 соответственно.

**2.2 Описание таблиц БД**

В таблице 1 представлена структура основных таблиц базы данных.

Таблица 1 "Структура основных таблиц БД"

|  |  |  |  |
| --- | --- | --- | --- |
| **Таблица** | **Атрибут** | **Тип данных** | **Примечание** |
| Catalogue  (Каталог) | ProductID  (Код продукта) | int | Первичный ключ, автоинкрементное |
| ProductCode  (Код продукта в каталоге фирмы-издателя) | nvarchar(30) | Уникальное |
| ArtistID  (Код исполнителя) |  | Внешний ключ из таблицы Artists, часть составного уникального поля |
|  | nvarchar(50) | Часть составного уникального поля |
| TypeProductID  (Код категории) | int | Внешний ключ из таблицы TypeProduct, часть составного уникального поля |
| GenreID  (Код муз. жанра) | int |  |
| PublisherID  (Код фирмы-издателя) | int | Внешний ключ из таблицы Publishers, часть составного уникального поля |
| YearRelease  (Год выпуска) | int |  |
| DateAddition  (Дата добавления в каталог) | smalldatetime |  |
| Description  (Описание) | nvarchar(4000) |  |
| TrackList  (Список композиций) | nvarchar(1000) |  |
|  | bit |  |
| BoughtQuantity  (Кол-во заказов) | int |  |
| UnitCost  (Цена) | money |  |
| ImageFile  (Файл обложки) |  |  |
| TypeProduct  (Категория продукта) | TypeProductID  (Код категории) | int | Первичный ключ, автоинкрементное |
| TypeProductName  (Название категории) | nvarchar(30) | Уникальное |
| Description  (Описание) | nvarchar(50) |  |
| Genres  (Музыкальные жанры) | GenreID  (Код муз. жанра) |  | Первичный ключ, автоинкрементное |
| GenreName  (Название жанра) | nvarchar(50) |  |
| Description  (Описание) | nvarchar(50) |  |
| Publishers  (Фирмы-издатели) | PublisherID  (Код фирмы-издателя) | int | Первичный ключ, автоинкрементное |
| PublisherName  (Название фирмы-издателя) | nvarchar(50) | Уникальное |
|  | nvarchar(50) |  |
| Artists  (Исполнители) | ArtistID  (Код испонителя) | int | Первичный ключ, автоинкрементное |
| ArtistName  (Название исполнителя) | nvarchar(50) | Уникальное |
| Description  (Описание) | nvarchar(50) |  |
| Responses  (Отзывы о продукции) |  | int | Часть составного первичного ключа, автоинкрементное |
| ProductID  (Код продукта) |  | Часть составного первичного ключа, внешний ключ из таблицы Catalogue |
| Rating  (Оценка) | tinyint |  |
| UserName  (Имя пользователя) | nvarchar(250) |  |
| Response  (Отзыв) | nvarchar(500) |  |
| DateAddition  (Дата добавления) | smalldatetime |  |
| FAQ  (Вопрос / Ответ) | FAQID  (Код) | int | Первичный ключ, автоинкрементное |
| UserName  (Имя пользователя) | nvarchar(50) |  |
|  | nvarchar(500) |  |
| Answer | nvarchar(500) |  |
| Orders  (Заказы) | OrderID  (Код заказа) | int | Первичный ключ, автоинкрементное |
|  | uniqueidentifier | Внешний ключ из таблицы aspnet\_Users |
| OrderDate  (Дата заказа) | smalldatetime |  |
| SendDate  (Дата отправки заказа) |  |  |
| ConditionOrderID  (Код состояния заказа) | int | Внешний ключ из таблицы ConditionOrder |
| TypePaymentID  (Код способа оплаты) | int | Внешний ключ из таблицы TypePayment |
| TotalPrice  (Цена заказа) | money |  |
| Discount  (Скидка) | int |  |
| OrderDetails | OrderID  (Код заказа) | int | Часть составного первичного ключа, внешний ключ из таблицы Orders |
| ProductID  (Код продукта) | int | Часть составного первичного ключа, внешний ключ из таблицы Catalogue |
| Quantity  (Кол-во) | int |  |
| UnitCost  (Цена) | money |  |
|  | ConditionOrderID  (Код состояния заказа) | int | Первичный ключ, автоинкрементное |
| ConditionName  (Наименование состояния заказа) | nvarchar(50) |  |
| Description  (Описание) |  |  |
| TypePayment  (Способы оплаты) | TypePaymentID  (Код способа оплаты) | int | Первичный ключ, автоинкрементное |

**3. Взаимодействие с БД**

**3.1 Архитектура ADO.NET**

На сегодняшний момент существует множество технологий доступа к данным – ODBC, DAO, ADO и многие другие. Среда.NET Framework имеет свою собственную технологию доступа к данным – ADO.NET.

ADO.NET состоит из управляемых классов, позволяющих приложениям.NET подключаться к источникам данных (обычно реляционным базам данных), выполнять команды, осуществлять выборку, обработку и хранение данных.

В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении.

Важным элементом автономного доступа к данным является контейнер для табличных данных, который не знает о СУБД. Такой контейнер представлен в ADO.NET классом DataSet или DataTable.

Архитектуру ADO.NET можно разделить на две фундаментальные части: подключённую и автономную. Все различные классы ADO.NET можно отнести к подключённой или автономной части. Единственным исключением является класс DataAdapter. Работающий как посредник между подключённой и автономной частями.

Подключённая часть содержит классы, которым для работы нужно доступное открытое соединение, чтобы взаимодействовать с источником данных. Сюда входят классы для установки подключения к источнику данных (Connection), управления транзакциями (Transaction), выполнения команд (Command) и другие.

К автономной части относятся классы, которым не требуется доступное открытое подключение. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении.

Далее на рисунке 4 представлена общая структура ADO.NET и то, как с помощью неё программа взаимодействует с источником данных.

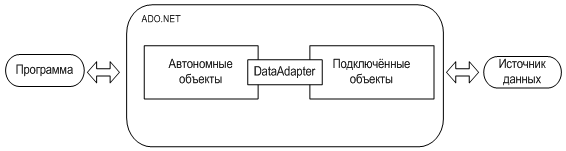


Рис. 4 "Общая структура ADO.NET и её место в обеспечении доступа к данным"

Отличительной особенностью ADO.NET являются то, что для доступа к разным базам данных используются разный набор классов. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. Классы автономной части ADO.NET являются одинаковыми для всех баз данных.

В данном пункте была рассмотрена технология доступа к данным ADO.NET, определено её место в обеспечении взаимодействия приложения с базой данных, а также рассмотрены основные классы, которые будут использоваться при разработке дипломного проекта.

**3.2 Компоненты доступа к данным**

В профессиональных, хорошо спроектированных приложениях код взаимодействия с базой данных никогда не встраивается непосредственно в код, лежащий в основе web-страницы. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. Чтобы выполнить операцию с базой данных, необходимо создать экземпляр класса и вызвать соответствующий метод.

При разработке дипломного проекта был использован именно такой подход. Для каждой таблицы базы данных, с которой будет взаимодействовать web-приложения, создан свой компонент доступа к данным, включающий все необходимые методы. Методы объявлены как статические, таким образом, вызывать их можно непосредственно через имя класса, не создавая его экземпляр. Каждый метод осуществляет обработку ошибок при взаимодействии с БД и информирует об этом web-страницу, из которой он был вызван.

Например, в компоненте данных OrdersDB, находятся все необходимые методы для управления заказами покупателей, взаимодействующие с таблицами базы данных Orders и OrderDetails. Вызов метода, который возвращает позиции заказа с указанным номером, в коде web-страницы будет выглядеть следующим образом:

OrdersDB. GetOrderDetails(orderID);

При этом весь код взаимодействия с базой данных находится в этом методе:

public class OrdersDB

{

…

public static DataSet GetOrderDetails (int orderID)

{

SqlConnection conn = new SqlConnection (WebConfigurationManager. ConnectionStrings ["connStr"].ConnectionString);

SqlCommand cmd = new SqlCommand ("GetOrderDetails", conn);

…………………………………………………;

cmd. Parameters. Add ("OrderID", SqlDbType. Int);

cmd. Parameters ["OrderID"].Value = orderID;

…………………………………………….;

SqlDataAdapter da = new SqlDataAdapter(cmd);

try

{

da. Fill(ds);

………….;

return ds;

}

catch (SqlException)

{

……………;

return null;

}

}

…

)

На рисунке 5 изображена диаграмма классов всех компонентов доступа к данным.

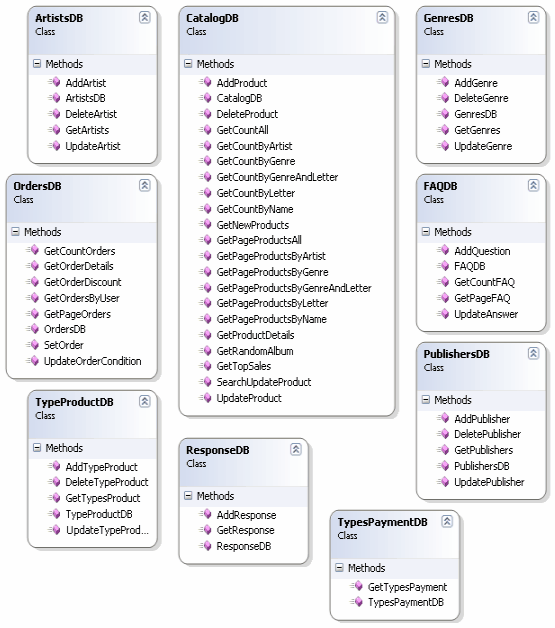


Рис. 5 "Диаграмма классов компонентов доступа к данным"

Примечание.

* ArtistDB – взаимодействие с таблицей Artists(Исполнители);
* OrdersDB – взаимодействие с таблицами Orders(Заказы) и OrderDetails (Детали заказа);
* TypeProductDB – взаимодействие с таблицей TypeProduct (Категории продукции);
* CatalogDB – взаимодействие с таблицей Catalogue(Каталог);
* ResponsesDB – взаимодействие с таблицей Responses(Отзывы);
* TypePaymentDB – взаимодействие с таблицей TypePayment (Способы оплаты);

**3.3 Привязка данных**

Помимо механизмов извлечения информации из источника данных, современные web-приложения нуждаются в удобном способе отображения этой информации на web-странице. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. Это означает, что не нужно вручную программировать логику циклического чтения строк, состоящих из многочисленного количества столбцов, и манипулирования отдельными элементами управления.

Например, чтобы отобразить на web-странице таблицу, содержащую позиции заказа, необходимо выполнить следующий код:

GridViewOrderDetails. DataSource = OrdersDB. GetOrderDetails(orderID); GridViewOrderDetails. DataBind();

Метод GetOrderDetails возвращает объект DataSet, содержащий результаты запроса к базе данных. Для того чтобы отобразить их на web-странице, используя привязку данных, необходимо присвоить DataSet специальному полю DataSource элемента управления (в данном случае GridView) и вызвать метод DataBind() элемента управления, чтобы выполнить привязку. В итоге на web-странице будет изображена таблица, содержащая позиции заказа с заданным номером.

Привязку данных можно также осуществлять с помощью элементов управления источниками данных. При разработке дипломного проекта использовался элемент управления источниками данных – ObjectDataSource, который для привязки данных использует компоненты доступа к данным. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. Ниже приведён пример использования элемента управления источником данных ObjectDataSource при отображении отзывов покупателей об альбоме с заданным номером:

<asp: ObjectDataSource ID= "ObjectDataSource1" SelectMethod= "GetResponse" TypeName= "ComponentsDB. ResponseDB" runat= "server">

<SelectParameters>

<asp: QueryStringParameter Name= "ProductID" QueryStringField= "ProductID" Type= "Int32" />

………………….

</asp: ObjectDataSource>

…

<asp: GridView ID= "ResponsesGridView" DataSourceID=" ObjectDataSource1" runat= "server" />

Всё это выполняется декларативно в файле дескрипторов web-страницы. При этом можно указывать, откуда ObjectDataSource должен брать параметры запроса, если они необходимы. В данном случае параметр ProductID берётся из строки запроса.

**3.4 Пример взаимодействия с БД**

Ниже приведён пример взаимодействия web-приложения с базой данных, используя технологию ADO.NET и компоненты доступа к данным. Web-страница HistoryOrders.aspx выводит на экран таблицу, содержащую все заказы данного пользователя.

На странице HistoryOrders.aspx находится элемент управления GridView, предназначенный для отображения табличных данных:

<asp: GridView ID= "GridViewOrders" AutoGenerateColumns= "False" EnableViewState= "False"

ShowFooter= "true" SkinID= "GridViewStyle" runat= "server">

<Columns>

<asp: BoundField DataField= "OrderID" HeaderText= "Номер заказа" />

<asp: BoundField DataField= "OrderDate" DataFormatString={0:dd.MM.yy}

………………………………………………………………………………

<asp: BoundField DataField= "SendDate" HeaderText= "Дата отправки" HtmlEncode= "False" DataFormatString=" {0:dd.MM.yy}" NullDisplayText= "Неизвестна"/>

<asp: BoundField DataField= "ConditionName" HeaderText= "Состояние заказа" />

…………………………………………………………………….

<asp: BoundField DataField= "TotalPrice" DataFormatString={0:c} HtmlEncode= "false" HeaderText= "Цена"/>

…………………

<ItemTemplate>

<asp: HyperLink ID= "OrderDetailsHyperLink" Text= "Подробности" NavigateUrl='<%# "~/ProtectedPages/HistoryOrderDetails.aspx? OrderID=" + …………………………….

runat= "server" CssClass= "GridViewLink"></asp: HyperLink>

</ItemTemplate>

<ItemStyle Width= "100" />

……………………….

</Columns>

</asp: GridView>

Для доступа к базе данных используется компонент доступа к данным – OrdersDB, для выборки данных – метод OrdersDB. GetOrdersByUser

public static DataSet GetOrdersByUser (Guid userID)

{

SqlConnection conn = new SqlConnection (WebConfigurationManager. ConnectionStrings ["connStr"].ConnectionString);

…………………………………………………………

cmd. CommandType = CommandType. StoredProcedure;

cmd. Parameters. Add ("UserId", SqlDbType. UniqueIdentifier);

cmd. Parameters ["UserId"].Value = userID;

………………………………………

SqlDataAdapter da = new SqlDataAdapter(cmd);

try

{

da. Fill(ds);

………….

