Министерство образования и науки РФ

Кольский филиал федерального государственного бюджетного образовательного

учреждения высшего профессионального образования

«Петрозаводский государственный университет»

Кафедра информационных систем

(Кафедра информационных технологий)

**Разработка Web-приложения по предоставлению услуг организации спортивных мероприятий**

Курсовая работа

студента 4 курса (гр.2)

очного отделения

факультета ИПМ

специальность 230201 –

*Информационные системы и*

*технологии*

**Васильевой Анны**

Научный руководитель:

к.т.н., доцент Тоичкин Н.А.

Оценка публичной защиты работы:

Апатиты 2015

**Оглавление**

[ВВЕДЕНИЕ 3](#_Toc435291579)

[1 ОБЩАЯ ЧАСТЬ 5](#_Toc435291580)

[1.1 Описание предметной области 5](#_Toc435291581)

[1.2 Характеристика и возможности современного «Web-приложения» 7](#_Toc435291582)

[1.3 Обзор Web-приложений, содержащих ремонтные услуги 10](#_Toc435291583)

[1.4 Архитектура Web-приложения и методы его разработки. 12](#_Toc435291584)

[1.6 Требования к разрабатываемому Web-приложению и постановка задач 14](#_Toc435291585)

[2 ПРОЕКТИРОВАНИЕ 17](#_Toc435291586)

[2.1 Используемые технологии и программные средства 17](#_Toc435291587)

[2.1.1 Язык UML 17](#_Toc435291588)

[2.1.2 MySQL 20](#_Toc435291589)

[2.1.3 PHP Designer 21](#_Toc435291590)

[2.1.4 PHP 21](#_Toc435291591)

[2.1.5 Denwer 23](#_Toc435291592)

[2.2 Проектирование с помощью CASE технологий. 23](#_Toc435291593)

[2.2.1 Диаграмма USE CASE 23](#_Toc435291594)

[2.2.2 Sequence диаграммы. 25](#_Toc435291595)

[2.2.3 Диаграмма состояний. 27](#_Toc435291596)

[2.4 Описание базы данных 28](#_Toc435291597)

[2.5 Разработка. 30](#_Toc435291598)

[2.5.1 Выбор CSS шаблона. 31](#_Toc435291599)

[2.5.2 Корректировка СSS шаблона. Создание графики. 31](#_Toc435291600)

[2.5.3 Разделение страницы на блоки. 32](#_Toc435291601)

[2.5.4 Создание новых страниц и наполнение их контентом. 33](#_Toc435291602)

[2.5.5 Создание БД 35](#_Toc435291603)

[2.5.6 Реализация администраторской страницы. 36](#_Toc435291604)

[2.6 Итоговая реализация сайта 37](#_Toc435291605)

[ЗАКЛЮЧЕНИЕ. 44](#_Toc435291606)

[СПИСОК ЛИТЕРАТУРЫ 46](#_Toc435291607)

#### ВВЕДЕНИЕ

Сегодняшний мир не может функционировать без новых, скоростных информационных способов связи, как между простыми людьми, так и между работниками и работодателями. Сегодня Интернет - это важнейшее средство массовых коммуникаций, стоящее по своей значимости в одном ряду с прессой, телефоном и телевидением.

Применение Интернета необходимо для многих сфер жизни, так как сеть Интернет предоставляет нам бесчисленное множество разнообразных возможностей. Ведь на наших глазах за последние несколько лет мировая сеть Интернет стремительно превратилась в одно из главных средств, используемых для общения, развлечений, рекламы, торговли, также она является источником разнообразной полезной информации для любой категории пользователей. Но еще одной из важных возможностей Интернета является быстрый и удобный поиск необходимых нам коммерческих услуг и предложений в своем регионе.

По статистике, на сегодняшний день, спортивные мероприятия занимают одну из первых строчек в поисковых системах. Ведь действительно, каждый день кто-то хочет организовать отдых, совмещая его с оздоровительной, физкультурной и спортивной деятельностью. Из-за этого в настоящее время всё больше и больше людей осознают необходимость разработки и поисковой оптимизации Web-приложения, которое могло бы без лишних усилий организовать спортивное мероприятие и выбрать профессионального инструктора.

Целью курсовой работы является разработка информационно-рекламного Web-приложения, по предоставлению организации спортивных мероприятий. Клиенты такого приложения смогут ознакомиться с подробным расписанием на каждый день по каждому виду спорта, а также заказывать услуги по проведению спортивных мероприятий на определенную дату. Для реализации такого Web-приложения необходимо решить следующие задачи:

* изучить предметную область и существующие аналоги;
* изучить архитектуру Web-приложения и методы его разработки на языке PHP;
* разработать требования к Web-приложению;
* разработать модели UML и структуру базы данных;
* реализовать и протестировать Web-приложение по предоставлению ремонтных услуг;

На данный момент в мире существует некоторое количество Web-приложений в сфере спортивных услуг, они отличаются между собой оформлением и спецификой. Однако стоит отметить, что проведенный в ходе исследования сравнительный анализ Web-сайтов, показал, что большинство из них не распространены на города с небольшим населением, а функционируют только в городах областного значения. В этом случае возникает проблема заказа услуг людям, проживающим в региональных городах.

Таким образом, становится очевидным необходимость создания Web-приложения в нашем регионе, специализирующегося на услугах по организации спортивных мероприятий, и распространяющегося на все города, включая региональные.

Это позволит пользователю сайта быстро и удобно найти все необходимые нам услуги в своем городе или регионе, а также очень просто заказать их, на выходя из дома.

# 1 ОБЩАЯ ЧАСТЬ

## Описание предметной области

Все сайты можно разделить как минимум на 9 групп.

1) Коммерческие сайты - к коммерческим следует отнести те сайты, которые непосредственно связаны с ведением бизнеса. Основной аудиторией коммерческого сайта являются действительные и потенциальные клиенты. По своему объему и типу коммерческий сайт может быть:

* Сайт-визитка - состоит буквально из пятидесяти страниц и содержит общие сведения о владельце сайта: его деятельность, реквизиты, контактная информация. То есть такой сайт действительно представляет собой расширенную визитную карточку предпринимателя или организации.
* Сайт-каталог - от сайта-визитки отличается в первую очередь наличием подробного прайс-листа, каталога продаваемой продукции, в котором могут быть перечислены подробные технические характеристики товара;
* Интернет-магазин - содержит каталог продукции, из которого пользователь может прямо на сайте выбрать и заказать нужные ему товары. То есть обязательным признаком интернет-магазина является возможность сделать онлайн заказ. Чаще всего эта возможность дополняется функцией «корзины», которая позволяет за один раз заказать несколько приглянувшихся товаров. сайт инфологическая модель;
* Промо-сайт - создается для презентации какого-либо конкретного товара или торговой марки и является частью общей маркетинговой программы по его продвижению. Содержит подробное описание технических характеристик и преимуществ товара, информацию о различных акциях, скидках и т. п.;
* Бизнес интернет-портал - обычно принадлежат крупным компаниям с широким диапазоном видов деятельности. Содержит очень большое количество страниц, охватывает большое количество тем. Отличительной чертой является предоставление пользователю широкого диапазона различных сервисов: почта, форум, чат и т. п.

2) Некоммерческие сайты - обычно осуществляют предоставление пользователям различной информации (информационные сайты) или оказание каких-либо онлайн услуг (веб-сервисы).

3) Личные страницы - так называемые homepages (домашние страницы), на которых человек размещает информацию о себе, фотографии и какую-либо еще информацию по своему желанию. В настоящее время практически полностью вытеснены блогами и социальными сетями.

4) Поисковые системы. В настоящее время большая часть поисковых систем превратились в полноценные интернет-порталы (Google, Яндекс, Рамблер). Тем не менее в качестве примера «чистой» поисковой системы можно назвать Nigma.ru.

5) Почтовые сервисы - все крупные почтовые сервисы либо стали крупными Интернет - порталами (например, Mail.ru), либо являются частью других порталов (например, Gmail.com как сервис, входящий в Google, или Яндекс.Почта как часть Яндекса).

6) Социальные сети - приобрели в последнее время просто фантастическую популярность. Концептуальный принцип таких веб-сайтов базируются на социологических теориях социальных сетей. Фактически их функции заключаются в возможности зарегистрированных пользователей находить в данной сети определенных людей и поддерживать с ними связь, организовываться в сообщества по интересам, а также в предоставлении пользователям возможности размещать собственный текстовый контент. Крупнейшими социальными сетями в мире являются Facebook и MySpace, а в России - «Вконтакте» и «Одноклассники».

7) Блоговые сервисы - Предоставляют площадки для ведения блогов, позволяют пользователям вести коллективные блоги (сообщества) и организовывать удобное взаимное чтение. Слово «блог» (blog) - это сокращение от web log - «веб-журнал». Фактически представляет из себя публичный онлайн дневник. В настоящее время блоги имеют очень широкий спектр использования: от бытовых заметок для нескольких друзей, до политической трибуны для тысяч людей. В качестве примеров блоговых сервисов можно привести livejournal.com, liveИнтернет.ru, сервис микроблогов twitter.com и другие.

8)Доски объявлений - фактически аналогичны обычным бытовым доскам объявлений или рекламным газетам. Каждый желающий может вывесить своё объявление, а все посетители сайта - прочитать его. Доски объявлений, как правило, поделены на несколько тематических разделов.

9) Онлайн энциклопедии - в подавляющем большинстве случае базируется на вики-принципе, который заключается в том, что любой пользователь может самостоятельно добавлять новые статьи и вносить изменения (правки) в уже существующие. Самой известной вики-энциклопедией является Wikipedia.

Очевидно, что большая часть перечисленных выше сайтов тем или иным способом занимаются зарабатыванием денег (размещением ли рекламы, или какими-либо платными услугами для пользователей). Однако такие сайты в первую очередь созданы не для получения прибыли, а для удовлетворения потребностей пользователей, поэтому их можно назвать некоммерческими. Впрочем, подобные разграничения весьма призрачны, так что многие из этих сайтов можно отнести к категории «условно некоммерческих».

Анализ предметной области является первым этапом построения информационной системы. Здесь осуществляется сбор информации об этой области, установление источников информации, выделение функциональных областей и процессов, происходящих в этих областях.

Знание предметной области проекта способствует глубокому пониманию сути проблемы или поставленной задачи.

Предметная область проекта укладывается в понятиях: сайт и услуги. Таким образом, предметная область проекта представляется как пространство функционирования сайта по организации предоставления ремонтных услуг.

## 1.2 Характеристика и возможности современного «Web-приложения»

Web-приложение представляет собой совокупность электронных документов, размещенных в сети Интернет, обладающих электронным адресом.

Сайт (от англ site - место; местонахождение, местоположение; позиция) так называют ресурсы глобальной компьютерной сети Интернет. Т.е. сайт в Интернете - это специальным образом структурированная информация, размещенная на сервере (компьютер, подключенный к сети) и открытая пользователям этой сети для свободного, авторизируемого или ограниченного доступа. Все многообразие информации в интернете, обозначаемой одним словом -"сайт", можно разделить по: задачам сайта, целям его создания и фунционирования (коммерческий, некоммерческий, информационный, рекламный), объемности (сайты-визитки, интернет-представительства), методу их создания и функционирования (динамические и статические).

Но Web-приложение представляет собой не только некоторый набор текстовой и графической информации, организованный в виде определенного количества Web-страниц, связанных между собой гипертекстовыми ссылками. Под этим сухим определением скрываются практически неограниченные возможности по организации присутствия фирмы в глобальной сети Интернет.

