Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение  
высшего профессионального образования

«Оренбургский государственный педагогический университет»

Физико-математический факультет

Кафедра информатики и методики преподавания информатики

**Махалова Ольга Александровна**

**Курсовая работа**

**по дисциплине «Администрирование информационных систем»**

**РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ ПО ПОДДЕРЖКЕ ДЕЯТЕЛЬНОСТИ АГЕНТСТВА НЕДВИЖИМОСТИ «вАШ ДОМ»**

Направление: 010500.62 Математическое обеспечение и администрирование информационных систем

Профиль: общий

Форма обучения: дневная

|  |  |
| --- | --- |
| Научный руководитель: | |
| Милохин Д. Б., старший преподаватель кафедры ИМПИ | |
|  |  |
| (оценка) | (подпись) |
| «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2014 г. | |

Оренбург 2014

**Оглавление**

[1. Введение 3](#_Toc391252752)

[2. Проектирование базы данных 4](#_Toc391252753)

[2.1 Определение сущностей и установление связей между ними 4](#_Toc391252754)

[2.2 Нормализация отношений 5](#_Toc391252755)

[2.3 Выбор инструментов и результат моделирования 7](#_Toc391252756)

[3. Программная реализация 11](#_Toc391252757)

[3.1 Язык программирования Perl, технология CGI, Apache HTTP-сервер, фронт-энд фреймворк Twitter Bootstrap 11](#_Toc391252758)

[3.2 Разработка приложения 13](#_Toc391252759)

[3.3 Интерфейс приложения 16](#_Toc391252760)

[4. Тестирование и заключение 23](#_Toc391252761)

[4.1 Тестирование конечного веб-приложения 23](#_Toc391252762)

[4.2 Вывод 24](#_Toc391252763)

[**Список литературы** 25](#_Toc391252764)

1. Введение

В данной курсовой работе описан процесс разработки базы данных для информационной поддержки деятельности агентства недвижимости «Ваш дом» с целью удобного слежения за поступлением новых предложений от собственников жилья, быстрой навигации среди этих предложений и обработки запросов клиентов. Информация о поступивших предложениях собственников и запросах клиентов заносится в базу данных. При получении новых предложений недвижимости каждому предложению присваивается свой индивидуальный код, в базе данных регистрируется следующая информация: код жилья, тип жилья, адрес, площадь, описание, цена и т.д.

**Целью** данной курсовой работы является разработка базы данных и клиентского веб-приложения для удобной работы с данными из базы данных.

**Объект** — разработка и администрирование информационных систем.

**Предмет** — кадровый учёт в конкретной организации.

Среди задач, которые предстоит выполнить, можно выделить следующие:

* исследовать предметную область;
* разработать модель данных;
* создать клиентское веб-приложение, которое будет отвечать требованиям гипотетической организации, для которой разрабатывалась база данных;
* Развернуть базу данных и веб-приложение на ОС Linux и иметь возможность работать с приложением через любые браузеры.

Курсовой проект отражает этапы создания базы данных на сервере СУБД MySQL и разработку клиентского приложения-оболочки для работы с созданной базой данных, которое будет использоваться администраторами из числа работников, а также, настройку сервера Apache2 для данного веб-приложения и базы данных.

2. Проектирование базы данных

2.1 Определение сущностей и установление связей между ними

Есть два типа записей: информация о жилье, предлагаемом для сдачи в аренду или продажи, и заявка о съёме или покупке жилья, удовлетворяющего некоторым условиям. Соответственно, есть те, кто их размещает: собственники и клиенты. Для заключения сделки должна сохраняться информация о клиенте, собственнике, заявке и размещении, а также, информация о том, когда была заключена сделка. По этим данным сформируем несколько сущностей.

**Пользователь**:

* идентификатор,
* ФИО,
* телефон,
* роль (клиент или собственник жилья),
* логин (адрес электронной почты),
* пароль,
* логическое поле, отображающее права (администратор или пользователь);

**Предложение**:

* номер контракта (он же идентификатор),
* дата,
* статус («активное», «неактивное»),
* тип услуги («сдам», «продам»),
* идентификатор владельца,
* тип жилья,
* район,
* количество комнат,
* адрес,
* цена,
* описание,
* площадь;

**Заявка**:

* номер заявки – идентификатор,
* дата,
* тип заявки («куплю», «сниму»),
* идентификатор клиента,
* тип жилья,
* район,
* площадь,
* количество комнат,
* цена «от»,
* цена «до»,
* статус («активная», «неактивная»);

**Сделка**:

* идентификатор, номер контракта (копируется номер предложения),
* код клиента,
* код собственника,
* дата заключения сделки.

Для разграничения прав доступа будет использоваться сущность «пользователь», которая содержит те три поля, необходимые для разграничения прав: логин, пароль и флаг прав: логическое значение, показывающее, обладает ли данный пользователь административными правами или нет.

2.1 Выбор инструментов и результат моделирования

Для разработки данной базы данных использовались СУБД MySQL 5.5, EMS Manager — менеджер баз данных. Использование данных программных продуктов позволило реализовать модель данных в виде, представленном на рис.1.1.

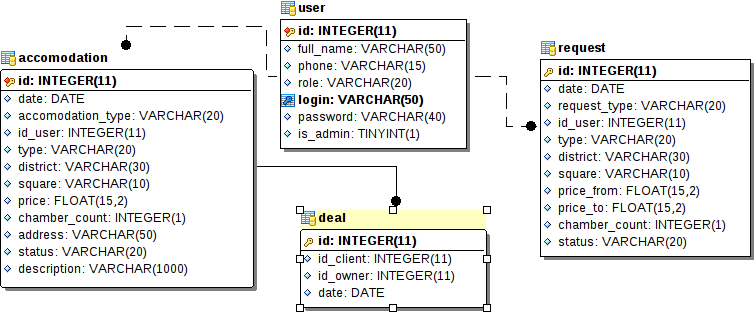


Рис. 1.1. Диаграмма сущностей, модель базы данных «Персонал»

