

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
"САМАРСКИЙ ГОСУДАРСТВЕННЫЙ ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ"

М.А. Колотилина

ИНТЕРНЕТ-ПРОГРАММИРОВАНИЕ

Учебное пособие

Самара
Издательство
Самарского государственного экономического университета
2019

УДК 004.41/42(075)

ББК 3973.2я7

К61

Рецензенты: кафедра "Высшая математика и информатика"
Самарского филиала МГПУ (зав. кафедрой кандидат
физико-математических наук, доцент С.Н. Богданов);
кандидат экономических наук, доцент Ю.В. Скибин

Издается по решению
редакционно-издательского совета университета

Колотилина, Мария Александровна.

К61 Интернет-программирование [Электронное издание] : учеб. пособие / М.А. Колотилина. - Самара : Изд-во Самар. гос. экон. ун-та, 2019. - 1 электрон. опт. диск. - Систем. требования: процессор Intel с тактовой частотой 1,3 ГГц и выше ; 256 Мб ОЗУ и более ; MS Windows XP/Vista/7/10 ; Adobe Reader ; разрешение экрана 1024×768 ; привод CD-ROM. - Загл. с титул. экрана. № госрегистрации: 0321903228
ISBN 978-5-94622-966-1

В пособии рассмотрены основы языков программирования HTML, CSS, JavaScript, а также системы управления контентом веб-сайтов (например, WordPress). Издание содержит практические примеры и рекомендации по созданию сайта, а также задание для самостоятельного выполнения.

Для студентов, обучающихся по направлению подготовки 38.03.05 "Бизнес-информатика".

УДК 004.41/42(075)
ББК 3973.2я7

ISBN 978-5-94622-966-1

© ФГБОУ ВО "Самарский государственный
экономический университет", 2019
© Колотилина М.А., 2019

ОГЛАВЛЕНИЕ

Введение	4
Глава 1. Введение в динамическое содержимое веб-страницы	7
1.1. Основы языка HTML.....	7
1.2. Архитектура	14
1.3. Обзор языков программирования	18
1.4. Обзор и выбор СУБД	23
1.5. Разработка БД.....	25
1.6. Создание БД в MySQL	31
1.7. Основы языка серверных сценариев PHP.....	32
1.8. Разработка шаблонов и главной страницы сайта	44
Глава 2. Основные системы управления контентом веб-сайтов	72
2.1. Виды CMS-систем, функции.....	72
2.2. Функционирование CMS	77
Задание	94
Список литературы	96
Приложение	98

ВВЕДЕНИЕ

Успешность деятельности любой компании во многом базируется на правильном и адекватном построении своей информационной инфраструктуры. Принципы ее построения зависят от ряда факторов, среди которых сфера и специфика деятельности компании, ее цели, размеры. Так, например, в настоящее время уже самые маленькие компании (с числом сотрудников в несколько человек) имеют в той или иной форме корпоративную сеть для обмена файлами между сотрудниками, но эффективность и таких небольших компаний может быть значительно увеличена при умелом использовании информационных систем, необходимых в настоящий момент. Развитие информационных технологий становится ядром преобразований в компании, а рынок информационных систем остается одним из самых быстрорастущих, предлагая все новые варианты построения информационной инфраструктуры.

В настоящее время можно отметить, что акцент в развитии информационных технологий смещается в сторону улучшения взаимоотношений с клиентами, также продолжает совершенствоваться класс интеллектуальных информационных технологий. Интеллектуальные программные системы способны непрерывно извлекать новые знания и изменять свою структуру и функции, вместе с компанией развиваться и адаптироваться к решаемым задачам и условиям внешней среды. Один из путей решения этих задач - внедрение информационных систем для автоматизации бизнеса, что характерно для клиентоориентированных компаний, где грамотные механизмы работы с различными экономическими субъектами пронизывают все процессы организации. Разработка и проектирование информационной системы предполагает создание единой базы контрагентов и различных функций, что позволяет вести полную историю сотрудничества с партнерами.

Информационные технологии автоматизации взаимоотношений субъектов экономики в настоящее время представляют собой удобный информационный инструмент и позволяют выработать эффективную основу для работы с клиентами. Таким образом, информационная система должна рассматриваться не только как концепция взаимодействия с партнерами компании, а скорее как система, помогающая выстроить долгосрочный клиентоориентированный бизнес. Результат применения этой концепции - рост объемов производства компаний, их мобильности на рынке, прибыльности и конкурентоспособности продукции.

В преддверии цифровизации РФ создаются условия для развития передовых отечественных информационно-коммуникационных технологий. Россияне в среднем в день пользуются Интернетом 4 ч и 40 мин при помощи компьютера или планшета, 1 ч и 23 мин - при помощи смартфона, а среднее время пользования социальными медиа в день составляет 1 ч и 52 мин. Приведенные данные показывают, насколько важными для человека становятся нахождение в сети, пользование социальными медиа. Современные компании это поняли, и на сегодняшний момент практически у всех фирм есть свои полноценные сайты либо сайты-визитки. Компании заказывают баннерную и контекстную рекламу, настраивают поисковую оптимизацию, но не все понимают, что таких действий недостаточно для эффективной активности в Интернете.

В данном учебном пособии подробно и поэтапно рассматривается процесс разработки электронного представительства компании посредством интернет-приложения на базе веб-сервера Apache, языка серверных сценариев PHP и системы управления базами данных MySQL. В процесс разработки приложения включаются следующие этапы:

- разработка архитектуры интернет-приложения;
- разработка структуры базы данных;
- разработка шаблонов и главной страницы;
- разработка системы регистрации и авторизации клиентов;
- разработка системы формирования заявки.

От читателя не требуется предварительных знаний по программированию для Интернета. В пособии имеется отдельная глава, посвященная основам языка гипертекстовой разметки HTML. Большая часть учебного пособия посвящена изучению серверной технологии PHP.

Предполагается, что читатель прослушал курс "Базы данных" и имеет представление о проектировании баз данных, о системах управления такими базами и о языке SQL. В пособии эти вопросы рассматриваются очень кратко - только в той степени, которая необходима для текущей задачи создания интернет-приложения.

Учебное пособие «Интернет-программирование», предназначенное для направления подготовки 38.03.05 "Бизнес-информатика", составлено в соответствии с требованиями ФГОС и компетентностного подхода в системе высшего образования. Целью разработки является текущий мониторинг степени усвоения студентами знаний, умений и навыков, а также профессиональных компетенций, формируемых дисциплиной "Интернет-программирование". Пособие включает в себя задание для текущего контроля сформированности у обучающихся следующих общепрофессиональных (ОПК-3) и профессиональных (ПК-10, ПК-16) компетенций.

Глава 1. ВВЕДЕНИЕ В ДИНАМИЧЕСКОЕ СОДЕРЖИМОЕ ВЕБ-СТРАНИЦЫ

1.1. Основы языка HTML

HTML (HyperText Markup Language) - это язык гипертекстовой разметки, на котором написаны практически все страницы Всемирной паутины. Команды данного языка называются тэгами и представляют собой ключевые слова в угловых скобках. Регистр букв не имеет значения. Большинство команд являются контейнерами, имеющими открывающий и закрывающий тэги. Базовая структура HTML-документа представлена следующими командами:

```
<html>
<header>
<title>Заголовок окна</title>
</header>
<body>Содержимое документа</body>
</html>
```

Тэг `<html></html>` является тэгом самого верхнего уровня в документе. Внутри него имеются заголовочная часть (`<header>`, содержащая, в частности, строку заголовка документа `<title>`) и тело документа (`<body>`).

Язык HTML, как следует из его названия, предназначен прежде всего для разметки, или форматирования, документа. В табл. 1.1 приведем несколько примеров команд форматирования.

Таблица 1.1

Команды форматирования

Команда	Расшифровка
<code></code> Жирный <code></code> шрифт	Жирный шрифт
<code><i></code> курсив <code></i></code>	Курсив
<code><u></code> подчеркнутый <code></u></code> шрифт	Подчеркнутый шрифт
верхний индекс: <code>A<sub>i</sub></code>	Верхний индекс: A^i
нижний индекс: <code>A<sup>i</sup></code>	Нижний индекс: A_i
<code><center></code> центрирование <code></center></code>	Центрирование

Для переноса на следующую строку используется команда `
` (без закрывающего тэга).

Для рисования горизонтальной линии применяется команда `<hr>` (также без закрывающего тэга).

Для задания заголовков различного размера существуют шесть команд:

от `<h1></h1>` (самый крупный) до `<h6></h6>` (самый мелкий).

Для создания маркированных и нумерованных списков используются команды (табл. 1.2).

Таблица 1.2

Построение списка

Команда	Расшифровка
<code></code> <code>понедельник</code> <code>вторник</code> <code>среда</code> <code></code>	понедельник вторник среда
<code></code> <code>понедельник</code> <code>вторник</code> <code>среда</code> <code></code>	понедельник вторник среда

Многие тэги могут содержать внутри себя дополнительные параметры. Параметр имеет имя и (почти всегда) значение. Так, например, тэг

```
<body bgcolor="red">
```

задает цвет фона документа (красный), а тэг

```
<body background="EULA.jpg">
```

задает графический файл, который будет использован в качестве фона для документа (в данном примере предполагается, что графический файл расположен в том же каталоге, что и HTML-документ).

Параметров внутри тэга может быть несколько. В примере

```
<body BACKGROUND="image/bg.png">
```

```
<link rel="stylesheet" type="text/css" href="js/jquery-ui-1.7.2.custom.css">
```


CSS (Cascading Style Sheets), или каскадные таблицы стилей, используются для описания внешнего вида документа, написанного языком разметки. Обычно CSS-стили нужны для создания и изменения стиля элементов веб-страниц и пользовательских интерфейсов, написанных на языках HTML и XHTML, но также могут быть применены к любому виду XML-документа, в том числе XML, SVG и XUL.

Каскадные таблицы стилей описывают правила форматирования элементов с помощью свойств и допустимых значений этих свойств. Для каждого элемента можно использовать ограниченный набор свойств, остальные свойства не будут оказывать на стиль никакого влияния.

Объявление стиля состоит из двух частей: из элемента веб-страницы (селектора) и команды форматирования (блока объявления). Селектор сообщает браузеру, какой именно элемент форматировать, а в блоке объявления (код в фигурных скобках) перечисляются форматированные команды - свойства и их значения (рис. 1.1).



Рис. 1.1. Структура объявления CSS-стиля

Другой пример с несколькими параметрами: тэг

```
<font face="Arial" size="+2" color="green">
```

будет выводить заданный текст зеленым цветом, шрифтом Arial и на 2 размера больше, чем текущий шрифт.

Кстати, для задания цвета удобно использовать формат #rrggbb, где rr, gg и bb - это двузначные шестнадцатеричные числа, представляющие собой интенсивность красной, зеленой и синей компонент цвета, соответственно. Например, #000000 - черный цвет, #ffffff - белый цвет, #ff0000 - ярко-красный, #880088 - умеренно-фиолетовый (смесь красного и синего).

Для размещения картинок используется тэг , например:

Параметр src задает имя графического файла, в качестве имени может быть использован и HTTP-адрес, например:

```

```

Параметр alt задает альтернативный текст, который будет выводиться при наведении курсора мыши на картинку [12].

Самой полезной командой, которая и позволила логически связать разрозненные документы в единую Всемирную паутину, является тэг <a> - "гиперссылка". Например, команда

```
<a href=" https://www.kv.by/post/1055062-15-yazykov-kotorye-nuzhno-znat-kazhdomu-programmistu" >
```

создает гиперссылку Языки программирования, которая выглядит как подчеркнутый текст синего (по умолчанию) цвета, при щелчке мышью по ней загружается страница с адресом, заданным в параметре href. Если требуется загрузить страницу из текущего каталога, то можно указывать только имя файла, например:

```
<a href=" zajav.phtml">Заявки</a>
```

В данном тэге можно также передавать параметры в вызываемую страницу следующим образом:

```
href="имяФайла?параметр1=значение1&параметр2=значение2".
```

Например:

```
<a href=" zajav.phtml?type=1&id_Book=25">Подать заявку</a>
```

Указанные параметры можно использовать в вызываемой странице.

Таблицы

Текст и другие элементы страницы выводятся в HTML-документ сплошным потоком, слева направо, сверху вниз. Для того чтобы упорядочить элементы страницы друг относительно друга, самым удобным способом являются таблицы (обрамление ячеек можно убрать).

Таблица создается тэгом <table></table>. Полезными параметрами для таблицы являются следующие:

- border - ширина рамки (если 0, то рамки нет);
- align - выравнивание самой таблицы по горизонтали ("right"; "left" или "center");
- width - ширина таблицы в процентах относительно ширины страницы ("100%") или в пикселах ("500");
- bgcolor - цвет фона в таблице.

Например, таблица без рамки, с шириной в 80% страницы и с выравниванием по правому краю может быть задана так:

```
<table border="0" align="right" width="80%">
```

Для создания строки внутри таблицы используется тэг

```
<tr></tr>.
```

Для создания ячейки внутри строки используется тэг

```
<td></td>.
```

Выравнивание содержимого ячейки, ширину ячейки и цвет фона можно задавать теми же параметрами `align`, `width` и `bgcolor`. Кроме того, иногда бывает удобно объединить несколько ячеек по горизонтали или вертикали: например, параметр `colspan=3` объединяет три ячейки по горизонтали, `rowspan=2` объединяет две ячейки по вертикали.

Таким образом, например, табл. 1.3 может быть создана с помощью следующего кода:

```
<table width="900" border="0" cellspacing="0" cellpadding="0">
```

```
<tr >
```

```
<td width="30">&nbsp;&nbsp;&nbsp;</td>
```

```
<td width="320" class="date1" >Полное наименование </td>
```

```
<td width="30">&nbsp;&nbsp;&nbsp;</td>
```

```
<td width="320" class="date1" >Артикул </td>
```

```
<td width="30">&nbsp;&nbsp;&nbsp;</td>
```

```
</tr>
```

```
<tr>
```

```
<td width="30">&nbsp;&nbsp;&nbsp;</td>
```

```
<td width="320" class="log">
```

```
<form method="post" id="signup">
```

```
<input type="text" size="60" value="Не указан" onFocus="doClear(this)"  
onBlur="doDefault(this)" name="nameprod" class="required" title="<br>  
Поле обязательно к заполнению"></td>
```

```
<td width="30">&nbsp;&nbsp;&nbsp;</td>
```

```
<td width="320" class="log">
```

```
<input type="text" size="60" value="Не указан" onFocus="do-  
Clear(this)" onBlur="doDefault(this)" name="art" class="required" ti-  
tle="<br>
```

```
Поле обязательно к заполнению">
```

```
</td>
```

Команды для форматирования таблиц

Команда	Расшифровка
<code><td width="320" class="date1">Полное наименование </td></code> <code><td width="30">&nbsp;</td></code>	Полное наименование
<code><td width="320" class="date1">Артикул </td></code> <code><td width="30">&nbsp;</td></code>	Артикул
<code><td width="320" class="date1">Единица измерения</td></code> <code><td width="30">&nbsp;</td></code>	Единица измерения
<code><td width="320" class="date1">Себестоимость</td></code> <code><td width="30">&nbsp;</td></code>	Себестоимость
<code><td width="320" class="date1">Оптовая цена</td></code> <code><td width="30">&nbsp;</td></code>	Оптовая цена
<code><td width="320" class="date1">Розничная цена</td></code> <code><td width="30">&nbsp;</td></code>	Розничная цена

Формы

Иногда веб-страницам следует передавать параметры (например: логин, пароль при авторизации, номер книги при добавлении ее в корзину и т.п.). Как уже говорилось, параметры можно передавать непосредственно в гиперссылке после имени файла через знак вопроса. Но этот способ не всегда удобен: во-первых, потому что указанные параметры будут отображаться в адресной строке браузера вместе с именем файла, что для пароля недопустимо; а во-вторых, потому что объем информации при этом ограничен. И вообще, через адресную строку не все типы данных можно передать (например, таким образом нельзя переслать целый файл). Поэтому в языке HTML существует такое понятие, как "форма". Форма - это контейнер (`<form>` `</form>`), содержащий элементы управления (текстовые поля, кнопки, флажки, радиокнопки, списки). Форма имеет следующие полезные параметры:

- `method` - способ (GET или POST), которым данные передаются на сервер (при использовании метода GET параметры передаются в адресной строке, как и в гиперссылке, а при использовании метода POST параметры передаются в теле запроса);

- `action` - файл, который будет загружаться как реакция на нажатие кнопки типа Submit.

Внутри контейнера могут располагаться указанные ниже *элементы*, каждый из которых должен иметь имя.

Текстовые поля:

`<input type="text" name="login">` - обычное текстовое поле;

`<input type="password" name="pass">` - поле для ввода пароля, при вводе символов в это поле на экране будут видны только "звездочки";

`<input type="hidden" name="id_book">` - скрытое поле, на экране его не видно, оно полезно для передачи служебной информации, которую пользователю видеть не следует.

Кнопки:

`<input type="button" name="mybutton" value="ОК">` - обычная кнопка;

`<input type="reset" name="reset" value="очистить">` - кнопка, очищающая все поля формы;

`<input type="submit" name="submit" value="отправить">` - кнопка для отправки данных на сервер: при нажатии этой кнопки вызывается веб-страница, имя которой задано в параметре `action` формы, при этом на сервер передаются значения элементов управления формы.

Флажки:

`<input type="checkbox" name="сCard" value="1">` Кредитная карта - независимый переключатель, имеющий два состояния - "включен" и "выключен". Если переключатель выключен, то его значение на сервер передаваться не будет. Для того чтобы включить флажок по умолчанию, следует указать атрибут `checked`.

Радиокнопки:

`<input type="radio" name="rCard" value="1" checked >Visa <input type="radio" name="rCard" value="2">MasterCard <input type="radio" name="rCard" value="3">American Express`

Для того чтобы зависимые переключатели рассматривались как единая группа, следует называть их одним именем. Если указан атрибут `checked`, то эта радиокнопка будет выбрана по умолчанию.

Списки:

`<select size="1" name="sCard">`

`<option value='1' selected > Visa </option> <option value='2'> MasterCard </option> <option value='3'> American Express </option> </select>`

Выпадающий список состоит из трех элементов. Их значение по умолчанию отмечается атрибутом `selected`.

Каким образом переданные данные анализируются на сервере, мы будем подробно рассматривать при изучении серверной технологии PHP [12].

1.2. Архитектура

Предполагается создание системы, основанной на клиент-серверной архитектуре, поэтому требования, предъявляемые к техническому обеспечению, состоят из требований к клиентской и серверной частям. Все компьютеры объединены в локальную сеть, что обеспечивает удовлетворение потребностей информационного обмена. Рабочие места сотрудников оснащаются принтерами для печати различных списков, отчетов, счетов и других выходных документов.

Сервер должен работать под управлением операционной системы Windows Server 2007-2010, так как она обладает повышенной степенью защиты, гибкостью настроек и максимальной производительностью.

Общая схема технического обеспечения компании представлена на рис. 1.2.

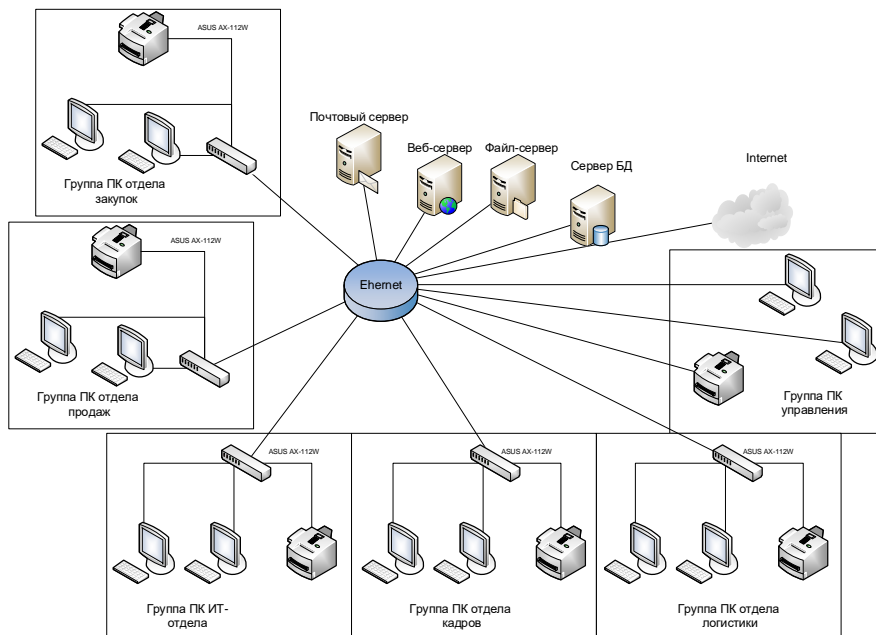


Рис. 1.2. Схема технического обеспечения типовой компании

Доступ в Интернет организован по выделенной линии по технологии ADSL со скоростью до 4 мбит/с.

В состав серверной фермы входят:

- файловый сервер;
- сервер баз данных;
- почтовый сервер;
- сервер управления;
- веб-сервер.

Защита сети осуществляется за счет применения антивирусной программы и программного брандмауэра. Кроме того, внедрена политика разделения прав доступа к ресурсам сети и выделены ее фронтальный участок, включающий в себя веб-сервер и рабочее место.

Для работы в системе рекомендуется использовать браузер Chrome. Статистика использования браузеров пользователями на территории России по данным <http://gs.statcounter.com/> представлена на рис. 1.3.

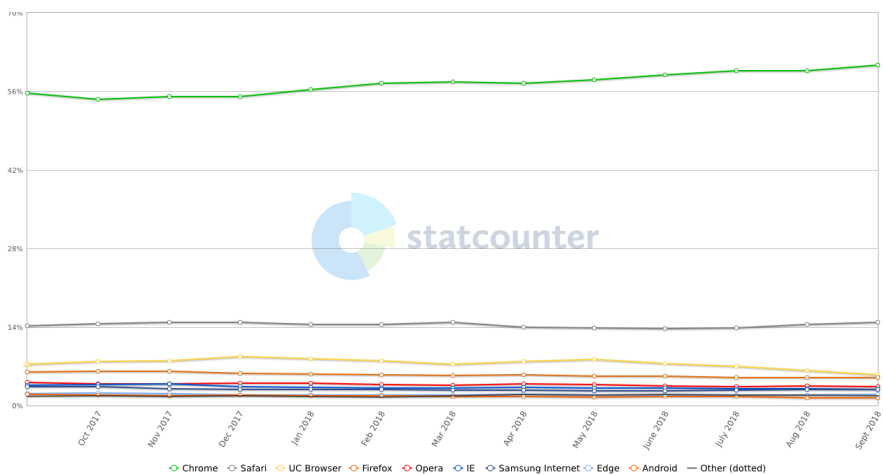


Рис. 1.3. Статистика использования браузеров в России

В соответствии с разработанной даталогической моделью в состав системы структурно должны входить следующие модули:

- модуль авторизации сотрудников;
- модуль работы со справочниками;
- модуль получения отчетных документов;
- модуль ввода данных о производственной деятельности.

Структурная схема приложения приведена на рис. 1.4.

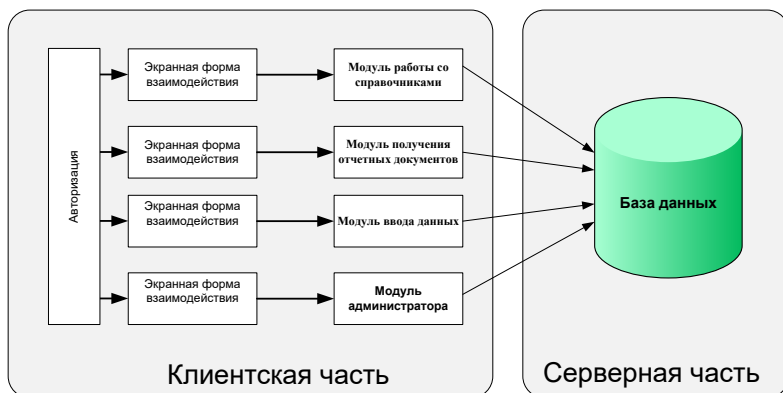


Рис. 1.4. Структурная схема информационной системы компании

Система состоит из четырех модулей: модуля работы со справочниками, модуля получения отчетных документов, модуля ввода данных, модуля администратора. Для хранения информации используется общая база данных.

Работа с системой осуществляется через любой браузер. Для работы необходимо установить локальный сервер в локальной сети предприятия, где будет также расположена база данных. Доступ к базе будет осуществляться с помощью набора адреса в адресной строке браузера.