Любому товару или услуге можно дать качественную оценку. Не составляют исключения и веб-сайты. Современное Web-приложение - это прежде всего выдержанный стиль - это когда каждая страничка имеет ярко выраженную принадлежность ко всему Web-приложению. Когда легко ориентироваться и поиск информации не сопряжен с опасностью заблудиться и потерять время. Когда странички загружаются менее, чем за минуту. Стильность Web-приложения достигается несколькими приемами. Можно выделить следующие элементы, участвующие в создании стиля:

* шрифт - в пределах сайта он должен иметь одинаковые характеристики - такие, как гарнитура (начертание), кегль (высота), цвет;
* абзац - желательно, чтобы преобладал какой-нибудь один из видов выравнивания на страничке, например, публикация сделана с отступом от левого края и выравниванием влево;
* цветовая схема web -сайта - она начинается с выбора тех трех цветов страницы, которые используются для представления обычного текста, ссылок и посещенных ссылок. Помимо этого, цветовая схема веб-сайта должна отражать общий фирменный стиль. Цветовая схема должна повторяться на всех страничках публикации, это создаст у посетителя ощущение связности сайта;
* Цвета ссылок должны быть сделаны таким образом, чтобы, с одной стороны, посетитель видел, что это ссылка, а с другой стороны, она бы не мешала читать основной текст;
* графическое оформление сайта - во-первых, оно должно укладываться в общую цветовую схему; во-вторых, должна быть продумана общую концепцию графического оформления. Все графические элементы можно разделить на два больших класса: рисованные и фотореалистические. Смешивание этих двух типов в оформлении нежелательно. Кроме этого, в случае, если используются на сайте фотографии в качестве иллюстраций, то перед использованием они должны быть обработаны - случае необходимости делается тоновая и цветовая коррекция, кадрирование. Данное оформление должно использоваться по всему сайту;
* навигация по сайту - именно она не дает посетителю запутаться в контенте (содержимом) веб-сайта. Для посетителя всегда должна быть предусмотрена возможность перехода на главную страничку веб-сайта. В случае, если на сайте используется навигационную графическая панель (выполненная в виде картинок), то обязательно должна быть текстовая копию, в противном случае посетитель сайта может не получить доступ к навигации по сайту в случае если его браузер не поддерживает загрузку картинок;
* сетикет (сетевой этикет) означает совокупность правил поведения и обхождения, принятых при использовании компьютерных сетей. Современные компьютерные сети порождают виртуальные города, виртуальные центры для гуляния, персональные страницы, веб-сайты различных компаний, правительственных, коммерческих и некоммерческими организаций. Бизнес-общество должно уважать весь этот огромный коммерческий потенциал. Сеть - это также персональный дом, окрестность, или даже город для миллионов. Правила бизнес-сетикета появились как руководство для установления и поддержки максимально эффективных взаимоотношений между коммерческой деятельностью и ее клиентами. Компании должны всегда помнить, что потенциальный покупатель всегда контролирует себя и имеет выбор.

Если говорить о современных и уже сложившихся правилах сетевого этикета, то можно выделить следующие:

* Внимательность к пользователям старых систем. Необходимо представлять текстовую версию веб-сайта на сплеш-странице (версии устаревших программных продуктов и систем не поддерживают современные технологии ). Поскольку пользователи платят за нахождение в сети, необходимо организовать простой и быстрый доступ к содержанию и информации;
* Использование сплэш-страницы для задания стиля и характера сайта. В тоже время, стиль веб-сайта должен соответствовать характеру компании;
* Целесообразно отображать дату и время последнего обновления вебсайта. Это говорит о том, что сайт используется практически, т.е. представленная на веб-сайте информация является актуальной;
* Минимум использования графики большого объема, анимации и аудио-программ, которые расточают время пользователя. Это связано с тем, что не все посетители имеют высокоскоростной канал доступа в Интернет, а так же с тем, что при просмотре сайтов с большим объемом контента возрастает сетевой трафик, а соответственно и расходы посетителей;
* Для организации доступа к любой информации, размещенной на сайте, должно действовать правило трех кликов - пользователь не захочет искать нужную ему страницу более чем тремя кликами;
* На сайте должна быть представлена информация о компании, которая обязательно включает e-mail, телефон, факс, физический адрес. Так же целесообразно предоставлять посетителем схему проезда до офиса, магазина и пр. компании;
* Если на сайте используются форумы, необходимо чтобы их темы обновлялись. Темы форума должны соответствовать интересам целевой аудитории веб-сайта;
* Необходимо предугадывать потребности пользователя. Например, если на веб-сайте используется база данных, должна быть и система поиска по ней;
* Для Web-сайта должен использоваться логичный URL. Пользователи не должны видеть Web-сайт, если он еще находится в стадии разработки. Негативное впечатление от недоделанного веб-сайта отпугнет посетителей и испортит не только его имидж, но и имидж компании;
* Если пользователи пришли воспользоваться какой-т услугой, представленной на веб-сайте, они должны попасть на нее с первой же страницы;
* Разработанная система ответов на e-mail письма создает положительное впечатления от веб-сайта в целом. Немедленный ответ говорит пользователям о том что их сообщение получено, и что веб­сайт действенен. Также необходимо сообщать пользователю примерные сроки ответа;
* При ссылке на другой веб-сайт, пользователь должен непосредственно направлятся на конкретную необходимую страницу, а не на главную страницу сайта;
* Любая информация должна быть размещена с соблюдением политики авторского права на графику и информацию. Не соблюдение этого правила приводит к различным видам ответственности.

## Обзор Web-приложений, содержащих услуги по организации спортивных мероприятий

На сегодняшний день, существует некоторое количество Web-сайтов в сети Интернет, содержащих в себе объявления по организации спортивных мероприятий. Каждому из которых можно дать качественную оценку. По каким принципам можно оценить сайт? Перечислим самые актуальные:

* Посещаемость.

Пожалуй, это главный и наиболее понятный показатель сайта. Всё прoсто: чем выше посещаемость ресурса, тем лучше. Различают посещаемость за сутки, неделю и месяц. Наиболее распространённый показатель - именно за сутки, а другие используются реже. Высокая посещаемость - то, к чему стремятся все владельцы. Посетителей никогда не бывает слишком много - это аксиома.

* Дизайн и удобство его использования.

Комфортный дизайн и удобство использования Web-приложений-еще один важный момент в оценке сайта. Если он удобен в использовании, то его посещают все чаще и чаще, если же не удобен, то практиковать его никто не станет. По мере роста количества Web-сайтов пользователи становятся все менее терпимы к тем из них, которыми сложно пользоваться. Поэтому каждая ошибка при разработке дизайна Web-приложения может привести к вымиранию сайта.

* Полезность.

Простыми словами — это эффективность приложения. Так как заходя на сайт, пользователь должен получить максимальное количество информации, и получить то, что ему нужно.

Ниже рассмотрим наиболее популярные Web-приложения услуг по организации спортивных мероприятий:

1. NashaLiga

Этот сайт позволяет выполнять следующие задачи:

* Самостоятельно (не автоматически) просматривать список доступных услуг;
* Отправить заявку на проведение спортивного мероприятия;
* Консультироваться по интересующим вопросам у специалистов;

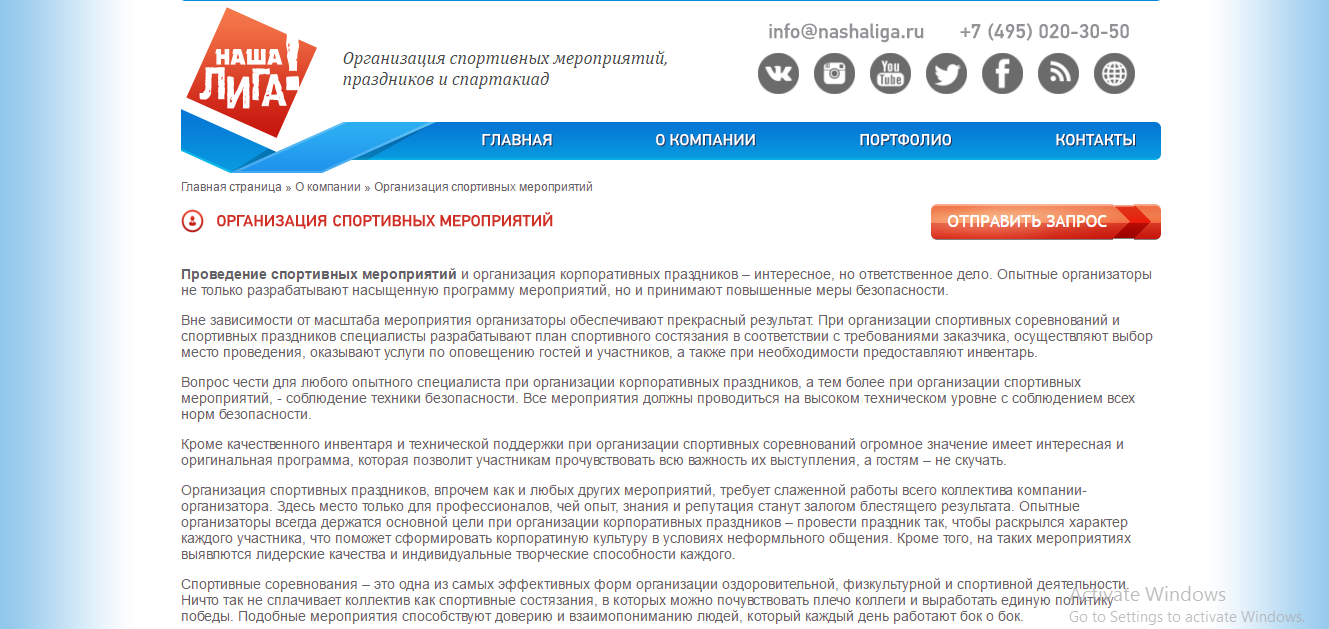


рис. 1.1. Web – приложение «NashaLiga»

1. Rusporting

Этот сайт позволяет нам:

* Организовывать спортивные праздники и турниры, спартакиады;
* Арендовать спортивный инвентарь;
* Выполнять экипировку команд;

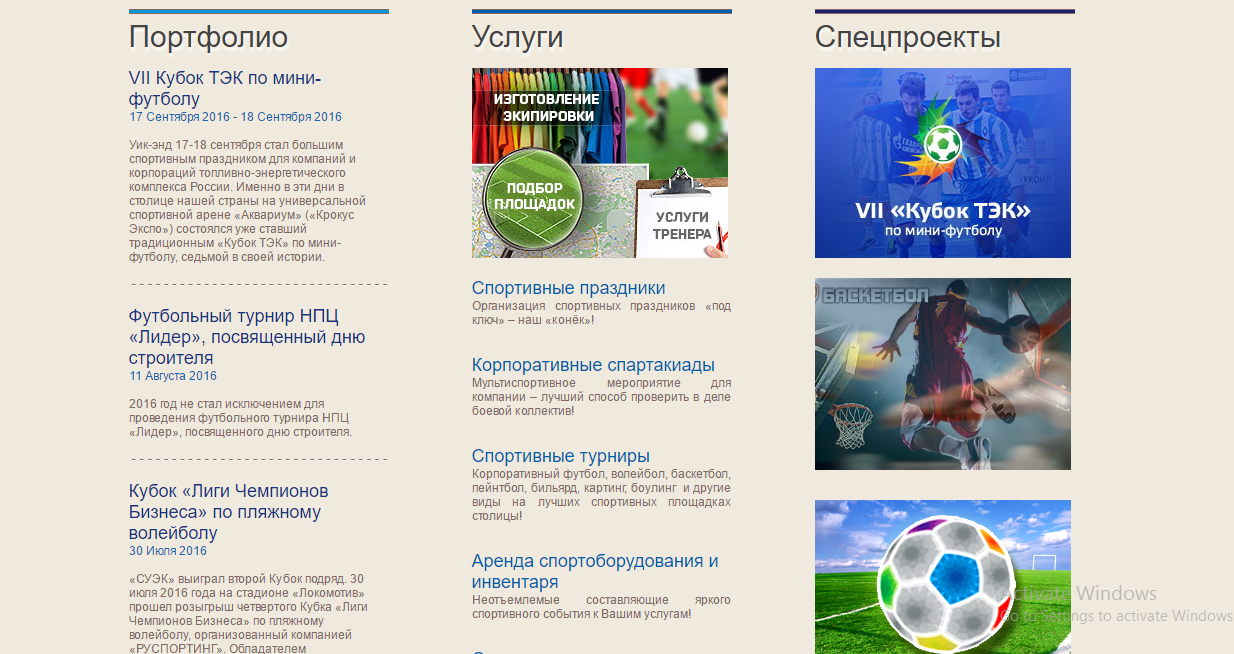


рис. 1.2. Web – приложение «Rusporting»

Конечно, функционал представленных выше Web-приложений очень широк, но все ли функции необходимы для людей, которым просто необходимо найти необходимые им ремонтные услуги в своем городе без других объявлений? Для поиска конкретных услуг, необходим отдельный сайт.

Сайт, разрабатываемый в данной работе ориентирован на узкую специализированную область применения, он не будет загружен лишними функциями. И конечно же, безусловным плюсом разрабатываемого в данной работе Web-сайта является его бесплатность.

## Архитектура Web-приложения и методы его разработки

Web-приложения представляют собой особый тип программ, построенных по архитектуре "клиент-сервер". Особенность их заключается в том, что само Web-приложение находится и выполняется на сервере - клиент при этом получает только результаты работы. Работа приложения основывается на получении запросов от пользователя (клиента), их обработке и выдачи результата. Передача запросов и результатов их обработки происходит через Интернет ( рис. 1.3).