}

catch (SqlException)

{

conn. Close();

return null;

}

}

Хранимая процедура GetOrdersByUser, находящаяся на сервере базы данных:

CREATE PROCEDURE GetOrdersByUser (@UserId UNIQUEIDENTIFIER) AS

SELECT OrderID, OrderDate, SendDate, ConditionName, TypePaymentName, TotalPrice FROM OrdersView

WHERE UserId = @UserId

ORDER BY OrderDate

GO

В файле кода web-страницы HistoryOrders.aspx.cs вызывается метод GetOrdersByUser класса OrdersDB:

MembershipUser \_user = Membership. GetUser (User. Identity. Name);

DataSet ds = OrdersDB. GetOrdersByUser((Guid)\_user. ProviderUserKey);

if (ds == null)

{

……………………………….

}

GridViewOrders. DataSource = ds;

GridViewOrders. DataBind();

Компонент доступа к данным вызывает хранимую процедуру на сервере базы данных, передавая в неё в качестве параметра код пользователя. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении, и если была обнаружена ошибка при взаимодействии с БД, то осуществляется переход на страницу информации об ошибке.

В результате на web-страницы будет изображена таблица, представленная на рисунке 6.



Рис. 6 "История заказов покупателя"

**4. Проектирование компонентов web-приложения**

**4.1 Общая структура web-приложения**

**4.1.1 Структура каталогов**

На рисунке 7 изображена полная структура каталогов разработанного web-приложения.

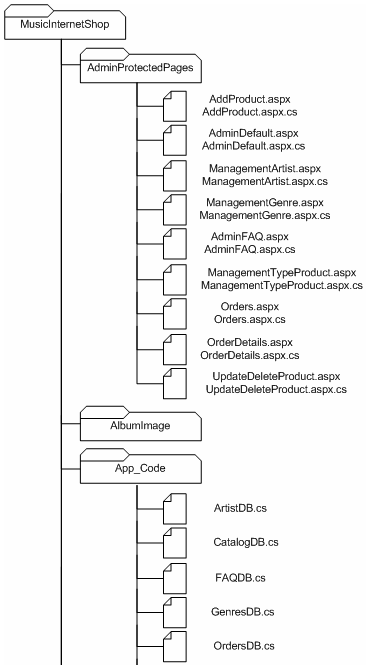


Рис. 7 лист 1 "Структура каталога web-приложения"

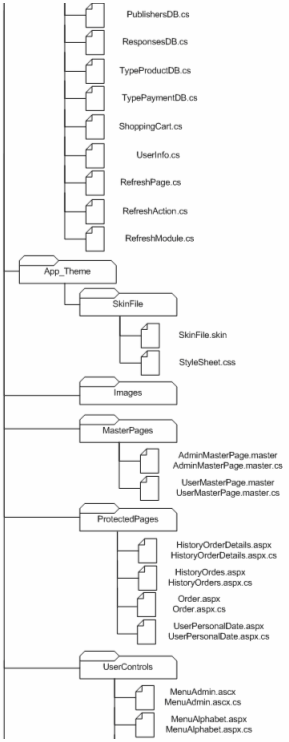


Рис. 7 лист 2 "Структура каталога web-приложения"

Подкаталог AdminProtectedPages содержит web-страницы (файлы дескрипторов.aspx и файлы кода.cs), доступные только аутентифицированным пользователям, чьи учётные записи относятся к роли Administrator. Подробнее о ролях рассмотрено в пункте №. 4.5.3

Назначение web-страниц подкаталога AdminProtectedPages:

* AddProduct.aspx – добавление новых альбомов в;
* AdminDefault.aspx – страница по умолчанию интерфейса администратор;
* ManagementArtist.aspx – добавление, удаление и редактирование информации об;
* ManagementGenre.aspx – добавление, удаление и редактирование информации о;
* AdminFAQ.aspx – просмотр и ответ на вопросы пользователей;
* ManagementTypeProduct.aspx – добавление, удаление и редактирование информации о;
* Orders.aspx – просмотр заказов;
* OrdersDetails.aspx – детали заказа, печать накладной заказа, изменение состояния выполнения заказа;
* UpdateDeleteProduct.aspx – редактирование и удаление данных об.

Подкаталог ImageAlbum содержит обложки музыкальных альбомов.

Подкаталог App\_Code содержит файлы пользовательских классов:

В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении.

Подкаталог App\_Theme содержит файл темы оформления и css-файл. (будут рассмотрены в пункте 4.2.3).

Подкаталог Images содержит изображения, используемые для оформления web-страниц.

Подкаталог MasterPages содержит файлы мастер-страниц, которые будут рассмотрены в пункте 4.2.1

Назначение web-страниц подкаталога MasterPages:

* UserMasterPage.master – мастер-страница интерфейса покупателя;
* AdminMasterPage.master – мастер-страница интерфейса администратора;

Подкаталог ProtectedPages содержит web-страницы (файлы дескрипторов.aspx и файлы кода.cs), доступные только аутентифицированным пользователям. Подробнее об аутентификации будет написано в пункте 4.5.2.

Назначение web-страниц подкаталога ProtectedPages:

* HistoryOrderDetails.aspx – подробности о;
* HistoryOrders.aspx – заказы покупателя;
* Order.aspx – оформление заказа;
* UserPersonalData.aspx – просмотр и редактирование личных данных покупателя, смена пароля, смена контрольного вопроса и ответа, управление подпиской;

Подкаталог UserControls, содержит файлы пользовательских элементов управления. Подробнее о пользовательских элементах управления написано в пункте 4.2.2.

Назначение пользовательских элементов управления:

* MenuAdmin.ascx – меню функций администратора;
* MenuAlphabet.ascx – алфавитное меню для поиска продукции по первой букве в названии;
* MenuGenres.ascx – панель поиска продукции по музыкальным жанрам;
* MenuUser.ascx – навигационная панель, отображает разные ссылки в завасимости от того аутентифицирован пользователь или нет, и к какой роли принадлежит его учётная запись;
* OrderSearch.ascx – панель поиска заказов;
* RandomAlbum.ascx – отображает случайный альбом из каталога;
* ShopInfo.ascx – навигационная панель;

В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении

Страницы, находящиеся в корневом каталоге web-приложения доступны всем пользователям.

Назначение web-страниц корневого каталога:

* AddQuestion.aspx – добавление вопроса администрации магазина;
* CatalogList.aspx – отображает список альбомов в зависимости от указанных критериев поиска;
* Default.aspx – страница по;
* ErrorPage.aspx – отображается при ошибке работы с;
* FAQ.aspx – отображает вопросы покупателей и;
* InfoShop.aspx – информация по работе Интернет-магазина;
* TypePayment.aspx – информация о способах оплаты;
* Login.aspx – страница входа в Интернет-магазин (аутентификация и авторизация);
* PasswordRecovery.aspx – восстановление пароля;
* ProductDetails.aspx – отображает подробности о выбранном альбоме, добавление отзывов об альбоме;
* Registration.aspx – страница регистрации;
* Responses.aspx – отображает отзывы покупателей о выбранном альбоме;
* ShoppingCart.aspx – отображает товары, находящиеся в покупательской корзине;

Также в корневом каталоге находится конфигурационный файл web.config и файл обработчиков глобальных событий web-приложения. Подробнее о назначении этих файлов описано в пункте 4.1.2.

**4.1.2 Файл конфигурации web-приложения web.config**

Файл web.config является xml-файлом, позволяющий задавать различные настройки web-приложения, такие как методы аутентификации и авторизации пользователей, настройка системы профилей, управление доступа к каталогам, строки подключения к источникам данных и многие другие.

Всё содержимое конфигурационного файла вкладывается в корневой элемент <configuration>. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении:

* Строка подключения к БД

<connectionStrings>

<add name= "connStr"

connectionString="Data Source=localhost; Integrated

……………………………………………………….

providerName= "System. Data. SqlClient"/>

</connectionStrings>

* Конфигурация интерфейса MemberShip API, подробнее о котором написано в пункте 4.5.1.

<membership defaultProvider= "MyMembershipProvider">

<providers>

<add name= "MyMembershipProvider"

type= "System. Web. Security. SqlMembershipProvider"

…………………………………….

applicationName="/MusicInternetShop"

enablePasswordRetrieval= "true"

enablePasswordReset= "true"

…………………………………….

requiresQuestionAndAnswer= "true"

requiresUniqueEmail= "true"

……………………………….

minRequiredPasswordLength= "7"/>

</providers>

</membership>

* Конфигурация системы аутентификации, подробнее о которой написано в пункте 4.5.2

<authentication mode= "Forms">

<forms name= "CookieName"

loginUrl= "Login.aspx"

…………………….

slidingExpiration= "true"

cookieless= "AutoDetect"

protection= "All"

…………………….

path="/">

</forms>

</authentication>

* Конфигурация интерфейса Roles API, подробнее о котором написано в пункте 4.5.3.

<roleManager

defaultProvider= "MySqlProvider"

………………………………….

cookieName=".MyRoleCookie"

cookieTimeout= "20"

cookieSlidingExpiration= "true"

………………………………

createPersistentCookie= "false">

<providers>

<add name= "MySqlProvider"

type= "System. Web. Security. SqlRoleProvider"

………………………………………

applicationName="/MusicInternetShop"/>

</providers>

</roleManager>

* Конфигурация интерфейса Profiles API, подробнее о котором написано в пункте 4.6

<profile defaultProvider= "MySqlProvider" automaticSaveEnabled= "false">

<providers>

<add name= "MySqlProvider"

type= "System. Web. Profile. SqlProfileProvider"

…………………………………………

applicationName="/MusicInternetShop"/>

</providers>

<properties>

<add name= "Cart" type= "ComponentsShoppingCart. ShoppingCart" serializeAs= "Binary" allowAnonymous= "true"/>

…………………………………………………………………….

<add name= "SumPurchase" type= "decimal" serializeAs= "Binary"/>

</properties>

</profile>

Также в файле web.config находятся элементы конфигурации SMTP-сервера, уровни доступа к различным подкаталогам, модель генерации html-кода. Более подробное описание каждого элемента конфигурации приведено в соответствующих пунктах пояснительной записки. Полный код файла web.config приведён в приложении 1.

Файл global.asax позволяет определять обработчики событий, реагирующие на глобальные события web-приложения, такие как начало и завершение запроса, моменты завершения аутентификации и авторизации и многие другие. При разработке дипломного проекта был определён обработчик Profile\_MigrateAnonymous, позволяющий переносить товары из временной корзины в корзину покупателя, когда пользователь входит в систему или регистрируется. Подробнее об этом написано в пункте 4.6.

**4.2 Компоновка и оформление web-страниц**

**4.2.1 Мастер-страницы**

Процесс создания web-приложения включает не только проектирование отдельных web-страниц, необходимо также обеспечить их интеграцию в завершённый, единообразный web-сайт. Каждая web-страница разрабатываемого web-приложения имеет содержит много одинаковых элементов, таких как различные заголовки, навигационные, поисковые и информационные панели. Для того чтобы использовать их на каждой web-странице, но при этом не размещать на ней их исходный код, а также обеспечить их одинаковое расположение, используются мастер-страницы.

Мастер-страницы представляют собой шаблоны web-страниц, которые могут определять фиксированное содержимое и объявлять часть web-страницы, в которую можно вставить динамическое содержимое. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. Элемент Content является дескриптором верхнего уровня для страницы содержимого, она не содержит дескрипторов <html>, <head> и <body>, они наследуются из соответствующей мастер-страницы.

Чтобы связать страницу содержимого с необходимой мастер-страницей необходимо выполнить следующие действия:

* В директиве Page страницы содержимого указать путь к файлу мастер-страницы: MasterPageFile= "~/MasterPages/AdminMasterPage.master";
* В дискрипторе Content страницы содержимого полю ContentPlaceHolderID присвоить идентификатор элемента ContentPlaceHolder, расположенному на;

Ниже приведён текст мастер-страницы AdminMasterPage.master, для web-страниц, входящих в интерфейс администратора музыкального Интернет-магазина:

<%@ Master Language= "C#" AutoEventWireup= "true" CodeFile= "AdminMasterPage.master.cs"

Inherits= "MasterPages\_AdminMasterPage"%>

<%@ Register Src="../UserControls/MenuAdmin.ascx" TagName= "MenuAdmin" TagPrefix= "uc"%>

<! DOCTYPE html PUBLIC "– //W3C //DTD XHTML 1.0 Transitional //EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

……………………………………………………………

<head runat= "server">

<title>Untitled Page</title>

</head>

<body>

<form id= "form1" runat= "server" style= "text-align: center;">

………………………………………………………….

<tr>

<td colspan= "2" style= "vertical-align: middle; text-align: center; height: 150px;">

……………………………………………………………………….

<div class= "HeaderPageFont2">Интерфейс администратора</div>

</td>

</tr>

<tr>

<td style= "width: 150px; vertical-align: top;">

…………………………………………….

</td>

<td style= "vertical-align: top; text-align: center; height: 700px; width: 850px;" class= "PageBackColor">

…………………………………………………………….

</asp: ContentPlaceHolder>

</td>

</tr>

<tr>

<td colspan= "2">

<span>copyright Луцкин Д.В. гр. №4231</span>

</td>

</tr>

………

</form>

</body>

</html>

Она определяет компоновку и расположение всех повторяющихся элементов и страницы-содержимого. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. Каждая страница содержимого для данной мастер страницы будет иметь следующий шаблон:

<%@ MasterPageFile= "~/MasterPages/AdminMasterPage.master" …%>

<asp: Content ID= "Content1" ContentPlaceHolderID= "ContentPlaceHolder1" Runat= "Server">

Содержимое соответствующей web-страницы.

</asp: Content>

На рисунках 8 и 9 представлена компоновка мастер-страниц AdminMasterPage.master и UserMasterPage.master, для web-страниц, входящих в интерфейс покупателя и администратора, соответственно.

4.2.2 Пользовательские элементы управления

Ещё одним средством стандартизации компоновки web-страниц ASP.NET, использованным при разработке дипломного проекта, являются пользовательские элементы управления (UserControl). Пользовательский элемент управления – это небольшая часть страницы, которая может содержать статический HTML-код и серверные элементы управления ASP.NET. Один и тот же пользовательский элемент управления может использоваться на множестве web-страниц. Применение пользовательских элементов управления значительно облегчает процесс отладки и модификации, а также позволяет более гибко использовать механизм кэширования. Подробнее о кэшировании пользовательских элементов управления описано в пункте 5.2.1.