Описание физической модели данных для каждой сущности можно представить в виде следующих таблиц:

Табл. 1.1. Описание сущности «Accomodation»

|  |  |  |  |
| --- | --- | --- | --- |
| **Поле** | **Тип** | **Описание** | **Пример** |
| data | Date | Дата объявления | 2014-04-01 |
| id | Integer | Номер контракта, ключевое поле | 1 |
| status | varchar | Показывает, доступно ли жильё для выбора | Неактивно или Активно |
| Accommodation\_type | varchar | Показывает тип предложения | Сдам или продам |
| id\_user | Integer | Внешний id собственника | 3 |
| district | varchar | район | Промышленный |
| type | varchar | Тип жилья | Коттедж |
| address | Varchar(40) | Домашний адрес | пр. Коростелёвых, 153 |
| price | float | Цена | 12000.00 |
| Chamber\_count | integer | Количество комнат | 2 |
| description | Text | Подробное текстовое описание жилья | Квартира в хорошем состоянии с отличным косметическим ремонтом, санузел раздельный и т.д. |
| square | varchar | Площадь | 42 |

Табл. 1.2. Описание сущности «Request»

|  |  |  |  |
| --- | --- | --- | --- |
| **Поле** | **Тип** | **Описание** | **Пример** |
| data | Date | Дата объявления | 2014-04-01 |
| id | Integer | Номер контракта, ключевое поле | 1 |
| status | varchar | Показывает, действительна ли заявка | Неактивно или Активно |
| request\_type | varchar | Показывает тип заявки | Куплю или сниму |
| id\_user | Integer | Внешний id Клиента | 3 |
| district | varchar | район | Промышленный |
| type | varchar | Тип жилья | Коттедж |
| Price\_from | float | Верхний потолок цены | 12000.00 |
| Price\_to | float | Нижний потолок цены | 14000.00 |
| square | varchar | Желаемая площадь | 42 |
| Chamber\_count | integer | Желаемое количество комнат | 2 |

Табл. 1.3. Описание сущности «User»

|  |  |  |  |
| --- | --- | --- | --- |
| **Поле** | **Тип** | **Описание** | **Пример** |
| id | Integer | Уникальный id пользователя, ключевое поле | 1 |
| Full\_name | Varchar | Ф.И.О. клиента | Лестрейндж Б.Л. |
| Phone | Varchar | Номер телефона | +79241232457 |
| Role | Varchar | Тип учётной записи | Клиент или владелец |
| Login | Varchar | Логин, адрес электронной почты для входа | [admin@example.com](mailto:admin@example.com) |
| Password | Varchar | Пароль. Хэшируется в md5 | 289dff07669d7a23de0ef88d2f7129e7 |
| Is\_admin | Boolean | Флаг прав. | 1 для администратора, 0 для пользователя |

Табл. 1.4. Описание сущности «Deal»

|  |  |  |  |
| --- | --- | --- | --- |
| **Поле** | **Тип** | **Описание** | **Пример** |
| Id | Integer | Ключ записи, копируется номер предложения | 1 |
| Id\_client | integer | Внешний id клиента | 3 |
| Id\_owner | integer | Внешний id собственника | 2 |
| data | Date | Дата заключения сделки | 2014-04-04 |

После создания модели были внесены тестовые значения, необходимые для дальнейшей разработки и тестирования программной оболочки для базы данных «Агентство».

3. Программная реализация

3.1 Язык программирования Perl, технология CGI, Apache HTTP-сервер, фронт-энд фреймворк Twitter Bootstrap

Прежде чем начинать разработку приложения, стоит выбрать язык программирования. Для разработки веб-приложения базы данных «Персонал» использовался язык Perl [4] и технология CGI — Common Gateway Interface [2]. Рассмотрим это подробнее.

Perl — язык программирования общего назначения, который был первоначально создан для манипуляций с текстом, но на данный момент используется для выполнения широкого спектра задач, включая системное администрирование, веб-разработку, сетевое программирование, игры, биоинформатику, разработку графических пользовательских интерфейсов.

Язык можно охарактеризовать скорее, как практичный (лёгкость в использовании, эффективность, полнота), чем красивый (элегантность, минималистичность). Главными достоинствами языка являются поддержка различных парадигм (процедурный, объектно-ориентированный и функциональный стили программирования), контроль за памятью (без сборщика мусора, основанного на циклах), встроенная поддержка обработки текста, а также большая коллекция модулей сторонних разработчиков — Comprehensive Perl Archive Network или CPAN

Согласно Ларри Уоллу, автору языка, у Perl есть два девиза. Первый — «Есть больше одного способа это сделать» («*There’s more than one way to do it*»), известный также под аббревиатурой TMTOWTDI. Второй слоган — «Простые вещи должны оставаться простыми, а сложные — стать выполнимыми» («*Easy things should be easy and hard things should be possible*»). Примечательно, что создатель языка лингвист по образованию, но в конечном счёте, стал программистом.

Программы на этом языке называются скриптами или сценариями, основной особенностью языка считаются его богатые возможности для работы с текстом, в том числе работа с регулярными выражениями, встроенная в синтаксис. Перл унаследовал много свойств от языков Си, AWK, скриптовых языков командных оболочек UNIX.

*Common Gateway Interface* — «общий интерфейс шлюза» — стандарт интерфейса, используемого для связи внешней программы с веб-сервером. Программу, которая работает по такому интерфейсу совместно с веб-сервером, принято называть шлюзом, хотя многие предпочитают названия «скрипт» (сценарий) или «CGI-программа». Сам интерфейс разработан таким образом, чтобы можно было использовать любой язык программирования, который может работать со стандартными устройствами ввода-вывода. Такими возможностями обладают даже скрипты для встроенных командных интерпретаторов операционных систем, поэтому в простых случаях могут использоваться даже командные скрипты. В нашем случае, понадобятся специальные настройки Apache HTTP-сервера — серверного приложения, в качестве основных особенностей которого: кроссплатформенность, модульность, кастомизация настроек посредством правки файлов конфигурации [1].