Состав каждого модуля аналогичен, что позволяет в случае модернизации либо изменения каких-либо сведений с максимальной скоростью внести корректировки в приложение.

Модуль работы со справочниками позволяет вносить изменения в справочники "Поставщики", "Клиенты", "Продукция", "Сырье", "Цеха". Для этого необходимо заполнить соответствующие поля в форме и нажать кнопку «Зарегистрировать». После нажатия кнопки выполняется SQL-запрос к базе данных, в результате чего в ней сохраняется введенная информация.

На одной странице с формой для ввода данных отображается как содержание соответствующего списка с указанием общего количества записей в нем, так и кнопка с возможностью удаления отдельной

записи. После нажатия кнопки запись помечается в таблице справочника как удаленная и не выводится на экран.

Модуль получения отчетных документов выполняет функции формирования результатных документов, реквизитный состав которых различен и перечислен был выше. Формирование документа происходит также после SQL-запроса к базе данных в виде таблицы с перечнем необходимых записей. Кроме того, отображаются результатные показатели, такие как общее количество и стоимость произведенной продукции.

Модуль ввода данных представлен формой учета производственной деятельности, при помощи которого осуществляется учет количества произведенной продукции каждым цехом. При этом выбор номера цеха и наименования продукции осуществляется из выпадающего списка во избежание ошибок оператора.

Модуль авторизации включает в себя форму авторизации и скрипт авторизации, проверяющий на соответствие введенные пароль и логин и существующие пароль и логин. Авторизация осуществляется путем сравнения введенных данных с данными в таблице базы данных. При совпадении пары пароль/логин осуществляется вход в систему, в противном случае система выдает сообщение об ошибке авторизации (рис. 1.5).

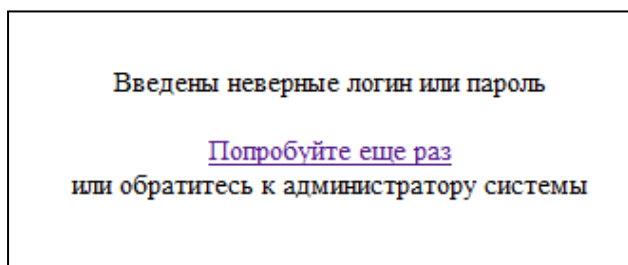


Рис. 1.5. Сообщение об ошибке авторизации

Модуль администратора предназначен для управления работой системы: для регистрации нового пользователя, смены пароля или логина, а также удаления/восстановления пользователя системы. Администратор имеет отдельную форму авторизации.

1.3. Обзор языков программирования

Многие задачи могут быть эффективно решены с помощью любого современного популярного языка программирования (рис. 1.6).



Рис. 1.6. Языки веб-программирования

Часто выбор языка и фреймворка (готового каркаса программы с основными модулями) определяется тем, какими знаниями обладают программисты, готовые реализовать данный проект.

В первую очередь, языки веб-программирования можно классифицировать на клиентские и серверные. Как следует из названия, клиентские языки используются для написания программ, выполняемых на стороне клиента (веб-браузер), а серверные языки - для программ, выполняемых на сервере.

Среди клиентских языков веб-программирования надо выделить JavaScript, который, как и HTML, лежит в основе многих веб-технологий (например, в основе популярной в последнее время технологии AJAX).

При создании веб-ориентированных систем, основанных на динамических страницах, есть несколько альтернатив в выборе интерпретатора языка программирования: ASP, JSP, Perl, PHP, Ruby, Python. Все данные варианты предоставляют возможности, необходимые для реализации приложения.

Однако ASP - это технология Microsoft, которая прежде всего ориентирована на взаимодействие с другими продуктами Microsoft, а кроме того, является коммерческой технологией. Недостатки языков JSP, Ruby и

Python – их небольшая распространенность в России и, как следствие, отсутствие необходимых для работы с ними квалифицированных специалистов на рынке труда. Perl уступает PHP по доле проектов и занимает область классических скриптовых языков ОС семейства Unix.

В табл. 1.4 по данным <http://www.tiobe.com/> указана степень распространения языков программирования по состоянию на сентябрь 2017 и 2018 гг.

Таблица 1.4

Распространение языков веб-программирования

Язык программирования	Место в рейтинге		Изменение позиции в рейтинге	Рейтинг, %	Изменение рейтинга, %
	2017 г.	2018 г.			
Ява	1	1		17,436	+ 4,75
C	2	2		15,447	+ 8,06
Python	3	5	▲	7,653	+ 4,67
C ++	4	3	▼	7,394	+ 1,83
Visual Basic .NET	5	8	▲	5,308	+ 3,33
C #	6	4	▼	3,295	-1,48
PHP	7	6	▼	2,775	+ 0,57
JavaScript	8	7	▼	2,131	+ 0,11
SQL	9	-	▲▲	2,062	+ 2,06
Objective-C	10	18	▲▲	1,509	+ 0,00
Delphi / Object Pascal	11	12	▲	1,292	-0,49
Рубин	12	10	▼	1,291	-0,64
MATLAB	13	16	▲▲	1,276	-0,35
Ассемблер	14	15	▲	1,232	-0,41
Стриж	15	13	▼	1,223	-0,54
Идти	16	17	▲	1,081	-0,49
Perl	17	9	▼▼	1,073	-0,88
p	18	11	▼▼	1,016	-0,80
PL / SQL	19	19		0,850	-0,63
Visual Basic	20	14	▼▼	0,682	-1,07

На основе табл. 1.4 составлен график изменения рейтингов языков программирования (рис. 1.7).

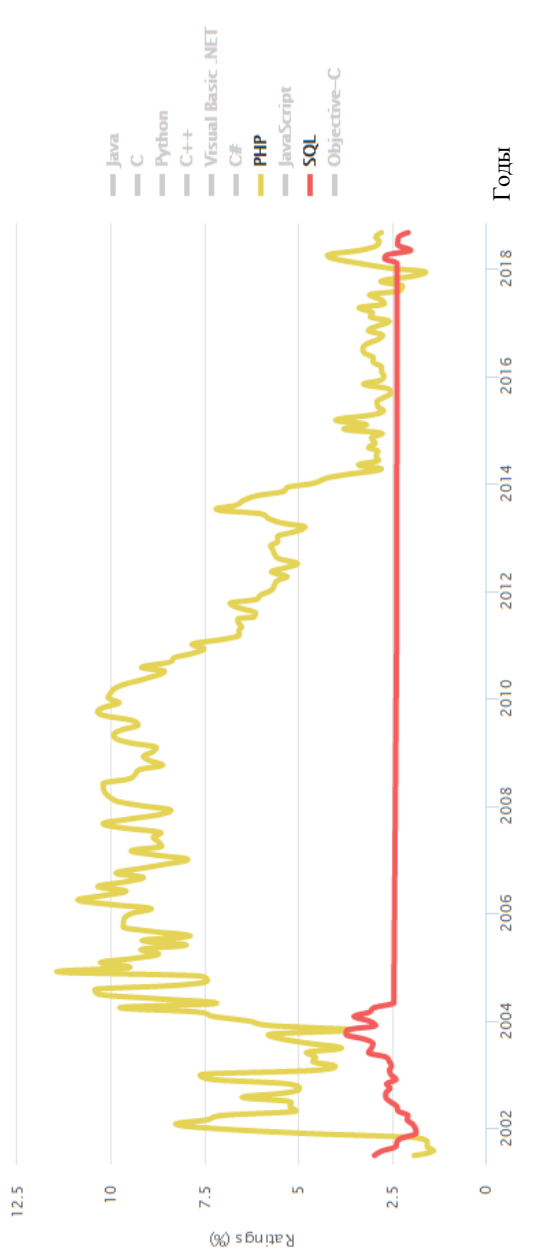


Рис. 1.7. График рейтинга языков веб-программирования

Впервые в своей истории Python появился в индексе ТЮВЕ top-3, на что действительно потребовалось много времени. В начале 1990-х гг. этот язык вошел в график. Затем прошло еще 10 лет, прежде чем Python впервые попал в индекс ТЮВЕ 10 лучших. После этого данный язык медленно, но верно подошел к топ-5 и в конечном итоге с появлением 3. Python язык становится все более распространенным, представляя собой уже первый выбор в университетах (для всех видов предметов, для которых требуется программирование), и теперь он также завоевывает индустриальный мир. Торговые точки Python просты в освоении и установке, легко развертываются. Другие интересные шаги в рассматриваемый период: Rust снизил рейтинг с 36-й до 31-й позиции, Groovy - с 44-й до 34-й, а Julia - с 50-й до 39-й.

ВАЖНАЯ ЗАМЕТКА: SQL снова добавлен в индекс ТЮВЕ с февраля 2018 г. Причина этого в том, что SQL, похоже, завершен. Как следствие, для языка нет недавней истории, и может показаться, что язык SQL растет очень быстро. Но здесь не тот случай.

Индекс программирования ТЮВЕ Community Index является показателем популярности языков программирования. Индекс обновляется раз в месяц. Рейтинги основаны на количестве квалифицированных инженеров по всему миру, курсов и сторонних поставщиков. Для расчета рейтингов используются популярные поисковые системы, такие как Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube и Baidu. Важно отметить, что индекс ТЮВЕ - это не лучший язык программирования или язык, на котором написано большинство строк кода.

Индекс может использоваться для проверки того, насколько ваши навыки программирования по-прежнему актуальны, чтобы принять стратегическое решение о том, какой язык программирования следует выбрать при создании новой системы программного обеспечения.

Одним из самых популярных языков веб-программирования является, безусловно, PHP. Его основными преимуществами выступают простой синтаксис, высокое быстродействие, поддержка большинством хостингов.

PHP - скриптовый язык программирования, созданный для генерации HTML-страниц на веб-сервере и для работы с базами данных. Группа специалистов по PHP состоит из множества людей, по своей инициативе работающих над ядром и расширениями PHP, над смежными проектами, такими как PEAR или документация языка. В настоящее время PHP поддерживается подавляющим большинством представителей хостинга и входит в LAMP – "стандартный" набор для создания веб-сайтов (Linux, Apache, MySQL, PHP) [8].

В области программирования веб-ориентированных систем PHP по-прежнему остается одним из популярнейших скриптовых языков (наряду с JSP, Perl и языками, используемыми в ASP) благодаря своей простоте, скорости выполнения, богатой функциональности и распространению исходных кодов на основе лицензии PHP.

PHP отличается наличием ядра и подключаемых модулей, "расширений": для работы с базами данных, сокетами, динамической графикой, криптографическими библиотеками, документами формата PDF. Синтаксис PHP подобен синтаксису языка C. Некоторые элементы, такие как ассоциативные массивы и цикл `foreach`, заимствованы из Perl.

Программирование в рамках данного проекта осуществляется на языке PHP. Основные критерии выбора - распространенность (заменяемость разработчиков), высокое качество документирования функций, отсутствие необходимости устанавливать на веб-сервер дополнительное программное обеспечение.

Для функционирования создаваемой автоматизированной системы необходим веб-сервер. Веб-сервер - это специальное программное обеспечение (сервер), принимающий HTTP-запросы от клиентов (обычно от браузеров) и выдающий им HTTP-ответы, обычно вместе с HTML-страницей, изображением, файлом, медиапоток или другими данными.

Веб-сервером называют как программное обеспечение, выполняющее функции веб-сервера, так и компьютер, на котором это программное обеспечение работает. Клиенты получают доступ к серверу по URL-адресу через веб-интерфейс. Дополнительными функциями многих веб-серверов являются:

- ведение журнала сервера про обращения пользователей к ресурсам;
- аутентификация пользователей;
- поддержка динамически генерируемых страниц;
- поддержка HTTPS для защищенных соединений с клиентами.

Существует два наиболее распространенных веб-сервера:

- Apache - веб-сервер с открытым исходным кодом, наиболее часто используется в Unix-подобных операционных системах (ОС);
- IIS - веб-сервер от компании Microsoft, как правило, используется на ОС семейства Microsoft Windows.

Основными достоинствами Apache считаются надежность и гибкость конфигурации. Веб-сервер позволяет подключать внешние модули для представления данных, использовать СУБД для аутентификации пользователей, модифицировать сообщения об ошибках. Недостатком наиболее часто называется отсутствие удобного стандартного интерфейса для администратора.

Сервер Apache может обслуживать большое количество клиентов, поддерживая их одновременную работу. Количество клиентов, которое может обслуживаться одновременно, ограничивается лишь используемыми аппаратными средствами и операционной системой. Сервер может быть легко сконфигурирован с помощью редактирования текстовых файлов или посредством использования одного из многочисленных инструментов с графическим интерфейсом. В соответствии со своей модульной архитектурой многие возможности, которые необходимы для работы некоторых приложений, реализуются в виде дополнительных модулей Apache. В целях поддержки таких возможностей для разработчиков модулей реализован хорошо документированный API. Модульность и существование бесплатных модулей позволяют легко создать мощный веб-сервер без изменения его исходного кода. Используя на сервере множество доступных скриптовых языков, можно легко создать любое веб-приложение. Для использования любого скриптового языка необходим только соответствующий подключаемый модуль.

Кроме того, данный сервер распространяется бесплатно, поэтому для разработки описываемой системы выберем именно его.

1.4. Обзор и выбор СУБД

В настоящее время для построения информационных систем применяются различные системы управления базами данных (СУБД), различающиеся как своими возможностями, так и требованиями к вычислительным ресурсам.

Разрабатываемая система является веб-ориентированной системой, поэтому она основана на клиент-серверной архитектуре, для ее функционирования необходима серверная СУБД. Одной из наиболее распространенных СУБД, используемых в веб-приложениях, является

MySQL. Эта СУБД динамично развивается и по умолчанию установлена на ОС FreeBSD. На ОС уже имеются все необходимые приложения, облегчающие администрирование MySQL, поэтому нет необходимости устанавливать стороннее ПО.

Кроме того, MySQL является одной из самых производительных систем на рынке и имеет хорошую документацию на русском языке. Продукт поддерживается разработчиками и продолжает развиваться.

В рамках данного проекта к СУБД предъявляется ряд специфических требований:

- поддержка транзакций;
- поддержка внешних ключей;
- корректная работа с русским языком;
- наличие инструмента администрирования с графическим интерфейсом;
- возможности доступа к данным средства языка запросов SQL;
- возможности резервного копирования базы данных.

MySQL 5.1 с включенной поддержкой таблиц типа InnoDB и утилитой PHPMyAdmin 3.1.4 удовлетворяет всем представленным требованиям.

На рынке серверных СУБД существуют и другие решения, ориентированные на работу веб-приложений. К ним в первую очередь относятся PostgreSQL, Microsoft SQL Server, Oracle. Однако, учитывая высокую стоимость решений от Oracle и Microsoft, их высокие требования к аппаратному обеспечению и потребность в окружении специфическим программным обеспечением, в качестве реальной альтернативы MySQL можно рассматривать только PostgreSQL [18]. Сравнительная характеристика MySQL и PostgreSQL представлена в табл. 1.5.

Таблица 1.5

Сравнение возможностей СУБД PostgreSQL и MySQL

Параметр	PostgreSQL	MySQL
1	2	3
ANSI SQL-совместимость	Близка к стандарту ANSI SQL	Следует некоторым стандартам ANSI SQL
Скорость работы	Медленнее	Быстрее
Вложенные SELECT	Да	Да
Транзакции	Да	Да, однако должен использоваться тип таблицы InnoDB

1	2	3
Внешние ключи	Да	Да, однако должен использоваться тип таблицы InnoDB
Представления	Да	Нет
Триггеры	Да	Нет
Поддержка Windows	Да	Да
ODBC	Да	Да
JDBC	Да	Да
Различные типы таблиц	Нет	Да

Как видно из табл. 1.5, PostgreSQL обладает некоторыми преимуществами перед MySQL. Так, в частности, PostgreSQL поддерживает представления и триггеры. Эта СУБД иногда показывает лучшие результаты производительности при пиковых нагрузках [13].

Данный проект отличается невысокими требованиями к СУБД, большие нагрузки не прогнозируются, а эксплуатация PostgreSQL требует дополнительных финансовых затрат. Здесь связано с тем, что для обеспечения высокой производительности *совместно* с высокой надежностью нужно обязательно использовать аппаратный дисковый контроллер со встроенной кэш-памятью записи значительного размера и с батареей (такие контроллеры недешевы, стоят от 7000 руб.). Кроме того, MySQL очень неприхотлива и нетребовательна в администрировании (по приблизительным оценкам, там раз в 10 меньше подводных камней, чем в PostgreSQL), а следовательно, малозатратна. Поэтому в рамках рассматриваемого проекта целесообразно использовать СУБД MySQL.

1.5. Разработка БД

Для формирования ранее указанных выходных документов в информационную систему вводятся следующие массивы информации:

- план поставок сырья;
- план реализации готовой продукции;
- номенклатура сырья;

- номенклатура готовой продукции;
- перечень поставщиков;
- перечень клиентов;
- заявки на закупку готовой продукции от клиентов.

Рассмотрим подробнее реквизитный состав массивов входной информации.

План поставок сырья формируется на основе достигнутых договоренностей с поставщиками и неразрывно связан с такими массивами, как перечень поставляемого сырья и перечень поставщиков.

В составе данного плана имеются следующие реквизиты:

- дата поставки;
- наименование сырья;
- наименование поставщика;
- количество поставляемого сырья;
- номер договора, по которому осуществляется поставка;
- стоимость поставляемого сырья.

План реализации готовой продукции формируется на основе подаваемых клиентами заявок и связан с такими массивами данных, как:

- дата реализации;
- наименование готовой продукции;
- количество готовой продукции;
- общая стоимость готовой продукции;
- наименование клиента;
- номер заявки, на основании которой осуществляется реализация готовой продукции.

В номенклатуре сырья содержатся следующие реквизиты:

- наименование сырья;
- единица измерения;
- закупочная цена за единицу измерения;
- срок хранения;
- дополнительные сведения о данном виде сырья.

В номенклатуре готовой продукции отражаются:

- тип готовой продукции;
- наименование;
- артикул;

- единица измерения;
- величина себестоимости;
- оптовая отпускная цена;
- розничная отпускная цена.

В перечень поставщиков включаются:

- полное наименование поставщика;
- краткое наименование поставщика;
- адрес размещения;
- юридический адрес;
- банковские реквизиты;
- контактное лицо;
- телефон;
- сайт;
- факс;
- дата начала сотрудничества.

В состав списка клиентов входят следующие данные:

- полное наименование клиента;
- краткое наименование клиента;
- адрес размещения;
- юридический адрес;
- банковские реквизиты;
- контактное лицо;
- телефон;
- сайт;
- факс;
- дата начала сотрудничества.

В состав заявки, подаваемой клиентом на закупку готовой продукции, входят:

- номер заявки;
- дата регистрации заявки;
- наименование клиента;
- наименование готовой продукции;
- количество готовой продукции;
- дата поставки готовой продукции;
- статус заявки.

Даталогическая модель потоков данных представлена на рис. 1.8.

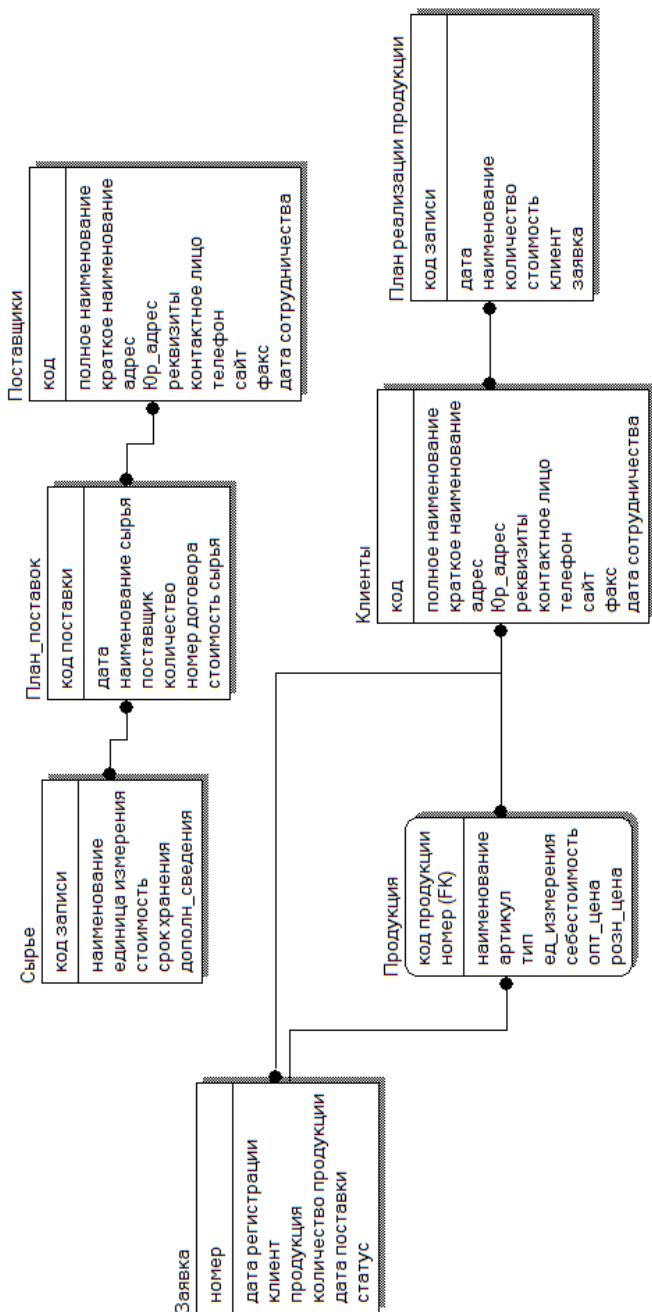


Рис. 1.8. Даталогическая модель

Кроме того, для обеспечения деятельности информационной системы в данной даталогической модели должны быть предусмотрены следующие сущности:

- пользователи;
- история;
- отчеты.

Пользователями должны быть указаны:

- код пользователя;
- дата регистрации;
- фамилия, имя, отчество;
- должность;
- статус;
- логин,
- пароль.

В системе предусматриваются три вида пользователей: менеджеры, которые вводят данные по всем видам учета; руководство, получающее отчеты по экономическим показателям производства; администратор системы, управляющий ее работой и производящий изменение исходных данных - регистрацию клиентов, поставщиков, формирование номенклатуры сырья и готовой продукции.

Сущность "История" предназначена для хранения истории посещения системы ее пользователями и содержит следующие реквизиты:

- код записи;
- дату и время посещения системы;
- код пользователя.

Отчеты предназначены для формирования отчетных документов и формируются по мере необходимости.

Информационная система должна иметь вид веб-приложения, благодаря чему обеспечиваются следующие преимущества:

- централизованное хранение данных;
- возможность одновременной работы нескольких пользователей;
- отсутствие необходимости установки программы на клиентские компьютеры, работа осуществляется с использованием любого браузера;
- высокая скорость работы приложения;
- возможность доступа к интерфейсу программы через Интернет. На рис. 1.9 представлена схема базы данных MySQL.