Как уже было сказано выше, пользовательские элементы управления, как и web-страницы могут содержать в себе статический HTML-код и серверные элементы управления ASP.NET, а также имеют такой же жизненный цикл. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. Далее приведён пример регистрации пользовательского элемента управления MenuGenre.ascx в мастер-странице UserMasterPage.master:

Таблица

Строка 1Столбец 1

Заголовок

Строка 3 Столбец 1

Нижний колонтитул

Строка 2

Столбец 1

Содержимое конкретной web-страницы

Строка 2 Столбец 1



Рис. 8 "Компоновка мастер-страницы AdminMasterPage.master"

Таблица

Строка 1Столбец 1

Заголовок

Строка 3 Столбец 1

Нижний колонтитул

Строка 2

Столбец 1

Содержимое конкретной web-страницы

Навигационные, информационные и поисковые панели

Навигационные, информационные и поисковые панели

Строка 2 Столбец 1

Рис. 9 "Компоновка мастер-страницы UserMasterPage.master"

<%@ Register Src="../UserControls/MenuGenres.ascx" TagName= "MenuGenres" TagPrefix= "uc"%>, где

* Src – путь к исходному файлу;
* TagPrefix – префикс дескриптора;
* TagName – имя дескриптора;

После этого элемент управления можно использовать на web-странице:

<uc: MenuGenres ID= "MenuGenres1" runat= "server" />

При разработке дипломного проекта были созданы различные пользовательские элементы управления, их полный список и назначение приведены в пункте 4.1.1.

**4.2.3 Каскадные таблицы стилей и темы оформления**

Для того чтобы web-приложение имело единообразный визуальный стиль, необходимо настроить цвет фона каждой web-страницы и пользовательского элемента управления, задать шрифт и стиль каждой кнопки и надписи. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. Поэтому для стандартизации элементов дизайна при разработке дипломного проекта использовались каскадные таблицы стилей (CSS) и темы оформления.

Каскадные таблицы стилей состоят из правил. Каждое из правил содержит набор атрибутов и их значений, определяющих формат элемента управления. Имя каждого правила состоит из двух частей. Часть правила, находящаяся слева от точки, указывает дескриптор, к которому оно будет применяться. Если перед точкой ничего не стоит, то правило может применяться к любому дескриптору. Справа от точки указывается уникальное имя, которое идентифицирует правило, оно называется именем класса CSS.

Например, правило HeaderSpanFont определяет шрифт Times New Roman, кегель 14, цвет – красный.

HeaderSpanFont

{

font-family: Times New Roman;

font-size: 14pt;

color: Red;

}

Это правило можно применить для форматирования текста элемента управления ASP.NET или HTML-дескриптора:

<div class=" HeaderSpanFont">Текст </div>

<asp: Label ID=" Label1" runat=" server" Text="" CssClass=""></asp: Label>

В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении.

В отличие от каскадных таблиц стилей темы основаны на элементах управления, а не на HTML-дескрипторах. Ниже приведён фрагмент файла темы SkinFile.skin, в котором определяется внешний вид элемента управления GridView:

<asp: GridView runat= "server" SkinID= "GridViewStyle" BorderWidth= "1" BorderColor= "#dbdbc6">

<HeaderStyle BackColor= "#A52A2A" Font-Names= "Times New Roman" Font-Size= "12" ForeColor= "White" />

<RowStyle BackColor= "#F5F5DC" Font-Names= "Times New Roman" Font-Size= "12" HorizontalAlign= "Center" />

<AlternatingRowStyle BackColor= "#dbdbc6" Font-Names= "Times New Roman" Font-Size= "12" HorizontalAlign= "Center" />

<FooterStyle BackColor= "Aqua" HorizontalAlign= "Left" Font-Names= "Times New Roman" Font-Size= "10" Font-Bold= "true" />

………………………………………………….

<PagerStyle HorizontalAlign= "Center" Font-Names= "Times New Roman" Font-Size= "12" BackColor= "#A52A2A" Font-Bold= "true" ForeColor= "White" />

</asp: GridView>

У элемента управления может быть определено несколько вариантов оформления, для этого каждому варианту необходимо задать уникальное имя с помощью атрибута SkinID.

Применение каскадных таблиц стилей и тем оформления позволяет лучше стандартизировать оформление web-приложения и намного упрощает процесс модификации в дальнейшем.

**4.3 Элементы управления проверкой достоверности**

На некоторых страницах web-приложения от пользователя требуется ввести определённую информацию, например, регистрация нового покупателя или добавление альбома в каталог. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. Для этого используется встроенный клиентский JavaScript-код. Однако проверка выполняется и на стороне сервера. Это необходимо, чтобы избежать возможных ошибок в случае вмешательства в JavaScript-код на стороне клиента. Эти элементы запускаются после загрузки страницы, но перед любыми другими событиями.

Ниже приведён пример использования элементов управления проверкой достоверности при разработке дипломного проекта:

<asp: TextBox ID= "Email" runat= "server" MaxLength= "50"></asp: TextBox> <asp: RequiredFieldValidator ID= "EmailRequired" runat= "server" ControlToValidate= "Email" ErrorMessage= "Вы забыли указать E-mail" ToolTip= "E-mail is required."

……. ……………………………………………………………………

<asp: RegularExpressionValidator ID= "EmailRegular" runat= "server" ValidationGroup= "CreateUserWizard1" ErrorMessage= "Недопустимый формат E-mail" ControlToValidate= "Email" ValidationExpression="\S+@\S+\.\S+">\*</asp: RegularExpressionValidator>

<asp: ValidationSummary HeaderText= "Неправильно заполненные поля:" ID= "ValidationSummary1" Font-Size= "10" DisplayMode= "BulletList" ValidationGroup= "CreateUserWizard1" runat= "server" />

RequiredFieldValidator проверяет, чтобы пользователь не забыл ввести адрес электронной почты.

RegularExpressionValidator проверяет, чтобы введённый адрес соответствовал определённому формату, который определён с помощью регулярного выражения.

ValidationSummary отображает итоговую информацию об ошибках.

Применение элементов управления проверкой достоверности позволяет предотвратить ввод некорректной информации, а также экономит ресурсы сервера и ускоряет обратную связь с пользователем.

**4.4 Класс web-страницы с возможностью отслеживания обновления**

Когда пользователь нажимает на кнопку "Обновить" на панели управления браузера, происходит обновление текущей страницы. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении.

При этом среда ASP.NET не может отличить обратный вызов, например при нажатии кнопки на web-страницы, от обновления. Иногда такое поведение может нарушить логику работы программы и исказить данные.

Например, в разработанном web-приложении покупатель может удалять товары из покупательской корзины, при этом он ставит галочку напротив этого товара и нажимает кнопку удалить. В обработчике этой кнопки определяется номер строки в GridView, где отмечена галочка на удаление, и товар удаляется из корзины. После этого страница отправляется обратно пользователю. Если теперь будет нажата кнопка "Обновить" на панели браузера, то будет повторён предыдущий запрос и, соответственно будет удалена ещё один товар. Таким образом, при обновлении страницы после операции удаления существует вероятность удалить ещё одну товар. Естественно, что такое поведение недопустимо. Для того чтобы обойти исправить такое поведение, необходимо отличать обновление страницы от обратной отсылки после нажатия кнопки.

Для этого используется новый класс web-страницы RefreshPage, унаследованный от класса System. Web.UI. Page, в котором добавлена возможность распознавания обновления. Для этого используется HTTP-модуль, осуществляющий предварительную обработку страницы. Далее рассмотрен алгоритм распознавания обновления страницы.

Для каждого запроса, обслуживаемого в контексте сеанса, выдается уникальный и возрастающий номер. Страница ASP.NET создает билет непосредственно перед созданием ответа и сохраняет его в специальном скрытом поле, которое передаётся в браузер. Когда пользователь передает новый запрос скрытое поле (если оно существует) автоматически добавляется к запросу на сервер.

На веб-сервере новый HTTP-модуль проверки перехватывает событие AcquireSessionState (этот метод вызывается перед тем, как для клиента будет получена информация, специфичная для сеанса, использована для заполнения коллекции Session), получает текущий билет из скрытого поля, а затем сравнивает его с кэшированным последним выданным билетом. Последний выданный билет сохраняется в состоянии сеанса. Если номер текущего билета больше, чем последний выданный номер, или оба они равны нулю, то это обычный запроc. Больше модуль проверки обновления не выполняет никаких действий и пропускает запросы в неизмененном виде.

Если последний выданный билет больше или равен текущему билету, то запрос определяется как обновление страницы. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. Срок жизни этих данных совпадает со сроком запроса; поэтому любые данные удаляются при создании ответа. Используя статическое свойство HttpContext. Current можно получить доступ к контексту обрабатываемого запроса из любого класса, вовлеченного в этот процесс.

Http-модуль проверки обновления создает новую запись IsPageRefreshed в коллекции Items. Эта логическая переменная обозначает, запрашивалась ли страница с помощью обычного запроса или с помощью обновления. В следующем примере показана реализация http-модуля проверки обновления:

public class RefreshModule: IHttpModule

{

public void Init (HttpApplication app)

{

app. AcquireRequestState += new EventHandler(OnAcquireRequestState);

}

private void OnAcquireRequestState (object sender, EventArgs e)

{

HttpApplication app = (HttpApplication) sender;

………………………….

 // Проверить на обновление

RefreshAction. Check(ctx);

return;

}

}

Класс RefreshAction содержит логику, определяющую, является ли текущий запрос обновлением страницы. Если он распознаётся как обновление, то в коллекции Items объекта HttpContext появляется новая запись IsPageRefreshed, которой присваивается значение true.

public static void Check (HttpContext ctx)

{

EnsureRefreshTicket(ctx);

 // Получение предыдущего номера из объекта Session

int lastTicket = GetLastRefreshTicket(ctx);

…………………………………………………….

int thisTicket = GetCurrentRefreshTicket(ctx);

 // Сравнение номеров

if (thisTicket > lastTicket || (thisTicket==lastTicket && thisTicket==0))

{

UpdateLastRefreshTicket (ctx, thisTicket);

………………………………………….

}

else

{

ctx. Items[PageRefreshEntry] = true;

}

}

Класс RefreshPage содержит свойство IsPageRefresh:

public bool IsPageRefresh

{

get {

object o = HttpContext. Current. Items [RefreshAction. PageRefreshEntry];

if (o == null)

{

return false;

}

return (bool) o;

}

}

Наследую класс web-страниц от класса RefreshPage можно использовать это свойство и определять обновление страницы. Таким образом, можно отслеживать обновление web-страницы при удалении товаров из покупательской корзины. Соответственно, если обнаружено обновление страницы, то товар не удаляется. Код обработчика представлен ниже:

protected void UpdateButton\_Click (object sender, EventArgs e)

{

if (! IsPageRefresh && IsValid)

{

int numDeleted = 0;

…………………………………………………………………

{

if (((CheckBox) (row. FindControl ("DeleteCheck"))).Checked)

{

Profile. Cart. RemoveAt (row. RowIndex – numDeleted);

……………….

continue;

}

string regular = "[^0–9]";

string text = ((TextBox) (row. FindControl ("TextBoxQuantity"))).Text;

MatchCollection collect = Regex. Matches (text, regular);

if ((collect. Count!= 0) || (text. Length == 0))

{

continue;

}

Profile. Cart [row. RowIndex – numDeleted].Quantity = Convert. ToInt32 (((TextBox) (row. FindControl ("TextBoxQuantity"))).Text);

}

Profile. Save();

Session ["IsUpdateCart"] = true;

}

Диаграмма классов, используемых для отслеживания обновления web-страниц, приведена на рисунке 10.

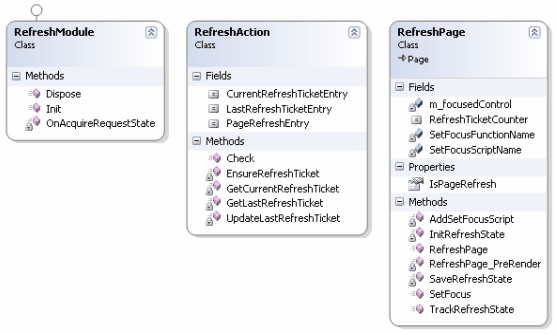


Рис. 10 "Диаграмма классов, используемых для отслеживания обновления web-страниц"

**4.5 Аутентификация и авторизация пользователей**

**4.5.1 Описание и настройка интерфейса Membership API**

Как уже было сказано в пункте 4.1.1, разработанное web-приложение содержит страницы, доступ к которым должен быть ограничен со стороны не аутентифицированных пользователей, либо аутентифицированных, но не обладающих необходимыми правами доступа.

Аутентификация – это процесс определения личности пользователя.

Авторизация – это процесс определения прав и ограничений, назначенных аутентифицированному пользователю. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. В данном пункте рассмотрен интерфейс Membership API и его применение при разработке дипломного проекта. Интерфейс Roles API описан в пункте 4.5.3.

Интерфейс Membership API предоставляет полный набор функций для хранения информации о пользователях и управления ею.

При разработке дипломного проекта использовались следующие классы, предоставляемые интерфейсом Membership API:

* Membership – предоставляет методы для управления пользователями;
* MembershipUser – предоставляет отдельного пользователя, записанного в хранилище данных. Этот объект содержит всю информацию о данном пользователе и возвращается несколькими методами класса Membership;

Также Membership API содержит набор элементов управления безопасность, предназначенных для создания web-страниц регистрации, входа в систему, смены и восстановления пароля. Все эти элементы полностью настраиваемы и расширяемы. По умолчанию каждый элемент работает на основе Membership API и аутентификации форм, но при необходимости любые события можно переопределить. Аутентификация форм рассмотрена в пункте 4.5.2.

Для того чтобы использовать интерфейс Membership API необходимо создать хранилище данных и настроить его на работу с ним. Для хранения удостоверений пользователей в базе данных web-приложения с помощью программы aspnet\_regsql.exe, поставляющейся вместе с.NET Framework 2.0, были созданы специальные таблицы: aspnet\_Membership, aspnet\_Users и aspnet\_Applications, а также хранимые процедуры для доступа к ним. Далее необходимо настроить поставщик Membership API, для обеспечения доступа к хранилищу данных. Это делается с помощью файла конфигурации web-приложения web.config:

<membership defaultProvider= "MyMembershipProvider">

<providers>

<add name= "MyMembershipProvider"

type= "System. Web. Security. SqlMembershipProvider"

………………………………………….

applicationName="/MusicInternetShop"

enablePasswordRetrieval= "true"

enablePasswordReset= "true"

maxInvalidPasswordAttempts= "5"

………………………………….

requiresUniqueEmail= "true"

passwordFormat= "Clear"

minRequiredPasswordLength= "7"/>

</providers>

</membership>

В элементе <membership><providers> определены следующие параметры настройки поставщика Membership:

* name – имя поставщика;
* applicationName – имя приложения, для которого поставщик управляет пользователями;
* type – тип поставщика. Выбирается в зависимости от используемого хранилища данных;
* connectionStringName – имя строки подключения к хранилищу данных. В данном случае используется строка поключения, описанная в элементе <connectionStrings>
* enablePasswordRetrieval – когда установлено в true, позволяет извлекать пароль из объекта MembershipUser;
* enablePasswordReset – когда установлено в true, включает функциональность для переустановки пользовательских паролей;
* maxInvalidPasswordAttempts – задаёт максимальное количество неудачных попыток входа в систему, прежде чем пользователь будет заблокирован;
* requiresQuestionAndAnswer – когда установлено в true, включает функциональность для восстановления забытого пароля используя контрольный вопрос и ответ, заданные при регистрации пользователя. Восстановленный пароль отправляется на почтовый ящик пользователя, указанный при регистрации;
* requiresUniqueEmail – указывает, должны ли адреса электронной почты быть уникальными для каждого пользователя webприложения;
* passwordFormat

После этого в web-приложении могут использоваться функции, предоставляемые Membership API. В следующем пункте подробно описана система аутентификации форм, основанная на интерфейсе Membership API, которая применяется в разработанном web-приложении.