Также, для упрощения верстки шаблонов приложения, использовался широко распространённый фронт-энд фреймворк Twitter Bootstrap, разработанный компанией Twitter и используемый на сегодняшний день в огромном количестве проектов. Сам фреймворк представляет собой набор css-стилей, иконок и jquery-скриптов для создания современных, красивых и простых в восприятии сайтов. Использование данного фреймворка упрощает задачу проектирования пользовательского интерфейса для сайта и позволяет сосредоточить внимание не на вёрстке интерфейсов приложения, а его программировании.

3.2 Разработка приложения

Разработанное приложение должно уметь:

* отображать все данные из сущностей «Предложение», «Заявка» и «Сделки» базы данных на отдельных страницах-формах, причём эти данные должны быть представлены в виде, удобном для чтения и быстрого ориентирования, иметь элементы управления для перехода к подробному описанию, кнопку удаления с подтверждением;
* в описании конкретного объявления должна быть возможность закрыть его, в случае, если по каким-то причинам оно должно быть закрыто или удалено, при этом, данное действие должно быть доступно только автору;
* Отображаться форма оформления запроса на жильё для клиента, также, для любого пользователя в личном кабинете должна быть отображена информация о сделках, в которых данный пользователь фигурирует: предложение или запрос, пользователь-автор и пользователь-клиент;
* авторизация и регистрация пользователей.

Вся разработка веб-приложения велась в Jetbrains Webstorm 8, доступной для бесплатного 30-дневного пользования, база данных — в EMS Manager, менеджере баз данных для СУБД MySQL, сервером для приложения является Apache HTTP-server, операционная система – Linux Mint 16.

Одной из неприятных особенностей Perl можно считать отсутствие встроенного шаблонизатора, в связи с чем обычный веб-скрипт на этом языке выглядит как мешанина из кода Perl с кусками верстки на HTML. Для того, чтобы решить данную проблему, потребовался модуль HTML::Template с cpan.org. Это один из шаблонизаторов, которым пользуются в данном языке чтобы отделить и обособить верстку страницы от программирования поведения скрипта. Существует множество различных шаблонизаторов для данного языка, но выбран был именно этот в связи с наличием хорошей подробной документации по нему, простоты использования и легковесности самой библиотеки. В отличии, например, от Template Toolkit, в котором теги самого шаблонизатора выглядят примерно так: [% FOREACH i = items %], в HTML::Template теги очень похожи на теги самого HTML: <TMPL\_LOOP NAME="items">, что несколько улучшает читабельность шаблона и позволяет быстрее ориентироваться в больших файлах.

Установка модулей в Perl производится средствами специального менеджера пакетов – cpan (аналогично названию библиотеки с модулями), для установки, как правило, вводится строка вроде: cpan install HTML::Template. Некоторые модули, а именно, все, которые отсутствуют в стандартном пакете языка, но требуемые для работы, такие как шаблонизатор, модуль подключения к СУБД и модуль хэширования в md5, были установлены именно таким образом, что сильно упростило задачу расширения языка.

Сам Perl, представленный в Linux, имеет версию 5.14.

Структура приложения выглядит следующим образом:

* bootstrap — директория с файлами фреймворка;
* cgi-bin — директория со всеми скриптами, которые и строят приложение. Содержит:
  + accomodation — директория содержит скрипты addAccomodation.pl, closeAccomodation.pl, showAccomodation.pl, deleteAccomodation.pl, getAllAccomodation.pl;
  + request — аналогично, скрипты addRequest.pl, showRequest.pl, closeRequest.pl, deleteRequest.pl, getAllRequest.pl. Стоит отметить, что скрипты с названием, начинающимся с add, добавляют запись в базу, getAll — читают все записи сущности, show — читают определённую запись, delete — удаляют запись. Это справедливо для всех скриптов в приложении;
  + deal — getAllDeal.pl, makeDeal.pl;
  + user — myOrders.pl
  + Также, основная директория cgi-bin содержит скрипты login\_new.pl (вход), logout.pl (выход), registration.pl (регистрация нового пользователя;
* Tmpl — директория со всеми шаблонами, используемыми в приложении. Содержит такие шаблоны, как footer.tmpl, header.tmpl, которые используются во всех других шаблонах и упрощают вёрстку в плане использования одного компонента во всех других формах (в шаблоне header, например, содержится верстка верхних элементов навигации, подключения скриптов и стилей. Использование такого подхода общепринято, т.к. не приходится вносить изменения в каждый отдельный шаблон, достаточно лишь изменить конкретный «кусочек», используемый везде), index.tmpl, выполняющий роль формы входа и регистрации, accomodation.tmpl, deals.tmpl и request.tmpl, используемые для представления всех записей сущностей «accomodation», «deal» и «request» соответственно в виде таблице, а также, возможности добавлять и удалять определённые записи из сущностей, шаблоны showAccomodation.tmpl, showRequest.tmpl используются для вывода полной информации о записи, myDeals.tmpl отображает список всех сделок с конкретным пользователем;
* В корне самого приложения лежат также файл index.html, представляющий собой «редирект» на форму входа приложения. Данный способ необходим для того, чтобы при входе на основной сайт, например, 127.0.0.1, localhost или [www.example.com](http://www.example.com), автоматически происходило перенаправление на страницу входа или любую другую страницу.

3.3 Интерфейс приложения

Большинство любых современных приложений создаются с учётом «**юзабилити»** – степени, с которой продукт может быть использован определёнными пользователями при определённом контексте использования для достижения определённых целей с должной эффективностью, продуктивностью и удовлетворённостью. Это значит, что программный продукт должен быть интуитивно понятен (элементы управления должны находиться там, где пользователь, скорее всего, ожидает их увидеть), при этом, все элементы должны быть действительно необходимыми (дублирование или избыточность лишь усложняет работу с ПО), а также, должен выглядеть достаточно минималистично и привлекательно. Потому было разработано приложение, которое бы как можно больше удовлетворяло бы этому термину.