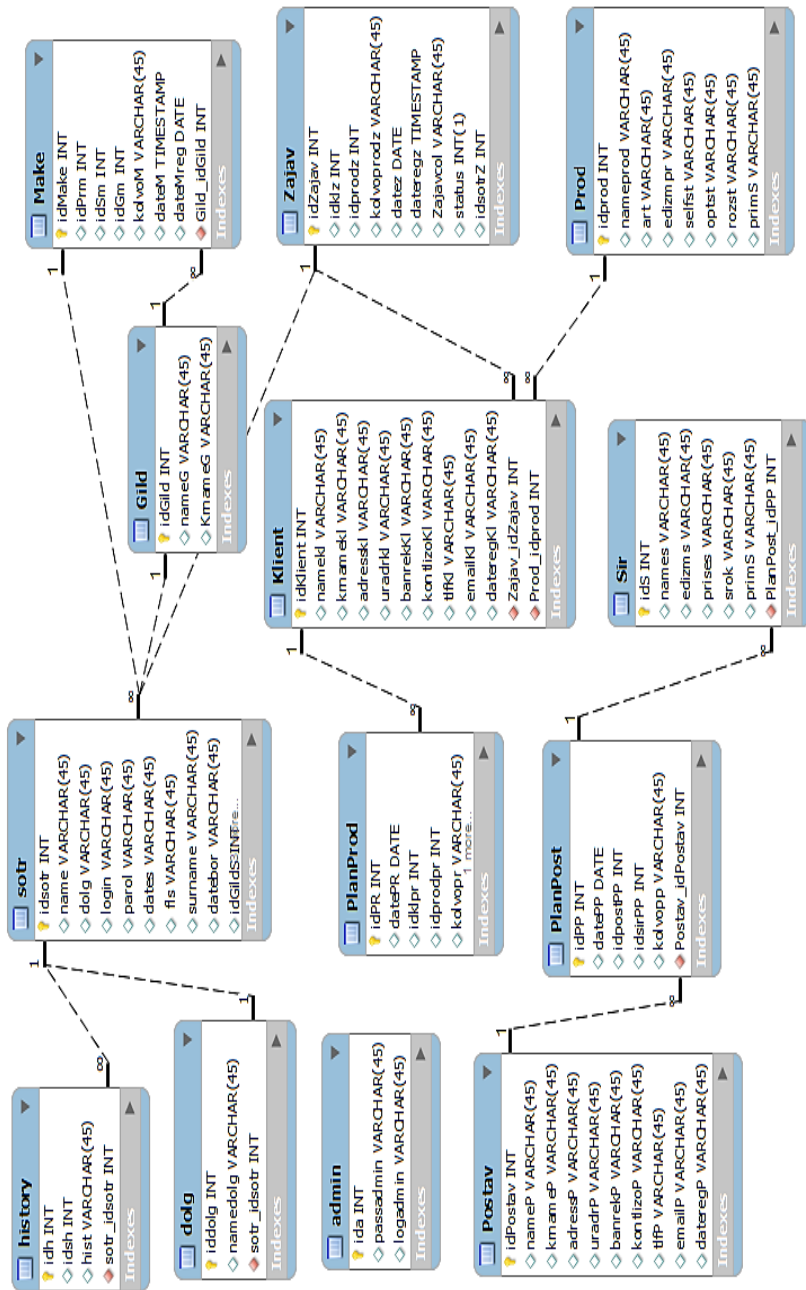


Рис. 1.9. Схема БД

1.6. Создание БД в MySQL

СУБД MySQL разработал Майкл (Монти) Видениус. Созданная им небольшая и очень эффективная для своего размера реляционная СУБД основана на традициях Hughes Technologies Mini SQL (mSQL).

Для работы с MySQL нужно выполнить следующие действия.

1. Разархивировать дистрибутив в произвольный каталог, запустить Setup.exe и установить MySQL в какой-нибудь каталог (например, C:\MySQL).

2. Если сервер установлен как служба, то он будет запущен автоматически. Если сервер установлен как приложение, то его следует запустить вручную. В любом случае полезно запустить программу WinMySQLAdmin.exe - простой интерфейс для администратора. На панели задач появится светофор: если он зеленый, то сервер запущен.

3. Запустить клиентскую часть MySQL - файл C:\MySQL\BIN\mysql.exe. Откроется консольное приложение, в котором можно набирать и выполнять команды MySQL. Для создания и заполнения базы данных нужно, как минимум:

1) создать базу данных командой CREATE DATABASE имя_БД;
(Не забывайте в конце команд ставить эту точку с запятой!),

2) выбрать текущую базу данных командой USE имя_БД; ,

3) создать таблицы командой CREATE TABLE ...; ,

4) заполнить таблицы данными командой INSERT INTO ...;
(INSERT - оператор языка SQL, который позволяет добавить строки в таблицу, заполняя их значениями. Значения можно вставлять, перечислив их в круглых скобках с помощью слова values через запятую или оператором select.). Можно написать файл-скрипт, содержащий команды MySQL, и выполнить его командой source полное_имя_скрипта (здесь точку с запятой ставить не надо!).

Сценарий создания и заполнения базы данных для нашего приложения может выглядеть следующим образом:

```
create database admin;
```

```
use admin;
```

```
create table dolg
```

```
( id_dolg int(11) primary key name_dolg, varchar(50), sotr_idsotr  
int(11));
```

```
insert into dolg (name_dolg) values ("Кладовщик"); insert into publishers  
(sotr_idsotr) values ("1");
```

```

create table Prod
( id_prod int(11) primary key name_prod, art edizmpr selfst optst rozst
  varchar(50), primP (text), udalPr int(11));

insert into Prod (name_ prod)
Values ("Труба ПНД техническая ");
insert into Prod (art)
Values ("4545454545");
insert into Prod (edizmpr)
Values ("Метр");
insert into Prod (selfst)
Values ("1000");
insert into Prod (optst)
Values ("1100");
insert into Prod (rozst)
Values ("1200 ");
insert into Prod (primP)
Values ("Труба техническая ");

```

1.7. Основы языка серверных сценариев PHP

Годом рождения PHP можно считать 1995 г., когда независимый программист Расмус Лерддорф написал скрипт для подсчета количества посетителей страницы со своим онлайн-резюме. Тот первый скрипт был написан на языке Perl, а так как средств для разработки интернет-приложений тогда было немного, то скрипт вызвал живой интерес программистов и пользователей во всем мире.

Сначала Лерддорф назвал свою программу Personal Home Page, а в дальнейшем, когда ее функции вышли далеко за пределы простого оформления домашней страницы, аббревиатуру PHP стали расшифровывать рекурсивно - как PHP Hypertext Processor.

Вскоре появилась версия PHP 2.0, написанная на языке C. В разработке версии 3.0 уже участвовала целая команда специалистов. Сейчас PHP - это не только язык серверных web-сценариев, но и творческое сообщество единомышленников, работающих над развитием своего программного продукта, который завоевал признание во всем

мире. По сведениям NETCRAFT (<http://www.netcraft.com>), PHP использует не более чем 1 000 000 хостов.

Данный язык, вобравший в себя лучшие черты Perl, C, Java, создавался специально для веб-приложений и получился очень удачным - лаконичным, простым, эффективным и надежным. Рассматриваемая в пособии версия 4.0 содержит такие возможности, как динамическое создание изображений, работу с PDF-файлами, поддержку многих форматов баз данных, интеграцию с XML, использование электронных платежных систем и т.п.

Существенным преимуществом языка PHP является тот факт, что в пределах одного web-документа можно перемешивать HTML-тэги и PHP-команды. Для того чтобы отделить PHP-код от HTML-тэгов, его ограничивают символами `<? ?>`. Работает все это следующим образом. Сначала на сервере выполняются PHP-команды. Они могут содержать обращение к базам данных, работу с файлами и прочие виды обработки информации. Очень удобными являются команды `print` и `echo`, которые выполняют печать непосредственно в HTML-документ. Документ после полной обработки на сервере отправляется пользователю и отображается у него в браузере. Пользователь никаких серверных команд уже не увидит - он увидит только результаты работы команд.

PHP может работать с разными веб-серверами (Apache, Microsoft IIS(PWS), Netscape Enterprize Server, Stronghold, Zeus) и, следовательно, на разных платформах (Unix, Solaris, FreeBSD, Windows 9x/NT/XP).

Рассмотрим установку PHP под конфигурацию Windows+Apache (может быть, экзотичную с точки зрения реального веб-сайта, но очень удобную для использования на локальном компьютере, особенно для учебных целей).

Прежде всего, следует установить и сконфигурировать Apache. Установка Apache, как и любой другой программы под Windows, не вызывает особых проблем. Если в процессе установки не были созданы (это зависит от версии) ярлыки для запуска и остановки программы в главном меню, можно создать их вручную. Для запуска используется файл `apache.exe`, для остановки - тот же файл с параметрами `apache.exe -k shutdown`. По умолчанию Apache устанавливается в каталог `C:\Program Files\Apache Group\Apache`.

После установки требуется открыть файл конфигурации `httpd.conf` (находится в подкаталоге `CONF`) и сделать в нем следующие

исправления: `ServerAdmin yourname@yoursite.com`. Это указывает почтовый адрес администратора сайта. Если сайт действующий, нужно указать реальный адрес.

В качестве имени сервера для локального компьютера следует использовать `localhost: ServerName localhost`.

Теперь можно запустить Apache через главное меню или проводник. Затем запустите браузер и введите адрес `http://localhost`. Если установка была выполнена успешно, то откроется страница с фирменным перышком Apache (рис. 1.10).



Рис. 1.10. Фирменный логотип Apache

Теперь можно устанавливать PHP. Распакуйте архив PHP в произвольный каталог, например, `C:\PHP`.

Найдите среди распакованных файлов `php.ini-dist` (в версии 3.0 его имя `php3.ini-dist`). Это файл конфигурации PHP. Скопируйте его в каталог `C:\WINDOWS` и переименуйте в `php.ini` (в версии 3.0 `php3.ini`).

Найдите файлы `php4ts.dll` и `Msvcr7.dll` и скопируйте их в каталог `C:\WINDOWS\SYSTEM`. Если файл `Msvcr7.dll` уже существует, переписывать его не надо.

Теперь нужно внести дополнительные изменения в файл `httpd.conf`. Найдите секцию, содержащую команды `ScriptAlias`. Здесь нужно создать строку `ScriptAlias /php4/ "C:/php/"`, означающую псевдоним для корневого каталога PHP.

Найдите секцию `AddType`. Добавьте или раскомментируйте, если они уже существуют, строки

```
AddType application/x-httpd-php .phtml .php
```

```
AddType application/x-httpd-php-source .phps
```

Данные строки объявляют, что файлы с PHP-сценариями могут иметь расширения `phtml` и `php`.

Найдите секцию `Action`. Добавьте в нее строку `Action application/x-httpd-php /php4/php.exe`. Указанная строка определяет, какое именно приложение будет выполнять PHP-сценарии.

После проведенных изменений в файле конфигурации следует остановить сервер `apache` и стартовать его заново.

Теперь можно протестировать PHP. Напишите файл с расширением phtml, содержащий следующий код:

```
<?
while (list($var, $value)=each($GLOBALS))
{
print "<br>$var => $value";
}
?>
```

символами <? ?> код сценария отделяется от HTML-текста) и поместите его в каталог HTDOCS - корневой каталог Apache. Теперь запустите браузер, просмотрите этот файл через адрес <http://localhost>. В строке поиска браузера можно увидеть длинный список глобальных переменных и их значений.

Специально для удобства разработчиков создан пакет Денвер, включающий в себя все необходимые нам программы - Apache, PHP и MySQL.

Типы данных

В PHP используются следующие:

- целые числа;
- вещественные числа;
- строки;
- логические величины;
- массивы;
- объекты.

Целые числа. Так же, как и в языке C, для целых чисел, кроме десятичного представления (1, 16, 255), допустимы восьмеричное (01, 020, 0377) и шестнадцатеричное (0x1, 0x10, 0xff).

Вещественные числа. Для чисел с десятичной точкой используются стандартная запись с фиксированной точкой (1.5, 0.999) и так называемая научная запись - с плавающей точкой (0.15e1, 9.99e-1).

Строки могут заключаться в двойные кавычки или в апострофы. Если в строке, заключенной в двойные кавычки, есть переменная, то вместо нее будет подставлено ее значение, например:

```
$year=2018;
$today="6 сентября $year года";
print $today;
Будет напечатано: 6 сентября 2018 года.
```

В двойных кавычках также интерпретируются и соответственно подставляются управляющие символы, такие как \n, \t, \r и подобные.

Логические величины. Нетрудно догадаться, что логический тип представлен значениями true false.

Массивы. Обычные массивы, называемые в PHP индексированными, можно создавать с помощью инициализации отдельных элементов:

```
$month[0]="Январь";  
$month[1]="Февраль";  
$month[2]="Март";
```

При этом можно даже не указывать индексы, элементы массива создаются последовательно:

```
$month[]="Январь";  
$month[]="Февраль";  
$month[]="Март";
```

Для инициализации также удобно использовать функцию array:

```
$month=array("Январь","Февраль","Март");
```

Кроме индексированных, в PHP используются ассоциативные массивы, в которых с каждым значением связывается строковый ключ:

```
$capital["Франция"]="Париж";  
$capital["Великобритания"]="Лондон"; $capital["Япония"]="Токио";
```

При использовании функции array задаются пары ключ/значение:

```
$capital=array("Франция"=>"Париж", "Великобритания"=>"Лондон",  
"Япония"=>"Токио");
```

Массивы могут быть также многомерными.

Объекты. Для определения класса используется команда class следующего вида:

```
class Light  
{  
var $isOn;  
function click($OnOff)  
{ $this->isOn=$OnOff; }  
}
```

Как видим, переменные элементы класса нужно объявлять с помощью ключевого слова **var**, при обращении к ним следует использовать ключевое слово **this**. В целом определение класса очень напоминает язык JavaScript. Создать объект какого-либо класса можно только с помощью команды `new`:

```
$bulb = new Light;
```

Переменные в PHP всегда начинаются со знака \$. Имя переменной может содержать буквы, цифры и некоторые специальные символы и должно начинаться с буквы или знака подчеркивания. Регистр букв учитывается.

Переменные не обязательно объявлять. Любая переменная объявляется неявно при первом использовании. Тип переменной также неявно определяется по типу хранящегося в ней значения, т.е. в языке PHP переменные слабо типизированы. Явно переменную можно объявить с помощью ключевого слова `var`.

По умолчанию любая переменная является локальной. Для того чтобы изнутри функции обратиться к глобальной переменной, нужно объявить ее с префиксом GLOBAL:

```
GLOBAL $x;
```

или использовать системный ассоциативный массив \$GLOBAL:

```
$GLOBALS["x"];
```

Статические переменные объявляются с ключевым словом `STATIC`:

```
STATIC $count=0;
```

Неявное преобразование (*juggling*) типа переменной происходит в том случае, если производится операция с переменными разных типов. Так, при выполнении арифметической операции над целым числом и строкой, представляющей собой целое число, результат будет целым числом. Если в операции участвуют целое и вещественное числа, результат будет вещественным числом, чтобы избежать потери точности.

Кроме неявного, можно использовать и явное преобразование типа `casting`. Чтобы явно преобразовать выражение, нужно перед ним указать в скобках ключевое слово типа (точно так же, как в языке C):

- `int` или `integer`;
- `real`, `double` или `float`;
- `string`;

- array;
- object.

Если переменная преобразуется в массив, она становится первым (т.е. нулевым) элементом массива.

Если переменная преобразуется в объект, она становится атрибутом объекта и ей назначается имя scalar.

Переменную можно определить как ссылку на другую переменную:

```
$x=0;
```

```
$y=&$x;
```

При этом \$y становится псевдонимом для \$x, и при изменении \$y будет изменяться также и \$x.

Массив \$GLOBALS, который уже упоминался, содержит множество полезных стандартных переменных. Их имена и текущие значения можно распечатать, если последовательно просмотреть весь этот глобальный массив:

```
while (list($var, $value)=each($GLOBALS))
{
print "<br>$var => $value";
}
```

Знак \$ перед именем переменной на самом деле представляет собой операцию макроподстановки, т.е. вместо имени переменной подставляется ее значение. Например:

```
$first="something";
```

```
$second="first";
```

```
print $($second);
```

Что будет напечатано? Очевидно, строка "something".

По окончании работы сценария все его переменные автоматически удаляются из памяти. Если же переменную нужно удалить раньше, можно использовать функцию unset:

```
unset ($MyArray);
```

Операции и операторы. В PHP используются операции, применяющие смесь из языков C и Perl (табл. 1.6).

Операции в PHP

Обозначение	Ассоциативность	Описание
()	-	Изменение приоритета
new	-	Создание экземпляра объекта
! NOT ~	П	Логическое и поразрядное отрицание
++ --	П	Инкремент, декремент
@	П	Маскировка ошибок
/ % *	Л	Деление, остаток, умножение
+ - .	Л	Сложение, вычитание, сцепление строк
<< >>	Л	Поразрядный сдвиг влево и вправо
< <= > >=	-	Меньше, меньше или равно, больше, больше или равно
== === != <>	-	Равно, идентично, не равно
& ^	Л	Поразрядные И, Исключающие ИЛИ, ИЛИ
&&	Л	Логически И, ИЛИ
?:	П	Тернарная операция
= += -= /= %= *= = . =	П	Присваивание
&= = <<= >>=		
AND XOR OR	Л	Логические И, исключающие ИЛИ, ИЛИ
// #	-	Однострочные комментарии
/* */	-	Многострочные комментарии

Операторы имеют два альтернативных синтаксиса (рис. 1.11)

Условный оператор	
if (условие) { блок } else { блок }	if (условие) блок else : блок endif;
Цикл с предусловием	
while (условие) { блок }	while (условие) блок endwhile;
Цикл с постусловием	
do { блок } while (условие);	do : блок while (условие);

Рис. 1.11. Синтаксис операторов (начало)

Цикл со счетчиком	
for (инициализация; условие выхода; действие по окончании ите- рации) { блок }	for (инициализация; условие выхода; действие по окончании ите- рации) : блок endfor;
Цикл для перебора массивов (нумерованного и ассоциативного)	
{ foreach (\$массив as \$элемент) : блок }	foreach (\$массив as \$элемент) : блок endforeach;
foreach (\$массив as \$ключ => \$элемент) { блок }	foreach (\$массив as \$ключ => \$эле- мент) : блок endforeach;
Оператор выбора	
switch (выражение) { case (условие1): блок case (условие2): блок ... case (условиеN): блок default: блок }	switch (выражение) : case (условие1): блок case (условие2): блок ... case (условиеN): блок default: блок endswitch;
Оператор прерывания цикла	
break;	
Оператор прерывания итерации цикла	
continue;	
Операторы печати	
print строка;	
echo строка;	

Рис. 1.11. Синтаксис операторов (окончание)

Функции. PHP - это процедурный язык с небольшими объектно-ориентированными возможностями, поэтому в нем большое внимание уделяется процедурам, или функциям, - в данном случае это одно и то же. Практически все в PHP реализовано через функции - от работы с

массивами до интеграции с XML и электронными платежными системами (приятное разнообразие после тотальной объектно-ориентированности!). Большинство функций можно применять без специального объявления, но некоторые пакеты специфических функций требуют упоминания в секции extensions файла php.ini. Разумеется, программист может писать и собственные функции.

Формат функции в PHP следующий:

```
function имя_функции ([$параметр1, $параметр2, ..., $параметрN])
{
    тело_функции
[return [возвращаемое_значение];]
}
```

Функции в PHP могут быть вложенными, т.е. одна функция может располагаться внутри другой (что недопустимо, например, в языке C). В этом случае внутреннюю функцию можно вызывать только после вызова внешней.

При вызове функций точно так же, как и при работе с переменными, можно применять макроподстановку \$:

```
function english()
{ print "Good morning!"; } function french()
{ print "Bon matin!"; }
$language="french";
$language(); // Вызов функции french
```

Массивы. Для работы с массивами разработано много полезных функций. Кратко рассмотрим самые часто используемые из них. Если тип параметра или возвращаемого значения функции не указан, значит, он может быть разный (табл. 1.7).

Таблица 1.7

Создание и изменение массивов

Оператор	Результат
1	2
array array ([значение1, значение2, ..., значение N]) void list (переменная1 [, переменная2, ... , переменнаяN])	Создает массив из указанных значений, используется в левой части оператора присваивания (в правой должен быть массив) и присваивает своим аргументам-переменным значения элементов этого массива
array range (int мин_значение, int макс значение)	Создает массив из целых чисел заданного диапазона

1	2
int array_push (array массив, значение1 [, значение2,..., значениеN])	Добавляет новые элементы в конец массива
void array_pop (array массив)	Удаляет элемент с конца массива
void array_shift (array массив)	Удаляет элемент с начала массива
void array_unshift (array массив, значение1 [, значение2,..., значениеN])	Добавляет новые элементы в начало массива
array array_reverse (array массив)	Возвращает перевернутый массив
array array_merge (array массив1, array массив2,... ,array массивN)	Объединяет несколько массивов в один и возвращает его
array array_slice (array массив1, int смещение [, int длина])	Возвращает "срез" массива, заданный смещением и длиной
Поиск в массиве	
bool in_array (элемент, array массив)	Проверяет, содержится ли элемент в массиве
Просмотр элементов массива	
reset (array массив)	Перемещает внутренний указатель массива на начало и возвращает его первый элемент
array each (array массив)	Возвращает пару ключ/значение из текущего элемента массива и перемещает указатель к следующему элементу
void end (array массив)	Перемещает внутренний указатель на конец массива
next (array массив)	Перемещает внутренний указатель на один элемент вперед и возвращает значение этого элемента; возвращает ложь, если вышли за пределы массива
prev (array массив)	Перемещает внутренний указатель на один элемент назад и возвращает значение этого элемента; возвращает ложь, если вышли за пределы массива
Размер массива	
int sizeof (array массив)	Возвращает размер массива
int count (переменная)	Если это массив, возвращает размер массива; если это скалярная переменная, возвращает 1; если переменная не существует, возвращает 0
array array_count_value (array массив)	Подсчитывает число вхождений каждого значения и возвращает пары значение/количество в виде ассоциативного массива

1	2
Сортировки	
sort rsort asort arsort ksort krsort usort uasort	Всевозможные сортировки на все случаи жизни - по возрастанию и по убыванию, по значениям и по ключам
void shuffle (array массив)	Перемешивает (на месте) значения элементов массива в случайном порядке

Строки. Это самый распространенный тип данных в PHP. Для работы со строками написано очень много функций, рассмотрим наиболее полезные из них (табл. 1.8).

Таблица 1.8

Редактирование строк

Оператор	Результат
1	2
string trim (string строка)	Удаляет лишние пробелы и символы \n, \t, \v, \b в начале и в конце строки
int strlen (string строка)	Возвращает длину строки
int strcmp (string строка1, string строка2)	Сравнивает строки и возвращает либо 0, либо положительное, либо отрицательное число (как в C)
int strcasecmp (string строка1, string строка2)	Сравнивает строки и возвращает либо 0, либо положительное, либо отрицательное число (как в C), при этом регистр символов не учитывается
void parse_str (string строка)	Выделяет в строке пары "переменная=значение" и присваивает данные значения реальным переменным. Удобно использовать при разборе строки параметров, полученной из URL
array explode (string разделитель, string строка [, int количество])	Разбивает строку, содержащую разделители, на подстроки и возвращает массив строк
string implode (string разделитель, array подстроки)	Объединяет подстроки из массива в единую строку, помещая между подстроками разделители (хотя подстроки можно сцепить просто операцией конкатенации ".")

1	2
int strpos (string строка, string подстрока [, int смещение])	Находит в строке первое вхождение подстроки и возвращает индекс первого символа подстроки (или false, если не найдено)
int strrpos (string строка, string символ)	Находит в строке последнее вхождение данного символа и возвращает его индекс (или 0, если не найдено)
string str_replace (string подстрока, string новая_подстрока, string строка)	Заменяет в строке все вхождения подстроки на новую подстроку
string substr (string строка, int начало [, int длина])	Выделяет подстроку из строки
int substr_count (string строка, string подстрока)	Подсчитывает количество вхождений подстроки в данную строку
string strtolower (string строка)	Преобразует все буквы к нижнему регистру
string strtoupper (string строка)	Преобразует все буквы к верхнему регистру
string ucfirst (string строка)	Преобразует к верхнему регистру первую букву
string ucwords (string строка)	Преобразует к верхнему регистру первую букву каждого слова

Также имеется набор функций для преобразования строк к HTML-формату и в обратную сторону [12].

Основы javascript. JavaScript - мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией языка ECMAScript (стандарт ECMA-262).

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

1.8. Разработка шаблонов и главной страницы сайта

Для корректного ввода информации в информационную систему используются формы, не позволяющие ввести излишнюю или неправильную информацию в систему.

В разрабатываемой системе применяются следующие виды форм:

- форма регистрации пользователей;

- форма авторизации;
- форма ввода данных;
- форма поиска;
- форма получения результатных данных.

Эскизы форм представлены на рис. 1.12 - 1.15.