**4.5.2 Аутентификация форм**

После того как соответствующим образом настроен интерфейс Membership API, можно реализовывать механизмы аутентификации и авторизации пользователей. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. ASP.NET обрабатывает cookie-набор и устанавливает контекст безопасности каждого запроса.

Аутентификация форм основана на мандатах. Это означает, что когда пользователь регистрируется, он получает так называемый мандат с базовой информацией о себе. Информация сохраняется в зашифрованном cookie-наборе, который прикрепляется к ответу, так что автоматически подтверждается при каждом последующем запросе.

Когда пользователь запрашивает web-страницу, недоступную анонимным пользователям, исполняющая система ASP.NET проверяет, доступен ли аутентифицирующий мандат формы. Если нет, то выполняется автоматическая переадресация на страницу входа в систему. На этой странице пользователь может ввести своё имя и пароль или перейти на страницу регистрации. Если пользователь успешно прошёл проверку, то исполняющая система ASP.NET автоматически устанавливает аутентифицирующий cookie-набор и переадресует пользователя на запрошенную страницу.

Как уже было сказано выше, аутентификация форм работает совместно с интерфейсом Membership API, это означает, что для создания страниц входа в систему и регистрации, можно использовать элементы управления безопасностью, такие как:

* Login – отображает текстовые поля имени пользователя и пароля, а также кнопку "Вход". Автоматически проверяет пользователя с помощью поставщика Membership;
* CreateUserWizard – в данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении;

Однако для того чтобы аутентификация форм работала необходимо настроить её в файле web.config. Ниже представлена настройка системы аутентификации форм для разработанного web-приложения:

<authentication mode= "Forms">

<forms name= "CookieName"

loginUrl= "Login.aspx"

……………………….

slidingExpiration= "true"

cookieless= "AutoDetect"

protection= "All"

……………….

path="/">

</forms>

</authentication>

В элементе <authentication> определены следующие параметры настройки:

* mode – указывает тип используемой аутентификации. Помимо аутентификации форм, существует windows-аутенификация, использующая для проверки учётные записи пользователей Windows;
* name – имя cookie-набора;
* loginUrl – определяет на какую страницу должен быть перенаправлен пользователь, чтобы войти в систему;
* timeout – период времени в минутах действительности cookie-набора. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. Если же они будут устаревать редко, то возникнет опасность их хищения и нерегламентированного использования;
* slidingExpiration – включает и отключает скользящее устаревание cookie-наборов. При включении устаревание cookie-наборов будет сбрасываться при каждом запросе страницы пользователем, то есть каждый запрос будет продлевать время жизни cookie-набора;
* cookieless – позволяет указать, где будут храниться мандаты аутентификации форм. Возможные значения UseCookie, UseUrl и AutoDetect. Использование cookie-набора требует, чтобы их использование поддерживал браузер клиента. В данном случае используется значение AutoDetect, устанавливающая применение cookie-наборов, если браузер клиента их поддерживает. В противном случае исполняющая система ASP.NET кодирует мандат аутентификации форм в запрошенный URL;
* protection – в данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении;
* defaultUrl – если пользователь был перенаправлен на страницу входа в систему с другой страницы web-приложения, то после успешного входа в систему он будет перенаправлен на запрошённую страницу. Однако, если пользователь сразу запросил страницу входа в систему, то после успешной аутентификации он будет направлен на страницу, указанную в этом параметре;

Система аутентификации позволяет отличить анонимных пользователей от аутентифицированных. Таким образом, мы можем закрыть доступ к определённым web-страницам и каталогам для анонимных пользователей. Чтобы это сделать необходимо в файле web.config в элементе <authorization> указать <deny users="?" />. Однако если аутентифицированные пользователи должны иметь разные уровни доступа к страницам, то необходимо как-то различать их. Для этого используется система авторизации и интерфейс Roles API, которые будут рассмотрены в следующем пункте. Также будет описано, как с помощью файла web.config открывать и закрывать доступ к разным каталогам и страницам приложения для разных групп аутентифицированных пользователей.

**4.5.3 Авторизация и настройка интерфейса Roles API**

Как уже было сказано в пункте 4.5.1, авторизацией называется процесс определения прав и ограничений, назначенных аутентифицированному пользователю. Другими словами, это процесс определения того, имеет ли аутентифицированный пользователь достаточные привилегии на выполнение того или иного действия. Таким действием может быть запрос web-страницы, доступ к ресурсу или выполнение специфичных для приложения задач.

Для того чтобы разграничить доступ разных пользователей к разным группам страниц web-приложения используется механизм ролей. Назначая права или ограничения для роли, мы тем самым назначаем их для каждого пользователя относящегося к этой роли. В разработанном web-приложении предусмотрено две роли:

* Users – к этой роли относятся покупатели музыкального Интернет-магазина. Когда покупатель регистрируется в Интернет-магазине, использую страницу регистрации, он автоматически приписывается к роли Users. Пользователи из группы Users имеют доступ ко всем общедоступным страницам, а также к страницам из подкаталога ProtectedPages. Описание этих web-страниц и их назначение приведено в пункте 4.1.1.
* Administrators – к этой роли относятся администраторы музыкального Интернет-магазина. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. Описание этих web-страниц и их назначение приведено в пункте 4.1.1.

Права доступа для каталогов указываются декларативно в файле web.config. Специальный HTTP-модуль UrlAutorizationModule просматривает эти правила и проверяет каждый запрос, чтобы гарантировать, что пользователю не будет доступен ресурс, доступ к которому для него закрыт. Авторизация этого типа называется авторизацией URL, потому что рассматривает только две детали – контекст безопасности пользователя и URL-ресурса, к которому пользователь пытается обратиться. Если доступ к странице запрещён, то пользователь будет перенаправлен на страницу входа в систему.

Правила авторизации указываются в элементе <authorization> файла web.config. Существует два типа правил: разрешить (allow) и запретить (deny).

В файле web.config разработанного web-приложения определены следующие правила авторизации:

1. Для всех web-страниц из корневого каталога открыт доступ для любого пользователя.

<authorization>

<allow users= "\*"/>

</authorization>

1. Ко всем web-страницам из подкаталога ProtectedPages доступ для не аутентифицированных пользователей, а также пользователей роли Administrators закрыт. К ним имеют доступ только пользователи, относящиеся к роли Users.

<location path= "ProtectedPages">

<system.web>

…………………

<deny users="?"/>

<deny roles= "Administrator"/>

<allow roles= "Users"/>

……………………

</system.web>

</location>

1. Ко всем web-страницам из подкаталога AdminProtectedPages доступ для не аутентифицированных пользователей, а также пользователей роли Users закрыт. К ним имеют доступ только пользователи, относящиеся к роли Administrators.

<location path= "AdminProtectedPages">

<system.web>

<authorization>

<deny users="?"/>

…………………….

<allow roles= "Administrator"/>

</authorization>

……………….

</location>

Для того чтобы обеспечить работу механизма авторизации на основе ролей необходимо настроить интерфейс Roles API. Как и в случае с Membership API с помощью программы aspnet\_regsql.exe в базе данных были созданы таблицы aspnet\_Roles и aspnet\_UsersInRoles, а также хранимые процедуры. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении.

После того как в базе данных созданы необходимые таблицы и хранимые, нужно сконфигурировать интерфейс Roles API и поставщик Roles в файле web.config. Для разработанного web-приложения настройка выглядит следующим образом:

<roleManager

defaultProvider= "MySqlProvider"

cacheRolesInCookie= "true"

cookieName=".MyRoleCookie"

cookieTimeout= "20"

……………………

cookieSlidingExpiration= "true"

cookieProtection= "All"

createPersistentCookie= "false">

<providers>

…………………………….

type= "System. Web. Security. SqlRoleProvider"

connectionStringName= "connStr"

applicationName="/MusicInternetShop"/>

</providers>

</roleManager>

В элементе <roleManager> определены следующие параметры настройки, многие из которых аналогичны, описанным в пункте 4.5.2:

* defaultProvider – указывает имя поставщика, используемого по;
* cacheRolesInCookie – если установлено в true, устанавливает сохранение информации о роли в cookie-наборе, вместо постоянного чтения информации о;
* cookieName – имя cookie-набора;
* cookieTimeout – время устаревания cookie-набора. Подробнее описано в пункте 4.5.2;
* cookieSlidingExpiration – включает и отключает скользящее устаревание cookie-наборов. Подробнее описано в пункте 4.5.2;
* cookieProtection – указывает уровень безопасности cookie-наборов. Подробнее описано в пункте 4.5.2;
* createPersistentCookie – если установлено в true, cookie-набор постоянно сохраняется на клиентской машине, иначе он существует только в течении сеанса и удаляется, когда пользователь закрывает браузер;

В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении.

В пунктах 4.5.1, 4.5.2 и 4.5.3 и были рассмотрены интерфейсы Membership API и Roles API, которые применялись для создание системы аутентификации и авторизации пользователей в разработанном web-приложении. На рисунке 11 представлена общая схема аутентификации и авторизации пользователей.

Клиентский запрос

Аутентифицирующий cookie-набор прикреплён?

Аутентифицирован?

Да

Нет

Форма входа в систему принимает имя и пароль пользователя

Аутентификация пользователя

Да

Нет

Авторизация пользователя

Авторизован?

Да

Нет

Разрешить доступ к ресурсу

Запретить доступ к ресурсу

Рис. 11 "Схема аутентификации и авторизации пользователей"

**4.6 Описание профилей пользователей**

Системы аутентификации и авторизации позволяют различать покупателей музыкального Интернет-магазина и разграничивать их права доступа к разным частям web-приложения. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. При разработке дипломного проекта для этих целей использовалось средство ASP.NET, которое называется профили.

Информация о профилях пользователей содержится в специальной таблице базы данных aspnet\_Profile, созданной с помощью программы aspnet\_regsql.exe. Средство профилей интегрировано с моделью аутентификации ASP.NET так, что информация из профиля автоматически извлекается для текущего пользователя, когда это необходимо и записывается в базу данных в конце текущего запроса, если эта информация изменялась. Таким образом, профили сохраняются в индивидуальных записях, каждая из которых уникально идентифицирована именем пользователя.

Для того чтобы использовать профили в web-приложении необходимо сконфигурировать поставщик и свойства профилей в файле web.config. Ниже приведён код настройки свойства профилей для разработанного web-приложения:

<profile defaultProvider= "MySqlProvider" automaticSaveEnabled= "false">

<providers>

<add name= "MySqlProvider"

type= "System. Web. Profile. SqlProfileProvider"

……………………………………………

applicationName="/MusicInternetShop"/>

</providers>

<properties>

<add name= "Cart" type= "ComponentsShoppingCart. ShoppingCart" serializeAs= "Binary" allowAnonymous= "true"/>

<……………………………………………………………………

<add name= "SumPurchase" type= "decimal" serializeAs= "Binary"/>

</properties>

</profile>

Настройка поставщика профилей аналогична настройке поставщика ролей и содержит информацию об имени и типе поставщика, строке подключения к базе данных и имени web-приложения.

Элемент <properties> специфицирует информацию, которая будет храниться в профиле. Помимо переменных стандартных типов C# в профиле можно сохранять также объекту пользовательских классов. Для хранения информации пользователя были разработаны три пользовательских класса. На рисунке 12 представлена диаграмма этих классов.

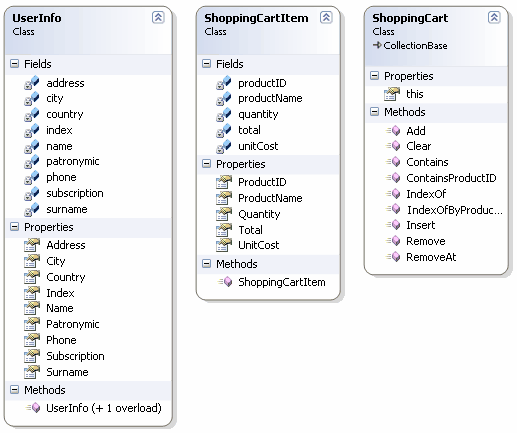


Рис. 12 "Диаграмм классов, используемых в профилях"

Объект класс UserInfo содержит информацию о фамилии, имени, отчестве покупателя, адресе и почтовом индексе, а также контактном телефоне.

В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении.

Переменные классов объявлены как приватные, доступ к ним осуществляется через общедоступные свойства. Это позволяет использовать механизм привязки данных, который описан в пункте 3.3. Исходный код классов приведён в приложении №3.

Также в профиле сохраняется отдельная переменная SumPurchase типа decimal, она хранит общую сумму покупок пользователя в Интернет-магазине и используется для определения скидки.

В элементе <properties> файла web.config приведены следующие настройки свойств профиля:

* name – имя свойства;
* type – тип свойства;
* serializeAs – формат, используемый при сериализации. Возможные значения String, Binary, Xml. В данном случае используется двоичный формат Binary;
* allowAnonymous – если установлено в true, то это свойство применяется также и в анонимных профилях. Об анонимных профилях будет описано ниже.

Среда ASP.NET предоставляет программный интерфейс Profiles API для управления профилями пользователей. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении.

Ниже приведён код добавления товара в корзину покупателя:

protected void ImageButtonAddToCart\_Click (object sender, ImageClickEventArgs e)

{

if (! IsPageRefresh && (Roles. IsUserInRole ("User") ||! User. Identity. IsAuthenticated))

{

……………………………………………….

{

if (\_item. ProductID == productID)

{

 // Если такой товар уже присутствует в корзине, то его кол-во увеличивается на единицу

\_item. Quantity++;

Profile. Save();

………………………….