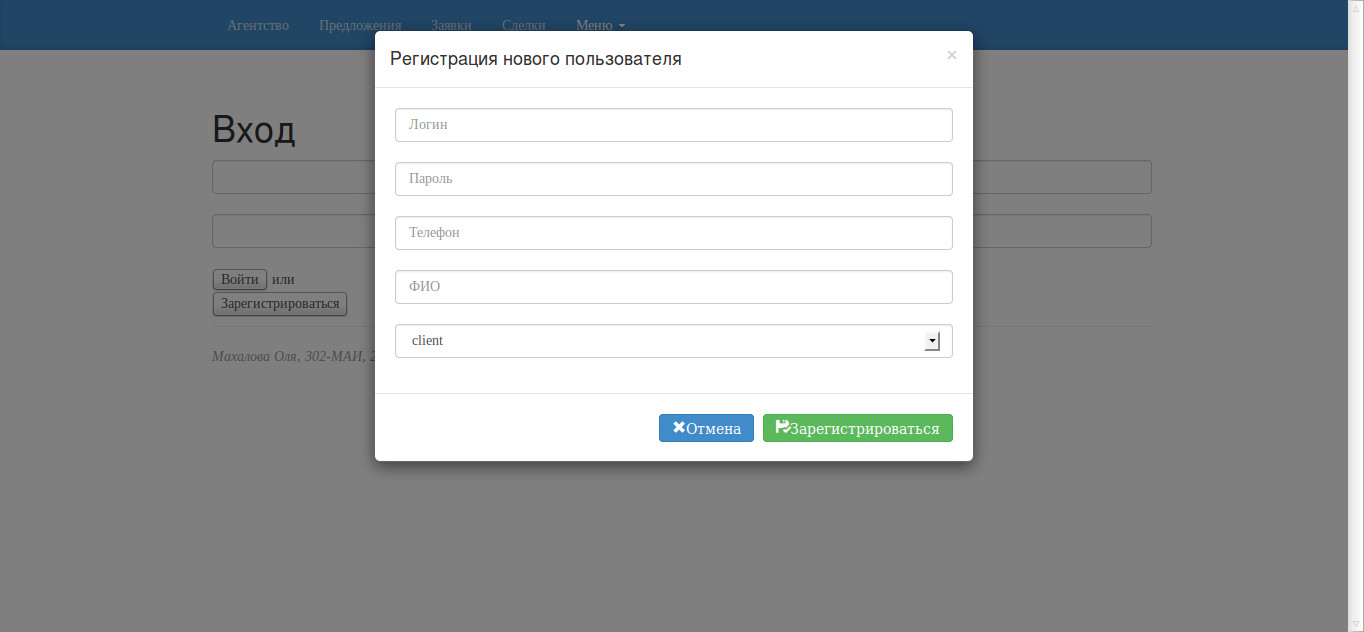


Рис. 3.1. Форма входа с открытым модальным окном регистрации.

На рис. 3.1. показан пример использования модальных окон в шаблонах. Данное действие позволяет использование одного шаблона для нескольких действий, а также, улучшает «юзабилити»: пользователю не нужно перезагружать страницу, чтобы получить окно для регистрации, в любой момент он может закрыть это окно и войти под уже существующим профилем.

Стоит сказать несколько слов о валидации и представлении некоторых форм ввода. В стандарте HTML5 содержатся встроенные средства для проверки валидности email’а. Данная возможность позволяет упростить валидацию электронной почты, достаточно задать тегу input значение типа = email. Для того, чтобы добавить маску на строку ввода пароля, также используется значение типа, в этот раз = password. Однако для проверки поля на отсутствие информации нужно написать небольшой JavaScript-сниппет:

<script type="text/javascript">

var name = document.getElementById("pass");

$(name).blur(function(){

if (name.value === "") {

name.setCustomValidity("Please fill out the field with your password");

}

});

</script>

Данный код позволяет запретить вход или регистрацию, если поле пароля пустое.

JavaScript и JQuery код используется ещё и для придания веб-приложению возможности выделения элементов управления сверху в зависимости от активной в данный момент формы. Например, при открытой форме отделов будет выделен соответствующий элемент «предложения» на верхней панели как на рис.3.2. Также, JQuery используется для блокирования кнопок «закрыть» и «сделать заказ», если статус объявления равен «inactive» (рис. 3.3).