Фамилия	Поле ввода данных	Подсказка
<input type="text"/>	<input type="text"/>	<input type="text"/>
Имя	Поле ввода данных	Подсказка
<input type="text"/>	<input type="text"/>	<input type="text"/>
Отчество	Поле ввода данных	Подсказка
<input type="text"/>	<input type="text"/>	<input type="text"/>
Должность	Поле ввода данных	Подсказка
<input type="text"/>	<input type="text"/>	<input type="text"/>
адрес	Поле ввода данных	Подсказка
<input type="text"/>	<input type="text"/>	<input type="text"/>
телефон	Поле ввода данных	Подсказка
<input type="text"/>	<input type="text"/>	<input type="text"/>
Кнопка отправки формы		
<input type="submit" value="Submit"/>		

Рис. 1.12. Эскиз формы регистрации пользователя

Логин	Поле ввода данных
<input type="text"/>	<input type="text"/>
Пароль	Поле ввода данных
<input type="text"/>	<input type="text"/>
Кнопка отправки формы	
<input type="submit" value="Submit"/>	

Рис. 1.13. Эскиз формы авторизации

Наименование	Выпадающий список	Подсказка
<input type="text"/>	<input type="text"/>	<input type="text"/>
Дата	Маска ввода ДД/ММ/ГГГГ	Подсказка
<input type="text"/>	<input type="text"/>	<input type="text"/>
Единица измерения	Поле ввода данных	Подсказка
<input type="text"/>	<input type="text"/>	<input type="text"/>
Стоимость	Поле ввода данных	Подсказка
<input type="text"/>	<input type="text"/>	<input type="text"/>
Количество	Поле ввода данных	Подсказка
<input type="text"/>	<input type="text"/>	<input type="text"/>
Поставщик	Поле ввода данных	Подсказка
<input type="text"/>	<input type="text"/>	<input type="text"/>
Кнопка отправки формы		
<input type="button" value="Submit"/>		

Рис. 1.14. Эскиз формы ввода данных

Наименование	Выпадающий список	Подсказка
<input type="text"/>	<input type="text"/>	<input type="text"/>
Кнопка отправки формы		
<input type="button" value="Submit"/>		
<input type="button" value="Экспорт в Excel"/>		

Рис. 1.15. Эскиз формы получения резульатной информации

Шаблоны. Применительно к веб-программированию шаблоном можно назвать часть веб-документа, которая используется в нескольких страницах. Шаблоны позволяют быстро проводить модификацию всего сайта: достаточно изменить информацию или оформление в шаблоне, и это отразится на всех страницах, использующих данный шаблон.

Для включения текста шаблона в документ используются функции `include` и `require` (табл. 1.9).

Работа с текстом в шаблоне

Функция	Результат
include (string имя файла)	Включает содержимое файла в сценарий
include_once (string имя_файла)	Включает содержимое файла в сценарий только один раз на протяжении сценария. Если файл уже был включен, то повторное включение игнорируется

Вызов данной в табл. 1.9 функции можно поместить в блок if-else, например:

```
<?
if ($user=="admin')
{
include("adminMenu.phtml");
}
else {
include("guestMenu.phtml ");
}
?>
```

Заметим, что фигурные скобки здесь обязательны, так как файл, включаемый в сценарий, как правило, состоит из нескольких строк (табл. 1.10).

Таблица 1.10

Работа в шаблоне

Функция	Результат
require (string имя_файла)	Включает содержимое файла в сценарий независимо от каких-либо условий. Даже если вызов функции находится в блоке if и условие ложно, файл все равно будет включен
require_once (string имя_файла)	Аналогично require, но включает содержимое файла в сценарий только один раз на протяжении сценария. Если файл уже был включен, то повторное включение игнорируется

Обычно в отдельный шаблон удобно выделить шапку страницы, а в другой шаблон - нижний колонтитул или средства навигации:

```
<? include("shapka.phtml"); ?>
текст HTML-документа
<? include("bottomMenu.phtml"); ?>
```

Проанализируем, какие элементы оформления в нашем электронном магазине обязательно будут появляться на каждой странице.

К верхнему шаблону можно отнести фоновый рисунок, логотип вместе с маленькой формой для авторизации, а также строку меню. Обратите внимание, что заголовок текущего режима работы - "Справочник", "Отчеты" и.т.п., на полоске соответствующего цвета также можно вынести в шаблон. Для этого заведем PHP-переменные \$color и \$title, которые будем инициализировать в соответствующих страницах, а распечатывать - в верхнем шаблоне.

Работа с системой начинается с авторизации пользователя, для чего применяется форма авторизации (рис. 1.16)

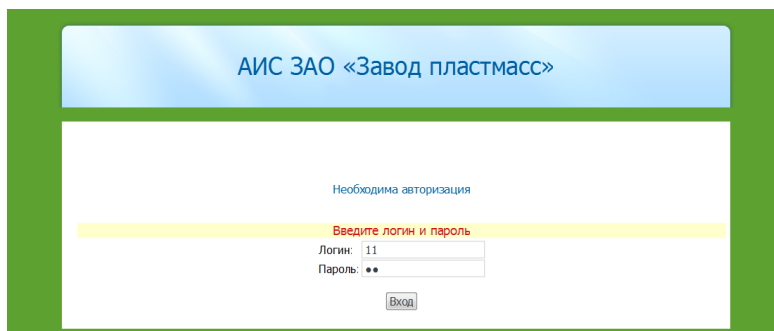


Рис. 1.16. Форма авторизации

При правильном вводе пароля и логина осуществляется вход в систему.

```
<?php
session_start();
require "dbconnect.php";
if(isset($_POST['parol']))
{
$res=mysql_query("SELECT IDsotr FROM sotr WHERE
login='".$_POST['login']."' AND parol='".$_POST['password']."' AND
udal=0");
if(mysql_num_rows($res))
{
while($prise_mas=mysql_fetch_row($res))
{
$_SESSION['var']="$prise_mas[0]";
Header ("Location: dannie.php");
$ccc=date("d/m/y G:i:s", time());
```



```

$hist=mysql_query("insert into history (idsh, hist)
values ('{$_SESSION['var]}','{$ccc}')");
}
}
else
{ //такого пользователя нет
echo "<center><br><br>Введены неверные логин или па-
роль<br><br>    <span><a href='login.php'>Попробуйте еще
раз</a></span> <br> или обратитесь к администратору системы<br>
";
}
}
?>

```

Итак, верхний шаблон header.phtml будет выглядеть следующим образом. Javascript подключается напрямую в HTML-файл. Самый простой способ - это написать javascript-команды внутрь тэга <script> где-нибудь в теле страницы. Когда браузер читает HTML-страничку и видит <script>, он первым делом читает и выполняет код, а только потом продолжает читать страницу дальше. Так, в следующем примере будет показано начало страницы, затем три раза выполнится функция alert, которая выводит окошко с информацией, а только потом появится остальная часть страницы.

Обычно javascript стараются отделить от собственно документа. Для этого его помещают внутрь тэга HEAD, а в теле страницы по возможности оставляется чистая верстка.

В следующем примере javascript-код только описывает функцию count_rabbits, а ее вызов осуществляется по нажатию на кнопку input. В примере использовались следующие элементы:

```
<script type="text/javascript"> ... </script>
```

Тэг <script> сообщает браузеру о том, что внутри находится исполняемый скрипт. Атрибут type говорит о том, что это javascript. Вообще говоря, атрибут может отсутствовать - разницы нет, но с точки зрения стандарта его следует указать.

Конструкция for использует обычный цикл, по синтаксису аналогичный другим языкам программирования.

Объявление var i - это объявление переменной внутри цикла, i - локальная переменная.

Функция alert выводит сообщение на экран и ждет, пока посетитель не нажмет ОК.

Пример программного кода:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
'http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

```
<html xmlns='http://www.w3.org/1999/xhtml'>

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

<title>ИС </title>
<link rel='stylesheet' type='text/css' href='css/index.css' />
<script type='text/javascript' src='js/jquery.min.js'></script>
<script type='text/javascript' src='js/jquery-ui.min.js'></script>
<script type='text/javascript' src='js/jquery-form.min.js'></script>
<script type='text/javascript' src='js/jquery-ifixpng.min.js'></script>
<script type='text/javascript' src='js/jquery-scrollable.min.js'></script>
<script type='text/javascript' src='js/index.js'></script>

</head>

<body>
  <br>

  <div id='main' class='noifixpng'>

    </div>
    <center>
    <div id='head'>
      <div id='head-m'>
        <div id='head-i'>

          <h1> <br>
          ИС </h1>

        </div>
      </div>
    </div>

    <div id='menu'>
      <div id='menu-m'>
        <div id='menu-i'>
          <ul>

            <li><a href='post.php'
class='c1'><span><i>Справочники</i></span></a>
              <div class='submenu'><div class='submenu-m'>
                <ul
                  class='c1'><li><div><a
href='post.php'>Поставщики</a></div></li>
                <li><div><a href='klient.php'>Клиенты
</a></div></li>
                <li><div><a
href='sirie.php'>Сырье</a></div></li>
```

```

href='produkt.php'>Продукция</a></div></li>
href='zecha.php'>Цеха</a></div></li>
class='c6'><span><i>Отчеты</i></span></a>
href='otz.php'>Заявки</a></div></li>
продаж</a></div></li>
<li><div><a href='otpost.php'>План поставок</a></div></li>
<li><div><a href='make.php'>Производство</a></div></li>
class='c3'><span><i>Ввод</i></span></a>
href='zajav.php'>Заявки </a></div></li>
<li><div><a href='prvvod.php'>Продажи</a></div></li>
<li><div><a href='postvod.php'>Поставки</a></div></li>
class='c4'><span><i>Данные</i></span></a>

```

```

class='c5'><span><i>Выход</i></span></a>
</li><a href='login.php'
</ul>
</div>
<div class='submenu-b'></div></div></li>
</div>
</div>
</div>
<br>
</body>
</html>

```

Для работы со справочниками используются формы ввода, приведенные на рис. 1.17 - 1.21.

При правильном вводе пароля и логина осуществляется вход в систему. Кроме того, в системе доступен модуль администратора.

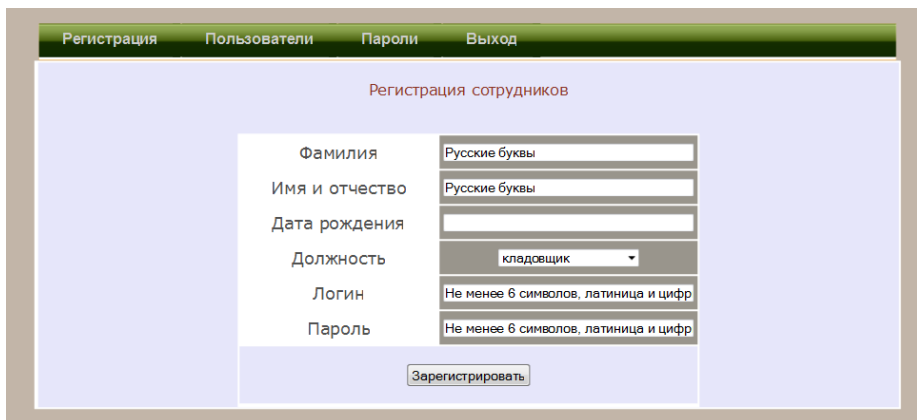


Рис. 1.17. Регистрация пользователей

При регистрации пользователя предусмотрена проверка правильности ввода данных (см. рис. 1.17).

Регистрация Пользователи Пароли Выход

Регистрация сотрудников

Фамилия Только русский алфавит

Имя и отчество Русские буквы
Только русский алфавит

Дата рождения Поле обязательно к заполнению.

Должность клавишник

Логин Не менее 6 символов, латиница и цифр
Поле обязательно к заполнению

Пароль Не менее 6 символов, латиница и цифр
Только латинский алфавит и цифры

Рис. 1.18. Регистрация сотрудников

Регистрация Пользователи Пароли Выход

Список пользователей

№	Фамилия	Имя и отчество	Дата рождения	Дата регистрации	Должность	Логин	Пароль	Удалить/восстановить
1	Иванов	Иван Иванович	1989-12-31	2010-04-01 23:57:25	клавишник	111111	111111	<input type="button" value="✖"/>
2	Петров	Александр Сергеевич	1970-01-10	2010-04-01 23:30:35	начальник склада	222	222	<input type="button" value="✖"/>
4	Сидоров	Иван Петрович	1980-12-12	2010-04-01 23:30:34	старший клавишник	10	10	<input type="button" value="✖"/>

Всего зарегистрировано пользователей 5

Рис. 1.19. Список пользователей

При удалении пользователя он отмечается цветом, тут же появляется возможность его восстановления (см. рис. 1.19).

Регистрация Пользователи Пароли Выход								
Список пользователей								
№	Фамилия	Имя и отчество	Дата рождения	Дата регистрации	Должность	Логин	Пароль	Удалить/восстановить
1	Иванов	Иван Иванович	1989-12-31	2010-04-01 23:37:25	кладовщик	111111	111111	
2	Петров	Александр Сергеевич	1970-01-10	2010-04-01 23:30:35	начальник склада	222	222	
4	Сидоров	Иван Петрович	1980-12-12	2010-04-01 23:30:34	старший кладовщик	10	10	

Всего зарегистрировано пользователей 5

Рис. 1.20. Выделение цветом удаленного пользователя

Регистрация Пользователи Пароли Выход	
Смена пароля и логина	
Пользователь	<input type="text" value="кладовщик Иванов Иван Иванович"/>
Новый пароль	<input type="text" value="Не менее 6 символов, латиница и цифр"/>
Новый логин	<input type="text" value="Не менее 6 символов, латиница и цифр"/>
<input type="button" value="Зменить"/>	

Рис. 1.21. Страница смены пароля

Разработка витрины электронного магазина на PHP и MySQL.

Для MySQL в PHP разработаны функции прямого доступа к данным.

Так как обычно подключения к БД используются множеством скриптов, то нередко настройки подключения выносятся в отдельный файл, благодаря чему легче их оперативно изменять.

Итак, создадим файл connection.php и добавим в него следующие строки:

1	<?php	Для подключения к MySQL из PHP нам надо указать настройки подключения: адрес сервера, логин, пароль, название базы данных и т.д.
2	\$host = 'localhost'; // адрес сервера	
3	\$database = 'compstore'; // имя базы данных	
4	\$user = 'root'; // имя пользователя	
5	\$password = '1234567'; // пароль	
6	?>	

Так как мы будем подключаться к серверу на локальной машине, то адресом сервера будет localhost. В качестве базы данных выберем созданную вначале базу данных.

По умолчанию на локальном сервере MySQL уже есть пользователь root, под которым мы и будем подключаться. И также нам необходим пароль, который был указан при установке MySQL.

Теперь мы можем подключиться к базе данных:

<pre><?php 1 require_once 'connection.php'; // подклю- 2 чаем скрипт 3 4 // подключаемся к серверу 5 \$link = mysqli_connect(\$host, \$user, 6 \$password, \$database) 7 or die("Ошибка " . mysqli_error(\$link)); 8 9 // выполняем операции с базой данных 10 11 // закрываем подключение 12 mysqli_close(\$link); ?></pre>	<p>Первым делом подключаем скрипт с настройками с помощью инструкции require_once.</p> <p>Для открытия подключения применяем функцию mysqli_connect(). Он принимает все конфигурационные настройки и подключается к серверу. В случае ошибки подключения срабатывает оператор die(), который выводит сообщение об ошибке и завершает работу скрипта. А в случае успешного подключения функция mysqli_connect() возвращает объект подключения в виде переменной \$link.</p> <p>После окончания работы подключение нужно закрыть. Для этого применяется функция mysqli_close(), которая в качестве параметра принимает объект подключения. Чтобы осуществить запрос к базе данных, нам надо использовать функцию mysqli_query(), которая принимает два параметра: объект подключения и строку запроса на языке SQL</p>
--	--

Например:

<pre><?php require_once 'connection.php'; // подключаем 1 скрипт 2 3 // подключаемся к серверу 4 \$link = mysqli_connect(\$host, \$user, 5 \$password, \$database) 6 or die("Ошибка " . mysqli_error(\$link)); 7 8 // выполняем операции с базой данных 9 \$query = "SELECT * FROM sotr"; 10 \$result = mysqli_query(\$link, \$query) or 11 die("Ошибка " . mysqli_error(\$link)); 12 if(\$result) 13 { 14 echo "Выполнение запроса прошло 15 успешно"; 16 } 17 18 // закрываем подключение mysqli_close(\$link); ?></pre>	<p>Функция <code>mysqli_query()</code> возвращает объект <code>\$result</code>, который содержит результат запроса. В случае неудачи данный объект содержит значение <code>false</code>.</p>
--	--

Функции обработки результатов запроса. Если запрос, выполненный с помощью функции `mysqli_query()` успешно выполнен, то в результате клиент получит набор записей, который может быть обработан следующими функциями PHP:

`mysqli_result()` - получить необходимый элемент из набора записей;

`mysqli_fetch_array()` - занести запись в массив;

`mysqli_fetch_row()` - занести запись в массив;

`mysqli_fetch_assoc()` - занести запись в ассоциативный массив;

`mysqli_fetch_object()` - занести запись в объект.

Просмотр справочников. Мы предполагали, что у сотрудников должна быть возможность работать со справочниками.

Для работы со справочниками используются формы ввода, приведенные на рис. 1.22 - 1.26:


```

        <tr>
            <td width="30">&nbsp;</td>
<td width="320" class="log">
<form method="post" id="signup">
<input type="text" size="60" value="Не указан" onFocus="doClear(this)"
onBlur="doDefault(this)" name="nameP" class="required" title="<br>
Поле обязательно к заполнению"></td>
    <td width="30">&nbsp;</td>
        <td width="320" class="log">
            <input type="text" size="60" value="Не указан" onFocus="doClear(this)"
onBlur="doDefault(this)" name="krnameP" class="required" title="<br>
Поле обязательно к заполнению">
        </td>
            <td width="30">&nbsp;</td>
</tr>

                <tr>
<td colspan="5">&nbsp;</td>
</tr>
<tr >
<td width="30">&nbsp;</td>
<td width="320" class="date1" >Фактический адрес </td>
<td width="30">&nbsp;</td>
        <td width="320" class="date1" >Юридический адрес </td>
            <td width="30">&nbsp;</td>
</tr>

                <tr>
            <td width="30">&nbsp;</td>
<td width="320" class="log">
<input type="text" size="60" value="Не указан" onFocus="doClear(this)"
onBlur="doDefault(this)" name="adressP" class="required" title="<br>
Поле обязательно к заполнению"></td>
    <td width="30">&nbsp;</td>
        <td width="320" class="log">
            <input type="text" size="60" value="Не указан" onFocus="doClear(this)"
onBlur="doDefault(this)" name="uradrP" class="required" title="<br>
Поле обязательно к заполнению">
        </td>

```

```

        <td width="30">&nbsp;</td>
</tr>

        <tr>
<td colspan="5">&nbsp;</td>
</tr>

<tr >
<td width="30">&nbsp;</td>
<td width="320" class="date1" >Банковские реквизиты </td>
<td width="30">&nbsp;</td>
    <td width="320" class="date1" >Контактное лицо </td>
    <td width="30">&nbsp;</td>
        </tr>

        <tr>
            <td width="30">&nbsp;</td>
<td width="320" class="log">
<input type="text" size="60" value="Не указан" onFocus="doClear(this)"
onBlur="doDefault(this)" name="banrekP" class="required" title="<br>
Поле обязательно к заполнению"></td>
    <td width="30">&nbsp;</td>
        <td width="320" class="log">
            <input type="text" size="60" value="Не указан" onFocus="doClear(this)"
onBlur="doDefault(this)" name="kontlizoP" class="required" title="<br>
Поле обязательно к заполнению">
        </td>
    <td width="30">&nbsp;</td>
</tr>

        <tr>
<td colspan="5">&nbsp;</td>
</tr>

        <tr >
<td width="30">&nbsp;</td>
<td width="320" class="date1" >Телефон </td>
<td width="30">&nbsp;</td>
    <td width="320" class="date1" >Электронная почта </td>
    <td width="30">&nbsp;</td>
        </tr>

```

```

        <tr>
            <td width="30">&nbsp;</td>
            <td width="320" class="log">
                <input type="text" size="60" value="Не указан" onFocus="doClear(this)"
                onBlur="doDefault(this)" name="tlfP" class="required" title="<br>
                Поле обязательно к заполнению"></td>
            <td width="30">&nbsp;</td>
            <td width="320" class="log">
                <input type="text" size="60" value="Не указан" onFocus="doClear(this)"
                onBlur="doDefault(this)" name="emailP" class="required" title="<br>
                Поле обязательно к заполнению">
            </td>
            <td width="30">&nbsp;</td>
        </tr>

```

```

        <tr>
            <td colspan="5">&nbsp;</td>
        </tr>

```

```

        <tr>
            <td colspan="5" class="pp"><input type="submit" name="post"
            value="Зарегистрировать" >
            </form></td>
        </tr>

```

```

        <tr>
            <td colspan="7">&nbsp;</td>
        </tr>
    </tr>
    <td colspan="7">

```

```

    <?php
    require "dbconnect.php";
    if(isset($_POST['post']))
    {
    $query1="insert into postav (nameP`,`krnameP`,`adressP`,`ura-
    drP`,`banrekP`,`kontlizoP`,`tlfP`,`emailP`)
    values
    ('".$_POST['nameP']. "','".$_POST['krnameP']. "','".$_POST['adressP']. "','.
    $_POST['uradrP']. "','".$_POST['banrekP']. "','".$_POST['kontlizoP']. "','

```

```

"$_POST['tlfP'].",
"$_POST['emailP'].")";
if(mysql_query($query1) ) /* если запрос успешно произведен */
{
echo "<p class='pp' width='200'>Регистрация успешно завершена. <br>
</p>
";
}
else
{
echo "<p class='prav'>Не получилось</p>
";
}
}

?>
</td>
</tr>
</table>
<?php
require "dbconnect.php";
$sod1=mysql_query("SELECT count( idPostav )
FROM postav
WHERE udalp=0");
while($sod_mas1=mysql_fetch_row($sod1))
{
$kolvo=$sod_mas1[0];
}
if ($kolvo>0)
{
echo "<table border='0' align='center' width='900'><tr><td class='table1'> Всего
зарегистрировано поставщиков &nbsp;  <b>$kolvo</b></td></tr> </table>\n";
}
?>
<?php
require "dbconnect.php";
$res=mysql_query("SELECT idPostav, `nameP`, `knameP`, `adressP`, `ura-
drP`, `banrekP`, `kontlizoP`, `tlfP`, `emailP` FROM `Postav` where udalp=0");
while($res_id=mysql_fetch_row($res))
{

```

```

echo " <center> <table> <tr>
<td width='45' class='tz'> $res_id[0] </td>
<td width='90' class='tz'> $res_id[1] </td>
<td width='90' class='tz'> $res_id[2] </td>
<td width='90' class='tz'> $res_id[3] </td>
<td width='90' class='tz'> $res_id[4] </td>
<td width='90' class='tz'> $res_id[5] </td>
<td width='90' class='tz'> $res_id[6] </td>
<td width='90' class='tz'> $res_id[7] </td>
<td width='90' class='tz'> $res_id[8] </td>
<td width='45' class='table3'><form method='post' name='udal' action='udal.php'>
<INPUT TYPE='hidden' NAME='udal' VALUE='$res_id[0]'>
<input type='image' src='image/udal.png' TITLE='Удалить' WIDTH='40'
HEIGHT='40' >
</form>
</td>
</tr> </table>\n";
}

```

?>

</div>

</div>

<div id='main' class='noifxpng'>

</div>

<center>

При нажатии кнопки «Зарегистрировать» введенная в поля формы информация заносится в базу данных.

Кроме того, в системе доступны справочники «Клиенты», «Сырье», «Продукция», «Цеха».

АИС ЗАО «Завод пластмасс»

→ Справочники → Счеты → Ввод → Данные → Вывод

Регистрация клиентов

Полное наименование	Краткое наименование
Московская кабельная компания	ОАО МКК
Фактический адрес	Юридический адрес
Москва, Ленинд, 12	Москва, Ленинд, 12
Банковские реквизиты	Контактное лицо
Сбербанк	Петров А. Н.
Телефон	Электронная почта
495-999-09-09	mkk@mkk.ru

[Зарегистрировать](#)

Всего зарегистрировано клиентов 1

1	Московская кабельная компания	ОАО МКК	Москва	Москва	Банк Москва	Юдин С. А.	444-35-66	moska@mail.ru	
---	-------------------------------	---------	--------	--------	-------------	------------	-----------	---------------	--

Рис. 1.23. Форма для заполнения справочника «Клиенты»

АИС ЗАО «Завод пластмасс»

→ Справочники → Счеты → Ввод → Данные → Вывод

Виды поставляемого сырья

Полное наименование	Единица измерения
Полипропилен «Рандом сополимер» PPRC 3/ППРС 3	тонна
Стоимость	Срок хранения
12000	3 года

Примечание

Это легкий и достаточно прочный сополимер из разряда термопластов. Полипропилен химически стоек к многим видам растворителей кислотного и щелочного типа. Полипропиленовые трубы и фитинг эксплуатируются при температурах от -10°С до +90°С.