Session ["IsUpdateCart"] = true;

return;

}

}

 // Если такой товар отсутствует в корзине, то он добавляется

ShoppingCartItem item = new ShoppingCartItem (productID, productName, unitCost, 1);

Profile. Cart. Add(item);

Profile. Save();

Session ["IsUpdateCart"] = true;

}

TrackRefreshState();

}

Как уже было сказано, выше свойство Cart может использоваться с анонимными профилями. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. Также необходимо разрешить анонимную идентификацию с помощью элемента <anonymousIdentification enabled= "true"/>. При этом сохраняется временный идентификатор пользователя, с помощью которого можно получить доступ к анонимному профилю. Механизм миграции анонимных профилей реализован через событие Profile\_MigrateAnonymous, которое обработано в файле Global.asax, описанном в пункте 4.1.2. Это событие инициируется всякий раз, когда становится доступным анонимный идентификатор и текущий пользователь аутентифицируется. Анонимный профиль извлекается с помощью вызова Profile. GetProfile() с передачей анонимного идентификатора, данные из него переносятся в постоянный профиль, а затем анонимный идентификатор и профиль удаляются. Соответствующий код приведён ниже:

void Profile\_MigrateAnonymous (Object sender, ProfileMigrateEventArgs pe)

{

ProfileCommon anonProfile = Profile. GetProfile (pe. AnonymousID);

if ((anonProfile. Cart. Count!= 0) && (Roles. IsUserInRole ("User")))

{

foreach (ComponentsShoppingCart. ShoppingCartItem \_item in anonProfile. Cart)

{

……………………………………………………………

{

Profile. Cart [Profile. Cart. IndexOfByProductID (\_item. ProductID)].Quantity += \_item. Quantity;

}

else

{

…………………………….

}

}

Profile. Save();

}

try

{

Session ["IsUpdateCart"] = true;

}

catch (HttpException)

{}

System. Web. Profile. ProfileManager. DeleteProfile (pe. AnonymousID);

AnonymousIdentificationModule. ClearAnonymousIdentifier();

}

**4.7 Основные элементы интерфейса музыкального Интернет-магазина**

**4.7.1 Регистрация нового покупателя**

Регистрация нового покупателя происходит на странице Registration.aspx, изображение которой представлено на рисунке 13. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. Для сохранения дополнительная информация о пользователе было переопределено событие OnCreatedUser, которое инициируется после того, как мастер создал нового пользователя. Вся дополнительная информация сохраняется в профиле пользователя.

Также введена защита от автоматической регистрации. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. Для реализации защиты было переопределено событие OnCreatingUser, которое инициируется перед тем как мастер создаёт нового пользователя. Если код с картинки не соответствует коду, указанному пользователю, то регистрация отменяется и выдаётся соответствующее сообщение.

Каждое поле для ввода связано с необходимыми элементами проверки достоверности для определения корректности вводимой пользователем информации. В случае обнаружения ошибок соответствующие поля отмечаются, и пользователю возвращается сообщение об ошибке.

**4.7.2 Личные данные покупателя**

Покупатель может изменять информацию о себе, сменить пароль и контрольный вопрос и ответ для восстановления пароля на странице UserPersonalData.aspx. При изменении личных данных информация, содержащаяся в профиле обновляется. Для смены пароля и контрольного вопроса и ответа, использовались методы предоставляемые интерфейсом MembershipAPI – ChangePassword() и ChangePasswordQuestionAndAnswer() соответственно. Изображения страницы UserPersonalData.aspx представлено на рисунке 14.

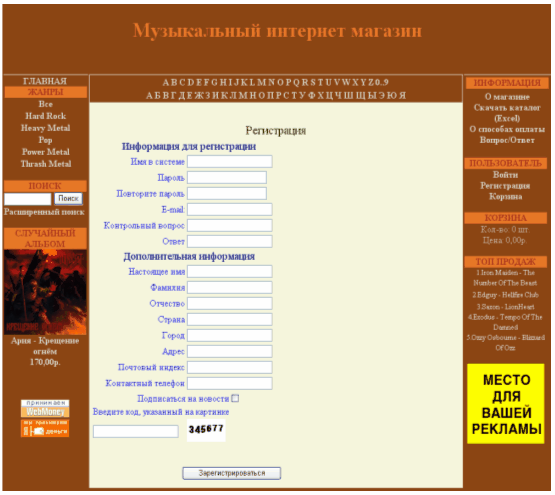


Рис. 13 "Страница регистрации нового покупателя"

**4.7.3 Каталог продукции**

Покупатель может просматривать каталог продукции на странице CatalogList.aspx. Переход на страницу осуществляется автоматически, когда покупатель выбирает один из критериев поиска товара, например, нажимает на ссылку в меню жанров. При этом на странице отображаются все товары, которые удовлетворяют этому критерию. Вывод осуществляется постранично по 15 записей. Для выборки данных используются методы, представленные в компоненте доступа к данным CatalogDB. На рисунке 15 представлено изображение страницы CatalogList.aspx, на которой выведены все альбомы, относящиеся к жанру Heavy Metal.

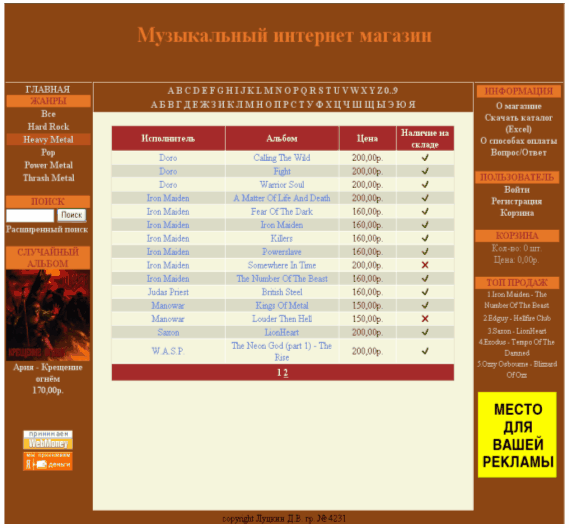


Рис. 15 "Страница каталога продукции"

На рисунке 16 представлено изображение страницы CatalogList.aspx с использованием расширенного поиска продукции.

**4.7.4 Страница подробностей о продукции**

Выбирая конкретный альбом на странице каталога, покупатель переходит на страницу ProductDetails.aspx, отображающую подробную информацию о выбранном альбоме, а также позволяющей добавить альбом в корзину товаров, просмотреть отзывы об альбоме или оставить свой. Изображение страницы ProductDetails.aspx представлено на рисунке 16. Ознакомившись с альбомом, покупатель может добавить альбом вы корзину товаров, нажав на иконку с изображением корзины.

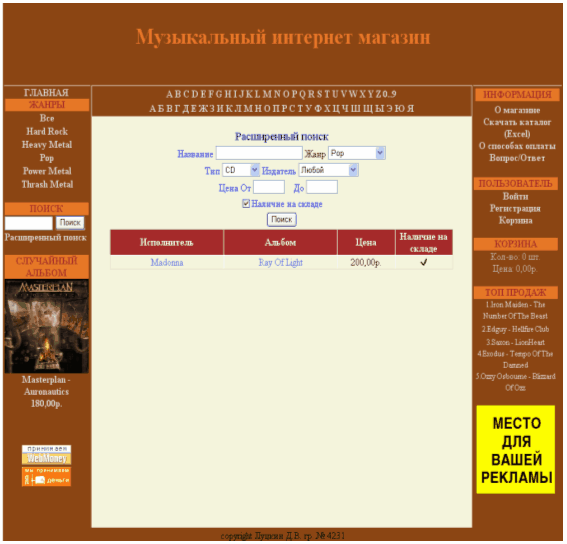


Рис. 15 "Страница каталога продукции с использованием расширенного поиска"

**4.7.5 Отзывы о продукции**

Нажав на ссылку "Просмотреть отзывы" покупателель перейдёт на страницу Responses.aspx, где на которой будут отображены все отзывы покупателей о данном альбоме. На рисунке 17 представлено изображение страницы Responses.aspx. Покупатель может также сам оставить отзыв об альбоме, для этого необходимо нажать на ссылку "Написать отзыв" и заполнить появившуюся форму. Оставлять отзывы об альбомах могут только аутентифицированные пользователи. Пример представлен на рисунке 18.

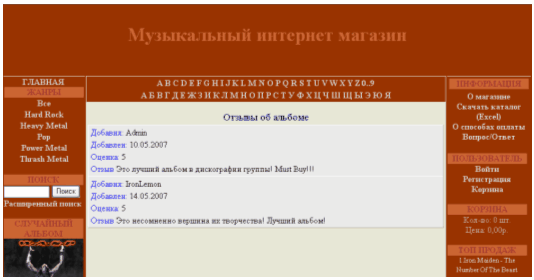


Рис. 17 "Отзывы об альбоме"

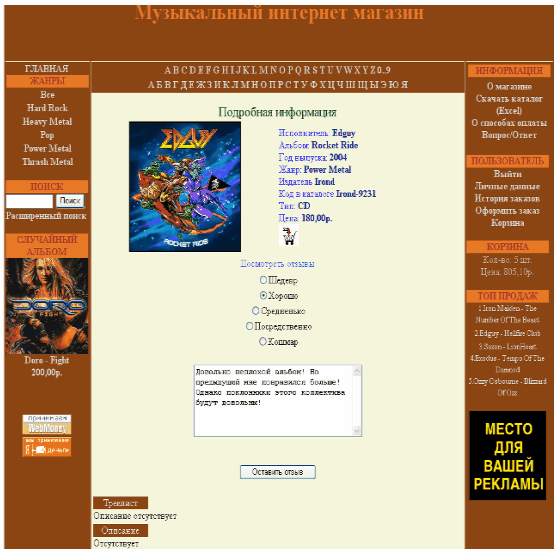


Рис. 18 "Добавление отзыва об альбоме"

**4.7.6 Корзина покупателя**

Перейдя по ссылке "Корзина", расположенной на панели "Пользователь", покупатель перейдёт на страницу ShoppingCart.aspx. На этой странице отображаются информация обо всех альбомах, находящихся в данный момент в корзине покупателя, а именно название, цена, количество. Также отображается начисленная скидка и итоговая цена всех альбомов с учётом этой скидки. Покупатель может изменять количество того или иного альбома в корзине, а также удалять их из корзины. Изображение страницы ShoppingCart.aspx приведёно на рисунке 19.

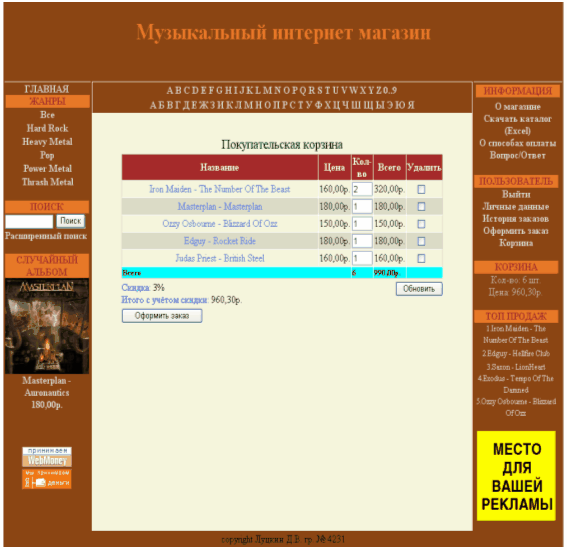


Рис. 19 "Корзина покупателя"

После того как покупатель определиться со списком желаемых альбомов, он может перейти к оформлению заказа, нажав на кнопку "Оформить заказ".

Кроме того, информация об общем количестве товаров в корзине и текущей цене заказа с учётом скидки выводится на панели "Корзина". Данная панель представляет собой пользовательский элемент управления ShoppingCart.ascx. Все изменения в корзине товаров также немедленно отражаются на этой панели.

**4.7.7 Оформление заказа**

Нажав на кнопку "Оформить заказ", покупатель переходит на страницу Order.aspx, изображение которой представлено на рисунке 20.

На ней кроме информации о товарах, находящихся в корзине и цене заказа, выводится информация о покупателе, которая будет использована администрацией магазина для доставки заказа. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении.

Если ничего менять не надо, покупатель выбирает желаемый способ оплаты заказа и нажимает на кнопку "Разместить заказ". После этого заказ размещён, и изменить его нельзя, а корзина товаров очищается.

**4.7.8 История заказов**

Покупатель может просматривать историю всех сделанных им заказов и следить за ходом их выполнения. Для этого необходимо перейти по ссылке "История заказов", размещённой на панели "Пользователь". После этого он перейдён на страницу HistoryOrders.aspx, изображённую на рисунке 21.



Рис. 21 "История заказов"

На странице выводится информация о каждом заказе, а именно дата заказа, состояние заказа, дата отправки на почту, тип оплаты, цена. Также выводится общая сумма всех заказов. Для того чтобы посмотреть подробности заказа необходимо перейти по ссылке "Подробности" напротив нужного заказа.

После этого покупатель будет перемещён на страницу HistoryOrderDetails.aspx, которая изображена на рисунке 22.

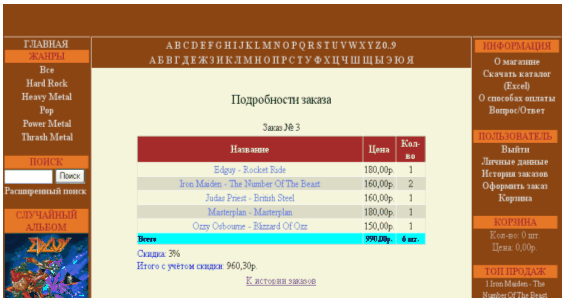


Рис. 22 "Подробности заказа"

На этой странице выводится информации о товарах заказа, а также скидка, которая была начислена на момент оформления заказа.

**4.7.9 Страница вопросов покупателей**

Любой пользователь как аутентифицированный, так и анонимный может задавать любые вопросы, касающиеся работы музыкального Интернет-магазина. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. Перейдя по ссылке "Задать свой вопрос" пользователь сможет задать свой вопрос, заполнив необходимые поля. Страница AddQuestion.aspx изображена на рисунке 24.