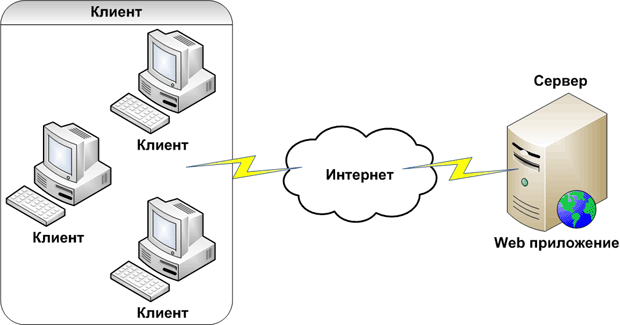


Рис. 1.3. Архитектура Web-приложения

Отображением результатов запросов, а также приемом данных от клиента и их передачей на сервер обычно занимается специальное приложение — браузер (Internet Explorer, Mozilla, Opera и т. д.). Как известно, одной из функций браузера является отображение данных, полученных из Интернета, в виде страницы, описанной на языке HTML, следовательно, результат, передаваемый сервером клиенту, должен быть представлен на этом языке.

На стороне сервера Web-приложение выполняется специальным программным обеспечением (Web-сервером), который и принимает запросы клиентов, обрабатывает их, формирует ответ в виде страницы, описанной на языке HTML, и передает его клиенту.

В процессе обработки запроса пользователя Web-приложение компонует ответ на основе исполнения программного кода, работающего на стороне сервера, Web-формы, страницы HTML, другого содержимого, включая графические файлы. В результате, как уже было сказано, формируется HTML-страница, которая и отправляется клиенту. Получается, что результат работы Web-приложения идентичен результату запроса к традиционному Web-сайту, однако, в отличие от него, Web-приложение генерирует HTML-код в зависимости от запроса пользователя, а не просто передает его клиенту в том виде, в котором этот код хранится в файле на стороне сервера. То есть Web-приложение динамически формирует ответ с помощью исполняемого кода — так называемой исполняемой части.

За счет наличия исполняемой части, Web-приложения способны выполнять практически те же операции, что и обычные Windows-приложения, с тем лишь ограничением, что код исполняется на сервере, в качестве интерфейса системы выступает браузер, а в качестве среды, посредством которой происходит обмен данными, — Интернет. К наиболее типичным операциям, выполняемым Web-приложениями, относятся:

* прием данных от пользователя и сохранение их на сервере;
* выполнение различных действий по запросу пользователя: извлечение данных из базы данных (БД), добавление, удаление, изменение данных в БД, проведение сложных вычислений;
* аутентифицирование пользователя и отображение интерфейса системы, соответствующего данному пользователю;
* отображение постоянно изменяющейся оперативной информации и т. д.

Типы клиентов в системе клиент-сервер:

Толстый клиент обладает полной функциональностью работы с данными сервера, обеспечивает режим многопользовательской работы, предоставляет возможность работы даже при обрывах связи с сервером, имеет возможность подключения к банкам данных без использования сети Интернет, обладает высоким быстродействием.

Однако широкие функциональные возможности "толстого клиента" часто несовместимы с политикой безопасности информационной системы и стоимость его чрезмерно высока.

При работе с ним возникают проблемы с удаленным доступом к данным, выражающиеся в сложности обновления данных, согласования их с другими клиентами и связанной с этим, неактуальностью данных.

Тонкий клиент, thin client в компьютерных технологиях — компьютер или программа-клиент в сетях с клиент-серверной или терминальной архитектурой, где большая часть задач по обработке информации перенесена на сервер и права доступа клиента строго ограничены. Примером тонкого клиента может служить компьютер с браузером, использующийся для работы с Web-приложениями. (картинки)

## 1.6 Требования к разрабатываемому Web-приложению и постановка задач

Все требования к приложениям, в том числе и веб-приложениям, разделяют на

функциональные и нефункциональные. Функциональные требования определяют ту

функциональность системы, которую разработчики должны построить, чтобы

пользователи смогли выполнить свои задачи в рамках своих бизнес-процессов.

Нефункциональные требования представляют собой описание характеристик приложения,

важных для пользователя при работе с системой. Рассмотрим, какие характеристики

задают нефункциональные требования к системе:

* надежность. Требования к надежности приложения определяются условиями функционирования приложения (параметры сервера, максимальное количество пользователей приложения) и допустимыми показателями качества работы системы в этих условиях (время обработки запроса пользователя к системе, количество отказов системы). Таким образом, надежное веб-приложение должно обеспечивать доступ ко всем функциям для пользователя при любых условиях;
* быстродействие приложения. Быстродействие определяется как среднее время обработки запроса пользователя к системе. Максимальным приемлемым временем отклика для веб-приложений считается 5 секунд;
* безопасность. Требование безопасности веб-приложения включает в себя: разграничение прав доступа к функциям и данным каждого компонента веб-приложения может, контроль уровня доступа компонентов и/или пользователей, авторизация и верификация пользователей;
* масштабируемость – это способность системы увеличивать свою производительность при повышенной нагрузке и добавлении ресурсов. Для пользователя масштабируемого веб-приложения должен оставаться незаметным момент (т.е. время отклика системы на запросы пользователя не должно заметно изменяться), когда возрастет нагрузка (например, к приложению получают доступ одновременно еще несколько пользователей), и при изменении конфигурации приложения (например, если на уровень бизнес-логики будет добавлен дополнительный компонент обработки данных).

Ключевой и первостепенный момент при создании Web-приложения - это, пожалуй, постановка задачи. Часто неправильно поставленные задачи или взаимонепонимание ведет к потере времени.

Что бы грамотно составить задачи разрабатываемого Web-приложения, необходимо обозначить, кто будет клиентом данного приложения? Что может получить клиент? Что мы хотим получить в конечном итоге?

Основных пользователей разрабатываемого Web-приложения можно проклассифицировать и разделить на группы.

В первую группу клиентов будут входить заказчики, то есть те люди, которым необходимы услуги в данной сфере.

Пользуясь данным Web-приложением, заказчик будет иметь возможность:

* быстро и удобно получать доступ ко всем возможным услугам по ремонту;
* консультироваться с исполнителем по всем необходимым вопросам;
* записываться на услугу, в необходимый заказчику день.

Ко второй группе пользователей разрабатываемого Web-приложения будут относиться исполнители ремонтных услуг, то есть люди, предоставляющие услуги в данной сфере.

Пользуясь данным Web-приложением исполнитель будет иметь возможность:

* рекламировать свои возможности в данной сфере;
* консультировать заказчиков по всем необходмым вопросам;
* вести учет заказчиков.

В третью группу пользователей Web-приложения будут входить администраторы. Администраторы сайта- это специалисты в области информационных технологий, которые могут поддерживать сайт в технически исправном состоянии.

Администраторы будут иметь возможность:

* создавать форумы;
* применить механизм отфильтровывания нецензурной лексики;
* следить за работой сервера;
* изменять дизайн сайта;
* удалять пользователей сайта;
* контролировать прибыль сайта.

К четвертой группе пользователей будут относиться посетители Web-сайта, которые будут иметь возможность:

* просматривать список услуг;
* консультироваться с исполнителями.

# 2 ПРОЕКТИРОВАНИЕ

## 2.1 Используемые технологии и программные средства

Для разработки Web-сайта используются объектно-ориентированные технологии, которые включают в себя применение языка UML на этапе проектирования Web-сайта. В качестве системы управления базой данных используется MongoDB. В качестве интегрированной среды разработки применяется Sublime 3. Для работы с сервером использовался open-source фреймворк Keystone, основанная на Node.js — программная платформа, основанная на движке V8. На клиенте использовались следующие технологии: React, CSS, Jade (шаблонизатор).

### 2.1.1 Язык UML

Унифицированный язык моделирования (Unified Modeling Language, UML) – это универсальный язык визуального моделирования систем. Хотя чаще всего UML ассоциируется с моделированием ОО программных систем, он имеет намного более широкое применение благодаря свойственной ему расширяемости. UML объединил лучшие современные технические приемы моделирования и разработки программного обеспечения. По сути, язык UML был задуман так, чтобы его можно было реализовать посредством его же инструментальных средств. Фактически это признание того, что большие современные программные системы, как правило, нуждаются в инструментальной поддержке. UML диаграммы легко воспринимаются и при этом без труда генерируются компьютерами. Важно понимать, что UML *не* предлагает нам какой-либо методологии моделирования. Конечно, некоторые методические аспекты подразу меваются элементами, составляющими модель UML, но сам UML предоставляет собой лишь визуальный синтаксис, который можно использовать для создания моделей.

Основная идея UML – возможность моделировать программное обеспечение и другие системы как наборы взаимодействующих объектов. Это, конечно же, замечательно подходит для ОО программных систем и языков программирования, но также очень хорошо работает и для бизнес-процессов и других прикладных задач.

В UML\_модели есть два аспекта:

**•** Статическая структура – описывает, какие типы объектов важны для моделирования системы и как они взаимосвязаны.

**•** Динамическое поведение – описывает жизненные циклы этих объектов и то, как они взаимодействуют друг с другом для обеспечения требуемой функциональности системы.

Эти два аспекта модели UML идут рука об руку, и ни один из них не является по настоящему полным без другого.

Объекты (и классы) будут подробно рассмотрены в главе 7. До тех пор будем считать, что объект единым целым блока данных и поведения. Иначе говоря, объекты содержат информацию и могут выполнять функции.

В UML существует четыре общих механизма, последовательно применяемых ко всему языку моделирования. Они описывают четыре стратегии подхода к моделированию объектов, которые в разных контекстах многократно применяются в UML. Это еще раз убеждает нас в простоте и элегантности структуры UML

Модели UML имеют, по крайней мере, два измерения: графическое, позволяющее визуализировать модель с помощью диаграмм и пиктограмм, и текстовое, состоящее из спецификаций различных элементов модели. Спецификации – это текстовые описания семантики элемента. Семантика элементов

модели фиксируется в спецификациях; без них можно только догадываться, что на самом деле представляет собой элемент.

Набор спецификаций – это суть модели. Спецификации формируют семантический задний план(*semantic backplane*), который объединяет модель и наполняет ее смыслом. Различные диаграммы – это просто представления или визуальные проекции этого плана.

Семантический задний план обычно сопровождается инструментом моделирования UML, предоставляющим доступ, просмотр и изменение спецификаций каждого элемента модели.

UML обеспечивает большую гибкость при создании моделей. В частности, модели могут быть:

**•** сокращенными – некоторые элементы присутствуют в заднем плане, но скрыты в той или иной диаграмме для упрощения представления;

**•** неполными – некоторые элементы модели могут быть полностью пропущены;

**•** несогласованными – модель может содержать противоречия.

Здесь важен сам факт ослабления требований к полноте и согласованности, поскольку, как вы заметите, со временем модель эволюционирует и неоднократно подвергается изменениям. Однако развитие всегда происходит по направлению к согласованным моделям, достаточно полнымдля создания программной системы.

Разработку моделей с помощью UML, как правило, начинают с графической модели, которая позволяет визуализировать систему, а затем по мере ее развития добавляют в задний план все больше и больше семантики. Однако модель можно считать полезной или полной, только если семантика модели *присутствует* в семантическом заднем плане. В противном случае модели не существует, есть просто бессмысленный набор блоков и пятен, соединенных линиями! Кстати, общую ошибку, совершаемую новичками в разработке моделей, можно назвать «смерть от диаграмм»: модель переполнена диаграммами, но недоопределена.

«Плюсы» UML

* Возможность полно описать статическую структуру архитектуры
* Изменчивость, возможность быстрого изменения архитектуры
* Частичная реализация последовательностей и активностей

Использование UML не ограничивается моделированием программного обеспечения. Его также используют для моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

UML позволяет также разработчикам программного обеспечения достигнуть соглашения в графических обозначениях для представления общих понятий (таких как класс, компонент, обобщение (generalization), объединение (aggregation) и поведение) и больше сконцентрироваться на проектировании и архитектуре.

Структуру диаграмм UML можно представить на диаграмме классов UML рис. 1.4.