[Зарегистрировать](#)

Всего видов сырья 5

Полипропилен «Рандом сополимер» PPRC 3/ППРС 3	тонна	1000	3 года	
---	-------	------	--------	--

Рис. 1.24. Форма для заполнения справочника «Сырье»

АИС ЗАО «Завод пластмасс»

← Справочник → Отчеты → Ввод → Данные → Вывод

Виды продукции

Полное наименование	Артикул
Труба ПНД теоническая 110М, 2 СЛ	45454545
Единица измерения	Себестоимость
метр	1000
Оптовая цена	Розничная цена
1100	1200

Описание

труба теоническая

Зарегистрировать

Всего видов продукции: 6

Рис. 1.25. Форма для заполнения справочника «Продукция»

АИС ЗАО «Завод пластмасс»

← Справочник → Отчеты → Ввод → Данные → Вывод

Цеха

Полное наименование	Краткое наименование
Производственный цех № 5	ЦВ05

Зарегистрировать

Всего цехов: 3




Производственный цех № 2	Цех №2	
Производственный цех № 3	Цех №3	
Производственный цех № 4	Цех №4	

Рис. 1.26. Форма для заполнения справочника «Цеха»

Формы списков приведены на рис. 1.27 - 1.31.

Всего зарегистрировано поставщиков: 2



5	Ульяновский химический завод	ЗАО УХЗ	Ульяновск	Ульяновск	ВТБ	Петров С. А.	333-45-56	ulhz@mail.ru	
4	Запорожский аминичный завод	ООО "ЗАЗ"	Украина, Запорожье	Украина, Запорожье	Банк Москва	Иванов А. А.	333-22-22	zaz@mail.ru	

Рис. 1.27. Список поставщиков

Всего зарегистрировано клиентов 1									
1	Московская кабельная компания	ОАО МКК	Москва	Москва	Банк Москва	Юдин С. А.	444-55-66	moskva@mail.ru	

Рис. 1.28. Список клиентов

Всего видов сырья 5				
Полипропилен «Фандом сополимер» РРРС 3/ППРС 3	тонна	1000	3 год	
Гонополимер полипропилена	тонна	1450	2 год	
Металлоценовый полипропилен	тонна	1678	1 год	
Блок-сополимер пропилена с этиленом	тонна	2400	1 год	
Статистический сополимер полипропилена	тонна	4789	8 месяцев	

Рис. 1.29. Список видов сырья

Всего видов продукции 6						
Труба ПНД техническая 110x4,2 СЛ	5656565656	метр	56	59	76	
Отвод ПНД 110*90	76043583	штука	23	34	45	
Муфта ПНД 160	67676767	штука	200	210	220	
Труба канализационная 110 с раструбом (L=1,5м)	9023849283	метр	100	110	120	
Труба ПБ 100 315x15 SDR 21 ГОСТ 18599-2001	0909090909	метр	1560	1670	1800	
Труба ПБ 100 500x36,8 SDR 13,6 ГОСТ 18599-200	507659865	метр	4500	4600	4700	

Рис. 1.30. Список видов продукции

Всего цехов 3		
Производственный цех № 2	Цех №2	
Производственный цех № 3	Цех №3	
Производственный цех № 4	Цех №4	

Рис. 1.31. Список цехов

Для ввода данных о производственной деятельности предприятия используется форма, представленная на рис. 1.32.

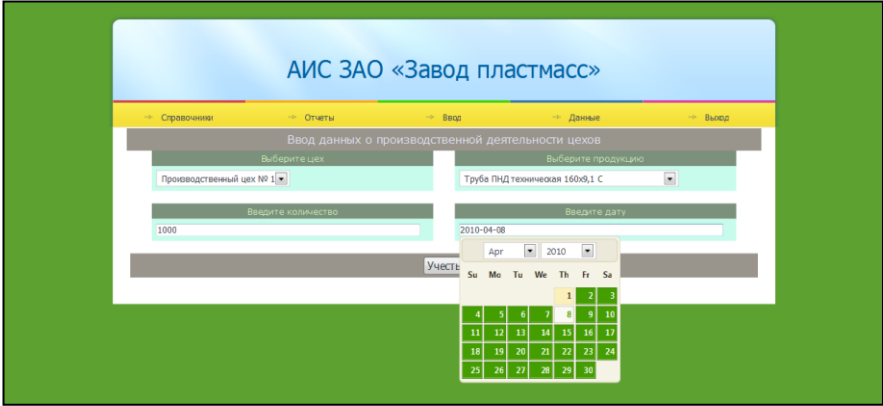


Рис. 1.32. Ввод данных о производственной деятельности

Формы ввода данных о заявках, планах продаж и планах поставок приведены на рис. 1.33, 1.34, 1.35, соответственно:

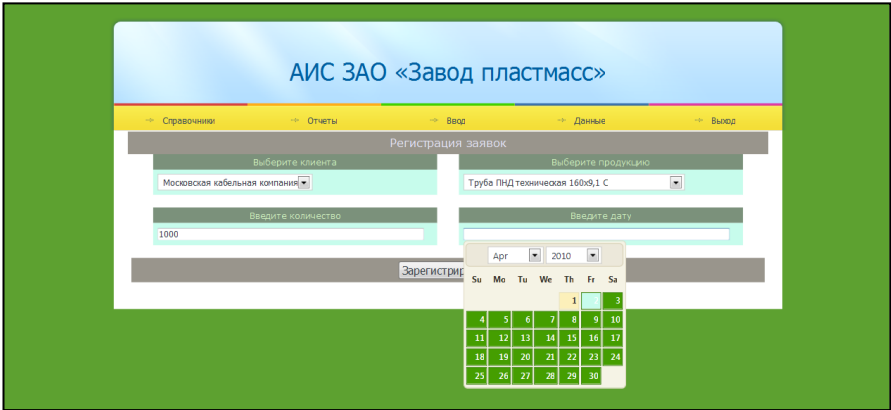


Рис. 1.33. Форма учета заявок на поставку продукции

```
<table width="900" border="0" cellspacing="0" cellpadding="0">
  <tr >
    <td width="30">&nbsp; </td>
    <td width="320" class="date1" >Выберите клиента</td>
    <td width="30">&nbsp; </td>
    <td width="320" class="date1" >Выберите продукцию </td>
    <td width="30">&nbsp; </td>
  </tr>
  <tr>
```

```

        <td width="30">&nbsp;</td>
<td width="320" class="log">
<form method="post" id="signup">
<select name="idklz"><center>
    <?php
    while ($klient=mysql_fetch_row($res_id1))
    {
    echo "<center><option value='$klient[0]'$>$klient[1]</option>\n";
    }
    ?>
    </select>
</td>
<td width="30">&nbsp;</td>
    <td width="320" class="log">
    <select name="idprodz"><center>
    <?php
    while ($prod=mysql_fetch_row($res_id2))
    {
    echo "<option value='$prod[0]'$>$prod[1]</option>\n";
    }
    ?>
    </select>
    </td>
        <td width="30">&nbsp;</td>
</tr>

                                <tr>
<td colspan="5">&nbsp;</td>
</tr>
<tr >
<td width="30">&nbsp;</td>
<td width="320" class="date1" >Введите количество</td>
<td width="30">&nbsp;</td>
    <td width="320" class="date1" >Введите дату</td>
    <td width="30">&nbsp;</td>
        </tr>

                                <tr>
    <td width="30">&nbsp;</td>
<td width="320" class="log">

```



```

</p>
";
}
else
{
echo "<p class='prav'>Не получилось</p>
";
}
}

```

Рис. 1.34. Форма для ввода плана продаж

Рис. 1.35. Форма для формирования плана поставок

Полный код сценария по каждому справочнику приведен в приложении.

Формы для получения отчетных документов и результаты этого приведены на рис. 1.36 - 1.39.

→ Справочники → Отчеты → Ввод → Данные → Выход

Список заявок

Введите необходимые реквизиты

Для поиска достаточно ввести одну или несколько букв (цифр) в любое поле

Дата регистрации заявки	Наименование клиента	Наименование продукции
<input type="text" value="2010-04-02"/>	<input type="text" value="МКК"/>	<input type="text" value="Труба"/>

Результаты поиска

Всего найдено заявок 5

Рег. номер	Дата	Клиент	Продукция	Ед. изм.	Кол-во	Стоимость	Общая стоимость
1	2010-04-01	ОАО МКК	Труба ПНД техническая 160х9,1 С	метр	1000	167	167000
2	2010-04-01	ОАО МКК	Труба канализационная 110 с раструбом (L=1.5м)	метр	1000	110	110000
3	2010-04-01	ОАО МКК	Отвод ПНД 110*90	штука	130	34	4420
4	2010-04-01	ОАО МКК	Труба ПНД техническая 110х4,2 СЛ	метр	3000	59	177000
5	2010-03-01	ОАО МКК	Труба ПНД техническая 160х9,1 С	метр	2000	167	334000

Рис. 1.36. Форма для получения и образец списка заявок

План продаж

Для ввода даты выберите дату, для просмотра продаж за месяц введите номер месяца в виде -03-

Дата или период продаж	Наименование клиента	Наименование продукции
<input type="text" value="2010-04-06"/>	<input type="text" value="МКК"/>	<input type="text" value="Труба"/>

Результаты поиска

Всего найдено записей 8

Итого на сумму 30564000 рублей

Рег. номер	Дата	Клиент	Продукция	Ед. изм.	Кол-во	Стоимость	Общая стоимость
6	2010-04-01	ОАО МКК	Труба ПНД техническая 160х9,1 С	метр	1000	167	167000
7	2010-04-02	ОАО МКК	Труба ПНД техническая 160х9,1 С	метр	2000	167	334000
8	2010-04-03	ОАО МКК	Труба ПНД техническая 110х4,2 СЛ	метр	3000	59	177000
9	2010-04-04	ОАО МКК	Отвод ПНД 110*90	штука	4000	34	136000
10	2010-04-05	ОАО МКК	Муфта ПНД 160	штука	5000	210	1050000
11	2010-04-06	ОАО МКК	Труба канализационная 110 с раструбом (L=1.5м)	метр	10000	110	1100000
12	2010-04-12	ОАО МКК	Труба ПЭ 100 500х36,8 SDR 13,6 ГОСТ 18599-200	метр	3000	4600	13800000
13	2010-04-12	ОАО МКК	Труба ПЭ 100 500х36,8 SDR 13,6 ГОСТ 18599-200	метр	3000	4600	13800000

Рис. 1.37. Форма для получения и образец плана продаж

АИС ЗАО «Завод пластмасс»

-- Справочники -- Отчеты -- Ввод -- Данные -- Вывод

План поставок

Для ввода даты выберите дату, для просмотра поставок за месяц введите номер месяца в виде -03-

Дата или период поставок	Наименование поставщика	Наименование сырья
2010-04-06	УХЗ	Полипропилен

Результаты поиска

Всего найдено записей 3

Итого на сумму 26623000 рублей

Рег. номер	Дата	Клиент	Продукция	Ед. изм.	Кол-во	Стоимость	Общая стоимость
1	2010-04-02	ЗАО УХЗ	Полипропилен «Фандом сополимер» РРРС 3/ППРС 3	тонна	1000	1000	1000000
2	2010-04-02	ЗАО УХЗ	Металлоцепаый полипропилен	тонна	1000	1678	1678000
3	2010-04-15	ООО "ЗАЗ"	Статистический сополимер полипропилена	тонна	5000	4789	23945000

Рис. 1.38. Форма для получения данных о производстве

-- Справочники -- Отчеты -- Ввод -- Данные -- Вывод

Данные о производстве

Для ввода даты выберите дату, для просмотра производства за месяц введите номер месяца в виде -03-

Дата или период производства	Номер цеха	Наименование продукции
2010-04-08	2	Труба

Результаты поиска

Всего найдено записей 2

Итого себестоимость 240600 рублей

Итого опт 290900 рублей

Итого розница 344600 рублей

Дата	Номер	Цех	Продукция	Количество	Себестоимость	Итого себестоимость	Итого по оптовым ценам	Итого по розничным ценам
2010-04-01	3	Цех №1	Труба ПНД техническая 160x9,1 С	1000	123	123000	167000	185000
2010-04-01	4	Цех №2	Труба ПНД техническая 110x4,2 СЛ	2100	56	117600	123900	159600

Рис. 1.39. Экранная форма данных о производстве

Глава 2. ОСНОВНЫЕ СИСТЕМЫ УПРАВЛЕНИЯ КОНТЕНТОМ ВЕБ-САЙТОВ

2.1. Виды CMS-систем, функции

Все сайты делятся на два типа:

- созданные на CMS - на системах управления сайтом (неважно, самописных, бесплатных или массовых коммерческих);
- сайты, функциональность которых задается программистами. Специалисты могут использовать только языки программирования или работать с фреймворками.

Сайт состоит из внешней и внутренней частей. Внешняя - это дизайн и контент, внутренняя часть - это база данных и административная панель. При разработке сайта на CMS необходимо создать только внешнюю часть - дизайн, сверстать его и «наложить на движок», а при самостоятельной разработке придется создавать и всю начинку.

Промежуточный вариант, который совмещает оба метода, - самописные системы управления контентом. В этом случае сначала с нуля создается скелет функций, а затем решение становится типовым, и уже на его основе создаются похожие сайты. Как правило, самописные CMS используются для решения довольно узких задач.

Функции CMS. Функции систем управления контентом можно разделить на несколько основных категорий.

1. Создание - предоставление авторам удобных и привычных средств создания контента.

2. Управление - хранение контента в едином *репозитории*. Это позволяет следить за версиями документов, контролировать, кто и когда их изменял, убеждаться, что каждый пользователь может изменить только тот раздел, за который он отвечает. Кроме того, обеспечивается интеграция с существующими информационными источниками и ИТ-системами. CMS поддерживает контроль над рабочим потоком документов, т.е. контроль за процессом их одобрения. Таким образом, управление контентом включает в себя хранение, отслеживание версий, контроль за доступом, интеграцию с другими информационными системами и *управление потоком* документов.

3. Публикация - автоматическое размещение контента на термине пользователя. Соответствующие инструменты автоматически адаптируют внешний вид страницы к дизайну всего сайта.

4. Представление - дополнительные функции, позволяющие улучшить форму представления данных; например, можно строить навигацию по структуре *репозитория*.

Системы управления контентом делятся на четыре основные категории, которые частично перекрываются:

1) системы управления исходными кодами традиционно поддерживают управление исходными кодами программ и часто предоставляют некоторый веб-интерфейс, который может использоваться внутри *корпоративной сети*, а также вне ее для параллельной работы с исходными кодами;

2) системы управления документами предназначены для организаций, оперирующих с большим количеством документов (например, офисы больших компаний, редакции и страховые компании);

3) системы управления веб-контентом представляют собой новую индустрию программных продуктов. Эти системы предназначены для разработки и управления веб-сайтами различной сложности. Обычно такие системы поддерживают и некоторый тип управления *потоками работ*;

4) системы *электронной коммерции* - обеспечивают хранение и управление электронными каталогами товаров. По сути, данные системы незначительно отличаются друг от друга. Самое главное отличие рассматриваемых систем - это использующие их люди.

Использование CMS предоставляет следующие преимущества.

1. Оперативное обновление информации - информацию публикует сотрудник, владеющий информацией, без дополнительных посредников в виде технических специалистов. CMS предназначены для автоматизации процесса публикации информации на веб-сайте, предоставляя пользователям возможность самим публиковать материалы в WWW и определять их визуальное представление, используя для этого стандартные средства, не требующие знания языка HTML и процедур, достаточно сложных для неспециалиста. С помощью CMS можно, не будучи профессиональным разработчиком, создавать и модифицировать *информационное наполнение* сайтов.

2. Снижение стоимости поддержки - обновление информации производится самостоятельно, нет необходимости оплачивать труд собственного или внешнего веб-мастера. Снижение стоимости происходит за счет уменьшения потерь времени на поиски документов, пресечения дублирования и ошибок, увеличения скорости связи с партнерами и клиентами.

3. Предоставление дополнительных сервисов пользователю. Некоторые сервисы (поиск, форумы, голосования и т.д.) требуют интерактивного взаимодействия с пользователем и уже реализованы в рамках CMS.

4. Уменьшение сроков и стоимости разработки - наиболее востребованная функциональность уже реализована в CMS и может быть сразу использована.

5. Повышение качества разработки - при разработке полностью или частично используются готовые модули, которые уже прошли неоднократное тестирование.

6. Снижение стоимости дальнейших модификаций - CMS дают возможность разделить данные и их представление. Это позволяет гораздо проще изменить внешний вид сайта, чем в случае со статическим сайтом.

Среди CMS-систем часто выделяют так называемые каркасы (content management framework, CMF) - инструментарию для создания системы.

Разработкой систем управления контентом занимаются многие компании, в том числе IBM, Microsoft, Oracle, Macromedia.

Стандартизация CMS. В последнее время начали появляться организации, пытающиеся объединить разработчиков CMS, создать единую *информационную среду* для потенциальных пользователей подобных систем, продвигать и утверждать единые стандарты. Прежде всего, это ассоциации OSCOM (*Open Source Content Management*) и CMSWatch.

























OSCOM утвердила такие стандарты, как WebDav, RSS, ATOM и JSR-170.




















В свою очередь, CMSWatch ежегодно выпускает отчет, включающий в себя обзор рынка CMS-систем, сравнение некоторых из них, описание жизненного цикла контента и управления им в CMS-системах.

Рейтинг предлагает полный список популярных в России CMS-систем управления сайтами. От верного выбора CMS зависит успешность будущего сайта, а правильно подобранный функционал движка обеспечит удобное и простое развитие ресурса. Самое простое решение данной проблемы - воспользоваться рейтингом CMS Magazine и выбрать лучшую CMS в зависимости от вида и тематики вашего сайта (табл. 2.1).

Таблица 2.1

Рейтинг CMS

№ п/п	Логотип CMS	Виды CMS ↓↑	тИЦ сайта ↓↑	Кол-во работ ↓ ↑	Суммарный тИЦ ↓↑	Средний тИЦ ↓↑
1	2	3	4	5	6	7
		CS-Cart	★	★	★	★
1		1С-Битрикс	14 000	35 057	2 162 674	62
2		WordPress ^{free}	120 000	13 453	328 038	24
3		Joomla! ^{free}	8000	10 491	210 721	20
4		Evolution CMS ^{free}	20	10 336	136 225	13
5		InSales	7300	9389	62 155	7
6		StoreLand	3000	9348	11 300	1
7		AMIRO.CMS	2700	8688	53 945	6
8		MODX ^{free}	120	7831	131 381	17
9		CS-Cart	475	7806	37 905	5
10		UMI.CMS	2500	5460	150 271	28
11		NetCat	2100	5212	172 601	33
12		Drupal ^{free}	0	5140	257 079	50
13		DIAFAN.CMS	600	4081	47 150	12
14		Shop-Script	2000	3711	81 270	22
15		HostCMS	3900	3559	96 880	27
16		Opencart ^{free}	2300	2999	34 172	11
17		VamShop	375	2718	59 665	22
18		TYPO3 CMS ^{free}	120	2596	38 705	15
19		Айтекс-CMS	130	2541	32 375	13
20		UralCMS	0	2474	21 330	9
21		Moguta.CMS ^{free}	800	1893	6830	4
22		CMS.CITRON	750	1881	22 750	12
23		PHPShop	900	1350	17 105	13
24		A5.ru ^{free}	2000	1242	2430	2

1	2	3	4	5	6	7
25		UniversalCMS	300	1171	3030	3
26		Canape CMS	550	1143	13 701	12
27		ADVANTSHOP	450	1010	14 550	14
28		ImageCMS Shop	375	899	11 705	13
29		SiteSoft.CMS	0	812	138 080	170
30		InstantCMS ^{free}	2300	768	10 785	14
31		Simpla	425	672	9845	15
32		SiteEdit	0	653	9610	15
33		django CMS ^{free}	30	590	38 423	65
34		DataLife Engine	4300	583	60 760	104
35		CMS Brane	0	533	8990	17
36		Shop2YOU	160	514	9790	19
37		Панель управления сайтом		463	12 105	26
38		ABO.CMS	210	434	22 700	52
39		CMS Sitebill	100	411	2750	7
40		ПростоСайт	160	404	10 245	25
41		"Платформа"		393	14 264	36
42		WebSitePro		388	5620	14
43		CMS S.Builder	90	380	13 960	37
44		PrestaShop ^{free}	2500	366	3880	11
45		ELiTES-CMS ^{free}	0	354	12 495	35
46		Текарт CMS/CMF ^{free}	10	350	12 776	37
47		R70.CMS		345	3680	11
48		ExpertPlus CMS ^{free}	250	325	6110	19
49		X3M.CMS	10	324	10 510	32
50		Joostina ^{free}	60	305	4790	16

2.2. Функционирование CMS

Основная идея систем управления контентом - разделение визуального дизайна сайта и его информационного наполнения. При создании сайта с помощью такой системы разрабатывается набор шаблонов страниц, в которых впоследствии размещается информация. В этом случае роль разработчиков (фактически это группа внедрения) ограничивается только созданием "начальной" информационной системы на основе системы управления контентом, затем пользователи сами публикуют требуемую информацию и определяют ее представление. Управление сайтом сводится к минимуму: администратору остается только управлять пользователями.

Пользователи CMS делятся на две группы: на создателей шаблонов страниц и авторов контента (информационного наполнения). Таким образом, одна группа пользователей создает структуру и оформление страниц, а другая наполняет его содержанием (рис. 2.1).

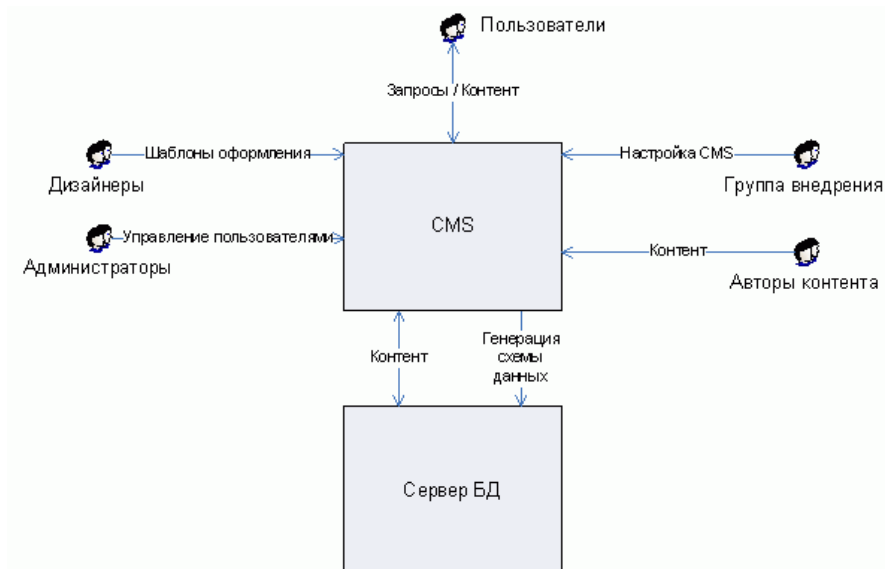


Рис. 2.1. Схема функционирования CMS

Функции систем управления контентом структурированы согласно жизненному циклу системы.

Сначала группа внедрения разворачивает ядро CMS и создает в СУБД информационное хранилище контента - БД. Далее

администратор предоставляет доступ к системе различным пользователям, затем создается контент, он публикуется и к нему применяются шаблоны оформления.

Рассмотрим CMS WordPress - это не просто платформа для ведения блогов. С ее помощью можно создавать совершенно разные сайты.

Любой шаблон состоит из 10 базовых файлов.