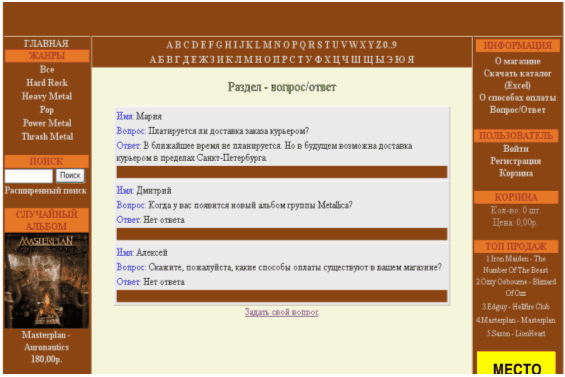


Рис. 23 "Вопросы покупателей"

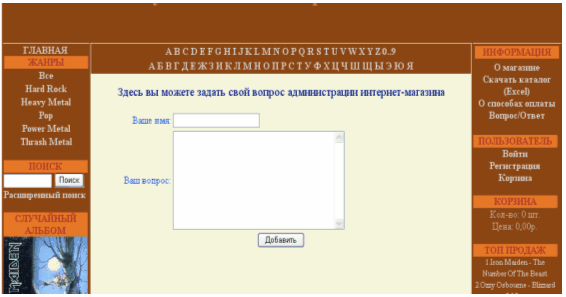


Рис. 24 "Размещение вопроса администрации музыкального Интернет-магазина"

**4.8 Основные элементы интерфейса администратора музыкального Интернет-магазина**

**4.8.1 Управление каталогом товаров**

Администратор музыкального Интернет-магазина может добавлять, модифицировать и удалять товары в каталоге. Чтобы добавить альбом в каталог необходимо перейти по ссылке "Альбомы -> Добавить альбом" в меню "Функции". После этого администратор будет перемещён на страницу AddProduct.aspx, изображённую на рисунке 25.

После заполнения необходимых полей и нажатия кнопки "Добавить", альбом будет добавлен в каталог. Описание товара и треклист могут вводиться как вручную так и из файла. Поддерживаются файлы с расширением.txt и xml.

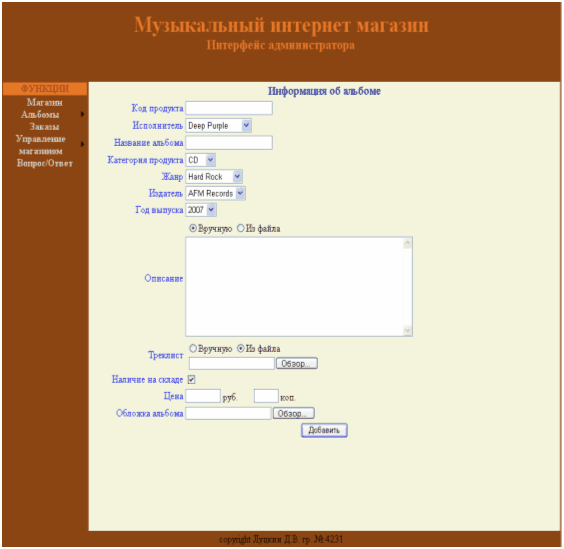


Рис. 25 "Добавление альбома в каталог"

Для описания альбома формат xml-файла должен быть следующим:

<description>

… описание…

</description>

Для треклиста альбома xml-файла должен быть следующим:

<tracklist>

<track> название песни №1</track>

<track> название песни №2</track>

…

</tracklist>

Если файл текстовый или песни вводятся вручную, каждое название песни следует располагать на новой строке.

Для того чтобы изменить или удалить альбом в каталоге необходимо перейти по ссылке "Альбомы -> Изменить информацию об альбоме / Удалить альбом" в меню "Функции". В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. Удалить можно только альбом, который ещё ни разу не заказывали. Это необходимо для поддержания целостности информации в истории заказов, а также в текущих заказов, находящихся в обработке. Если в данный момент альбом отсутствует на складе, необходимо просто снять галочку "Наличие на складе".

На рисунке 26 изображена страница UpdateDeleteProduct.aspx.

Также администратор может управлять структурой каталога, а именно добавлять изменять и удалять информацию об исполнителях, музыкальных жанрах, категориях продукции и фирмах-издателях. Для этого необходимо выбрать соответствующий подпункт в элементе меню "Управление магазином". На рисунке 27 изображена страница ManagementArtist.aspx, предназначенная для управления исполнителями. Другие страницы управления каталогом действуют аналогично.

Удалить какой-то элемент на страницах управления каталогом можно только, если он не используется для описания свойств товара. Это требуется для подержания ссылочной целостности БД. Поэтому следует осторожно пользоваться этими инструментами и исправлять обнаруженную ошибку как можно раньше.

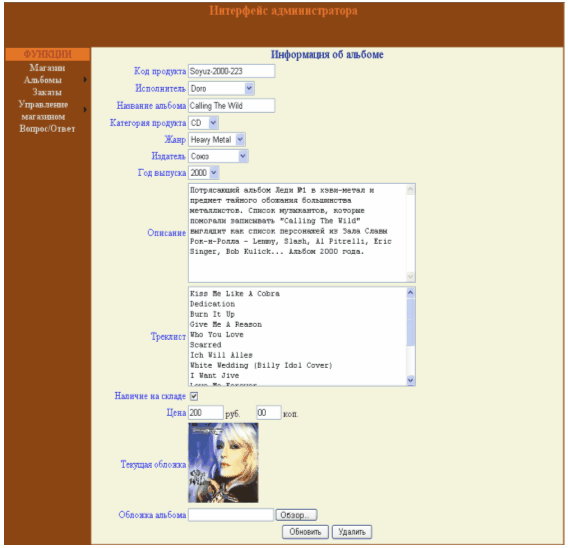


Рис. 26 "Изменение и удаление альбомов в каталоге"

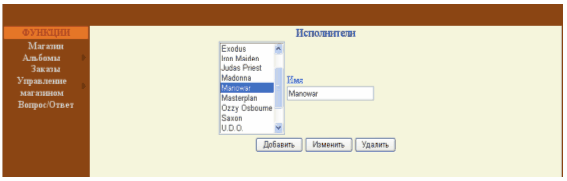


Рис. 27 "Управление исполнителями"

**4.8.2 Управление заказами**

Для управления заказами администратор должен перейти по ссылке "Заказы" в меню "Функции". После этого он будет перемещён на страницу Orders.aspx. Используя панель поиска он может отсортировать заказы по дате и состояния выполнения. Изображение страницы Orders.aspx приведено на рисунке 28.

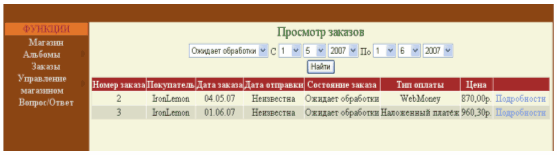


Рис. 28 "Заказы покупателей"

Она содержит общую информацию о заказе. Чтобы посмотреть детали заказа, необходимо перейти по ссылке "Подробности" напротив соответствующего заказа. После этого администратор будет перемещён на страницу OrderDetails.aspx. Здесь администратор может изменить состояние выполнения, а также распечатать бланк заказа. Изображение страницы OrderDetails.aspx приведено на рисунке 29.

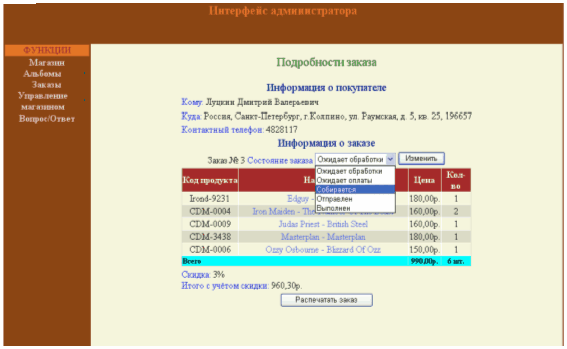


Рис. 29 "Подробности заказа"

**4.8.3 Ответы на вопросы покупателей**

Администратор может также отвечать на вопросы покупателей. Для этого необходимо перейти по ссылке "Вопрос / Ответ" в меню "Функции" на страницу AdminFAQ. Изображение страницы приведено на рисунке 30.



Рис. 30 "Страница ответов на вопросы покупателей"

**5. Производительность и масштабируемость web-приложения**

**5.1 Способы повышения производительности и масштабируемости**

Для любого web-приложения, которое одновременно обслуживает большое число пользователей, важна скорость работы. Особенно это важно для приложений электронной коммерции, в конкурентной борьбе за клиентов это имеет огромное значение. Ведь если пользователю придется подолгу ждать загрузки страницы, то он, скорее всего, предпочтёт пользоваться услугами другого Интернет-магазина, который, предлагая аналогичные услуги, обеспечивает более высокую производительность.

Существует много различных способов повышения производительности, как аппаратных, так и программных. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении.

Одним из важнейших способов является кэширование. Кэширование – это техника хранения в памяти копии некоторой информации, которую дорого пересоздавать, например, результат запроса к базе данных. Используя механизм кэширования, повышается как производительность, так и масштабируемость. Производительность растёт благодаря значительному снижению времени, необходимого на извлечение информации, а масштабируемость – за счёт того, что обходятся узкие места в системе – такие как соединение с базой данных. В результате приложение может обслужить больше запросов страниц одновременно при меньшем количестве операций с базой данных.

В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении.

В данной главе была обозначена важность обеспечения высоких показателей производительности и масштабируемости для систем электронной коммерции, а также приведены некоторые способы их достижения, такие как кэширование и разбиение на страницы. В следующих разделах будут приведены конкретные способы, которые использовались при разработке web-приложения "Музыкальный Интернет-магазин".

**5.2 Применение механизма кэширования**

**5.2.1 Кэширование вывода**

В ASP.NET поддерживается два типа кэширования, наиболее простой из них кэширование вывода. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. Кэширование вывода обеспечивает максимальный прирост производительности, так как исключаются все накладные расходы, связанные с выполнением кода.

При разработке дипломного проекта этот способ кэширования применялся для некоторых пользовательских элементов управления, например, пользовательский элемент управления TopSales.ascx, который отображает пять наиболее продаваемых товаров. Он отображается на каждой странице, доступной пользователю, и если не применять кэширование, то при каждом запросе любой страницы любым пользователем необходимые данные извлекались бы из базы данных. Кэширование вывода применяется декларативно:

<%@ OutputCache Duration=" 3600" VaryByParam=" None" Shared=" true"%>

Если добавить эту строку в файл дескрипторов TopSales.ascx, то при первом запросе сгенерированный HTML-код пользовательского элемента управления будет сохранён в памяти на один час, и при последующих запросах пользователей будет вставляться в мастер-страницу в готовом виде, без обращения к базе данных. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. Если пользовательский элемент управления используется на нескольких страницах, то по умолчанию, в КЭШе сохраняется копию для каждой страницы отдельно. Чтобы не хранить лишние копии, указан параметр Shared = "true". Таким образом, в кэше сохраняется одна копия элемента для всех страниц, где он используется.

Приведённый выше пример, показывает как, применяя кэширование вывода можно в десятки и сотни раз уменьшить частоту обращения к базе данных, что позволит web-приложению одновременно обрабатывать большее запросов пользователей и с более высокой скоростью.

**5.2.2 Кэширование данных**

Несмотря на простоту применения и эффективность кэширования вывода, иногда применять его нельзя. Так как кэшированная информация, страница или пользовательский элемент управления, хранятся в виде готового HTML-кода, то мы не имеем возможности взаимодействовать с ней на стороне сервера. В таком случае необходимо применять кэширование данных. Например, можно сохранять в памяти результаты запроса к базе данных. При следующем запросе страницы, обращения к базе данных не будет, вместо этого необходимая информация будет взята из кэша, однако код серверной стороны выполнится и HTML-код будет сгенерирован заново.

Кэширование данных широко применялось при разработке дипломного проект. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. При каждом последующем запросе обращение к базе данных не происходит, а информация прямо из КЭШа вставляется в элементы управления.

Ниже приведён пример наполнения списка "Музыкальные жанры":

DataSet ds;

if (Cache ["GenresList"]!= null)

{

ds = (DataSet) Cache ["GenresList"];

}

else

{

ds = GenresDB. GetGenres();

……………….

{

Response. Redirect ("~/ErrorPage.aspx");

}

Cache. Insert ("GenresList", ds, null, DateTime. Now. AddMinutes(60), TimeSpan. Zero);

}

GenreList. DataSource = ds;

…………………………

Если необходимые данные в кэше существуют, то они используются для наполнения списка, если их нет, например, это первый запрос или истёк срок нахождения элемента в кэше, то выполняется соответствующий запрос к базе данных. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении.

Как уже было сказано выше, при кэшировании данных происходит выполнение кода на сервере, так мы можем узнать, какие критерии для поиска выбрал пользователь, произвести соответствующий поиск и вывести результат.

Если запросы к базе данных осуществляются с помощью элементов управления источниками данных, то в этом случае также можно использовать механизм кэширования, так как они поддерживают встроенное кэширование данных. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. Ниже приведён пример настройки кэширования элемента управления источником данных, осуществляющего поиск по первой букве в названии продукции:

<asp: ObjectDataSource ID= "ObjectDataSource3" runat= "server"

TypeName= "ComponentsDB. CatalogDB" SelectCountMethod= "GetCountByLetter"

………………………………………………………………………………………….

OnSelected= "ObjectDataSource\_Selected" EnableCaching= "true" CacheDuration= "360">

<SelectParameters>

<asp: QueryStringParameter Name= "Letter" QueryStringField= "Letter" Type= "Char" />

</SelectParameters>

</asp: ObjectDataSource>

…………………………………………………………………………….

CacheDuration= "360" – устанавливает срок нахождения данных в кэше.

В предыдущих двух разделах был рассмотрен механизмы кэширования в ASP.NET (кэширование данных и кэширование вывода), в данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении, обеспечив тем самым наибольшее быстродействие и эффективность использование ресурсов сервера.

**5.3 Настраиваемое разбиение на страницы с помощью ObjectDataSource**

Как уже было сказано в пункте 4.7.3 информация из каталога выводится на экран в виде таблицы (GridView) постранично по 15 записей. Элементы управления GridView и ObjectDataSource поддерживают автоматическое разбиение на страницы, однако нагрузка на базу данных при этом не уменьшается. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. При этом на базу данных выпадает большая нагрузка, связанная с тем, что необходимо выбирать каждый раз весь объём информации.

Чтобы обойти это ограничение при разработке дипломного проекта использовалось настраиваемое разбиение на страницы, при котором из базы данных выбиралась информация только для текущей страницы. Настраиваемое разбиение на страницы, используется совместно с кэшированием данных, поэтому в лучшем случае, обращение к базе данных вообще не потребуется.

Для примера рассмотрим процесс автоматического разбиения на страницы при отображении каталога альбомов, отсортированных по жанру.

Необходимо настроить элемент управления источником данных на автоматическое кэширование:

<asp: ObjectDataSource ID= "ObjectDataSource1" runat= "server"

TypeName= "ComponentsDB. CatalogDB" SelectCountMethod= "GetCountByGenre" SelectMethod= "GetPageProductsByGenre" EnablePaging= "true" OnSelected= "ObjectDataSource\_Selected" EnableCaching= "true" CacheDuration= "360">

<SelectParameters>

<asp: QueryStringParameter Name= "GenreID" QueryStringField= "GenreID"

Type= "Int32" />

</SelectParameters>

</asp: ObjectDataSource>

где

EnablePading = "true" – включает разбиение на страницы в источнике данных.