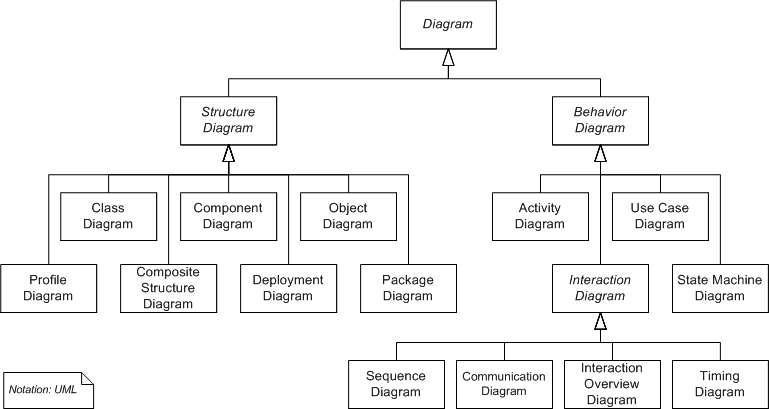


рис. 1.4. Диаграмма классов

Диаграмма классов (Class diagram) — статическая структурная диаграмма, описывающая структуру системы, она демонстрирует классы системы, их атрибуты, методы и зависимости между классами.

Существуют разные точки зрения на построение диаграмм классов в зависимости от целей их применения:

* концептуальная точка зрения — диаграмма классов описывает модель предметной области, в ней присутствуют только классы прикладных объектов;
* точка зрения спецификации — диаграмма классов применяется при проектировании информационных систем;
* точка зрения реализации — диаграмма классов содержит классы, используемые непосредственно в программном коде (при использовании объектно-ориентированных языков программирования).

### 2.1.2 MongoDB

MongoDB  -  документоориентированная система управления базами данных (СУБД) с открытым исходным кодом, не требующая описания схемы таблиц. Написана на языке C++.

Основные возможности:

* Документоориентированное хранение (JSON-подобная схема данных)
* Javascript как язык для формирования запросов
* Динамические запросы
* Поддержка индексов
* Профилирование запросов
* Атомарная операция
* Эффективное хранение двоичных данных больших объёмов, например, фото и видео
* Журналирование операций, модифицирующих данные в базе данных
* Поддержка отказоустойчивости и масштабируемости: асинхронная репликация, набор реплик и распределения базы данных на узлы
* Может работать в соответствии с парадигмой MapReduce
* Полнотекстовый поиск, в том числе на русском языке, с поддержкой морфологии

СУБД управляет наборами JSON-подобных документов, хранимых в двоичном виде в формате BSON. Хранение и поиск файлов в MongoDB происходит благодаря вызовам протокола GridFS. Подобно другим документоориентированным СУБД ([CouchDB](https://ru.wikipedia.org/wiki/CouchDB" \o "CouchDB) и др.), MongoDB не является реляционной СУБД. В СУБД:

Нет такого понятия, как «транзакция». Атомарность гарантируется только на уровне целого документа, то есть частичного обновления документа произойти не может.

Отсутствует понятие «изоляции». Любые данные, которые считываются одним клиентом, могут параллельно изменяться другим клиентом.

В MongoDB реализована асинхронная репликация в конфигурации «ведущий — ведомый» основанная на передаче журнала изменений с ведущего узла на ведомые. Поддерживается автоматическое восстановление в случае выхода из строя ведущего узла. Серверы с запущенным процессом mongod должны образовать кворум, чтобы произошло автоматическое определение нового ведущего узла. Таким образом, если не используется специальный процесс-арбитр (процесс mongod, только участвующий в установке кворума, но не хранящий никаких данных), количество запущенных реплик должно быть нечётным.

### 2.1.3 Keystone

Keystone – CMS, использующая Express - минималистичный и гибкий веб-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений. Для связи с СУБД MongoDB используется ODM Mongoose. Данная CMS предоставляет админ-панель, где администратор может управлять всеми документами MongoDB, такими как, добавление, редактирование, удаление, фильтрация. Имеет встроенную систему типов, для конфигурирования моделей. Предоставляет готовые решения для валидации форм, управление сессиями, отправкой e-mail.

### 2.1.4 JavaScript

JavaScript® (часто просто JS) — это легкий, интерпретируемый, объектно-ориентированный язык с функциями первого класса. Наиболее широкое применение находит как язык сценариев веб-страниц, но также используется и в других программны продуктах, например, node.js или Apache CouchDB. JavaScript это прототипно-ориентированный, мультипарадигменный язык с динамической типизацией, который поддерживает объектно-ориентированный, императивный и декларативный (например, функциональное программирование) стили программирования.

JavaScript является объектно-ориентированным языком, но используемое в языке прототипирование обуславливает отличия в работе с объектами по сравнению с традиционными класс-ориентированными языками. Кроме того, JavaScript имеет ряд свойств, присущих функциональным языкам — функции как объекты первого класса, объекты как списки, карринг, анонимные функции, замыкания — что придаёт языку дополнительную гибкость.

Несмотря на схожий с Си синтаксис, JavaScript по сравнению с языком Си имеет коренные отличия:

* объекты, с возможностью интроспекции;
* функции как объекты первого класса;
* автоматическое приведение типов;
* автоматическая сборка мусора;
* анонимные функции.

### 2.1.5 Node.JS

Node.js – платформа, основанная на JavaScript движке [V8](https://code.google.com/p/v8/), и предлагающая асинхронное API для работы с сетью и диском. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения (при помощи NW.js, AppJS или Electron для Linux, Windows и Mac OS) и даже программировать микроконтроллеры (например, tessel и espruino).

## 2.2 Проектирование с помощью CASE технологий.

Описание функционального назначения программного комплекса представлено с помощью диаграмм CASE- средства Star UML.

### 2.2.1 Диаграмма USE CASE

На рис.2.1 представлена диаграмма вариантов использования. Этот вид диаграмм позволяет создать список операций, которые выполняет система и описывает функциональное назначение системы, то есть то, что система должна делать в процессе своего функционирования. Часто этот вид диаграмм называют диаграммой функций, потому что на основе набора таких диаграмм создается список требований к системе и определяется множество выполняемых системой функций.

Диаграмма вариантов использования состоит из актеров, для которых система производит действие, и собственно действие Use Case, которое описывает то, что актер хочет получить от системы. Актер обозначается значком человечка, а Use Case - овалом.

Каждая такая диаграмма или, как ее обычно называют, каждый Use case – это описание сценария поведения, которому следуют действующие лица (Actors).

Данный тип диаграмм используется при описании процессов автоматизируемой предметной области, определении требований к будущей программной системе. Отражает объекты, как системы, так и предметной области, и выполняемые ими задачи.

В нашем случае на диаграмме отображены четыре действующих лица:

* Клиент
* Инструктор
* Администратор

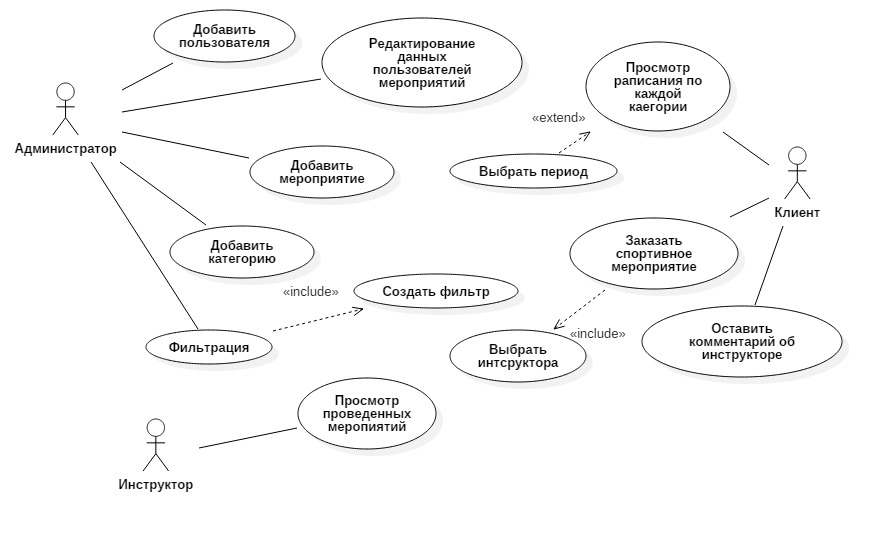
На диаграмме присутствует полное отображение всех выполняемых действующими лицами операций, связи и взаимодействия между ними.

Рис 2.1. Диаграмма use case

Опишем представленную выше диаграмму use case в виде таблицы.

Таблица 2.1

|  |  |
| --- | --- |
| Актер | Действия |
| Клиент | 1. Просматривать расписание по каждому виду спортивного мероприятия 2. Добавлять заявку на проведение спортивного мероприятия 3. Просматривать информацию по каждому из мероприятий 4. Оставлять комментарий об инструкторе |
| Инструктор | Просмотр проведенных мероприятий |
| Администратор | 1. Добавление, редактирование, удаление пользователей 2. Добавление, редактирование, удаление спортивных категорий (информации по ним) 3. Добавление, редактирование, удаление спортивных мероприятий 4. Создание фильтров (по каждому полю любой из моделей) |

### 2.2.2 Sequence диаграммы.

Для моделирования взаимодействия объектов во времени в языке UML используются диаграммы последовательности.

На диаграмме последовательности изображаются исключительно те объекты, которые непосредственно участвуют во взаимодействии и не показываются возможные статические ассоциации с другими объектами. Для диаграммы последовательности ключевым моментом является именно динамика взаимодействия объектов во времени.

В UML каждое взаимодействие описывается совокупностью сообщений, которыми участвующие в нем объекты обмениваются между собой. Сообщение представляет собой законченный фрагмент информации, который отправляется одним объектом другому. Прием сообщения инициирует выполнение определенных действий, направленных на решение отдельной задачи тем объектом, которому это сообщение отправлено.

Сообщения не только передают некоторую информацию, но и требуют или предполагают выполнения ожидаемых действий от принимающего объекта.

Для построения диаграммы Sequence, необходимо определить какие объекты будут принимать участие во взаимодействии. Для этого необходимо провести объектную декомпозицию предметной области. Выделим объекты в нашей предметной области и отметим их обязанности, результат сведем в таблицу.

После того как мы определили основные объекты, представим их взаимодействие на диаграмме Sequence

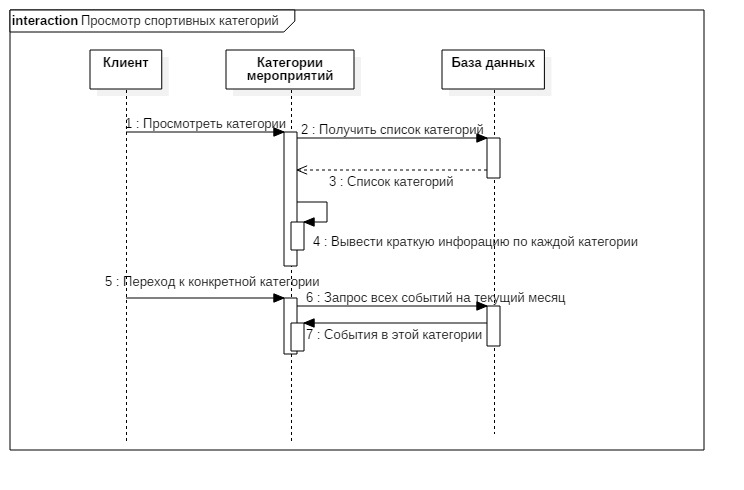


Рис. 2.2. Диаграмма Sequence – Просмотр категорий

Следующий сценарий показывает выбор услуги и связь с исполнителем. Посетитель выбирает нужную услугу из каталога, в результате чего происходит запрос в Базу данных для получения информации об услуге и исполнителе. Если услуга и исполнитель обнаружены, то производится связь с исполнителем, иначе появляется диагностическое сообщение об отсутствии услуг.