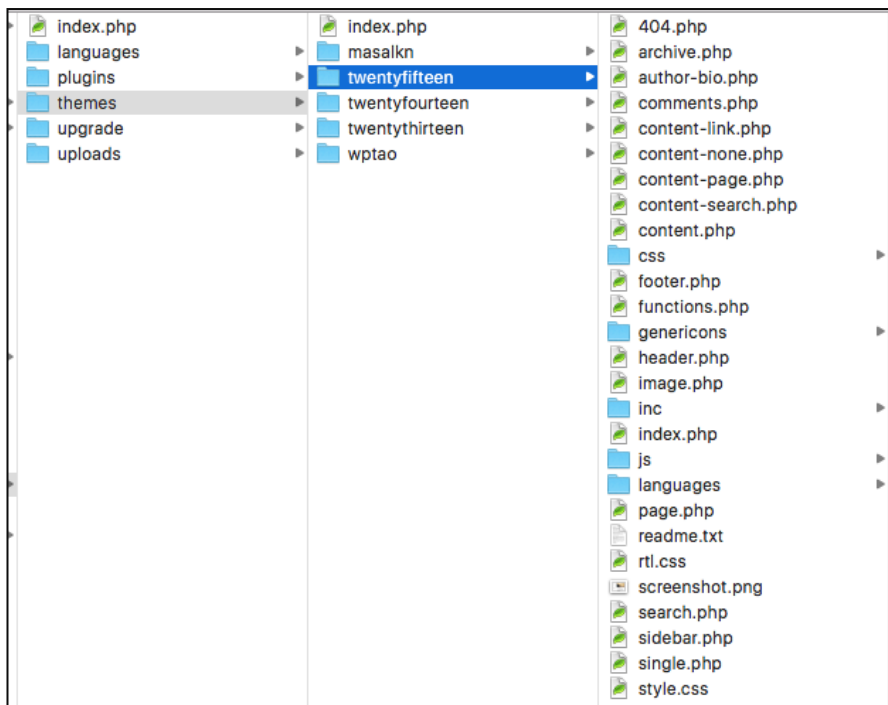


Рис. 2.2. Пример структуры WordPress-шаблона

Базовые файлы:

style.css - файл таблицы стилей шаблона. Этот файл должен быть у любой темы, так как именно он отвечает за ее объявление, а также может хранить дополнительную информацию: имя автора шаблона, версию шаблона, домашнюю страницу автора и т.д. Ну и конечно же, прямое назначение - это все CSS-стили;

index.php - файл отвечает за отображение главной страницы WordPress-шаблона. При верстке своего макета всегда помните, что главная страница может быть как статичной, так и динамичной;

single.php - файл, отвечающий за вывод каждого отдельного поста вашей темы. В качестве примера можно привести данную статью, которую вы читаете;

page.php - файл отвечает за формирование статичных страниц. Обычно это страницы о контактах, об авторе, о ресурсе и т.д.;

header.php - файл формирует шапку сайта и хранит в себе все важные метатэги для продвижения;

footer.php - файл отвечает за отображение подвала сайта;

sidebar.php – файл формирует отображение сайдбара, или, по-русски, боковой колонки блога или сайта;

functions.php - файл, отвечающий за дополнительный функционал шаблона. Например: вывод последних комментаторов на моем блоге формируется именно в этом файле;

comments.php - файл отвечает за отображение комментариев у шаблона;

404.php – файл отвечает за отображение страницы с 404-й ошибкой;

search.php - файл, отвечающий за отображение страницы поиска.

Необязательные файлы:

category.php - файл отвечает за отображение анонсов в категориях (если файла нет, то формирование осуществляется за счет файла `index.php`);

tag.php - файл отвечает за отображение анонсов на странице тэгов (если файла нет, то формирование осуществляется за счет файла `index.php`);

taxonomy.php - файл отвечает за отображение анонсов на страницах таксономии (если файла нет, то формирование осуществляется за счет файла `index.php`);

author.php - файл отвечает за отображение анонсов статей определенного автора (если файла нет, то формирование осуществляется за счет файла `index.php`);

attachment.php - файл отвечает за вывод прикрепленного файла;

searchform.php - файл отвечает за формирование формы поиска.

Если в вашем шаблоне присутствуют файлы из необязательного списка, то вы должны знать, что они всегда имеют приоритет выше, чем файл `index.php`. Это можно объяснить тем, что `index.php` - общий случай, а, допустим, `tag.php` - уже частный и, тем более, конкретный. Следовательно, его и нужно воспринимать.

Папки обычно служат для хранилища определенного типа файлов. Например: папка `image` хранит в себе файлы изображений, которые относятся к данной теме, папка `JS` - файлы скриптов, а папка `inc - php` файлы, которые включаются (инклюдятся) в существующие файлы темы.

Такого рода папок может быть сколько угодно, и называться они могут по-разному. Суть их в том, чтобы устранить бардак в шаблоне и привести все к какой-либо иерархии. Обычно этот файл содержит скриншот шаблона, чтобы в админке, когда вы выбирали тему для сайта или блога, могли видеть ее в качестве миниатюры. Размер такого файла должен быть 880x660 px (рис. 2.3).

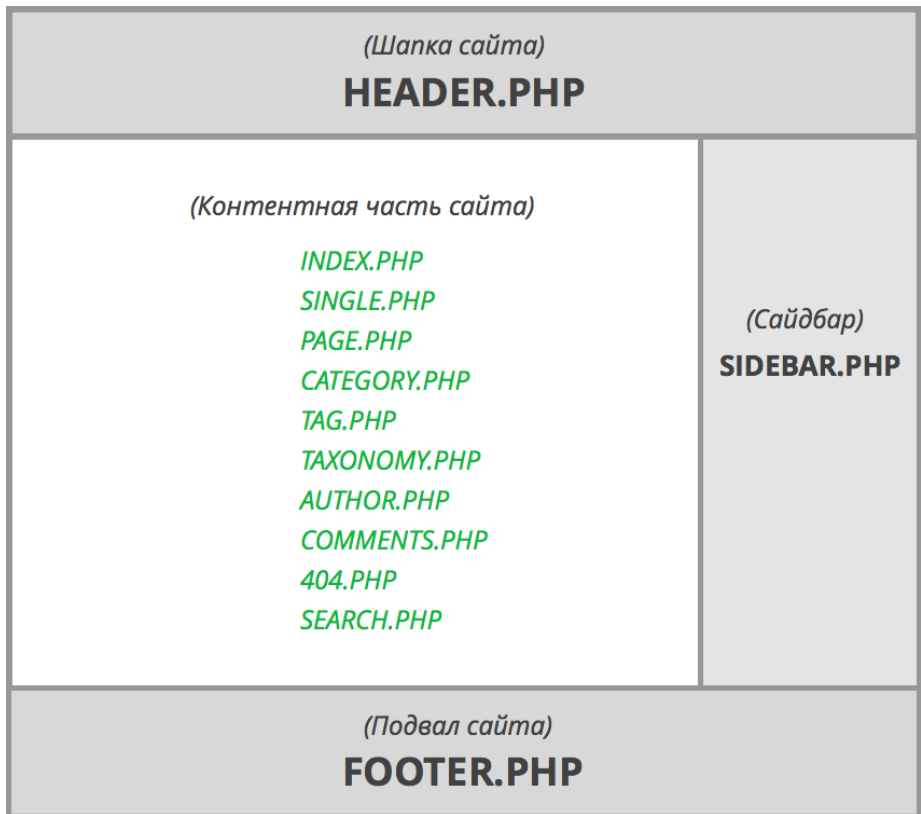


Рис. 2.3. Визуальное представление структуры шаблона

Структура, которая представлена на рис. 2.3, соответствует структуре разрабатываемого портала. Несмотря на то, что шаблоны разных авторов выглядят по-разному, все они имеют в своей основе вышеперечисленные файлы.

Ниже приведено несколько примеров того, что можно сделать после более детального знакомства с платформой:

- интернет-магазин. Существует множество плагинов, которые можно использовать (например, бесплатный плагин WooCommerce);
- веб-сайт для бизнеса. Используя WordPress, можно создать профессиональный веб-сайт, не зная языков программирования;
- форумы. Существуют плагины, помогающие создать собственную доску объявлений, на основе которой можно быстро организовать сообщество;
- целевую страницу (landing page). Это также посадочная страница - веб-страница, основной задачей которой является сбор контактных данных целевой аудитории.

Работа в CMS-системе WordPress начинается с выбора шаблона, что является важным шагом для каждого веб-мастера. Во многом при выборе шаблона новички руководствуются внешним видом (дизайном), ведь именно он определяет, как будет выглядеть сайт. Никто сразу не задается вопросом, как шаблон может облегчить или усложнить работу над сайтом. Те же, кто уже ведет свой сайт, знают, что дизайн - это важный фактор, но далеко не первостепенный, и неправильный выбор здесь способен подкинуть много дополнительных работ.

Когда речь идет о выборе шаблона, должны учитываться следующие критерии:

- 1) скорость;
- 2) адаптивность;
- 3) SEO-оптимизация;
- 4) дизайн и функциональность;
- 5) возможности поддержки и обновления.

От указанных факторов зависит, сколько времени, сил и средств вы потратите на доработку и продвижение сайта. Важно проверить шаблон до принятия его окончательного использования на сайте.

Рассмотрим пример создания посадочной страницы (лендинга) на примере создания Инвестиционного портала (рис. 2.4).



Рис. 2.4. Тематическая страничка

Сайт-одностраничник (лендинг) призван усилить вовлеченность посетителей в диалог с компанией и завоевать их доверие.

Конверсия Landing page на 60% зависит от традиционных правил создания эффективной посадочной страницы, цель которой вызвать доверие человека и в чем-то его убедить.

Основные элементы Landing page (Landing - пейзаж):

- 1) заголовок;
- 2) описание оффера;
- 3) картинка или видео;
- 4) отзывы, логотип или знак качества;
- 5) лид-форма (выполнение целевого действия на лендинге.

Например, отправка заявки через форму или нажатие на кнопку, чтобы перейти к следующему шагу);

6) кнопка к призыву.

Разберем каждый блок посадочной страницы по отдельности.

1. Тематическая картинка. Должен быть сильный главный экран, поскольку его увидят 80% людей, и лишь 20% прокрутят страничку вниз. Все самое главное (триггеры и т.д.) размещают здесь.

Триггеры на сайте вызывают острую потребность в товаре и желание купить его здесь и сейчас. Объясняйте посетителю все, что может вызвать у него сомнения.

Важно упрощение взаимодействия пользователя и сайта. Не надо заставлять человека делать лишние шаги. Каждый дополнительный шаг повышает вероятность потери пользователей и препятствует созданию полноценного сообщества, хотя человек инстинктивно тянется к группе.

Заголовки отражают пользу для человека, поиск, релевантные слова.

Призыв к действию должны выделить, сделать заметным.

Ценность для человека - вызвать доверие и убедить человека.

Один слой - одна единица смыслов.

Нужны короткие тексты. Длинные тексты не хочется читать, они выглядят устрашающе. Короткое читать легко, поэтому их читают.

Рекомендуется разбивать текст на короткие блоки (рис. 2.5).

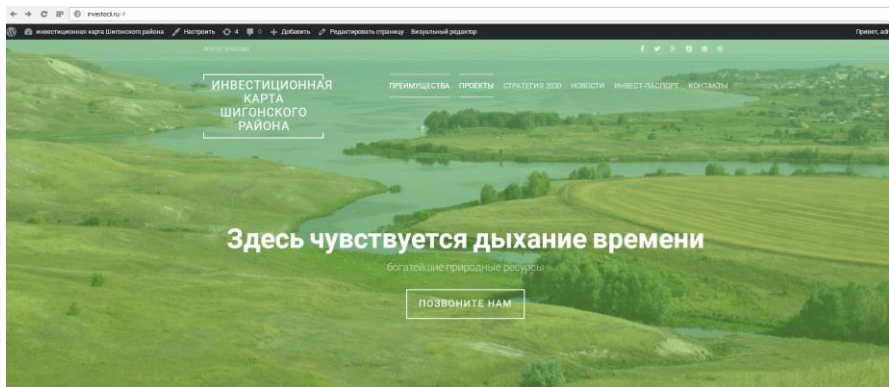


Рис. 2.5. Главный заголовок (тематическая страничка)

Можно добавить герб, флаг, социальные сети и ссылки на любой ресурс, прямую линию (телефон, работающий в режиме 24/7).

2. Описание оффера - ценности и пользы вашего маркетингового предложения, товара и услуги (рис. 2.6). Здесь должны быть:

- 1) перечень стратегических приоритетных проектов, рассказ о них;
- 2) место для новых идей бизнеса и общественных проектов;
- 3) место применения новых технологий, входа в цифровое будущее;
- 4) новые схемы привлечения инвесторов.

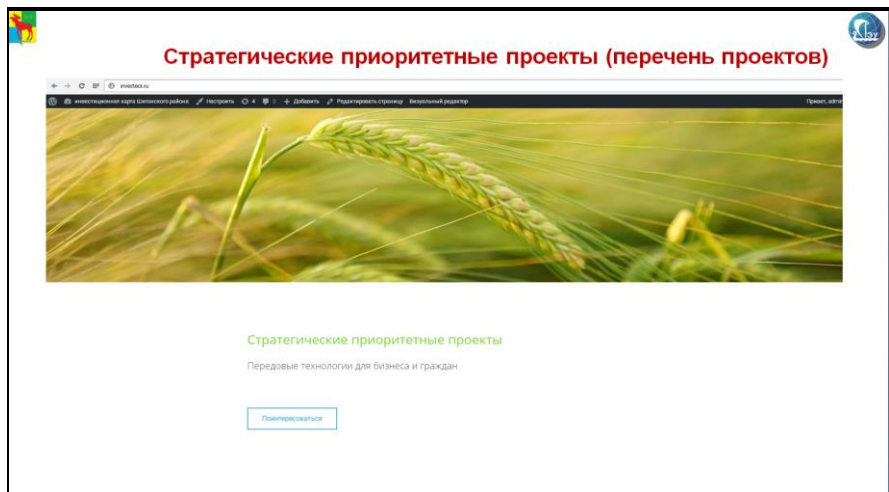


Рис. 2.6. Описание оффера

3. Картинка или видео. Социальные доказательства (рис. 2.7). Этот блок указывает на стратегию роста (можно выделить активных

людей района "Люди роста"). Развитие социальных блоков на интернет-порталах привело к возникновению феноменов, которые называют по-разному: мудрость толпы, краудсорсинг, викиномика, общественная поддержка, паутина соучастия [17, 18, 20].

Термин "краудсорсинг" (от англ. crowd - толпа, source - источник) был впервые использован Джеффом Хоувом [20] для обозначения нового способа выполнения сложного задания путем его передачи во внешнюю среду организации.

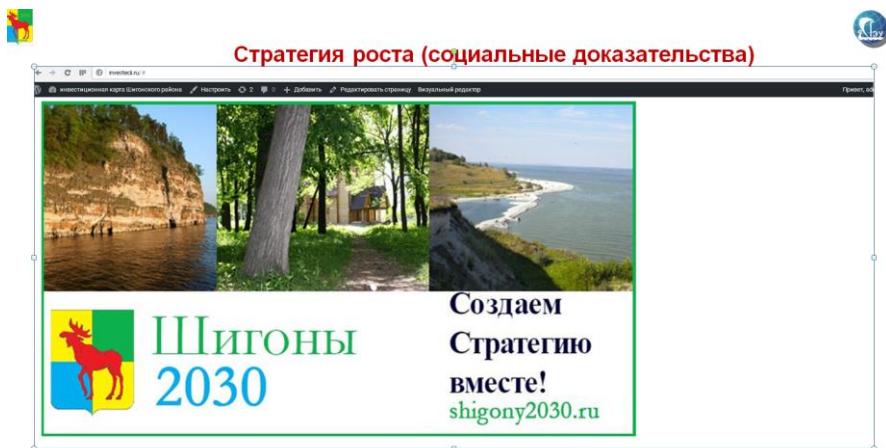


Рис. 2.7. Социальные доказательства

4. Отзывы, логотип или знак качества. Этот блок знакомит посетителей с наиболее интересной и актуальной информацией о Шигонском районе Самарской области. История села Шигоны тесно связана с историей села Усолье, и нужно рассказать о местных достопримечательностях с точки зрения развития здесь туризма. История района очень богатая, особенно стоит вспомнить усадьбу графа В.П. Орлова-Давыдова в Усолье (рис. 2.8).

Можно добавить ФИО знаменитостей, которые жили в данном районе, публикации этих людей, отзывы посетителей, источники материалов и цитаты из СМИ, условия конкурсов, также важно рассказать о наградах, подарках, призах.



Яркие достопримечательности

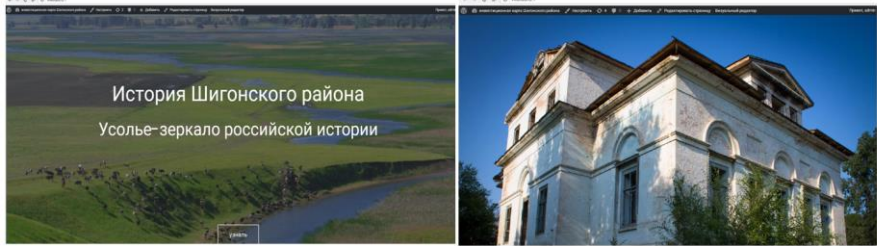


Рис. 2.8. Яркие достопримечательности Шигон

5. Лид-форма применяется для выполнения целевого действия на лендинге. Стоит сделать акцент на основные экономические показатели (рис. 2.9).



Рис. 2.9. Экономические показатели

По итогам анализа передовых инвестиционных порталов в качестве ключевых экономических показателей отражают:

- валовой региональный продукт;
- объем производимой в районе продукции, которая экспортируется;
- наименования торгово-экономических партнеров;
- объем прямых инвестиций в день.

Одним из инструментов развития инвестиционной привлекательности и инвестиционного климата территорий является электронная инвестиционная карта, на которой отмечаются следующие ключевые объекты территории:

- потенциальные области;
- промышленные объекты;
- исторические памятники культурного наследия;
- сельское хозяйство;
- агропромышленный комплекс;
- лесничество.

Нанесение объектов на электронную карту осуществляется с помощью контрастного восприятия (цифры, иконки, графики, сравнительные диаграммы).

6. Кнопка к призыву. В этом блоке разрабатывается форма обратной связи (с кнопкой призыва к действию "Стать инвестором"). Грамотно разработанная форма обратной связи обеспечивает 50% успеха (рис. 2.10).

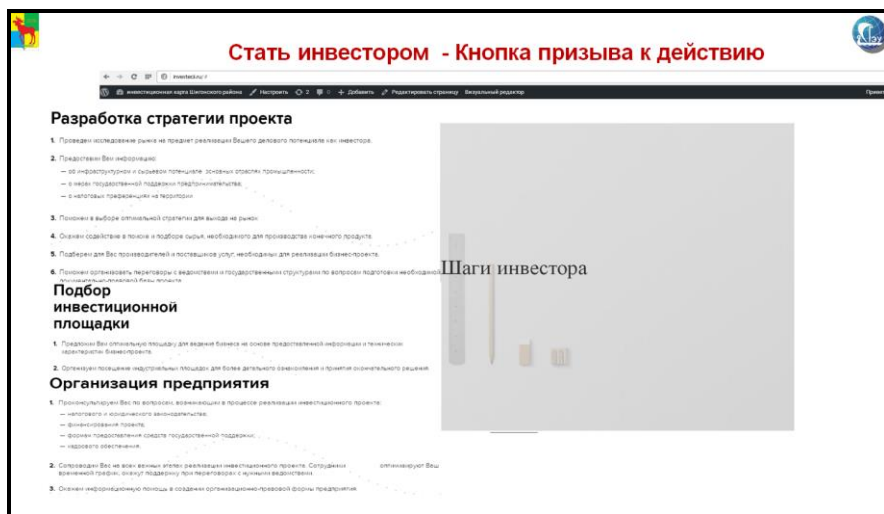


Рис. 2.10. Кнопка к призыву

Одним из критериев эффективности лендинга является показатель "количество посетителей, которые темой не заинтересовались". На форме обратной связи стоит сократить количество вопросов или перефразировать вопросы коротко (рис. 2.11).

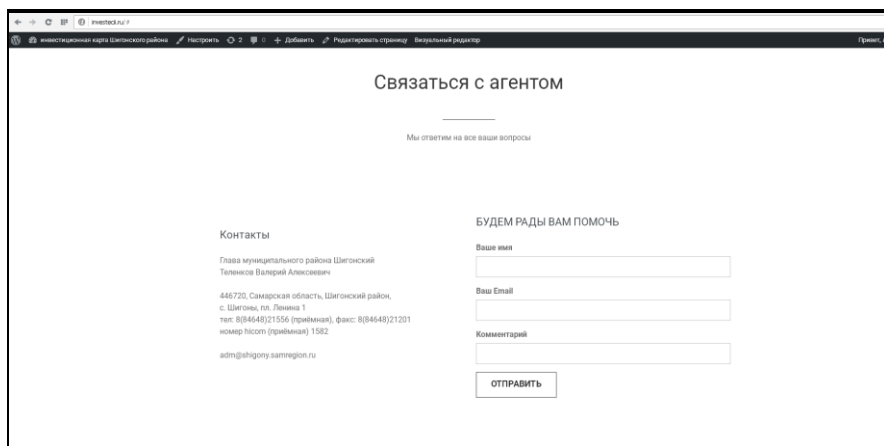


Рис. 2.11. Форма обратной связи

Адаптивный дизайн ворвался в интернет-индустрию моментально. Отсутствие мобильной версии сайта может в корне снизить его посещаемость. Так, с ежедневным ростом количества обладателей мобильных телефонов повышается и необходимость отзывчивого (адаптивного) дизайна.

Адаптивный дизайн - это такой вид веб-дизайна, который подстраивает сайт под разрешение экранов мобильных гаджетов для комфортного просмотра его страниц. Благодаря такой адаптивности исчезает потребность увеличивать страницу на маленьком экране мобильного телефона и передвигать ее пальцем влево-вправо, чтоб прочесть информацию мелким шрифтом.

На WordPress есть отличный пример качественного отзывчивого дизайна - тема Divi 2.0. Просто откройте эту тему на мобильном устройстве. Она сразу же приобретет ширину экрана и компактно отобразит все элементы дизайна в читабельном виде. Тема поведет себя иначе, если вы будете изменять размеры окна браузера на своем компьютере: от ширины экрана зависит местоположение многих элементов, на лету трансформируются CSS-стили сайта. Чтобы было более понятно, тут применяются **breakpoint** на разных этапах уменьшения размера. Это контрольные точки, по мере достижения которых в стилях оформления страницы происходят изменения, которые условно называются **CSS media query** - это заявление в таблице стилей, которое используется для вызова других заявлений и основано на размере экрана того устройства, под которое создается адаптивный

дизайн. Чем меньше его ширина, тем больше сайт оптимизирован под мобильные устройства типа iPhone или Android.

Дизайн меняет свою ширину на экране, потому что изначально разработчики оптимизировали ее под разные типы устройств.

Для портала использован такой же подход, благодаря чему дизайн отображается отзывчиво как на планшете, так и на экране мобильного телефона (рис. 2.12).



Рис. 2.12. Мобильная версия портала

Если тестировать все посещаемые сайты на разных устройствах, то через некоторое время это войдет в привычку. Отдельные сайты абсолютно по-разному выводятся на экран на разных устройствах.

Например, Google использует отличающиеся таблицы стилей в зависимости от ширины экрана устройства. Такой метод требует дополнительного кода CSS и, помимо стандартного CSS, применяется еще и на таких крупных сайтах, как Google, Amazon, Yahoo.

Также в самообразовании в сфере отзывчивого дизайна помогают:

- чтение тематических блогов и руководств;
- прохождение курсов;
- посещение воркшопов;
- пополнение знаний о CSS3 и использовании Media Queries;
- самостоятельное внедрение знаний на практике;
- создание, пусть и простого, сайта с помощью основ HTML и CSS для понимания принципа их работы.

Для создания адаптивной версии сайта на WordPress необходимо использовать **CSS3 Media queries**, о которых сегодня уже упоминалось.

Стандартное заявление media-query выглядит таким образом:

```
1      @media (max-width: 700px) {
2      .container {
3      width: 100%;
4      margin: 0 auto;
5      }
6      }
```

С помощью правила @media для веб-сайта можно создать разные шаблоны под различные виды экранов, версию для печати, мобильного телефона или же планшета. Можно использовать сразу несколько видов и определить любую нужную ширину, применив media query.

В коде есть элемент @media с ограничением (max-width: 700px), и это значит, что если разрешение экрана или окно браузера достигает 700 пикселей или меньше, то применяются параметры, указанные в скобках.

Лучше, конечно, использовать данные правила @media без фанатизма и поменьше.

Для портала будут использованы основы HTML и CSS, чтоб показать на примере, как создаются правила адаптивного дизайна внутри кода. Нужно только применить стили в файле **style.css**, чтобы изменить дизайн сайта.

Код HTML:

```
<!DOCTYPE html>
<head>
  <meta charset="utf-8">
  <title>Elegant Themes Responsive Web
Design - Demo</title>
  <meta name="viewport" con-
tent="width=device-width, initial-scale=1">
  <link rel="stylesheet" type="text/css"
href="style.css">
</head>
0 <body>
  <div class="container">
1 <h1>Responsive Web Design</h1>
  <ul>
2 <li><a href="#">Sample Link
1</a></li>
3 <li><a href="#">Sample Link
2</a></li>
4 <li><a href="#">Sample Link
3</a></li>
5 <li><a href="#">Sample Link
4</a></li>
6 <li><a href="#">Sample Link
5</a></li>
7 </ul>
  </div>
8 </body>
</html>
9
0
```

Выше приведен пример стандартного кода. Метатэг viewport используется для того, чтобы заявить браузеру о выведении кода в соответствии с шириной экрана девайса. Часть кода div содержит ссылки.