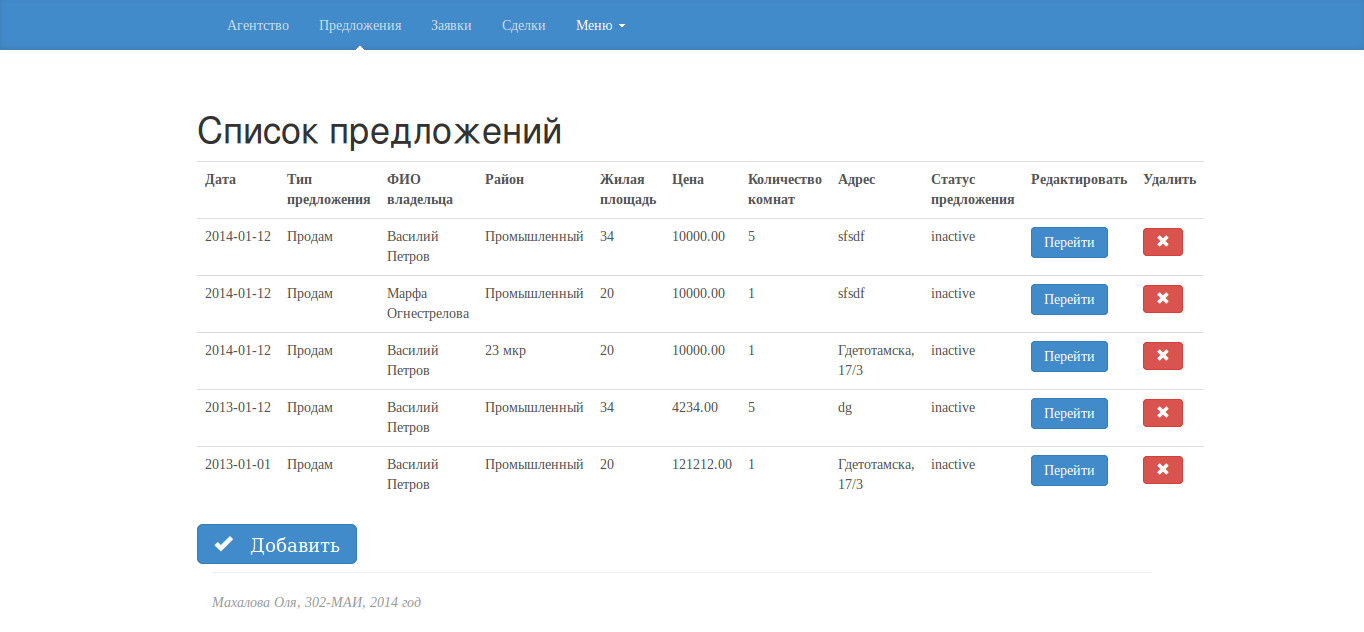


Рис.3.2. Форма «Предложения», шаблон accomodation.tmpl

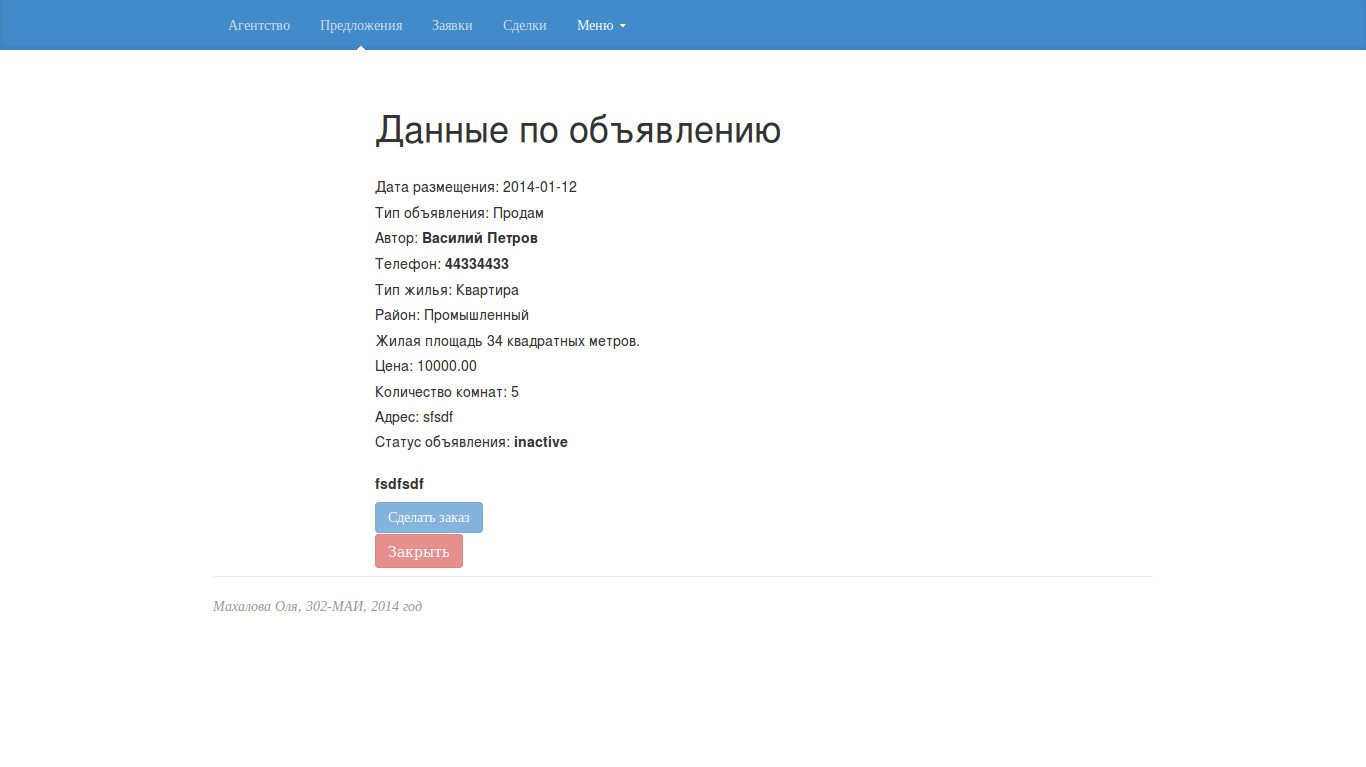


Рис.3.3. Форма «Показать предложение» с заблокированными кнопками, зависящими от статуса