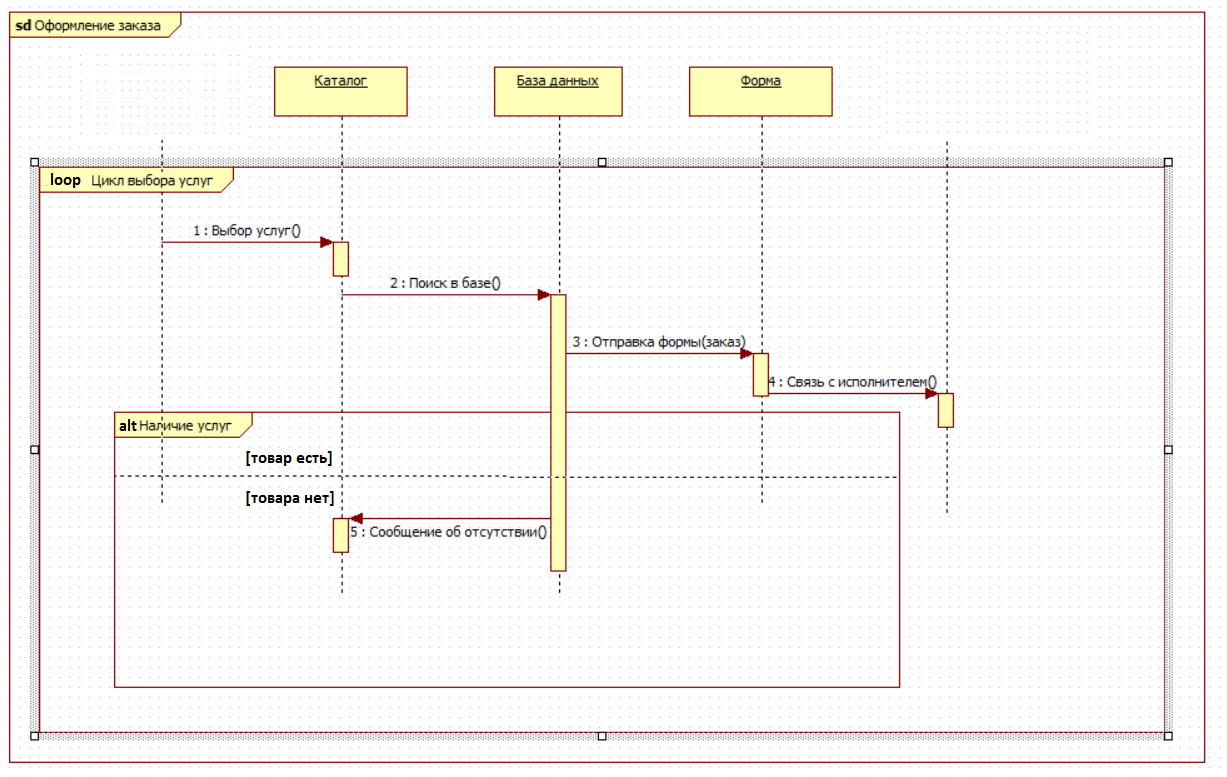
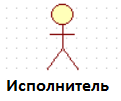
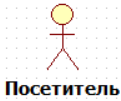


рис. 2.3.Диаграмма Sequence – Связь с исполнителем

### 2.2.3 Диаграмма состояний

Диаграмма состояний (statechart diagram) – диаграмма, которая представляет конечный автомат.

Главное назначение диаграммы состояний - описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение моделируемой системы в течение всего ее жизненного цикла.

Диаграмма состояний представляет динамическое поведение сущностей, на основе спецификации их реакции на восприятие некоторых конкретных событий. Системы, которые реагируют на внешние действия от других систем или от пользователей, иногда называют реактивными. Если такие действия инициируются в произвольные случайные моменты времени, то говорят об асинхронном поведении модели.

Диаграммы состояний чаще всего используются для описания поведения отдельных систем и подсистем. Они также могут быть применены для спецификации функциональности экземпляров отдельных классов, т. е. для моделирования всех возможных изменений состояний конкретных объектов.

Диаграмма состояний по существу является графом специального вида, который служит для представления конечного автомата. Конечный автомат (state machine) - модель для спецификации поведения объекта в форме последовательности его состояний, которые описывают реакцию объекта на внешние события, выполнение объектом действий, а также изменение его отдельных свойств. Основными понятиями, характеризующими конечный автомат, являются состояние и переход. Ключевое различие между ними заключается в том, что длительность нахождения системы в отдельном состоянии существенно превышает время, которое затрачивается на переход из одного состояния в другое. Предполагается, что переход объекта из состояния в состояние происходит мгновенно.

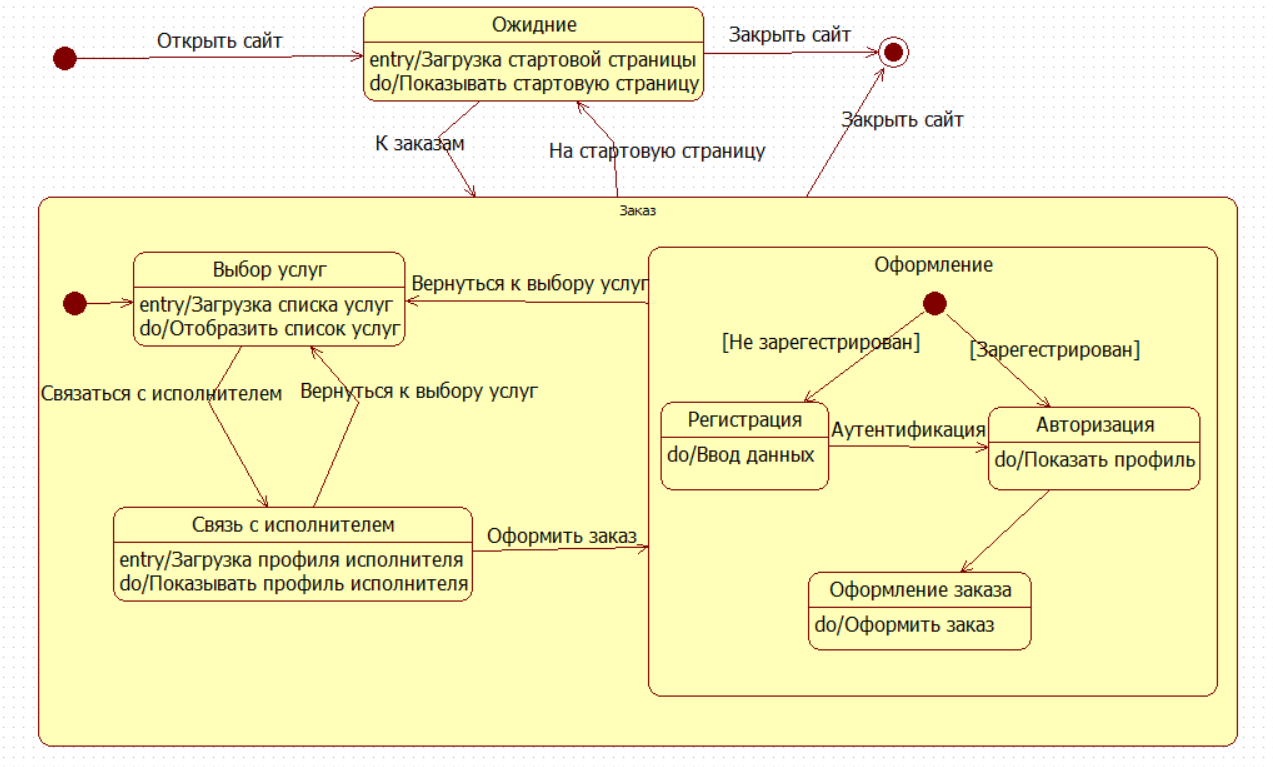


Рис 2.2. Диаграмма состояний

## 2.4 Описание базы данных

Схема системы базы данных (от англ. *Database scheme*) – структура базы данных, описанная на формальном языке, поддерживаемом системой управления базами данных (СУБД). В реляционных базах данных схема определяет таблицы, поля в каждой таблице, а также отношения между полями и таблицами.

Представленная ниже база данных спроектирована и реализована средствами Microsoft Access и содержит 5 таблиц: заказчик, исполнитель, вид услуг, заказы, исполнитель.

На рис 2.3 представлена инфологическая модель схемы базы данных.

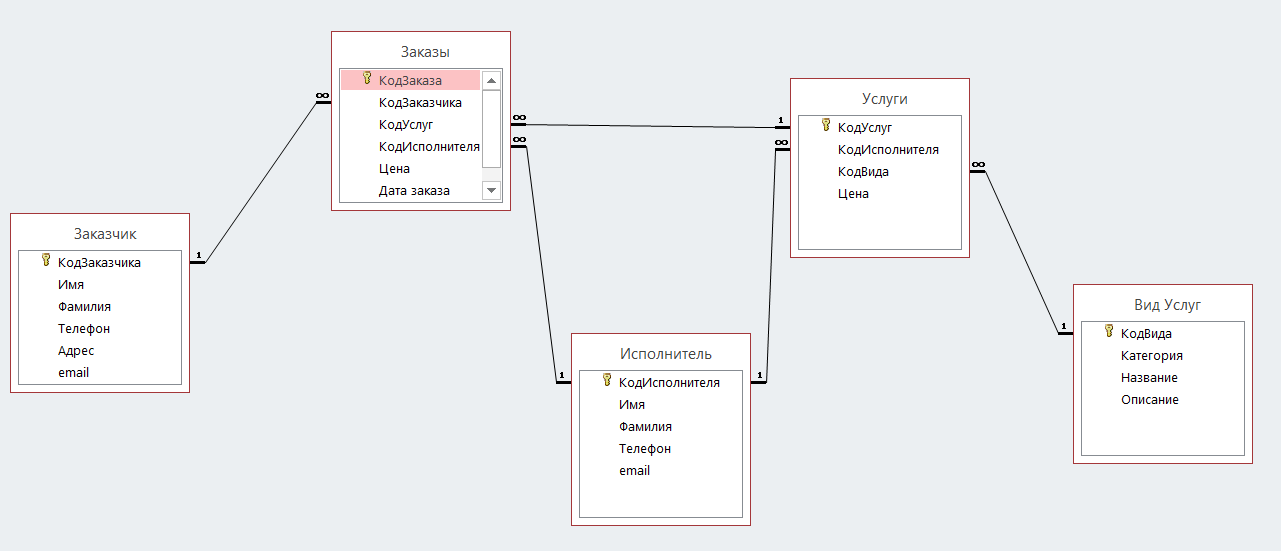


Рис 2.3. Инфологическая модель схемы базы данных

Описание структуры таблицы заказчик.

Таблица 2.3

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| КодЗаказчика | Счетчик | id заказчика |
| Имя | Текстовый | Имя заказчика |
| Фамилия | Текстовый | Фамилия заказчика |
| Телефон | Числовой | Телефон заказчика |
| Адрес | Текстовый | Адрес заказчика |
| e.mail | Текстовый | e.mail заказчика |

Описание структуры таблицы исполнитель.

Таблица 2.4

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| КодИсполнителя | Счетчик | id заказчика |
| Имя | Текстовый | Имя исполнителя |
| Фамилия | Текстовый | Фамилия исполнителя |
| Телефон | Числовой | Телефон исполнителя |
| e.mail | Текстовый | e.mail исполнителя |

Описание структуры таблицы услуги.

Таблица 2.5

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| КодУслуг | Счетчик | id услуги |
| КодВида | Числовой | id вида услуг |
| КодИсполнителя | Числовой | id исполнителя |
| Цена | Денежный | цена за услугу |

Описание структуры таблицы заказы.

Таблица 2.6

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| КодЗаказа | Счетчик | id заказа |
| КодЗаказчика | Числовой | id заказчика |
| КодИсполнителя | Числовой | id исполнителя |
| КодУслуг | Числовой | id услуги |
| Цена | Денежный | цена за услугу |
| Дата заказа | Дата и время | дата заказа |
| Дата исполнения | Дата и время | дата исполнения |

Описание структуры таблицы вид услуг.

Таблица 2.7

|  |  |  |
| --- | --- | --- |
| Название | Тип | Описание |
| КодВида | Счетчик | id вида |
| Категория | Текстовый | категория услуг |
| Название | Текстовый | название услуг |
| Описание | Текстовый | описание услг |

## 2.5 Разработка

Для того, чтобы создать свой сайт, необходимо выделить краткую последовательность шагов, представленную ниже.

1) Выбор CSS шаблона.

2) Корректировка СSS шаблона. Создание графики.

3) Разделение страницы на блоки (логические части).

4) Создание новых страниц и наполнение их контентом.

5) Создание БД.

6) Реализация администраторской страницы.

Рассмотрим каждый шаг более подробно.

### 2.5.1 Выбор CSS шаблона.

В последнее время шаблоны CSS стали очень популярными. Шаблоны с CSS – это готовые дизайны сайтов, которые содержат в исходниках необходимые CSS-файлы со стилями для редактирования и быстрого изменения внешнего вида.

Особенность готовых шаблоны с CSS – это отличная возможность создать удобный и легко настраиваемый сайт с гибким дизайном.