Вот как отображается страница в Chrome:

Responsive Web Design

- [Sample Link 1](#)
- [Sample Link 2](#)
- [Sample Link 3](#)
- [Sample Link 4](#)
- [Sample Link 5](#)

Оставить страницу в таком виде нельзя, поэтому используем CSS.

Код CSS:

```
1 <!DOCTYPE html>
2 <head>
3   <meta charset="utf-8">
4   <title>Elegant Themes Responsive Web Design - Demo</title>
5   <meta name="viewport" content="width=device-width, initial-
6   scale=1">
7   <link rel="stylesheet" type="text/css" href="style.css">
8 </head>
9 <body>
10  <div class="container">
11  <h1>Responsive Web Design</h1>
12  <ul>
13    <li><a href="#">Sample Link 1</a></li>
14    <li><a href="#">Sample Link 2</a></li>
15    <li><a href="#">Sample Link 3</a></li>
16    <li><a href="#">Sample Link 4</a></li>
17    <li><a href="#">Sample Link 5</a></li>
18  </ul>
19 </div>
20 </body>
</html>
```

После применения стиля страница выглядит уже иначе (рис. 2.13).



Рис. 2.13. Страница после применения стиля

Как данный дизайн будет выглядеть на экране мобильного устройства? Скорее всего, ссылки не будут отображаться в один ряд на маленьком разрешении экрана.

Необходимо использовать **media query**, что поможет подстроить страницу под изменения в CSS. Для этого нужно добавить нижеуказанный код CSS внизу файла **style.css**.

```
1 @media (max-width: 970px) {  
2  
3 .container {  
4   width: 100%;  
5   margin: 0 auto;  
6   padding:20px 0;  
7 }  
8 .container ul {  
9   padding:10px;  
10 }  
11 .container ul li {  
12   display:block;  
13   padding:10px 0;  
14 }  
15 }
```

Теперь, когда код добавлен, можно изменить размер окна браузера, чтобы увидеть эффект адаптивности (рис 2.14).

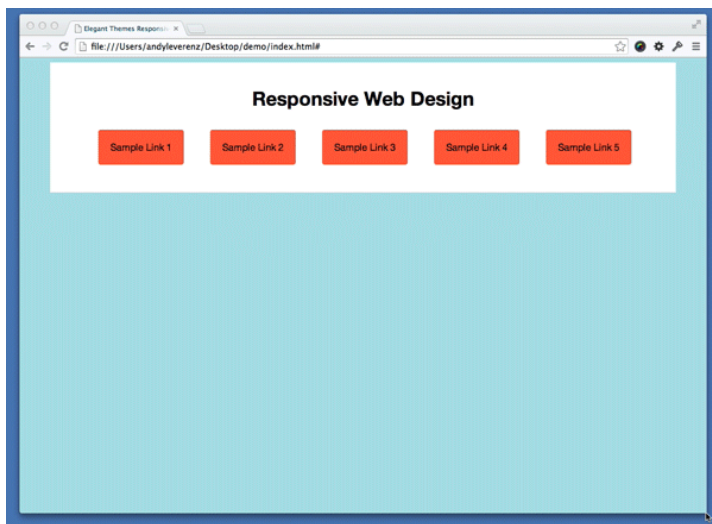


Рис. 2.14. Эффект адаптивности

Можно настроить еще несколько элементов, как только браузер приобретет минимальную ширину. Использование media query поможет улучшить результат. Добавьте эту часть внизу файла **style.css**:

```
1 @media (max-width: 480px) {  
2   .container h1 {  
3     font-size: 22px;  
4   }  
5  
6 }
```

Понемногу дизайн приблизился к желаемому конечному виду (рис. 2.15).

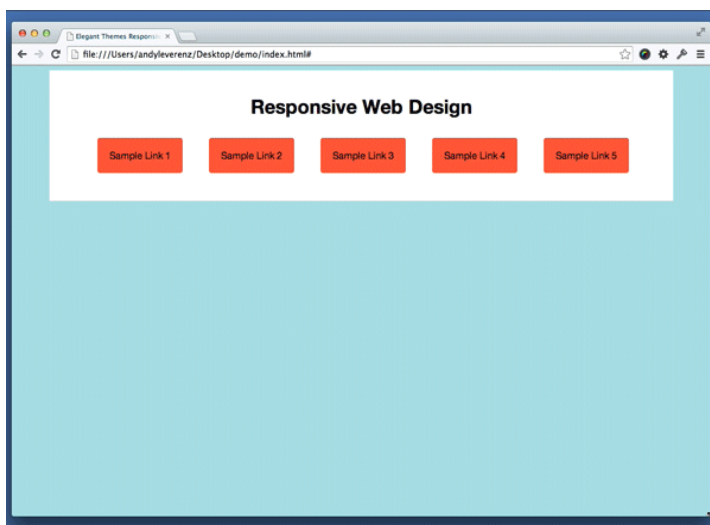


Рис. 2.15. Адаптивный дизайн

Теперь адаптивный дизайн полностью готов. Этот принцип можно использовать для любого сайта независимо от CMS.

ЗАДАНИЕ

Выберите предметную область и на основе CMS-системы разработайте сайт.

К разработке веб-сайта предъявлены следующие требования:

- 1) обязательные мультимедийные составляющие: текст, звук, графика, видео;
- 2) минимальное количество страниц - 7;

3) обеспечение интерактивности для пользователей.

Для выполнения задания поставлены следующие задачи:

1) определить понятие и функции сайта;

2) на основе полученной информации разработать сайт, отвечающий всем критериям мультимедийности.

СПИСОК ЛИТЕРАТУРЫ

1. Автоматизированные информационные системы, базы и банки данных. Вводный курс [Текст] : учеб. пособие. - Москва : Гелиос АРВ, 2011. - 368 с.

2. *Астелс, Д.* Практическое руководство по экстремальному программированию [Текст] : пер. с англ. / Дэвид Астелс, Гранвилл Миллер, Мирослав Новак. - Москва : Вильямс, 2010. - 320 с.

3. *Баженова, И.Ю.* Основы проектирования приложений баз данных [Текст] / И.Ю. Баженова. - Москва : Бином. Лаборатория знаний : Интернет-университет информационных технологий, 2010. - 328 с.

4. *Вигерс, К.* Разработка требований к программному обеспечению [Текст] : пер. с англ. / К. Вигерс. - Москва : Русская Редакция, 2010. - 576 с.

5. *Гвоздева, Т.В.* Проектирование информационных систем [Текст] / Т.В. Гвоздева, Б.А. Баллод. - Москва : Феникс, 2012. - 512 с.

6. ГОСТ Р ИСО 9126-93. Оценка программной продукции. Характеристики качества и руководства по их применению [Электронный ресурс]. - Москва : Изд-во стандартов, 1994. – Режим доступа : <http://docs.cntd.ru/document/1200009076>.

7. *Илюшечкин, В.М.* Основы использования и проектирования баз данных [Текст] / В.М. Илюшечкин. - Москва : Юрайт, 2010. - 224 с.

8. *Крейн, Д.* AJAX в действии [Текст] : учебник / Дейв Крейн, Эрик Паскарелло, Даррен Джеймс. - Москва : Вильямс, 2011. - 490 с.

9. *Кузин, А.В.* Базы данных [Текст] / А.В. Кузин, С.В. Левонисова. - Москва : Академия, 2010. - 320 с.

10. *Кузнецов, С.Д.* Основы баз данных [Текст] / С.Д. Кузнецов. - Москва : Бином. Лаборатория знаний : Интернет-университет информационных технологий, 2011. - 488 с.

11. *Незнанов, А.А.* Программирование и алгоритмизация [Текст] / А.А. Незнанов. - Москва : Академия, 2010. - 304 с.

12. *Пинягина, О.В.* Разработка электронного магазина на PHP и MySQL [Текст] / О.В. Пинягина ; Казан. гос. ун-т. - Казань : 2010. - 108 с.

13. *Пирогов, В.Ю.* Информационные системы и базы данных. Организация и проектирование [Текст] / В.Ю. Пирогов. – Санкт-Петербург : БХВ-Петербург, 2012. - 528 с.

14. Рейтинг digital-агентств [Электронный ресурс]. - Режим доступа: ratings.cmsmagazine.ru/cms_analyt-ics/?sk=tcy&bt=&st=&ctl=&dy=&cpp=50&&so=desc&pn=2 (дата обращения: 18.10.2018).

15. Реляционные базы данных: практические приемы оптимальных решений. - Санкт-Петербург : БХВ-Петербург, 2012. - 400 с.

16. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. ГОСТ 19.701-90 (ИСО 5807-85) [Текст] / Гос. комитет СССР по управлению качеством продукции и стандартам. - Москва, 1992. - 22 с.

17. *Тапскотт, Д.* Викиномика. Как массовое сотрудничество изменяет все [Текст] / Д. Тапскотт, Э. Уильямс. - Москва : BestBusinessBooks, 2009. - 392 с.

18. *Шуровьески, Д.* Мудрость толпы [Текст] / Д. Шуровьески. - Москва : Вильямс, 2007. - 304 с.

19. AJAX и PHP. Разработка динамических веб-приложений [Текст] : учебник / Кристиан Дари [и др.]. - Москва : Символ Плюс, 2011. - 289 с.

20. *Howe, J.* The Rise of Crowdsourcing [Electronic resource] / J. Howe. – Режим доступа: <http://www.wired.com/wired/archive/14.06/crowds.html> (дата обращения: 01.04.2019).

21. *Флэнаган, Д.* JavaScript. Подробное руководство [Текст] : учебник / Дэвид Флэнаган. - Москва : Символ Плюс, 2010. - 240 с.

ПРИЛОЖЕНИЕ

Листинг программных модулей

Dannie.php

```
<?php
session_start();
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd>
<head>
<meta http-equiv='content-type' content='text/html; charset=windows-1251' />

<title>АКУ ЗАО «Завод пластмасс»</title>
<link rel='stylesheet' type='text/css' href='css/index.css' />
<link rel="stylesheet" type="text/css" href="js/jquery-ui-1.7.2.custom.css">
<script type="text/javascript" src="js/jquery-1.3.2.min.js"></script>
<script type="text/javascript" src="js/jquery-ui-1.7.2.custom.min.js"></script>
<script type="text/javascript" src="js/ui.datepicker-uk.js"></script>
<script type="text/javascript" src="js/jquery.validate.min.js"></script>
<script type="text/javascript" src="js/rounded-corners-min.js"></script>
<link rel="stylesheet" type="text/css" href="style.css">
<script type="text/javascript" >
$(document).ready(function(){
    $('#datepicker1').datepicker({
        changeMonth:true,
            changeYear:true,
            yearRange:"1950:2000",
            dateFormat:"yy-mm-dd",
    }
});
$('#datepicker2').datepicker({
    changeMonth:true,
        changeYear:true,
        yearRange:"1950:2010",
        dateFormat:"yy-mm-dd",
    }
});
$('#datepicker').datepicker({
    changeMonth:true,
```

```

        changeYear:true,
        yearRange:"1995:2010",
        dateFormat:"yy-mm-dd",
    }
);
$.validator.addMethod('validName', function (value) {
    var result = true;
    var iChars = " !@#$%^&*()+=-
[]\\';,/{ }|\"':<>?"+йцукенгшщзхъфывапролджэячсмитьбюЙЦУКЕНГШЩЗХЪ
ФЫВАПРОЛДЖЭЯЧСМИТЬБЮ";
    for (var i = 0; i < value.length; i++) {
        if (iChars.indexOf(value.charAt(i)) != -1) {
            return false;
        }
    }
    return result;
}, "");
$.validator.addMethod('validNamer', function (value) {
    var result = true;
    var iChars = " !@#$%^&*()+=-[]\\';,./{}|\"':<>?012789"+"qwertyui-
opasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM"+" ";
    for (var i = 0; i < value.length; i++) {
        if (iChars.indexOf(value.charAt(i)) != -1) {
            return false;
        }
    }
    return result;
}, "");

$('#signup').validate({
    rules: {
        login: {
            minlength:6,
            validName: true,
        },
        pass: {
            minlength:6,
            validName: true,
        },
        conpass: {
            equalTo:'#pass'
        }
    },
});

```

```
messages: {
  login: {minlength: "Не менее шести символов",
  validName: "Только латинский алфавит и цифры",
  },
  pass: {minlength: "Не менее шести символов",
  validName: "Только латинский алфавит и цифры",
  },
  conpass: {equalTo: "Пароли не совпадают"
  }
  },
  });
$(this).corner();
});
</script>
<SCRIPT Language=JavaScript >
//Скрипт очищающий форму от текста при нажатии на нее курсора
function doClear(theText) { if (theText.value == theText.defaultValue) {
theText.value = "" } }
function doDefault(theText) { if (theText.value == "") { theText.value =
theText.defaultValue } }
</script>

</head>

<body>
  <?php include "menu.php"; ?>

  <div id='content'>
    <div id='content1'>
      <!-- вот здесь вся инфа-->
      <p class="pp">
Ввод данных о производственной деятельности цехов<br>
</p>
<?php
require "dbconnect.php";
$res_id1=mysql_query("SELECT idgild, nameG FROM `Gild`")
or die(mysql_error());
$gild=mysql_num_rows($res_id1);
$res_id2=mysql_query("SELECT idprod, nameprod FROM `prod`")
or die(mysql_error());
$prod=mysql_num_rows($res_id2);
?>

<table width="900" border="0" cellspacing="0" cellpadding="0">
```

```

<tr >
<td width="30">&nbsp;&nbsp;&nbsp;</td>
<td width="320" class="date1" >Выберите цех</td>
<td width="30">&nbsp;&nbsp;&nbsp;</td>
    <td width="320" class="date1" >Выберите продукцию </td>
    <td width="30">&nbsp;&nbsp;&nbsp;</td>
    </tr>
        <tr>
            <td width="30">&nbsp;&nbsp;&nbsp;</td>
<td width="320" class="log">
<form method="post" id="signup">
<select name="idGm"><center>
    <?php
    while ($gild=mysql_fetch_row($res_id1))
    {
    echo "<center><option value='$gild[0]'$>$gild[1]</option>\n";
    }
    ?>
    </select>
</td>
<td width="30">&nbsp;&nbsp;&nbsp;</td>
    <td width="320" class="log">
    <select name="idPrm"><center>
    <?php
    while ($prod=mysql_fetch_row($res_id2))
    {
    echo "<option value='$prod[0]'$>$prod[1]</option>\n";
    }
    ?>
    </select>
    </td>
    <td width="30">&nbsp;&nbsp;&nbsp;</td>
</tr>
        <tr>
            <td colspan="5">&nbsp;&nbsp;&nbsp;</td>
        </tr>
<tr >
<td width="30">&nbsp;&nbsp;&nbsp;</td>
<td width="320" class="date1" >Введите количество</td>
<td width="30">&nbsp;&nbsp;&nbsp;</td>
    <td width="320" class="date1" >Введите дату</td>
    <td width="30">&nbsp;&nbsp;&nbsp;</td>
    </tr>

```

```

        <tr>
            <td width="30">&nbsp;</td>
            <td width="320" class="log">
                <input type="text" size="60" value="Не указан" onFocus="doClear(this)"
onBlur="doDefault(this)" name="kolvoM" class="required" title="<br>
Поле обязательно к заполнению"></td>
            <td width="30">&nbsp;</td>
            <td width="320" class="log">
                <input type="text" size="60" value="Не указан" onFocus="do-
Clear(this)" onBlur="doDefault(this)" name="dateM" id="datepicker2" class="re-
quired" title="<br>
Поле обязательно к заполнению">
            </td>
            <td width="30">&nbsp;</td>
        </tr>

        <td colspan="5">&nbsp;</td>
    </tr>

    <tr>
        <td colspan="5" class="pp"><input type="submit" name="post" value="Учесть"
>
    </td>
    </tr>
    </tr>
    <tr>
        <td colspan="7">&nbsp;</td>
    </tr>
    <tr>
        <td colspan="7">

        <?php
require "dbconnect.php";
if(isset($_POST['post']))
{
$query1="insert into make (`idPrm`,`idSm`,`idGm`,`kolvoM`,`dateM`)
values
('".$_POST['idPrm']."','".$_SES-
SION['var']','".$_POST['idGm']."','".$_POST['kolvoM']."','".$_POST['dateM']."')";
if(mysql_query($query1) ) /* если запрос успешно произведен */
{
echo "<p class='pp' width='200'>Запись сохранена <br>
</p>
";
}
}

```

```
else
{
echo "<p class='prav'>Не получилось</p>";
}
}
```

```
?>
</td>
</tr>
</table>
</div>
</div>
<br>
<div id='main' class='noifxpng'>
</div>
<center>
```

```
<br>
```

```
</body>
```

```
</html>
```

Klient.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
'http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd'>
```

```
<html xmlns='http://www.w3.org/1999/xhtml'>
```

```
<head>
```

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

```
<title>АСУ ЗАО «Завод пластмасс»</title>
```

```
<link rel='stylesheet' type='text/css' href='css/index.css' />
```



```

}, "");
$.validator.addMethod('validNamer', function (value) {
    var result = true;
    var iChars = "!@#%&*()+=-[]\\';,./{}|\":<?012789"+"qwertyui-
opasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM"+" ";
    for (var i = 0; i < value.length; i++) {
        if (iChars.indexOf(value.charAt(i)) != -1) {
            return false;
        }
    }
    return result;
}, "");

```

```

    $('#signup').validate({
        rules: {
            login: {
                minlength: 6,
                validName: true,
            },
            pass: {
                minlength: 6,
                validName: true,
            },
            conpass: {
                equalTo: '#pass'
            }
        },
        messages: {
            login: { minlength: "Не менее шести символов",
                validName: "Только латинский алфавит и цифры",
            },
            pass: { minlength: "Не менее шести символов",
                validName: "Только латинский алфавит и цифры",
            },
            conpass: { equalTo: "Пароли не совпадают"
            }
        },
    });
$(this).corner();
});

```

```
</script>
```

```
<SCRIPT Language=JavaScript >
```

```
//Скрипт очищающий форму от текста при нажатии на нее курсора
```

```
function doClear(theText) { if (theText.value == theText.defaultValue) {
theText.value = "" } }
function doDefault(theText) { if (theText.value == "") { theText.value =
theText.defaultValue } }
</script>

</head>

<body>
  <?php include "menu.php"; ?>

  <div id='content'>
    <div id='content1'>
      <!-- вот здесь вся инфа-->
    <p class="pp">
      Регистрация клиентов <br>
    </p>
    <?php
    /*      require "dbconnect.php";
    $res_id2=mysql_query("SELECT * FROM `country`")
    or die(mysql_error());
    $strana=mysql_num_rows($res_id2);
    $res_id4=mysql_query("SELECT * FROM `sex`")
    or die(mysql_error());
    $sex=mysql_num_rows($res_id4);*/
    ?>

    <table width="900" border="0" cellspacing="0" cellpadding="0">
      <tr >
        <td width="30">&nbsp;</td>
        <td width="320" class="date1" >Полное наименование </td>
        <td width="30">&nbsp;</td>
        <td width="320" class="date1" >Краткое наименование </td>
        <td width="30">&nbsp;</td>
      </tr>
      <tr>
        <td width="30">&nbsp;</td>
        <td width="320" class="log">
          <form method="post" id="signup">
            <input type="text" size="60" value="Не указан" onFocus="doClear(this)"
            onBlur="doDefault(this)" name="nameP" class="required" title="<br>
            Поле обязательно к заполнению"></td>
            <td width="30">&nbsp;</td>
            <td width="320" class="log">
```

```
<input type="text" size="60" value="Не указан" onFocus="doClear(this)" onBlur="doDefault(this)" name="knameP" class="required" title="<br>
```

```
Поле обязательно к заполнению">
```

```
</td>
```

```
<td width="30">&nbsp;&nbsp;&nbsp;</td>
```

```
</tr>
```

```
<tr>
```

```
<td colspan="5">&nbsp;&nbsp;&nbsp;</td>
```

```
</tr>
```

```
<tr >
```

```
<td width="30">&nbsp;&nbsp;&nbsp;</td>
```

```
<td width="320" class="date1" >Фактический адрес </td>
```

```
<td width="30">&nbsp;&nbsp;&nbsp;</td>
```

```
<td width="320" class="date1" >Юридический адрес </td>
```

```
<td width="30">&nbsp;&nbsp;&nbsp;</td>
```

```
</tr>
```

```
<tr>
```

```
<td width="30">&nbsp;&nbsp;&nbsp;</td>
```

```
<td width="320" class="log">
```

```
<input type="text" size="60" value="Не указан" onFocus="doClear(this)" onBlur="doDefault(this)" name="adressP" class="required" title="<br>
```

```
Поле обязательно к заполнению"></td>
```

```
<td width="30">&nbsp;&nbsp;&nbsp;</td>
```

```
<td width="320" class="log">
```

```
<input type="text" size="60" value="Не указан" onFocus="doClear(this)" onBlur="doDefault(this)" name="uradrP" class="required" title="<br>
```

```
Поле обязательно к заполнению">
```

```
</td>
```

```
<td width="30">&nbsp;&nbsp;&nbsp;</td>
```

```
</tr>
```

```
<tr>
```

```
<td colspan="5">&nbsp;&nbsp;&nbsp;</td>
```

```
</tr>
```

```
<tr >
```

```
<td width="30">&nbsp;&nbsp;&nbsp;</td>
```

```
<td width="320" class="date1" >Банковские реквизиты </td>
```

```
<td width="30">&nbsp;&nbsp;&nbsp;</td>
```

```
<td width="320" class="date1" >Контактное лицо </td>
```

```
<td width="30">&nbsp;&nbsp;&nbsp;</td>
```

```

        </tr>
        <tr>
            <td width="30">&nbsp;&nbsp;&nbsp;</td>
            <td width="320" class="log">
                <input type="text" size="60" value="Не указан" onFocus="doClear(this)"
                onBlur="doDefault(this)" name="banrekP" class="required" title="<br>
                Поле обязательно к заполнению"></td>
            <td width="30">&nbsp;&nbsp;&nbsp;</td>
            <td width="320" class="log">
                <input type="text" size="60" value="Не указан" onFocus="do-
                Clear(this)" onBlur="doDefault(this)" name="kontlizoP" class="required" ti-
                tle="<br>
                Поле обязательно к заполнению">
            </td>
            <td width="30">&nbsp;&nbsp;&nbsp;</td>
        </tr>
        <tr>
            <td colspan="5">&nbsp;&nbsp;&nbsp;</td>
        </tr>
        <tr>
            <td width="30">&nbsp;&nbsp;&nbsp;</td>
            <td width="320" class="date1" >Телефон </td>
            <td width="30">&nbsp;&nbsp;&nbsp;</td>
            <td width="320" class="date1" >Электронная почта </td>
            <td width="30">&nbsp;&nbsp;&nbsp;</td>
        </tr>
        <tr>
            <td width="30">&nbsp;&nbsp;&nbsp;</td>
            <td width="320" class="log">
                <input type="text" size="60" value="Не указан" onFocus="doClear(this)"
                onBlur="doDefault(this)" name="tlfP" class="required" title="<br>
                Поле обязательно к заполнению"></td>
            <td width="30">&nbsp;&nbsp;&nbsp;</td>
            <td width="320" class="log">
                <input type="text" size="60" value="Не указан" onFocus="do-
                Clear(this)" onBlur="doDefault(this)" name="emailP" class="required" title="<br>
                Поле обязательно к заполнению">
            </td>
            <td width="30">&nbsp;&nbsp;&nbsp;</td>
        </tr>
        <tr>

```