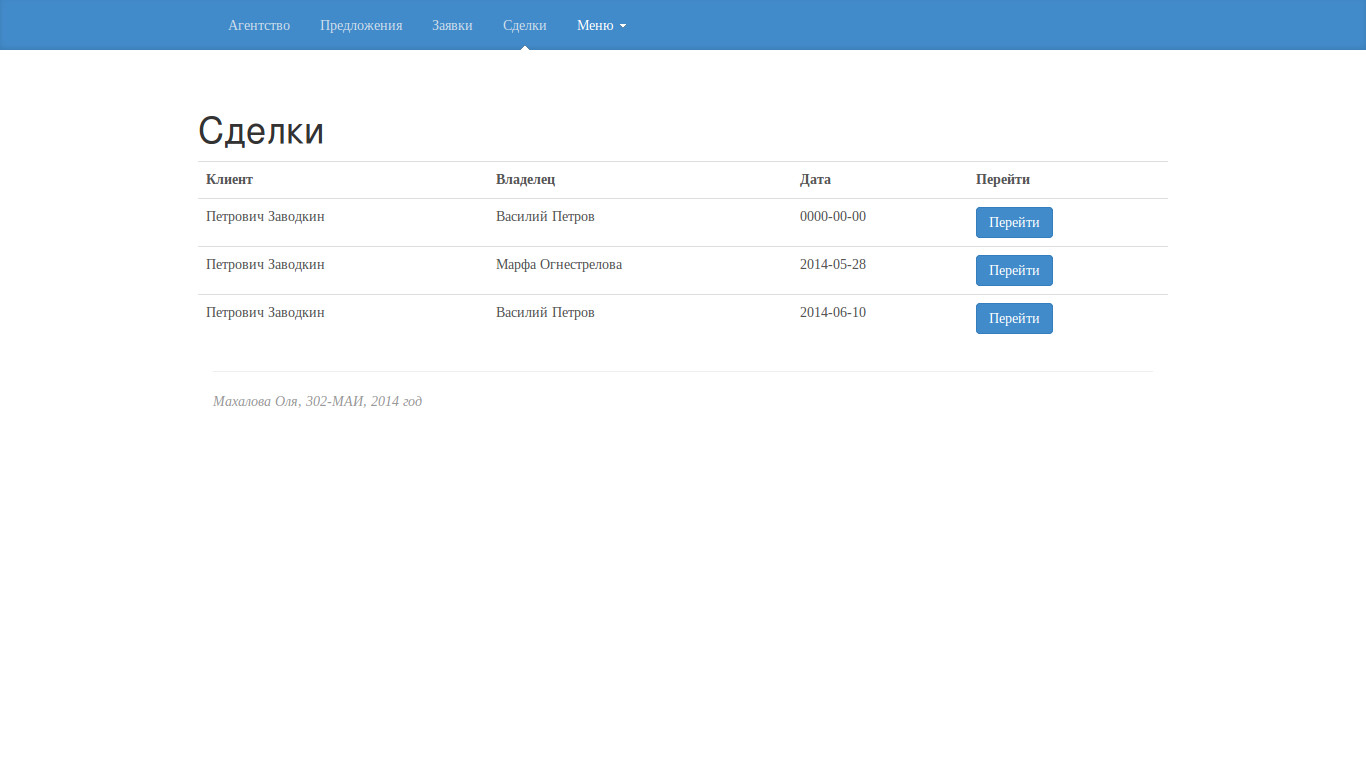


Рис.3.4. Список сделок. При нажатии на «перейти» осуществляется переход к подробному описанию размещения.

Формы добавления сущностей оформлены в виде модальных окон, что, как раньше уже было сказано, позволяет ввод данных без перезагрузки страницы или перехода на другую. Одна из таких форм представлена на рис.3.5.

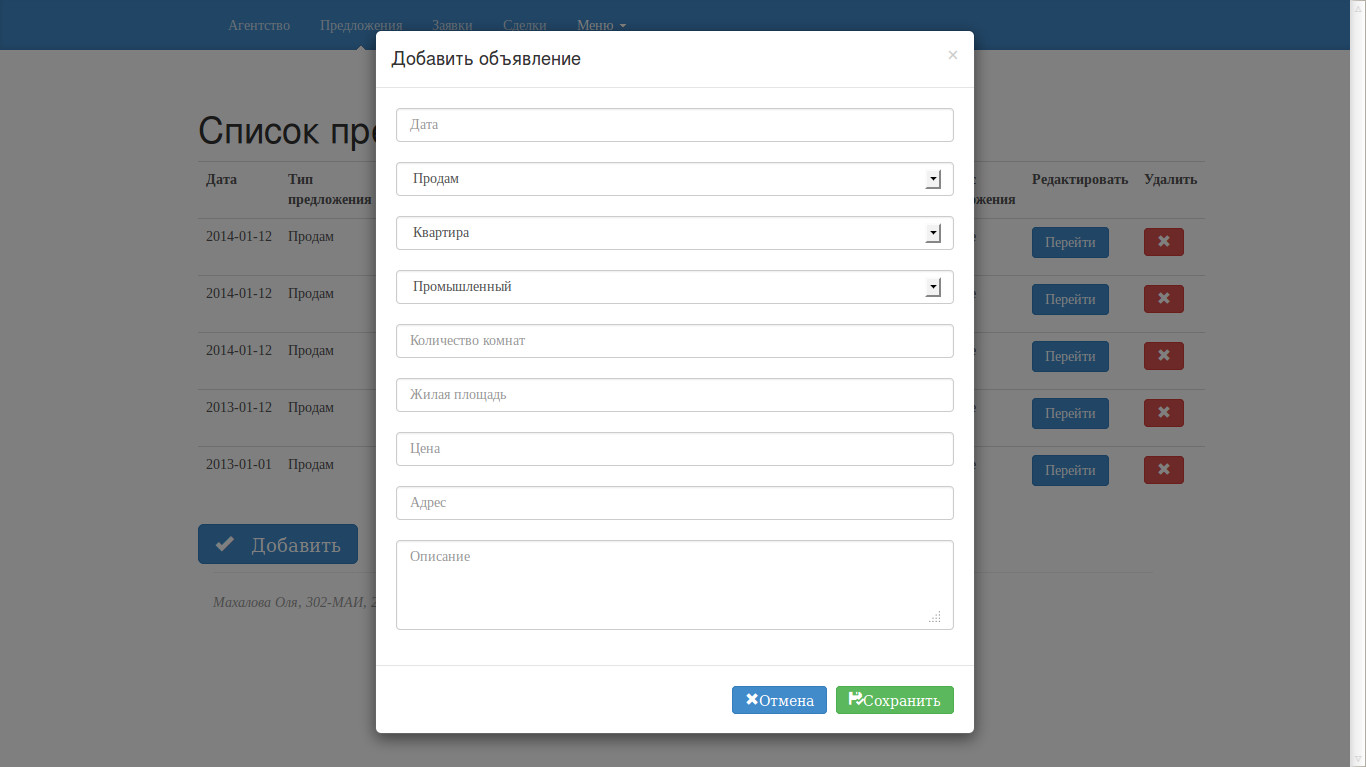


Рис.3.5. Форма добавления нового предложения.