Свой шаблон, на основе которого создавалось Web-приложение, был скачен с сайта

<http://www.freecsstemplates.org>

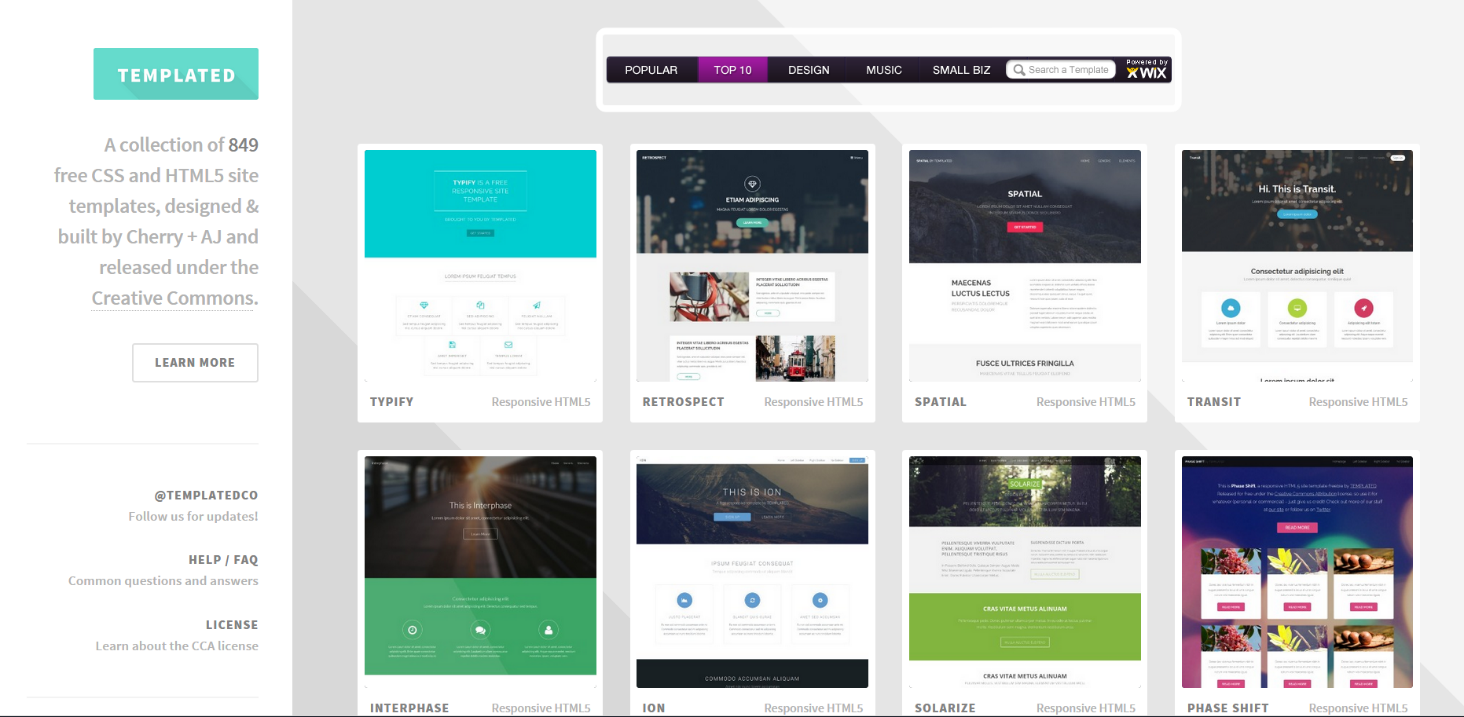


рис. 2.4. Сайт css шаблонов

### 2.5.2 Корректировка СSS шаблона. Создание графики.

Для того, что бы откорректировать css шаблон, необходимо внести изменения в файл style.css.

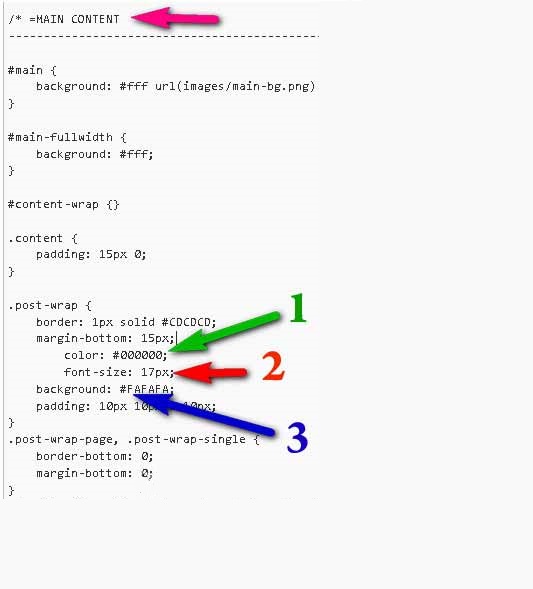


рис. 2.5. Корректировка css шаблона

Main Content переводится как — главное содержание

Стрелка №1 указывает на строку, которая отвечает за цвет шрифта

Стрелка под №2 указывает на строку, которая отвечает за размер шрифта в статьях (эту строку пришлось добавить, потому что ее вообще не было)

Стрелка под №3 указывает на строку, которая отвечает за цвет фона в тексте.

Создание графики подразумевает собой добавление изображений на Web-страницы. Для этого, необходимо изначально изменить их размер.

### 2.5.3 Разделение страницы на блоки.

Разобьём страницу на 14 логических частей:

1. admindata (содержит данные о пользователе admin)
2. car (логическая часть-ремонт машин)
3. clothes (логическая часть-ремонт одежды)
4. footer ( содержит данные о нижней панели)
5. ifadmin (содержит данные об авторизации пользователя admin)
6. ifenter (содержит данные об авторизации остальных пользователей)
7. leftbar (содержит данные о левой панели)
8. menu ( содержит данные о меню Web-приложения)
9. meta (содержит метаданные)
10. reg (содержит данные о процедуре регистрации заказчиком)
11. regisp ((содержит данные о процедуре регистрации исполнителем)
12. rightbar (содержит данные о правой панели)
13. roominf ( логическая часть-ремонт квартир)
14. teknikint (логическая часть-ремонт одежды)

Для того, чтобы разбить страницу на логические части, необходимо создать новую папку «parts», в которой мы будем хранить файлы с логическими блоками. Что бы соединить эти части с главным файлом, используется команда <?php require\_once("parts/название файла");?>

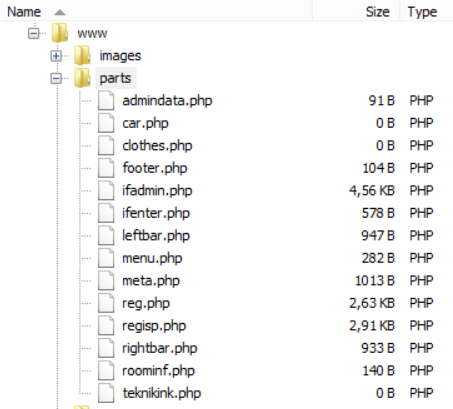


рис. 2.6. Логические части

### 2.5.4 Создание новых страниц и наполнение их контентом.

Типичная веб-страница представляет собой текстовый файл в формате HTML, который может содержать ссылки на файлы в других форматах (текст, графические изображения, видео, аудио, мультимедиа, апплеты, прикладные программы, базы данных, веб-службы и прочее).

Создаем остальные страницы на основе файла index.php и сохраняем в папку www.

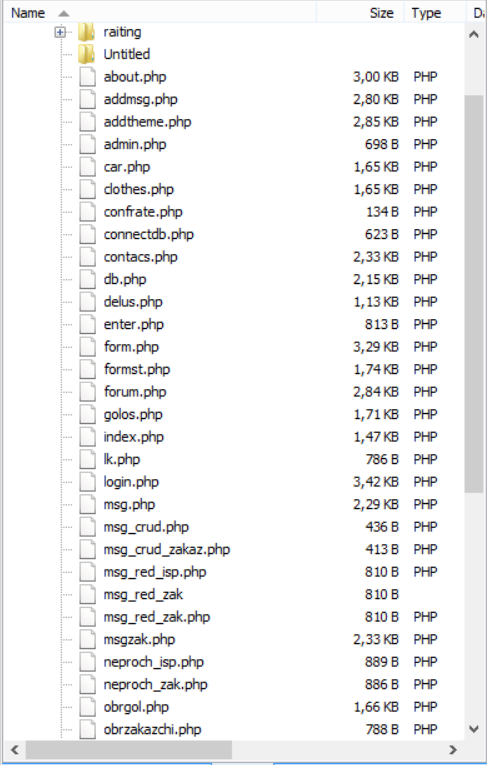


рис. 2.7. а Новые страницы и наполнения

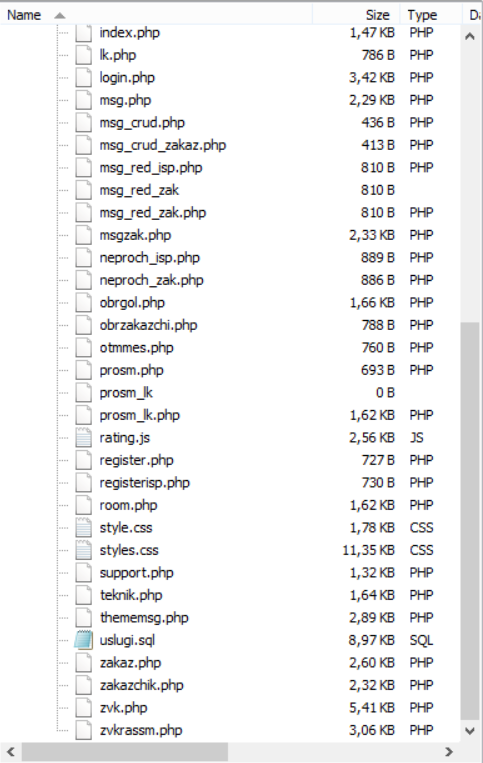


рис. 2.7. б Новые страницы и наполнения

### 2.5.5 Создание БД

Создание баз данных и использование их в своих веб страницах – это достаточно мощный инструмент, решающий множество задач веб разработки.

Что бы создать базу данных, необходимо войти на страницу администрирования базы данных MySQL. Для того, чтобы зайти на страницу администрирования баз данных, в командной строке Вашего браузера введите следующий адрес: localhost/tools/phpmyadmin/.

Для того, чтобы создать новую базу данных нажмем на верхнюю вкладку «Базы данных» и перед нами на центральном поле откроется список всех имеющихся баз данных MySQL. Нам же нужно создать новую. Для этого в поле «Создать базу данных» впишем название создаваемой базы и нажмем на кнопку «Создать».

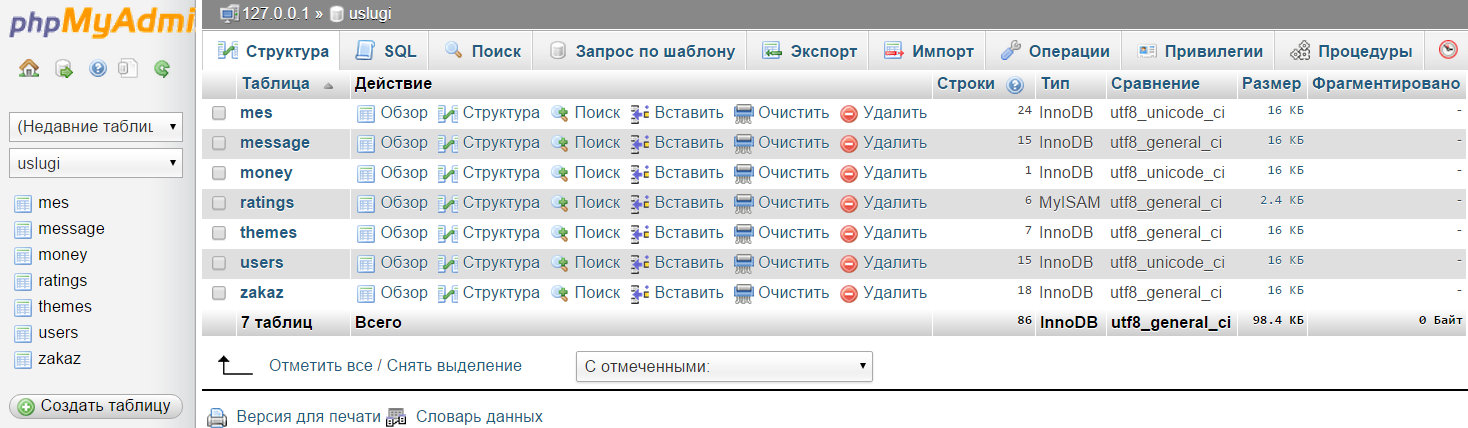
После чего, создаются таблицы.