```

require "dbconnect.php";
$sod1=mysql_query("SELECT count( idklient )
FROM klient
WHERE udalkl=0");
while($sod_mas1=mysql_fetch_row($sod1))
{
$kolvo=$sod_mas1[0];
}
if ($kolvo>0)
{
echo "<table border='0' align='center' width='900'><tr><td class='table1'> Всего
зарегистрировано клиентов &nbsp;&nbsp;&nbsp;<b>$kolvo</b></td></tr> </table>\n";
}
?>
<?php
require "dbconnect.php";
$res=mysql_query("SELECT idklient, `namekl`, `knamekl`, `adresskl`, `ura-
drkl`, `banrekkl`, `kontlizokl`, `tlfkl`, `emailkl` FROM `Klient` where udalkl=0");
while($res_id=mysql_fetch_row($res))
{
echo " <center> <table> <tr>
<td width='45' class='tz'> $res_id[0] </td>
<td width='90' class='tz'> $res_id[1] </td>
<td width='90' class='tz'> $res_id[2] </td>
<td width='90' class='tz'> $res_id[3] </td>
<td width='90' class='tz'> $res_id[4] </td>
<td width='90' class='tz'> $res_id[5] </td>
<td width='90' class='tz'> $res_id[6] </td>
<td width='90' class='tz'> $res_id[7] </td>
<td width='90' class='tz'> $res_id[8] </td>
<td width='45' class='table3'><form method='post' name='udal' ac-
tion='udal1.php'>
<INPUT TYPE='hidden' NAME='udal' VALUE='$res_id[0]'>
<input type='image' src='image/udal.png' TITLE='Удалить' WIDTH='40'
HEIGHT='40' >
</form>
</td>
</tr> </table>\n";
}
?>
</div>
</div>
<br>

```

```
<div id='main' class='noifxpng'>
```

```
    </div>
```

```
        <center>
```

```
<br>
```

```
</body>
```

```
</html>
```

Otpost.php

```
<?php
```

```
session_start();
```

```
?>
```

```
<html>
```

```
<title>Поиск</title>
```

```
<head >
```

```
<body BACKGROUND="image/bg.png">
```

```
<link rel="stylesheet" type="text/css" href="js/jquery-ui-1.7.2.custom.css">
```

```
<script type="text/javascript" src="js/jquery-1.3.2.min.js"></script>
```

```
<script type="text/javascript" src="js/jquery-ui-1.7.2.custom.min.js"></script>
```

```
<script type="text/javascript" src="js/ui.datepicker-uk.js"></script>
```

```
<script type="text/javascript" src="js/jquery.validate.min.js"></script>
```

```
<script type="text/javascript" src="js/rounded-corners-min.js"></script>
```

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

```
<script type="text/javascript" >
```

```
$(document).ready(function(){
```

```
    $('#datepicker1').datepicker({
```

```
        changeMonth:true,
```

```
        changeYear:true,
```

```
        yearRange:"1950:2000",
```

```
        dateFormat:"yy-mm-dd",
```

```
    }
```

```
});
```

```
$('#datepicker2').datepicker({
```

```
    changeMonth:true,
```

```
    changeYear:true,
```

```
    yearRange:"1950:2010",
```

```
    dateFormat:"yy-mm-dd",
```

```

    }
  );
  $('#datepicker').datepicker({
    changeMonth:true,
      changeYear:true,
        yearRange:"1995:2010",
          dateFormat:"yy-mm-dd",
    }
  );
  $.validator.addMethod('validName', function (value) {
    var result = true;
    var iChars = " !@#%&*()+=-[]\\';/{}|\":<>?"+ "йцукенгшщзхъфывапролджэячсмитьбюЙЦУКЕНГШЩЗХЪФЫВАПРОЛДЖЭЯЧСМИТЬБЮ";
    for (var i = 0; i < value.length; i++) {
      if (iChars.indexOf(value.charAt(i)) != -1) {
        return false;
      }
    }
    return result;
  }, "");
  $.validator.addMethod('validNamer', function (value) {
    var result = true;
    var iChars = " !@#%&*()+=-[]\\';/{}|\":<>?012789"+ "qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM"+" ";
    for (var i = 0; i < value.length; i++) {
      if (iChars.indexOf(value.charAt(i)) != -1) {
        return false;
      }
    }
    return result;
  }, "");

  $('#signup').validate({
    rules: {
      login: {
        minlength:6,
        validName: true,
      },
      pass: {
        minlength:6,
        validName: true,
      },
    }
  });

```



```

conpass: {
equalTo: '#pass'
}
},
messages: {
login: {minlength: "Не менее шести символов",
validName: "Только латинский алфавит и цифры",
},
pass: {minlength: "Не менее шести символов",
validName: "Только латинский алфавит и цифры",
},
conpass: {equalTo: "Пароли не совпадают"
}
},
});
$(this).corner();
});
</script>
<SCRIPT Language=JavaScript >
//Скрипт очищающий форму от текста при нажатии на нее курсора
function doClear(theText) { if (theText.value == theText.defaultValue) {
theText.value = "" } }
function doDefault(theText) { if (theText.value == "") { theText.value =
theText.defaultValue } }
</script>
<link rel="stylesheet" type="text/css" href="style.css">
<br><br>
<br>
<center>
<!-- начало фоновой таблицы -->
<?php
include "menu.php";
?>
<div id='content'>
<div id='content1'>
<p class="pp">
План поставок<br>
</p>
<!-- начало основной таблицы -->

<table align="center" cellspacing="2" cellpadding="2" border="0"
bgcolor="ffffff" width="910">
<tr><td colspan="3"><p class='tr'>Для ввода даты выберите дату, для про-
смотрa поставок за месяц введите номер месяца в виде -03-</p> </td>

```



```
<td><center>$sod_mas[3]</a></td>
<td><center> $sod_mas[4]</td></td>
<td><center>$sod_mas[5]</td><td><center>$sod_mas[6]</td><td><cen-
ter>$sod_mas[7]</td></tr>\n";
}
echo "<p class='pp'> Итого на сумму $a рублей </p>";
}
else
{
echo "Не получилось";
}
}
?>

<br>

<div>&nbsp;

</div>
</div>

<!-- конец фоновой таблицы -->

</body>

</html>

Product.php

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
'http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd'>

<html xmlns='http://www.w3.org/1999/xhtml'>

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

<title>АСУ ЗАО «Завод пластмасс»</title>
<link rel='stylesheet' type='text/css' href='css/index.css' />
```



```

}, "");
$.validator.addMethod('validNamer', function (value) {
    var result = true;
    var iChars = " !@#$%^&*()+=-[]\\|';,./{}|\"':<>?012789"+"qwertyui-
opasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM"+" ";
    for (var i = 0; i < value.length; i++) {
        if (iChars.indexOf(value.charAt(i)) != -1) {
            return false;
        }
    }
    return result;
}, "");

```

```

    $('#signup').validate({
        rules: {
            login: {
                minlength: 6,
                validName: true,
            },
            pass: {
                minlength: 6,
                validName: true,
            },
            conpass: {
                equalTo: '#pass'
            }
        },
        messages: {
            login: {minlength: "Не менее шести символов",
                validName: "Только латинский алфавит и цифры",
            },
            pass: {minlength: "Не менее шести символов",
                validName: "Только латинский алфавит и цифры",
            },
            conpass: {equalTo: "Пароли не совпадают"}
        }
    });
$(this).corner();
});

```

</script>

<SCRIPT Language=JavaScript >

//Скрипт очищающий форму от текста при нажатии на нее курсора

```

function doClear(theText) { if (theText.value == theText.defaultValue) {
theText.value = "" } }
function doDefault(theText) { if (theText.value == "") { theText.value =
theText.defaultValue } }
</script>

</head>

<body>
  <?php include "menu.php"; ?>

  <div id='content'>
    <div id='content1'>
      <!-- вот здесь вся инфа-->
      <p class="pp">
        Виды продукции <br>
      </p>
      <?php
      /*      require "dbconnect.php";
      $res_id2=mysql_query("SELECT * FROM `country`")
      or die(mysql_error());
      $strana=mysql_num_rows($res_id2);
      $res_id4=mysql_query("SELECT * FROM `sex`")
      or die(mysql_error());
      $sex=mysql_num_rows($res_id4);*/
      ?>

      <table width="900" border="0" cellspacing="0" cellpadding="0">
        <tr >
          <td width="30">&nbsp;</td>
          <td width="320" class="date1" >Полное наименование </td>
          <td width="30">&nbsp;</td>
          <td width="320" class="date1" >Артикул </td>
          <td width="30">&nbsp;</td>
        </tr>
        <tr>
          <td width="30">&nbsp;</td>
          <td width="320" class="log">
            <form method="post" id="signup">
              <input type="text" size="60" value="Не указан" onFocus="doClear(this)"
onBlur="doDefault(this)" name="nameprod" class="required" title="<br>
Поле обязательно к заполнению"></td>
            <td width="30">&nbsp;</td>
            <td width="320" class="log">

```

```
<input type="text" size="60" value="Не указан" onFocus="do-
Clear(this)" onBlur="doDefault(this)" name="art" class="required" title="<br>
Поле обязательно к заполнению">
</td>
<td width="30">&nbsp;</td>
</tr>

<tr>
<td colspan="5">&nbsp;</td>
</tr>
<tr >
<td width="30">&nbsp;</td>
<td width="320" class="date1" >Единица измерения</td>
<td width="30">&nbsp;</td>
<td width="320" class="date1" >Себестоимость</td>
<td width="30">&nbsp;</td>
</tr>
<tr>
<td width="30">&nbsp;</td>
<td width="320" class="log">
<input type="text" size="60" value="Не указан" onFocus="doClear(this)"
onBlur="doDefault(this)" name="edizmpr" class="required" title="<br>
Поле обязательно к заполнению"></td>
<td width="30">&nbsp;</td>
<td width="320" class="log">
<input type="text" size="60" value="Не указан" onFocus="do-
Clear(this)" onBlur="doDefault(this)" name="selfst" class="required" title="<br>
Поле обязательно к заполнению">
</td>
<td width="30">&nbsp;</td>
</tr>

<td colspan="5">&nbsp;</td>
</tr>
<tr >
<td width="30">&nbsp;</td>
<td width="320" class="date1" >Оптовая цена</td>
<td width="30">&nbsp;</td>
<td width="320" class="date1" >Розничная цена</td>
<td width="30">&nbsp;</td>
</tr>
<tr>
```



```

        <td width="30">&nbsp;&nbsp;&nbsp;</td>
<td width="320" class="log">
<input type="text" size="60" value="Не указан" onFocus="doClear(this)"
onBlur="doDefault(this)" name="optst" class="required" title="<br>
Поле обязательно к заполнению"></td>
    <td width="30">&nbsp;&nbsp;&nbsp;</td>
        <td width="320" class="log">
            <input type="text" size="60" value="Не указан" onFocus="do-
Clear(this)" onBlur="doDefault(this)" name="rozst" class="required" title="<br>
Поле обязательно к заполнению">
        </td>
        <td width="30">&nbsp;&nbsp;&nbsp;</td>
</tr>

        <tr>
<td colspan="5">&nbsp;&nbsp;&nbsp;</td>
</tr>

        <tr >
<td width="30">&nbsp;&nbsp;&nbsp;</td>
<td class="date1" colspan="3">Описание </td>
        <td width="30">&nbsp;&nbsp;&nbsp;</td>
        </tr>
        <tr>
            <td width="30" >&nbsp;&nbsp;&nbsp;</td>
<td class="log" colspan="3">
<textarea ROWS=3 COLS=140 name="primP" class="required" title="<p
class='log'>Укажите наименование</p>"> </textarea>
        </td>
        <td width="30">&nbsp;&nbsp;&nbsp;</td>
</tr>

        <tr>
<td colspan="5">&nbsp;&nbsp;&nbsp;</td>
</tr>

        <tr>
<td colspan="5" class="pp"><input type="submit" name="post"
value="Зарегистрировать" >
        </form></td>
</tr>

        <tr>
<td colspan="7">&nbsp;&nbsp;&nbsp;</td>

```

```

</tr>
<tr>
<td colspan="7">

        <?php
require "dbconnect.php";
if(isset($_POST['post']))
{
$query1="insert            into            prod            ( nameprod`,`art`,`ediz-
mpr`,`selfst`,`optst`,`rozst`,`primP` )
values            (".$_POST['nameprod'].",".$_POST['art'].",".$_POST['ediz-
mpr'].",".$_POST['self-
st'].",".$_POST['optst'].",".$_POST['rozst'].",".$_POST['primP'].")";
if(mysql_query($query1) ) /* если запрос успешно произведен */
{
echo "<p class='pp' width='200'>Регистрация успешно завершена. <br>
</p>
";
}
else
{
echo "<p class='prav'>Не получилось</p>
";
}
}

?>
</td>
</tr>
</table>
<?php
require "dbconnect.php";
$sod1=mysql_query("SELECT count( idprod )
FROM prod WHERE udalPr=0");
while($sod_mas1=mysql_fetch_row($sod1))
{
$kolvo=$sod_mas1[0];
}
if ($kolvo>0)
{
echo "<table border='0' align='center' width='900'><tr><td class='table1'> Всего
видов продукции &nbsp;&nbsp;&nbsp;<b>$kolvo</b></td></tr> </table>\n";
}
?>

```

```
<?php
require "dbconnect.php";
$res=mysql_query("SELECT `nameprod`,`art`,`edizmpr`,`selfst`,`optst`,`rozst`,
idprod FROM `prod` where udalPr=0");
while($res_id=mysql_fetch_row($res))
{
echo " <center> <table> <tr>
<td width='400' class='tz'> $res_id[0] </td>
<td width='100' class='tz'> $res_id[1] </td>
<td width='100' class='tz'> $res_id[2] </td>
<td width='100' class='tz'> $res_id[3] </td>
<td width='100' class='tz'> $res_id[4] </td>
<td width='100' class='tz'> $res_id[5] </td>
<td width='50' class='table3'><form method='post' name='udal' ac-
tion='udal3.php'>
<INPUT TYPE='hidden' NAME='udal' VALUE='$res_id[6]'>
<input type='image' src='image/udal.png' TITLE='Удалить' WIDTH='40'
HEIGHT='40' >
</form>
</td>
</tr> </table>\n";
}

?>

</div>
</div>

<br>

<div id='main' class='noifixpng'>

</div>

<center>

<br>
</body>
</html>
```

Zajav.php

```
<?php
session_start();
?>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd>
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
```

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

```
<title>АКУ ЗАО «Завод пластмасс»</title>
```

```
<link rel='stylesheet' type='text/css' href='css/index.css' />
```

```
    <link rel="stylesheet" type="text/css" href="js/jquery-ui-1.7.2.cus-
tom.css">
```

```
    <script type="text/javascript" src="js/jquery-1.3.2.min.js"></script>
```

```
    <script type="text/javascript" src="js/jquery-ui-1.7.2.cus-
tom.min.js"></script>
```

```
    <script type="text/javascript" src="js/ui.datepicker-
uk.js"></script>
```

```
    <script type="text/javascript" src="js/jquery.validate.min.js"></script>
```

```
    <script type="text/javascript" src="js/rounded-corners-min.js"></script>
```

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

```
    <script type="text/javascript" >
```

```
    $(document).ready(function(){
```

```
        $('#datepicker1').datepicker({
```

```
        changeMonth:true,
```

```
        changeYear:true,
```

```
        yearRange:"1950:2000",
```

```
        dateFormat:"yy-mm-dd",
```

```
    };
```

```
    );
```

```
    $('#datepicker2').datepicker({
```

```
    changeMonth:true,
```

```
    changeYear:true,
```

```
    yearRange:"1950:2010",
```

```
    dateFormat:"yy-mm-dd",
```

```
    };
```

```
    );
```

```
    $('#datepicker').datepicker({
```

```
    changeMonth:true,
```

```
    changeYear:true,
```

```
    yearRange:"1995:2010",
```

```
    dateFormat:"yy-mm-dd",
```

```
    }
```

```

    );
    $.validator.addMethod('validName', function (value) {
    var result = true;
    var iChars = "!@#%&*()+=-[]\\';/{}|\"<?\"+\"йцукенгшщзхъфывапролджэячсмитьбюЙЦУКЕНГШЩЗХЪФЫВАПРОЛДЖЭЯЧСМИТЬБЮ";
    for (var i = 0; i < value.length; i++) {
    if (iChars.indexOf(value.charAt(i)) != -1) {
    return false;
    }
    }
    return result;
    }, "");
    $.validator.addMethod('validNamer', function (value) {
    var result = true;
    var iChars = "!@#%&*()+=-[]\\';,./{}|\"<?012789\"+\"qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVCBVM\"+\" ";
    for (var i = 0; i < value.length; i++) {
    if (iChars.indexOf(value.charAt(i)) != -1) {
    return false;
    }
    }
    return result;
    }, "");

```

```

$ ('#signup').validate({
  rules: {
    login: {
      minlength: 6,
      validName: true,
    },
    pass: {
      minlength: 6,
      validName: true,
    },
    compass: {
      equalTo: '#pass'
    }
  },
  messages: {
    login: { minlength: "Не менее шести символов",
    validName: "Только латинский алфавит и цифры",
  },

```

```
pass: { minlength: "Не менее шести символов",
validName: "Только латинский алфавит и цифры",
},
compass: { equalTo: "Пароли не совпадают"
}
},
});
$(this).corner();
});
</script>
<SCRIPT Language=JavaScript >
//Скрипт очищающий форму от текста при нажатии на нее курсора
function doClear(theText) { if (theText.value == theText.defaultValue) {
theText.value = "" } }
function doDefault(theText) { if (theText.value == "") { theText.value =
theText.defaultValue } }
</script>

</head>

<body>
<?php include "menu.php"; ?>

<div id='content'>
    <div id='content1'>
        <!-- вот здесь вся инфа-->
    <p class="pp">
Регистрация заявок <br>
</p>
<?php
require "dbconnect.php";
$res_id1=mysql_query("SELECT idklient, namekl FROM `klient`")
or die(mysql_error());
$klient=mysql_num_rows($res_id1);
$res_id2=mysql_query("SELECT idprod, nameprod FROM `prod`")
or die(mysql_error());
$prod=mysql_num_rows($res_id2);
?>

<table width="900" border="0" cellspacing="0" cellpadding="0">
<tr >
<td width="30">&nbsp;</td>
<td width="320" class="date1" >Выберите клиента</td>
<td width="30">&nbsp;</td>

```



```

<input type="text" size="60" value="Не указан" onFocus="doClear(this)"
onBlur="doDefault(this)" name="kolvoprodz" class="required" title="<br>
Поле обязательно к заполнению"></td>
<td width="30">&nbsp;</td>
<td width="320" class="log">
<input type="text" size="60" value="Не указан" onFocus="do-
Clear(this)" onBlur="doDefault(this)" name="datez" id="datepicker2" class="re-
quired" title="<br>
Поле обязательно к заполнению">
</td>
<td width="30">&nbsp;</td>
</tr>

<td colspan="5">&nbsp;</td>
</tr>

<tr>
<td colspan="5" class="pp"><input type="submit" name="post"
value="Зарегистрировать" >
</form></td>
</tr>

<tr>
<td colspan="7">&nbsp;</td>
</tr>
<tr>
<td colspan="7">

<?php
require "dbconnect.php";
if(isset($_POST['post']))
{
$query1="insert into zajav ( `idklz`, `idprodz`,`kolvoprodz`,`datez`,`idsotrZ`,`sta-
tus` )
values ( '$_POST['idklz'].','.$_POST['idprodz'].','.$_POST['kol-
voprodz'].','.$_POST['datez'].','.{$_SESSION['var']}',1)";
if(mysql_query($query1) ) /* если запрос успешно произведен */
{
echo "<p class='pp' width='200'>Заявка принята <br>
</p>
";
}
else
{

```



```
echo "<p class='prav'>Не получилось</p>";
}
}

?>
</td>
</tr>
</table>
</div>
</div>

<br>

<div id='main' class='noifxpng'>

</div>

<center>

<br>
</body>
</html>
```

Zecha.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
'http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd'>
```

```
<html xmlns='http://www.w3.org/1999/xhtml'>
```

```
<head>
```

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

```
<title>АКУ ЗАО «Завод пластмасс»</title>
```

```
<link rel='stylesheet' type='text/css' href='css/index.css' />
```

```
<link rel="stylesheet" type="text/css" href="js/jquery-ui-1.7.2.custom.css">
```

```
<script type="text/javascript" src="js/jquery-1.3.2.min.js"></script>
```

```
<script type="text/javascript" src="js/jquery-ui-1.7.2.custom.min.js"></script>
```

```
<script type="text/javascript" src="js/ui.datepicker-uk.js"></script>
```

```
<script type="text/javascript" src="js/jquery.validate.min.js"></script>
```

```
<script type="text/javascript" src="js/rounded-corners-min.js"></script>
```

```

<link rel="stylesheet" type="text/css" href="style.css">
<script type="text/javascript" >
$(document).ready(function(){
    $('#datepicker1').datepicker({
        changeMonth:true,
            changeYear:true,
                yearRange:"1950:2000",
                    dateFormat:"yy-mm-dd",
        }
    );
    $('#datepicker2').datepicker({
        changeMonth:true,
            changeYear:true,
                yearRange:"1950:2010",
                    dateFormat:"yy-mm-dd",
        }
    );
    $('#datepicker').datepicker({
        changeMonth:true,
            changeYear:true,
                yearRange:"1995:2010",
                    dateFormat:"yy-mm-dd",
        }
    );
    $.validator.addMethod('validName', function (value) {
var result = true;
var iChars = " !@#%&*()+=-[]\|';,/{}|\":<?\"+'йцукенгшщзхъфывапролджэячсмитьбюЙЦУКЕНГШЩЗХЪФЫВАПРОЛДЖЭЯЧСМИТЬБЮ";
for (var i = 0; i < value.length; i++) {
    if (iChars.indexOf(value.charAt(i)) != -1) {
        return false;
    }
}
return result;
}, "");
$.validator.addMethod('validNamer', function (value) {
var result = true;
var iChars = " !@#%&*()+=-[]\|';,/{}|\":<?012789\"+\"qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM\"+\" ";
for (var i = 0; i < value.length; i++) {
    if (iChars.indexOf(value.charAt(i)) != -1) {
        return false;
    }
}
return result;
}, "");

```

```

    }
  }
  return result;
}, "");

    $('#signup').validate({
      rules: {
        login: {
          minlength:6,
          validName: true,
        },
        pass: {
          minlength:6,
          validName: true,
        },
        conpass: {
          equalTo: '#pass'
        }
      },
      messages: {
        login: {minlength:"Не менее шести символов",
          validName: "Только латинский алфавит и цифры",
        },
        pass: {minlength:"Не менее шести символов",
          validName: "Только латинский алфавит и цифры",
        },
        conpass: {equalTo:"Пароли не совпадают"}
      },
    });

    $(this).corner();
  });
</script>
<SCRIPT Language=JavaScript >
//Скрипт очищающий форму от текста при нажатии на нее курсора
function doClear(theText) { if (theText.value == theText.defaultValue) {
theText.value = "" } }
function doDefault(theText) { if (theText.value == "") { theText.value =
theText.defaultValue } }
</script>

</head>

<body>

```

```

<?php include "menu.php"; ?>

<div id='content'>
    <div id='content1'>
        <!-- вот здесь вся инфа-->
    </div>
<p class="pp">
Цеха<br>
</p>
<?php
/*      require "dbconnect.php";
$res_id2=mysql_query("SELECT * FROM `country`")
or die(mysql_error());
$strana=mysql_num_rows($res_id2);
$res_id4=mysql_query("SELECT * FROM `sex`")
or die(mysql_error());
$sex=mysql_num_rows($res_id4);*/
?>

<table width="900" border="0" cellspacing="0" cellpadding="0">
<tr >
<td width="30">&nbsp;</td>
<td width="320" class="date1" >Полное наименование </td>
<td width="30">&nbsp;</td>
    <td width="320" class="date1" >Краткое наименование </td>
    <td width="30">&nbsp;</td>
</tr>
    <tr>
        <td width="30">&nbsp;</td>
<td width="320" class="log">
<form method="post" id="signup">
<input type="text" size="60" value="Не указан" onFocus="doClear(this)"
onBlur="doDefault(this)" name="nameG" class="required" title="<br>
Поле обязательно к заполнению"></td>
    <td width="30">&nbsp;</td>
        <td width="320" class="log">
            <input type="text" size="60" value="Не указан" onFocus="do-
Clear(this)" onBlur="doDefault(this)" name="krnameG" class="required" ti-
tle="<br>
Поле обязательно к заполнению">
        </td>
    <td width="30">&nbsp;</td>
</tr>
</tr>
</tr>

```


Учебное издание

Колотилина Мария Александровна

ИНТЕРНЕТ-ПРОГРАММИРОВАНИЕ

Учебное пособие

Руководитель издательской группы О.В. Егорова
Редактор Г.И. Конева
Корректор Л.И. Трофимова
Компьютерная верстка А.Н. Славкиной, А.Ю. Девяткиной

Подписано к изданию 20.03.2019. Печ. л. 8,44
ФГБОУ ВО "Самарский государственный экономический университет".
443090, Самара, ул. Советской Армии, 141.