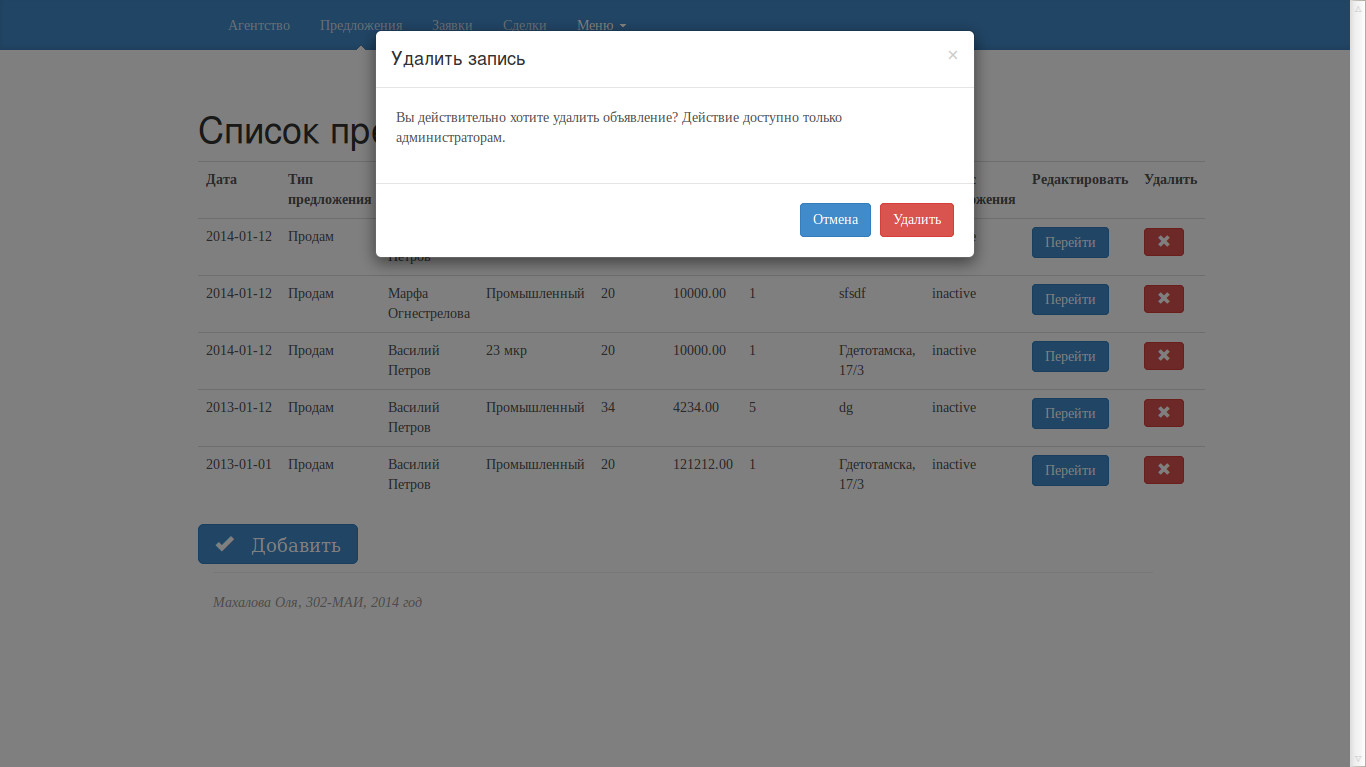


Рис.3.6. Форма «предложения» и модальное окно подтверждения удаления.

Удаление записей из базы данных сопровождается подтверждением о данном действии, так что пользователь, случайно нажавший на кнопку удаления, сможет отменить это действие и не лишится важных данных.

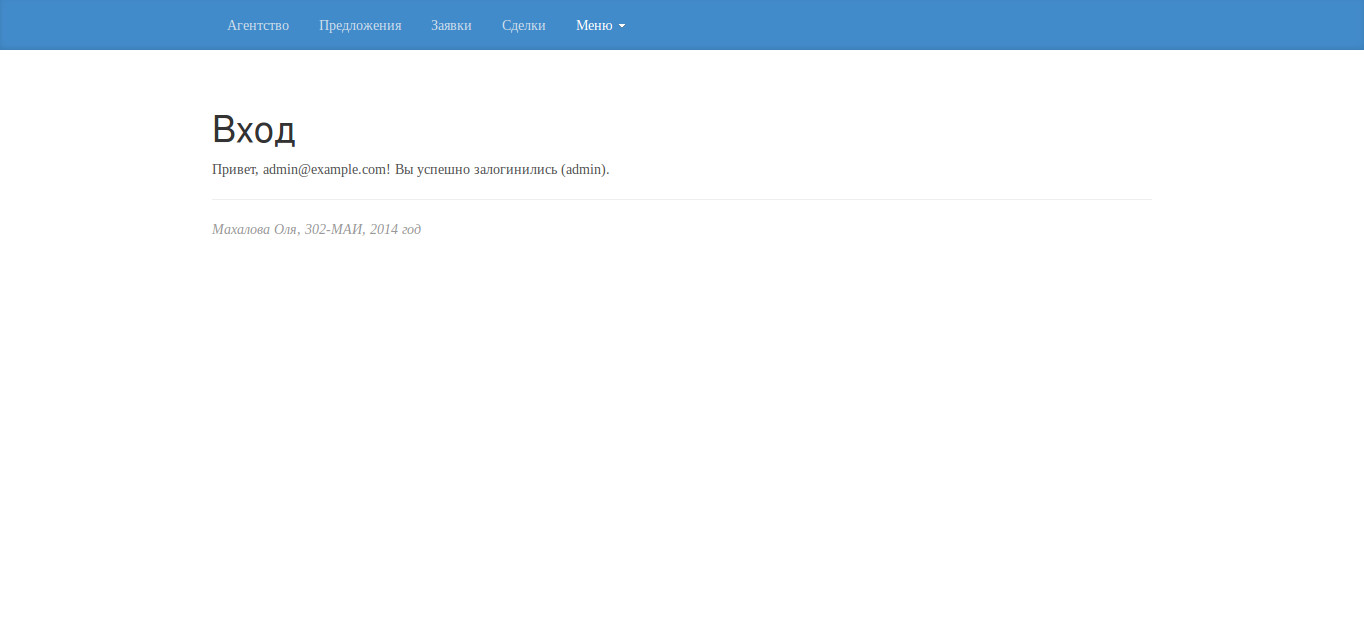


Рис.3.7. Та же форма логина и регистрации, но с другой логикой.