рис. 2.8. Таблицы

### 2.5.6 Реализация администраторской страницы.

Административная часть сайта представляет собой отдельную часть сайта. Вход в которую возможен только пользователям имеющим доступ.

По сути это дополнительный сайт, со своим дизайном, функциями. Задачи к админке ставятся исходя из функционала сайта.

Так если на сайте присутствует модуль регистрации пользователей, то в админке существует модуль учета этих пользователей, как правило, с возможностью ручного изменения. Например, сделать активным, не активным, удалить или добавить полномочия.

Файл admin.php - сама админка, к которому привязывается файл ifadmin.php (Показывает, что на странице вы вошли как админ, в нем же формируются ссылки на личные кабинеты пользователей) Начало:

<?php

if($\_COOKIE["pass"]!=="54321"){

sleep(1);

if(isset($\_POST["pass"])){

setcookie("pass",$\_POST["pass"],);

die("Обновите страницу");

}

?>

<html><head><title>Админка</title></head><body>

<form method="post">

<input type="password" name="pass" value="">

<input type="submit" name="submit" value="54321">

</form></body></html>

<?php

exit();}

?>

## 2.6 Итоговая реализация сайта

1. Главная страница

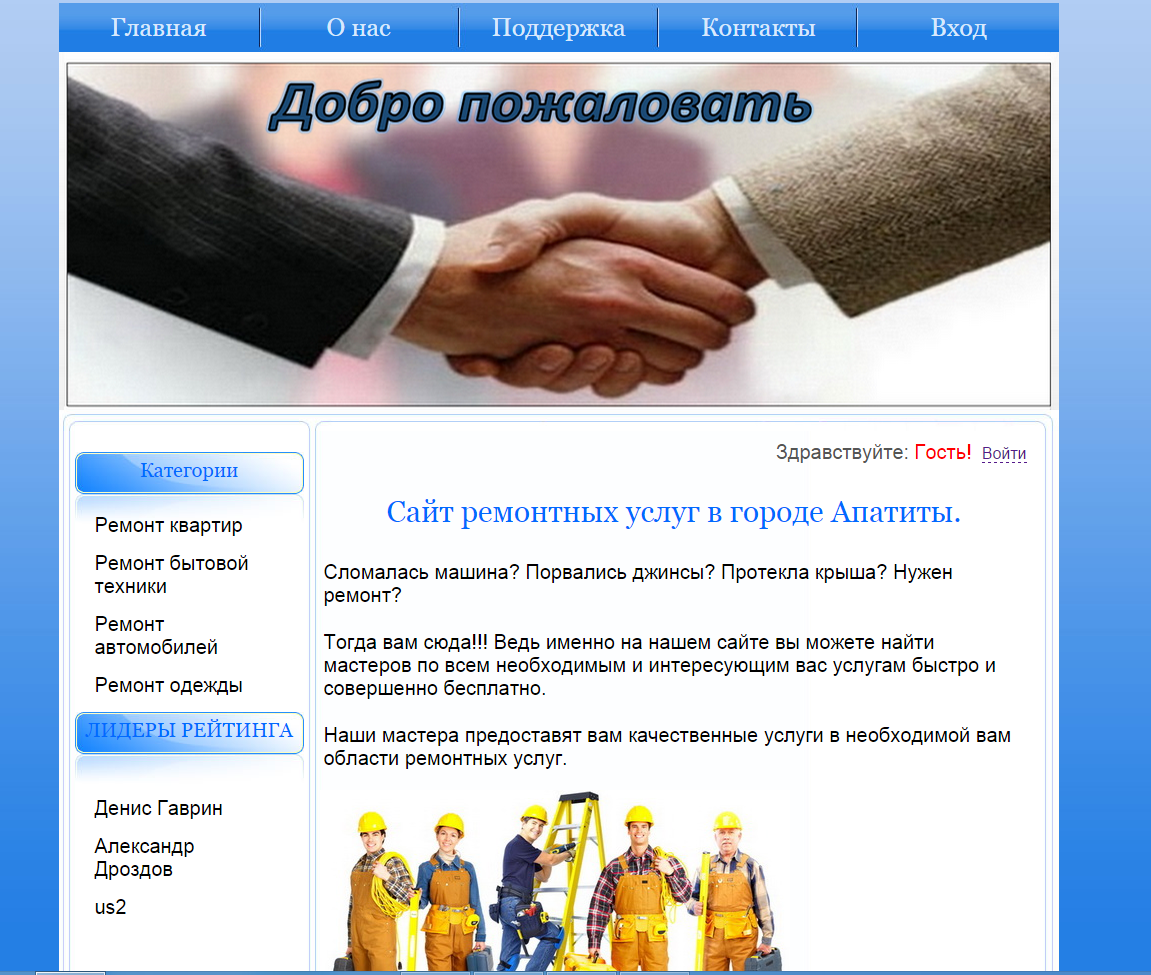


рис. 2.9. Главная страница сайта

2) Форма для входа

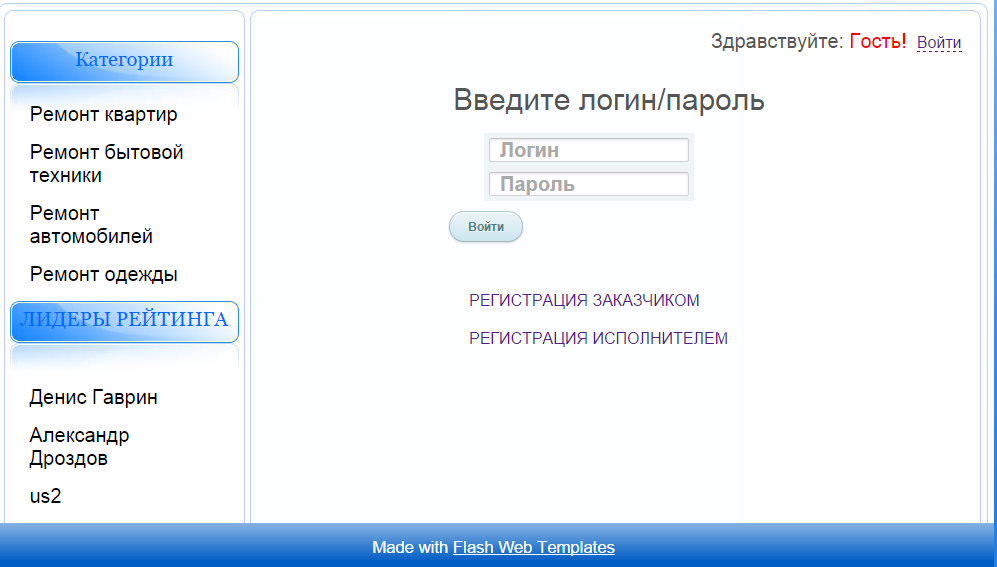


рис. 2.10. Форма для входа

3) Форма для регистрации исполнителем

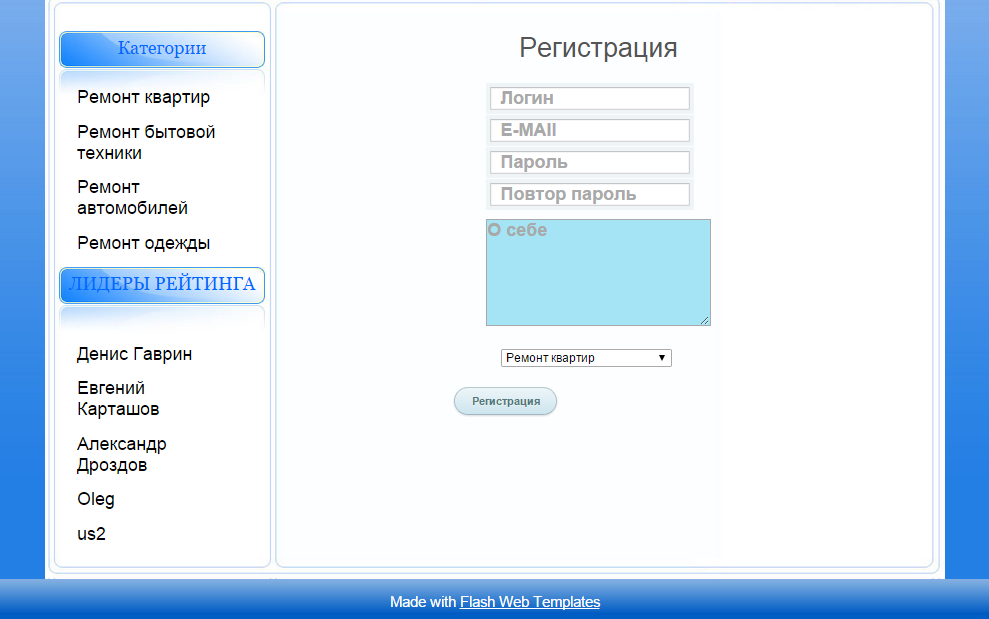


рис. 2.11.Форма для регистрации исполнителем

Для заказчиков форма остается та же, только отсутствует пункт с выбором категорий и окно «О себе».

4) Список исполнителей, в одной из категорий.

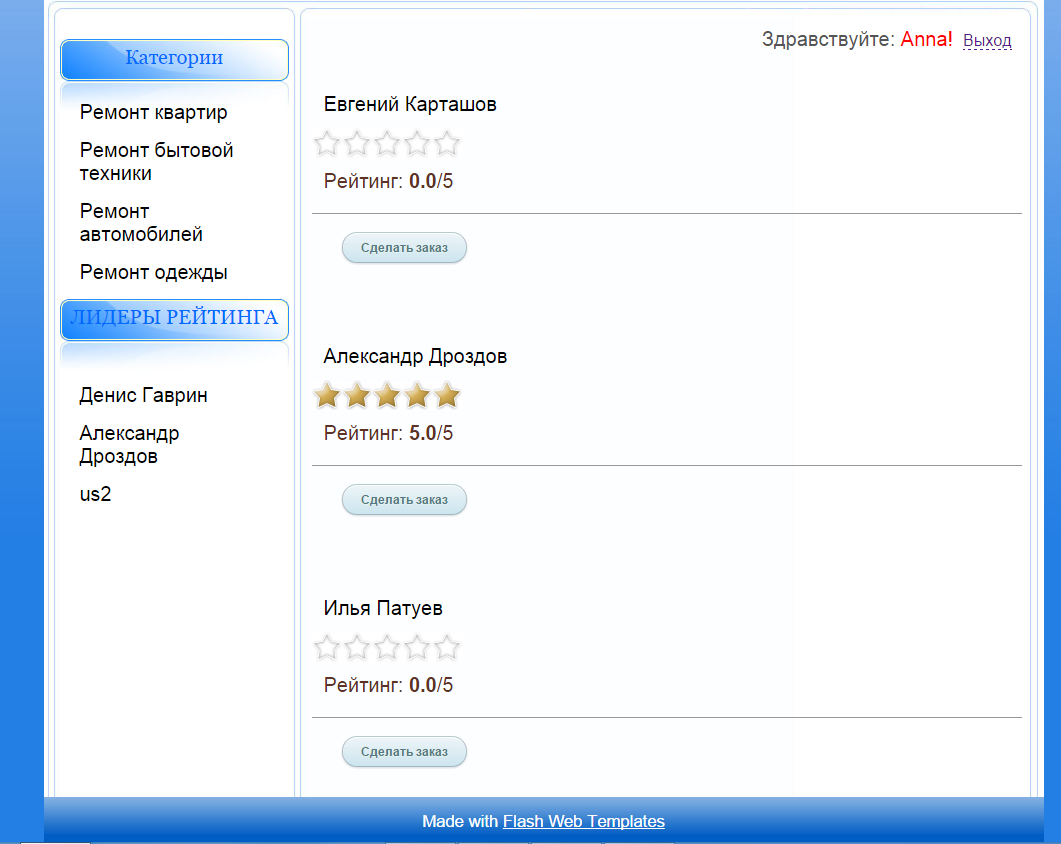


рис. 2.12. Список исполнителей, в одной из категорий

Здесь представлен список всех исполнителей, относящихся к одной из категорий ремонта.