SelectCountMethod = "GetCountByGenre" – метод для подсчёта общего кол-ва строк в таблице. Данный метод необходим, для того чтобы GridView мог создать корректное количество ссылок страниц на панели навигации.

В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении.

Метод GetCountByGenre:

public static int GetCountByGenre (int GenreID)

{

SqlConnection conn = new SqlConnection (WebConfigurationManager. ConnectionStrings ["connStr"].ConnectionString);

SqlCommand cmd = new SqlCommand ("GetCountByGenre", conn);

……………………………………………………

cmd. Parameters. Add ("GenreID", SqlDbType. Int);

cmd. Parameters ["GenreID"].Value = GenreID;

int count=-1;

try

{

conn. Open();

count = (int) cmd. ExecuteScalar();

}

……………

{

conn. Close();

}

finally

{

conn. Close();

}

return count;

}

Хранимая процедура GetCountByGenre

CREATE PROCEDURE GetCountByGenre (@GenreID INT)

AS

……………………………….

WHERE GenreID = @GenreID

GO

Метод GetPageProductsByGenre вызывает хранимую процедуру GetPageProductsByGenre, которая возвращает данные для текущей страницы GridView.

Метод GetPageProductsByGenre:

public static DataSet GetPageProductsByGenre (int startRowIndex, int maximumRows, int GenreID)

{

SqlConnection conn = new SqlConnection (WebConfigurationManager. ConnectionStrings ["connStr"].ConnectionString);

SqlCommand cmd = new SqlCommand ("GetPageProductsByGenre", conn);

cmd. CommandType = CommandType. StoredProcedure;

cmd. Parameters. Add ("Start", SqlDbType. Int);

cmd. Parameters ["Start"].Value = startRowIndex;

……………………………………………………

cmd. Parameters ["Count"].Value = maximumRows;

cmd. Parameters. Add ("GenreID", SqlDbType. Int);

cmd. Parameters ["GenreID"].Value = GenreID;

DataSet ds = new DataSet();

SqlDataAdapter da = new SqlDataAdapter();

………………………….

try

{

conn. Open();

da. Fill(ds);

}

catch (SqlException)

{

conn. Close();

return null;

}

finally

{

conn. Close();

}

return ds;

}

Хранимая процедура GetPageProductsByGenre:

CREATE PROCEDURE GetPageProductsByGenre (@Start INT, @Count INT, @GenreID INT)

AS

CREATE TABLE #TempProductsList

(

……………….

ProductID INT,

ArtistName NVARCHAR(50),

ArtistID INT,

……………………….

UnitCost MONEY,

Presence BIT

)

INSERT INTO #TempProductsList

(ProductID, ArtistName, ArtistID, AlbumName, UnitCost, Presence)

SELECT ProductID, ArtistName, Catalogue. ArtistID, AlbumName, UnitCost, Presence

………………………….

WHERE GenreID = @GenreID AND Catalogue. ArtistID = Artists. ArtistID

ORDER BY ArtistName

DECLARE @FromNum INT

DECLARE @ToNum INT

SET @FromNum = @Start

……………………………

SELECT ProductID, ArtistName, ArtistID, AlbumName, UnitCost, Presence FROM #TempProductsList

WHERE Num >= @FromNum AND Num <= @ToNum

GO

В данном пункте был рассмотрен метод настраиваемогого разбиения на страницы с использованием GridView и ObjectDataSource, который позволяет выбирать из базы данные только конкретной страницы, а не целиком для всей таблицы. Тем самым, снижая объём данных, выбираемых из базы данных.

**6. Развёртывание web-приложения и модели компиляции**

**6.1 Развёртывание web-приложения**

В самом простом варианте процесс развёртывания web-приложения ASP.NET заключается в копировании структуры каталога вашего приложения и его файлов на целевой сервер. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. Ниже описаны все действия, необходимые для развёртывания web-приложения "Музыкальный Интернет магазин":

* Прежде всего, нужно убедиться, что на целевой ЭВМ установлено необходимое ПО (ОС Windows XP Professional, Windows 2000 Server или Windows Server 2003, IIS 5.0 или 6.0 в зависимости от ОС.NET Framework 2.0, СУБД SQL Server 2000). Если какой-то компонент отсутствует, то его необходимо установить.
* В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении.
* Необходимо скопировать структуру каталога приложения и его файлы на целевой сервер.
* Создать виртуальный каталог на основе каталога приложения. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. Если на сервере установлено несколько версий.NET Framework, необходимо установить для виртуального каталога именно версию 2.0, это можно сделать в его свойствах на вкладке ASP.NET.
* Установить права учётной записи Windows для пользователя рабочего процесса ASP.NET на доступ к каталогам приложения. В IIS 5.0 рабочий процесс называется aspnet\_wp.exe и по умолчанию работает от имени учётной записи ASP.NET Machine Account, в IIS 6.0 – w3wp.exe, учётная запись – Network Service. На корневой каталог приложения необходимо дать право на чтение, а на папку AlbumImage дополнительно установить права на запись и изменение. Это можно сделать на вкладке "Безопасность" в свойствах папки.
* Далее нужно создать и сконфигурировать базу данных для приложения. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении.
* Последним шагом, возможно, понадобится внести какие-то изменения в файл web.config, например, модифицировать строку соединения с базой данных. По умолчанию подразумевается, что используется сервер localhost и используется интегрированная аутентификация для подключения к базе данных.

Таким образом, для развёртывания web-приложения "Музыкальный Интернет-магазин" необходимо скопировать каталог с приложением на целевой сервер, сконфигурировать виртуальный каталог и задать необходимые права учётной записи ASP.NET, а также создать и сконфигурировать базу данных.

**6.2 Модели компиляции**

В предыдущем разделе был подробно описан процесс развёртывания web-приложения. В связи с этим необходимо упомянуть ещё один аспект – модели компиляции. Web-приложение всегда выполняется в виде скомпилированной версии, что позволяет повысить производительность. При развёртывании мы можем выбрать, где именно будет происходить компиляция.

Если просто скопировать каталог приложения на целевой сервер со всеми файлами дескрипторов и файлами кода, то компиляция будет выполняться на целевом сервере. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. Но при этом динамическая компиляция происходит при первом запросе после изменения, поэтому для него потребуется больше времени, чем для последующих запросов.

Существует также возможность развёртывать web-приложение уже в скомпилированном виде, например, если необходимо скрыть доступ к исходным кодам. Эта модель получила название предварительной компиляции.

Чтобы предварительно скомпилировать web-приложение, необходимо использовать отдельный инструмент для компиляции, называемый aspnet\_compiler.exe, который храниться в каталоге Microsoft.NET Framework. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении. Далее каталог с приложением можно копировать на целевой сервер, и выполнять шаги, описанные в предыдущем пункту, никаких дополнительных действий выполнять не нужно.

**7. Разработка бизнес-плана**

**7.1 Резюме**

Целью дипломного проекта является создание программного продукта под общим названием "Музыкальный Интернет-магазин". Данный продукт предназначен для организации работы магазина по продаже музыкальной продукции, осуществляющую свою деятельность только в сети Интернет или использующий её в качестве дополнительного канала сбыта продукции.

В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес, указанный в конце дипломной работы, в заключении.

В ходе оценки экономической эффективности инвестиционного проекта с помощью критерия NPV (чистая приведённая стоимость), было установлено, что значение NPV выходит на положительную величину уже с первого года реализации, а за весь жизненный цикл проекта, который составляет 2 года, значение NPV составит 082512 руб., что свидетельствует об экономической целесообразности проекта.

В заключение, были проанализированы возможные риски при реализации проекта и определены возможные меры по их устранению.

**7.2 Анализ положения дел в отрасли**

На сегодняшний день электронная торговля, является очень выгодным и доходным видом бизнеса. Всё больше фирм стараются иметь своё представительство в сети Интернет, так как это даёт дополнительный канал сбыта продукции, делает доступными новые сервисы, как для продавца, так и покупателя, открывает широкие возможности для рекламы и маркетинговых исследований. Кроме того, торговля через Интернет позволяет сэкономить на аренде торговых площадей и персонале.

Потенциальными покупателями разрабатываемого продукта являются частные предприниматели, которые хотят осуществлять продажу музыкальной продукции через Интернет, а также существующие музыкальные магазины, которые хотят открыть новый канал сбыта. Ещё одними потенциальными покупателями являются музыкальные студии и издатели, которые хотят напрямую, без посредников, продавать в Интернет свою продукцию.

**7.3 Суть разрабатываемого проекта**

**7.3.1 Назначение**

Программный продукт "Музыкальный Интернет-магазин" предназначен для выполнения следующих функций:

Со стороны продавца:

* Хранить информацию о товарах, заказах, личные данные пользователя;
* Возможность управления каталогом продукции;
* Прием заказов от покупателей;
* Возможность отвечать на вопросы покупателей;

Со стороны покупателя:

* Предоставление покупателю полную информацию о продукции;
* Предоставление удобной и быстрой системы поиска продукции по различным критериям поиска;
* Позволять осуществлять заказ продукции;
* Возможность задавать вопросы по работе магазина;
* Возможность оставлять отзывы о продукции;

**7.3.2 Форма реализации**

Разработка web-приложения будет вестись на ПК под ОС Windows XP Professional. Для разработки данного программного продукта будет использоваться среда программирования Microsoft Visual Studio 2005, в качестве СУБД будет использоваться Microsoft SQL Server 2000.

**7.3.3 Преимущества перед аналогами**

Разрабатываемый программный продукт будет иметь следующие преимущества перед аналогами:

* Красивый, эргономичный и интуитивно-понятный интерфейс;
* Высокая производительность и масштабируемость за счёт применения самых новейших технологий в области web-программирования;
* Безопасность данных приложения от несанкционированного доступа;
* Высокая надёжность за счёт встроенной обработки ошибок и контроля корректности вводимой информации;

**7.4 План маркетинга**

**7.4.1 Анализ конкурентоспособности товара**

Таблица 2 "Характеристики товара"

|  |  |
| --- | --- |
| **Описание товара и его применения** | **Под товаром понимается web-приложение "Музыкальный Интернет магазин". Система разрабатывается как эффективный механизм для продажи через Интернет музыкальной продукции.** |
| Отличительные или уникальные свойства товара | – Красивый, эргономичный и интуитивно-понятный интерфейс;  – Высокая производительность и масштабируемость за счёт  применения самых новейших технологий в области web-  программирования;  – Безопасность данных приложения от несанкционированного доступа;  – Высокая надёжность за счёт встроенной обработки ошибок и контроля  корректности вводимой информации; |
| Любые лицензионные и патентные права. | Для функционирования приложения требуется операционная система Microsoft Windows XP Professional, Windows 2000 Server или Windows Server 2003. Web-сервер IIS 5.0 или ISS 6.0. В качестве СУБД требуется Microsoft SQL Server 2000. Также необходимо наличие среды.NET Framework 2.0 |
| Потенциальные возможности в будущем. | На основе модели системы можно расширять ассортимент продукции или перепрофилировать для продажи других товаров через Интернет. |

**7.4.2 Анализ рынка сбыта**

Результаты сегментации рынка и прогнозируемый объём продаж представлены в таблице 3 и на рисунке 31.

Таблица 3 "Характеристики сегментов рынка"

|  |  |  |  |
| --- | --- | --- | --- |
| **Сегмент** | **Описание сегмента** | **Прогнозируемый объем продаж** | **Прогнозируемая цена продажи** |
| Музыкальные Интернет-магазины | Интернет-магазины осуществляющие коммерческую деятельность только в Интернет | 3 шт. в год | 70000 руб. |
| Музыкальные магазины | Существующие музыкальные магазина, стремящиеся открыть дополнительный канал сбыта продукции. | 1 шт. в год | 70000 руб. |
| Звукозаписывающие студии и издатели. | Средние и маленькие звукозаписывающие студии и издатели, стремящиеся открыть дополнительный канал рекламы и сбыта продукции. | 2 шт. в год | 70000 руб. |

**17%**

**33 %**

**50 %**



Рисунок 31 "Планируемое распределение объёмов продаж по сегментам рынка"

**7.4.3 Разработка стратегии ценообразования и стратегии продвижения товара**

Проведённые исследования показывают, что на рынке существует серьёзная конкуренция. Многие разработчики программного обеспечения предлагают аналогичный товар. Поэтому в качестве стратегии продвижения товара на рынок была выбрана стратегия быстрого проникновения на рынок. Основные методы, которые будут использоваться для продвижения товара на рынок, приведены в таблице 4.

Таблица 4 "Применение стратегии продвижения информационного продукта на рынок"

|  |  |
| --- | --- |
| **Элементы стратегии** | **Действия** |
| Льготы и скидки | – Оплата в рассрочку  – Предоставление скидок участникам выставок и презентаций, на которых демонстрировалась программа |
| Рекламная деятельность | – Реклама в Интернете  – Создание общедоступного демонстрационного сайта |
| Консультация пользователей | – Развёртывание web-приложения на сервере клиента  – Настройка производительности  – Поддержка и сопровождение |
| Ознакомительный маркетинг | - |

**7.5 Производственный план**

Перечень технологических процессов:

* Анализ технического задания
* Изучение документации по данному заданию
* Выбор программной среды и технологий
* Разработка программного проекта
* Реализация программного проекта
* Тестирование и отладка
* Документирование

**7.6 Планирование разработки и определение себестоимости программного обеспечения**

**7.6.1 Планирование разработки**

Планируемый календарный график работ приведен в таблице 5.

Таблица 5 "Календарный график работ"

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Действия** | **Ai** | **Mi** | **Bi** | **Moi** | **Di** |
| 1. Разработка технического задания | 3 | 4 | 5 | 4 | 0,33 |
| 2. Уточнение технического задания, выбор программной среды и методов разработки ПО | 2 | 3 | 4 | 3 | 0,33 |
| 3. Изучение технической документации | 5 | 6 | 7 | 6 | 0,33 |
| 4. Обучение технологии программирования и освоение среды разработки | 5 | 6 | 7 | 6 | 0,33 |
| 5. Разработка алгоритмов | 10 | 12 | 03 | 01,833 | 0,50 |
| 6. Разработка ПО | 05 | 17 | 09 | 07 | 0,67 |
| 7. Тестирование и отладка ПО | 1 | 2 | 3 | 2 | 0,33 |
| 8. | 5 | 6 | 8 | 0,1667 | 0,50 |
| **Итого** | **06** | **06** | **06** | **06** | **0,22** |

Ai – оценка продолжительности действий при благоприятном стечении обстоятельств;

Bi – оценка продолжительности действий при неблагоприятном стечении обстоятельств;

Mi –;

Mоi – ожидаемая величина длительности действий;

Di – стандартное отклонение для каждого действия, характеризующее степень неопределённости выполнения работ за ожидаемое время;

Mоi= (Ai+4Mi+Bi)/6;

Di = (Bi-Ai)/6;

Итоговая оценка затрат на разработку программного продукта:



Стандартное отклонение от итоговой оценки:



На рисунке 32 представлена диаграмма Ганта, иллюстрирующая процесс разработки программного продукта.