Так как использовался шаблонизатор, имеется возможность задать поведение для одного и того же шаблона в зависимости от переданных данных, что, однако, относится всё ещё к вёрстке, а не программированию, и это является ещё одним плюсом использования принципа разделения кода и представления (пример на рис.3.7: если войти в систему, форма входа изменится на форму приветствия, однако сам шаблон остаётся прежним, другой пример — отображение кнопки «закрыть» происходит лишь для того пользователя, который является автором объявления). В современных веб-фреймворках такой принцип, чаще всего, относится к схеме Model-View-Controller, MVC, когда модели объекта, логика поведения и представление полностью абстрагированы друг от друга физически (хотя и не логически). В данном приложении, можно сказать, использовался принцип View-Controller, что вполне оправдано, так как оно небольшое.

Последний важный момент — настройка сервера для запуска приложения. Конфигурационный файл для Apache HTTP-server выглядит следующим образом:

<VirtualHost \*:80>

ServerAdmin admin@localhost

DocumentRoot /var/www

ScriptAlias /var/cgi-bin/

<Directory "/var/cgi-bin/\*">

AllowOverride None

Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch

Require all granted

</Directory>

ErrorLog ${APACHE\_LOG\_DIR}/error.log

CustomLog ${APACHE\_LOG\_DIR}/access.log combined

</VirtualHost>

Здесь задаётся поведение сервера: при запросе вида, к примеру, <http://localhost/login_new.pl>, сервер будет автоматически передавать скрипт интерпретатору, а позже выдавать получившийся результат пользователю.

4. Тестирование и заключение

4.1 Тестирование конечного веб-приложения

В ходе тестирования конечного приложения в разных ОС и браузерах было выявлено следующее:

* Верстка выглядит одинаково в браузерах IE 11, Firefox 27+, Chrome и Chromium 35+ различных семейств ОС, за исключением значения date атрибута type тега input: лишь Chrome/Chromium отображает виджет выбора даты для тега вида <input type=«date»…/>. Впрочем, всё в соответствии с рис. 4.1;
* За счёт фреймворка Bootstrap был создан современный и качественный интерфейс без особых усилий, который, в том числе, работает одинаково почти везде что крайне важно при конечном тестировании;
* Скорость работы удовлетворительная ввиду высокой скорости работы самого языка, СУБД и сервера Apache, разницы в различных семействах ОС не замечено;
* При переносе приложения из одной ОС в другую требуется заново устанавливать модули, а в случае перехода в Windows желательно устанавливать пакет strawberry вместо встроенных средств пакетов Denver или XAMPP (которые также желательно ставить чтобы иметь уже настроенный Apache-сервер) или аналогичный для удобной работы с Perl.

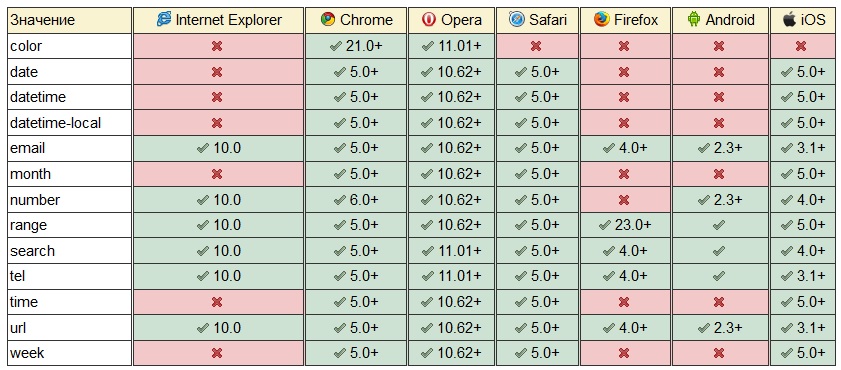


Рис. 4.1. Поддержка браузерами значений атрибута type для тега input.

4.2 Вывод

В процессе выполнения данного курсового проекта были выполнены все поставленные задачи: создана модель, создано полноценное веб-приложение, а также, освоены навыки программирования на языке Perl и улучшены навыки администрирования сервера Apache для ОС семейств Linux и Windows. Разработанные база данных и веб-приложение вполне может сгодиться для работы в реальных условиях, с небольшими модификациями или без них, а созданная структура приложения вполне может сгодиться каркасом для других похожих небольших оболочек к базам данных, она легко изменяется и имеет модульную структуру. Дизайн вёрстки, за счёт следования принципу KISS, также легко изменяется, что явный плюс в случае, если нужна модификация интерфейса.

На диске содержатся исходные коды веб-приложения, база данных и некоторые конфигурационные файлы. Исходные файлы приложены в том виде, в котором они использовались для развёртывания на ОС Linux Mint 16.

# **Список литературы**

x

|  |  |
| --- | --- |
| 1. | Apache // Wikipedia. URL: http:/​/​ru.wikipedia.org/​wiki/​Apache (дата обращения: 22.06.2014). |
| 2. | CGI // Wikipedia. URL: http:/​/​ru.wikipedia.org/​wiki/​CGI (дата обращения: 22.06.2014). |
| 3. | CRUD // Wikipedia. URL: http:/​/​ru.wikipedia.org/​wiki/​CRUD (дата обращения: 22.06.2014). |
| 4. | Perl // Wikipedia. URL: http:/​/​ru.wikipedia.org/​wiki/​Perl (дата обращения: 22.06.2014). |
| 5. | Дейт К.Д. Введение в системы баз данных, Пер. с англ. 8-е изд. М.: Вильяме, 2005. 1328 с. |
| 6. | Прохоренок Н.А. Разработка Web-сайтов с помощью Perl и MySQL. СПб.: БХВ-Петербург, 2009. 25-292 с. |

x