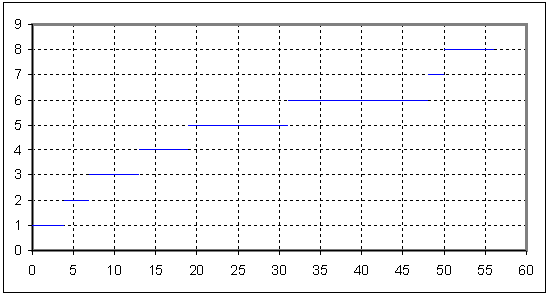


Рис. 32. "Диаграмма Ганта процесса разработки"

**7.6.2 Определение себестоимости программного обеспечения**

В разработке программного обеспечения принимает участие 1 человек, который выполняет все функции системного архитектора, программиста и дизайнера.

Зарплата сотрудника составляет 35000 руб.

Для разработки программного продукта необходимо закупить требуемое аппаратное и программное обеспечение, список которого представлен в таблице 6.

Таблица 6 "Аппаратное и программное обеспечение необходимое для разработки"

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Тип оборудования** | **Наименование** | **Модель** | **Кол-во, шт.** | **Цена за 1 шт., руб.** | **Стоимость, руб.** |
| Аппаратное обеспечение | Процессор | Athlon 64 3000+ | 1 | 0700 | 0700 |
| Материнская плата | Asus A8N-E | 1 | 0500 | 0500 |
| Жёсткий диск | Western Digital, Sata, 80Гб | 1 | 0500 | 0500 |
| Оперативная память | Samsung, DDR 400, 512 мб | 2 | 1000 | 2000 |
| Принтер | HP deskjet 3820 | 1 | 0500 | 0500 |
| Программное обеспечение | Операционная система | Windows XP Professional | 1 | 0600 | 0600 |
| ПО ведения  документации | Open Office | 1 | 200 | 200 |
| Среда разработки | Microsoft Visual Studio 2005 Standart Edition | 1 | 5000 | 5000 |
| СУБД | Microsoft SQL Server 2000 Standart Edition | 1 | 3000 | 3000 |
| **ИТОГО:** | | | | | **07000** |

Плановая себестоимость разработки С определяется по формуле:

, где



З – заработная плата разработчика с начислениями на социальные нужды;

А – амортизация ПЭВМ и другого оборудования;

Спр – прочие производственные затраты;

Заработная плата разработчика проекта рассчитывается по формуле:

, где



Зо – среднемесячная заработная плата,

m – количество рабочих дней в месяце (m = 22),

tp – время разработки.

Амортизационные отчисления при линейном методе расчета амортизации составят:

где



НА – годовая норма амортизации, НА = 12%;

ЦВТ – балансовая стоимость вычислительной техники (ВТ),

256 – среднее количество рабочих дней в году

Прочие производственные затраты включают затраты на техническое обслуживание, ремонт ВТ и расходные материалы и определяются по формуле:

,



где

Нпр – процент прочих производственных затрат от первоначальной стоимости ВТ, Нпр = 4%.

Проведение расчётов:

З = 05000 \* (1 + 0,395) / 02 \* 06 = 024282 руб.

А = (0,12 \* 07000) / 056 \* 06 = 082 руб.

Спр = (0,04 \* 07000) / 056 \* 06 = 094 руб.

Плановая себестоимость составит при КН = 0,2

С = (024282 + 082 + 094)\*(1 + 0,2) = 050070 руб.

**7.7 Оценка экономической эффективности инвестиционного проекта**

**7.7.1 Критерии эффективности коммерческих инвестиций**

Наиболее распространённым критерием эффективности инвестиций в мировой практике является чистая приведённая стоимость проекта (NPV), представляющая собой приведённую (современную) стоимость будущих потоков денежных средств, генерируемых инвестиционным проектом за жизненный цикл.

, где



t – интервал, год;

Дt – чистый поток денежных средств, генерируемых проектом на t-м интервале инвестиционного периода, руб.;

i – ставка сравнения (ставка дисконта) в долях, i = 13%.;

– коэффициент дисконтирования (дисконтный множитель по ставке i для периода планирования t).



Чистый поток денежных средств Дt, генерируемых инвестиционным проектом за каждый год жизни проекта рассчитывается по формуле:

, где



ЧПt – годовая чистая прибыль от реализации продукции, созданной инвестиционным проектом, руб.;

Аt – годовые амортизационные отчисления, руб.;

ИЗt –.

Годовая чистая прибыль может быть рассчитана как:

, где



ВР – годовой объём продажи продукции, руб.;

ЭР – годовые эксплуатационные затраты, руб.;

Т –.

Инвестиционные затраты определяются суммой затрат в основной и в оборотный капитал:

, где



ИЗоб – инвестиционные затраты в оборотный капитал (начальный оборотный капитал), руб. Инвестиционные затраты (капитальный бюджет) представлены в таблице 7.

Таблица 7 "Инвестиционные затраты (капитальный бюджет)"

|  |  |
| --- | --- |
| **Вид капиталовложений** | **Всего, руб.** |
|
| Сметная стоимость проектно-изыскательных работ (плановая себестоимость), руб. | 050070 |
| Стоимость оборудования, руб. | 07000 |
| **Итого** | **077070** |

Жизненный цикл данного проекта составляет два года. Жизненный цикл инвестиционного проекта – это длительность существования проекта. Он начинается с момента начала разработки и заканчивается после снятия с производства и утилизации.

Жизненный цикл инвестиционного проекта приведен на рисунке 33.

Жизненный цикл проекта

Интервалы планирования

Осуществление

инвестиций

Разработка

0

1

2

Рисунок 33 "Жизненный цикл инвестиционного проекта"

**7.7.2 Расчёт экономической эффективности инвестиционного проекта**

Прогнозируемый годовой объём продаж по данным маркетингового анализа составит 6 шт. В соответствии с рекомендациями маркетинговых исследований выбираем метод формирования цены "издержки плюс надбавки" с учётом цен конкурентов. Цены у конкурентов на аналогичные изделия составляют около 100000 руб. Чтобы составить достойную конкуренцию существующим товарам, выходить на новый рынок планируется с ценой 70000 руб./шт., что должно обеспечить достойную конкурентоспособность нашего изделия и привлечь внимание потенциальных покупателей.

В этом случае годовой объём продаж составит:

ВР = 70000 \* 0 = 020000 руб.

Годовые эксплуатационные (операционные) расходы:

ЭР = 05000\*0,5\*6= 005000 руб.

Годовая чистая прибыль:

ЧП = (020000 – 005000) \* (1 – 0,24) = 040000 руб.

Сумма инвестиционных затрат составит:

ИЗО = 077070 руб.

Если предположить, что жизненный цикл товара составит 2 года и за этот срок стоимость инвестиционных затрат в основной капитал (077070 руб.) должна быть возмещена для инвесторов, то годовые амортизационные отчисления составят 08535 руб. в год при равномерной амортизации.

Если предположить, что для того, чтобы запустить проект, требуется 10,5% годовых эксплуатационных расходов, то:

ИЗоб = 005000 \* 0,105 = 01025 руб.

Общая сумма инвестиционных затрат составит:

ИЗ = 077070 + 01025 = 088095 руб.

Расчет NPV представлен далее в виде таблицы 8.8.

Из таблицы 8 следует, что проект выходит на положительную величину NPV уже на первом году. За срок жизненного цикла Tж равный 2 года, значение NPV составит 382500 руб., что свидетельствует об экономической целесообразности проекта.

Таблица 8 "Укрупнённый прогноз потоков денежных средств проекта"

|  |  |  |  |
| --- | --- | --- | --- |
| **Показатели** | **Интервал планирования, год** | | |
| **0** | **1** | **2** |
| 1. Годовой объем продаж |  | 020000 | 020000 |
| 2. Инвестиционные затраты | -088095 |  |  |
| 3. Годовые эксплуатационные расходы |  | -005000 | -005000 |
| 4. Валовая прибыль (п. 1 – п. 3) |  | 015000 | 015000 |
| 5. Налог на прибыль (24% от п. 4) |  | 05600 | 05600 |
| 6. Чистая прибыль (п. 4 – п. 5) |  | 039400 | 039400 |
| 7. Амортизация, руб. |  | 08535 | 08535 |
| 8. Чистый поток денежных средств (п. 6 + п. 7) |  | 027935 | 027935 |
| 9. Дисконтный множитель i = 0,1 | 1 | 0,91 | 0,83 |
| 10. Приведенный чистый поток денежных средств (п. 8 \* п. 9) |  | 098420 | 072185 |
| 11. Чистая приведенная стоимость будущих потоков денежных средств NPV (п. 10 нарастающим итогом) | – 088095 | 010326 | 082512 |

**7.8 Анализ рисков и неопределенностей**

**7.8.1 Возможность ошибки при постановке или анализе задачи**

Данный риск присутствует при реализации любого программного проекта. Он заключается в том, что разработчики программного продукта могут неправильно понять требования заказчика. Учитывая, что на поздних стадиях реализации проекта стоимость изменений очень высокая, то такие ошибки могут привести к увеличению длительности разработки, срыву сроков и обязательств по бюджету и даже к отмене проекта.

Для предотвращения данной проблемы можно предпринять следующие действия:

* провести дополнительное обучение разработчиков в предметной области. Это позволит избежать проблемы за счет того, что разработчикам будет проще разговаривать с клиентом с использованием терминов предметной области;
* согласовать график регулярных встреч с заказчиком для презентации продукта.

**7.8.2 Срыв графика работ в результате ошибки в оценке времени разработки**

Срыв графика работ может произойти в результате одной из двух причин:

* разработка ведётся работниками, не имеющими необходимой квалификации;
* неправильная постановка задачи перед исполнителями.

Для предотвращения срыва графика работ можно предпринять следующие действия:

* при оценке длительности привлечь экспертов, компетентных в;
* привлечь специалистов более высокой квалификации и обладающих опытом в.

**Заключение**

В результате работы над дипломным проектом в соответствие с требованиями, изложенными в пункте 1.2, было разработано web-приложение, представляющее собой интернет – магазин по продаже музыкальных CD и DVD дисков. Все требования по функциям web-приложения выполнены.

В качестве основной технологии для разработки использовалась одна из самых современных и функциональных web-технологий на сегодняшний день – ASP.NET 2.0.

Благодаря использованию различных серверных и пользовательских элементов управления, применению каскадных таблиц стилей и тем оформления, удалось создать привлекательный, функциональный, удобный и интуитивно-понятный пользовательский интерфейс, с возможностью быстрой модернизации в случае необходимости.

С самого начала разработки особенное внимание уделялось обеспечению высокого быстродействия web-приложения при одновременной работе с большим количеством пользователей. Применение компонентов доступа к данным, кэширования, настраиваемого разбиения на страницы позволило обеспечить высокую производительность и масштабируемость разработанного web-приложения.

Также был рассмотрен план маркетинга и определена прогнозируемая прибыль от реализации web-приложения. Были рассмотрены вопросы, связанные с безопасностью и санитарно-гигиеническими условиями труда на рабочем месте пользователя ПЭВМ.

В результате работы над дипломным проектом был получен бесценный опыт программирования на основе технологии ASP.NET, что позволит в будущем создавать высокопроизводительные, профессиональные web-приложения.

В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома. Отсутствуют или не соответствуют некоторые необходимые формулы, расчеты, диаграммы и таблицы. Для получения оригинальной версии работы пишите на электронный адрес: [BarS-Kovd (собака) yandex.ru](mailto:BarS-Kovd@yandex.ru). К работе прилагается презентация, интернет-магазин, исходные тексты программ, речь к защите.

**Список источников**

1. Microsoft ASP.NET 2.0 с примерами на C# 2005 для профессионалов. – М.: ООО "Издательский дом Вильямс", 2006. – 1408 c.
2. C# 2005 для профессионалов. – М.: ООО "Издательский дом Вильямс", 2006. – 1376 c.
3. Visual C#.NET. Полное руководство. – К.: "ВЕК", 2004. – 960 с.
4. Microsoft ADO.NET 2.0 для профессионалов. – М.: ООО "Издательский дом Вильямс", 2006. –560 c.
5. Дэниел Эймор "Internet-магазины и закупочная деятельность"/ "Электронный бизнес. Эволюция и / или революция" изд. "Вильямс" 2001 г. С. 291–302.
6. Пирогов С.В. – "Обзорных крупных проектов" / "Электронная коммерция" Москва 2003 – С. 69–61.
7. Балабанов И.Т. – "Торговля через виртуальный магазин" / "Электронная коммерция"/ 2004 г. С. 195–197.
8. Паршенцев А.А. Проблема и перспективы развития электронных магазинов // Маркетинг в России и за рубежом. – 2000. – №3. – С. 84–89.
9. Петров В.Н. – "Информационные системы" СПб.: Питер, 2003. – 688 с.: ил.
10. Т.С. Карпова. – "Базы данных: Модели, разработка, реализация" СПб.: Питер, 2002. – 304 с.: ил.
11. Иванова Г.С. – "Основы программирования" Учебник для вузов. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2001. – 392 с., ил.
12. Костяев, Р.А. Бизнес в Интернете: финансы, маркетинг, планирование / Р.А Костяев – СПб. БХВ-Петербург, 2002. – 630 с.: ил.
13. Попов В.М. Глобальный бизнес и информационные технологии: современная практика и рекомендации/В.М. Попов, Р.А. Маршавин, С.И. Ляпунов; под общ. ред. В.М. Попова. – М.: Финансы и статистика, 2001. – 238 с.
14. Холмогоров В. Интернет – маркетинг. Краткий курс. 2-е издание/ В. Холмогоров. – СПб.: Питер 2002. – 272 с.: ил.
15. Юрасов А.В. Электронная коммерция: учеб. пособие./А.В. Юрасов – М.: Дело. 2003. – 480 с
16. Белов Н.А. Безопасность жизнедеятельности/ Н.А. Белов – М.: знание, 2000. – 364 с.
17. Комиссарова И, Перелетова И, Петрова Е.К. Интернет-магазин от А до Я: полное пошаговое руководство по созданию, управлению/ Эксмо – 2008.
18. [www.gotdotnet.ru](http://www.gotdotnet.ru)
19. www.aspnetmania.com