5) Оформление заказа

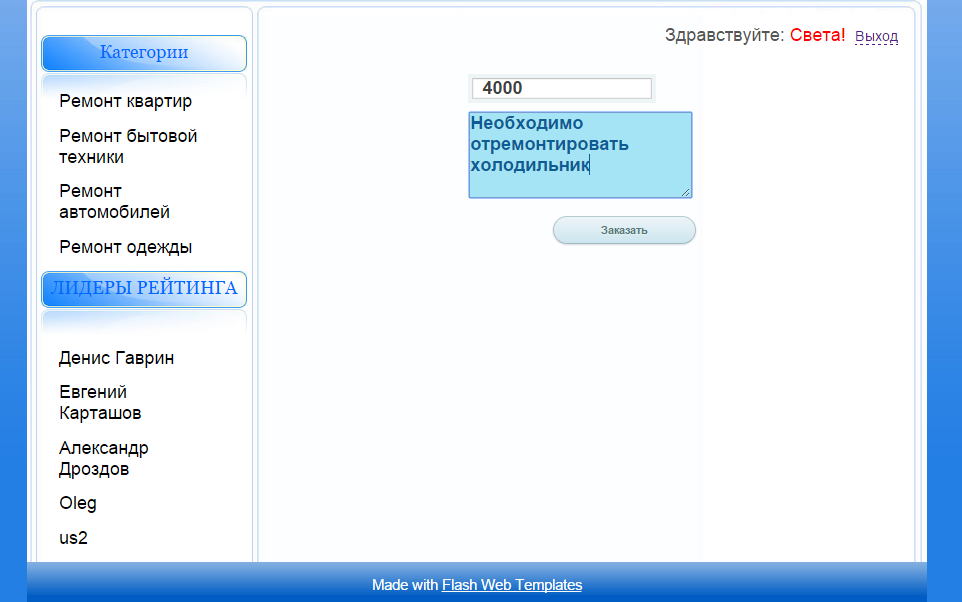


рис. 2.13. Оформление заказа

6) Личный кабинет заказчика

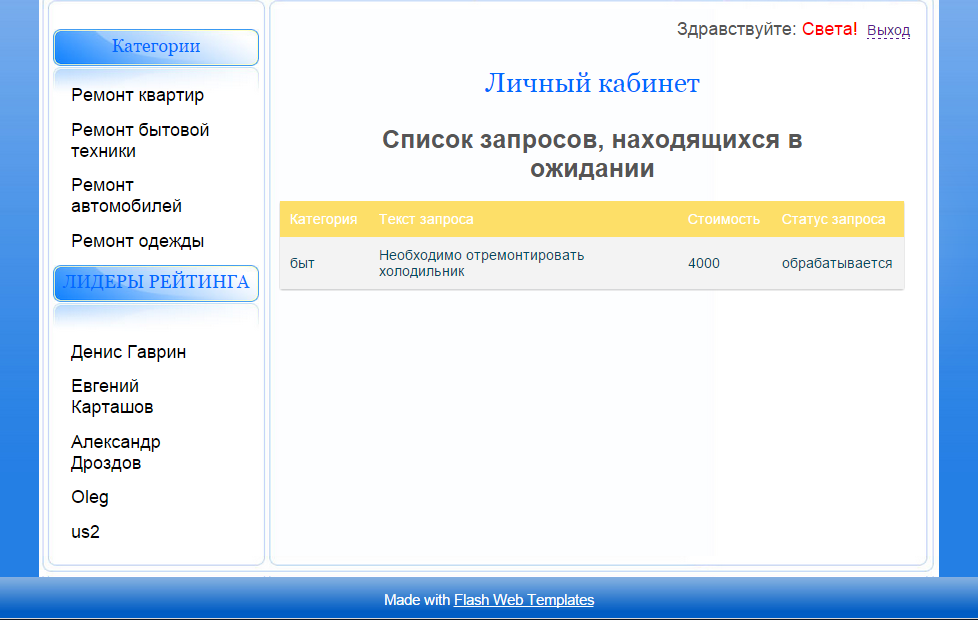


рис. 2.14. Личный кабинет заказчика

7) Личный кабинет исполнителя

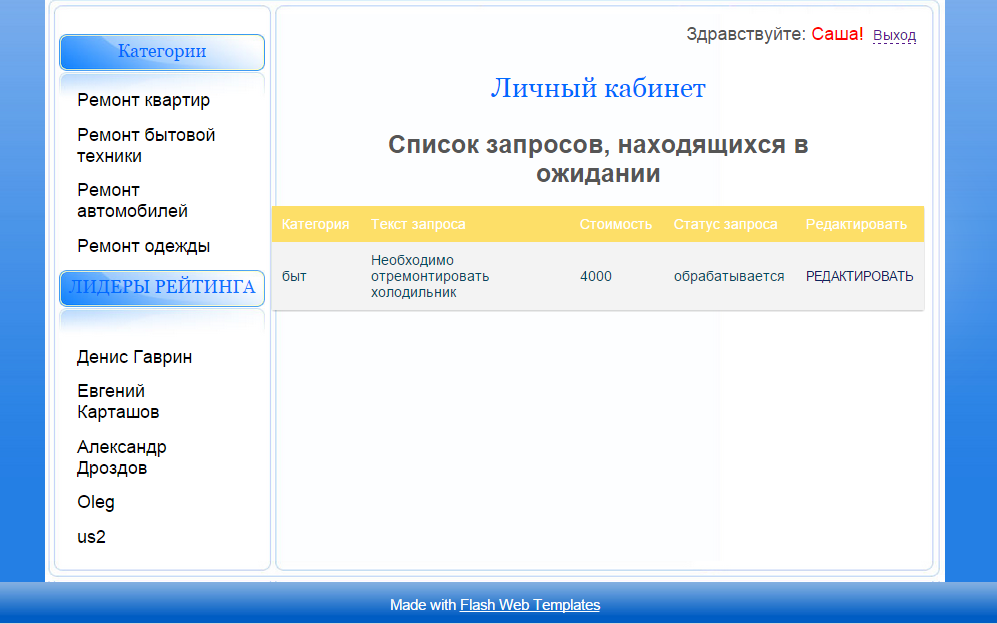


рис. 2.15. Личный кабинет исполнителя

8) Редактирование заказа исполнителем

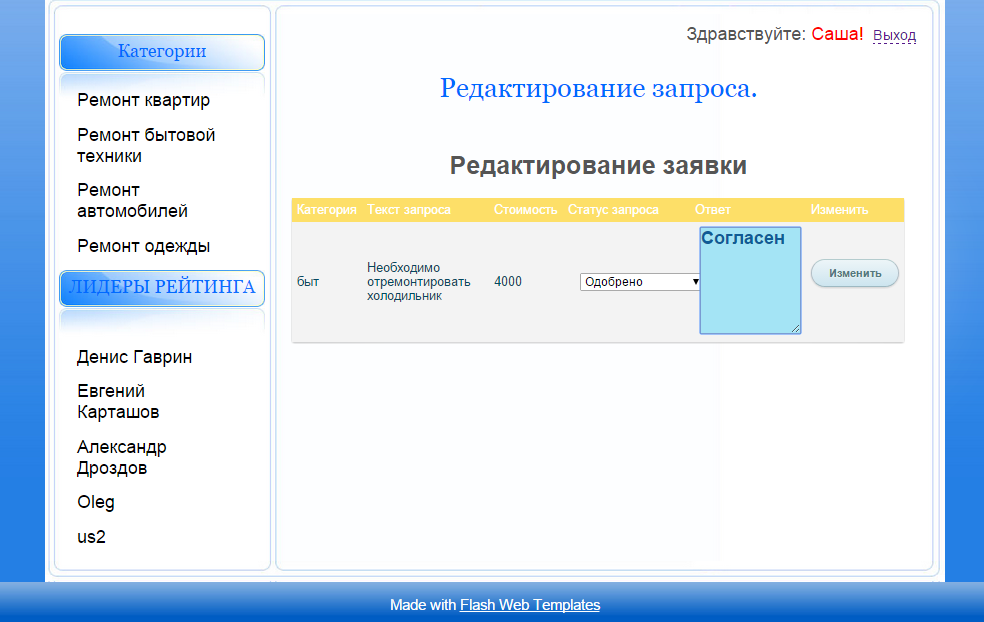


рис. 2.16. Редактирование заказа исполнителем

9) Отправка сообщения от исполнителя к заказчику

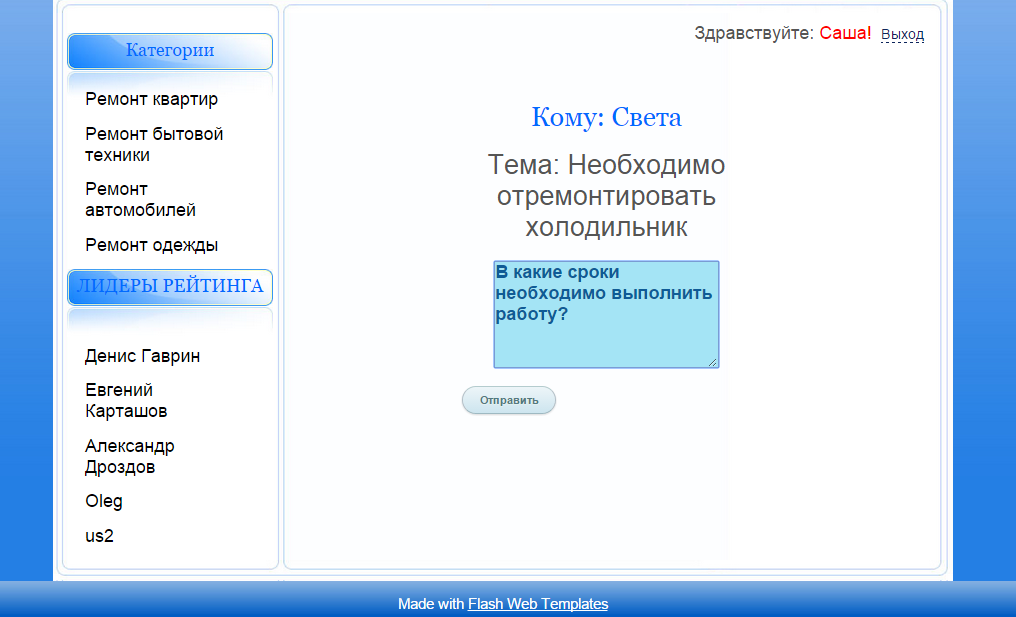


рис. 2.17. Отправка сообщения от исполнителя к заказчику

10) Отправка сообщения от заказчика к исполнителю

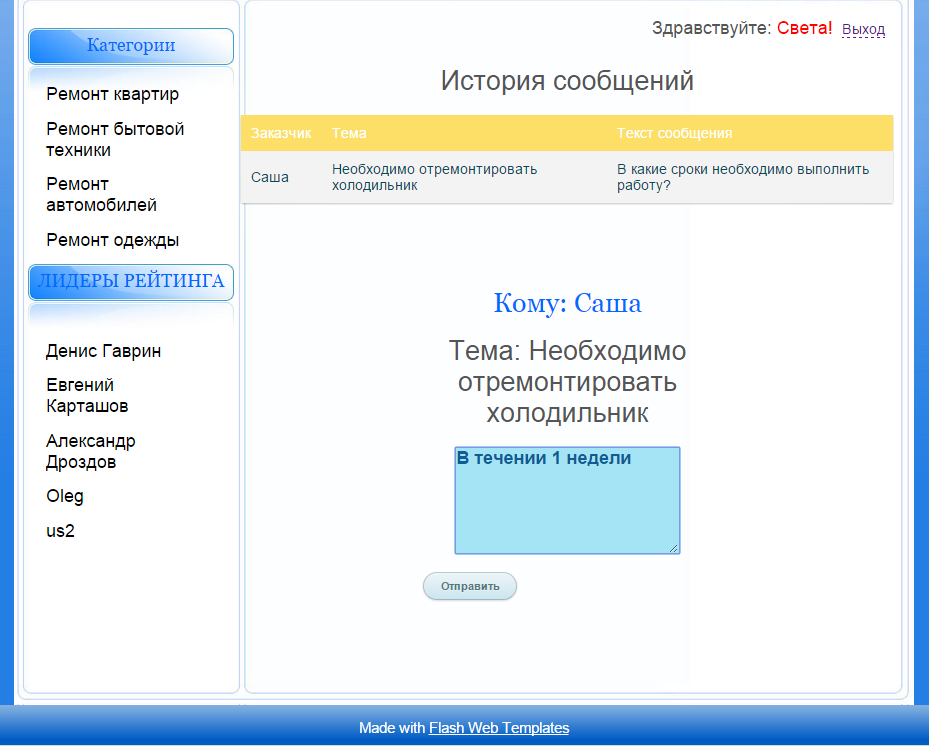


рис. 2.18. Отправка сообщения от заказчика к исполнителю

11) Личный кабинет администратора

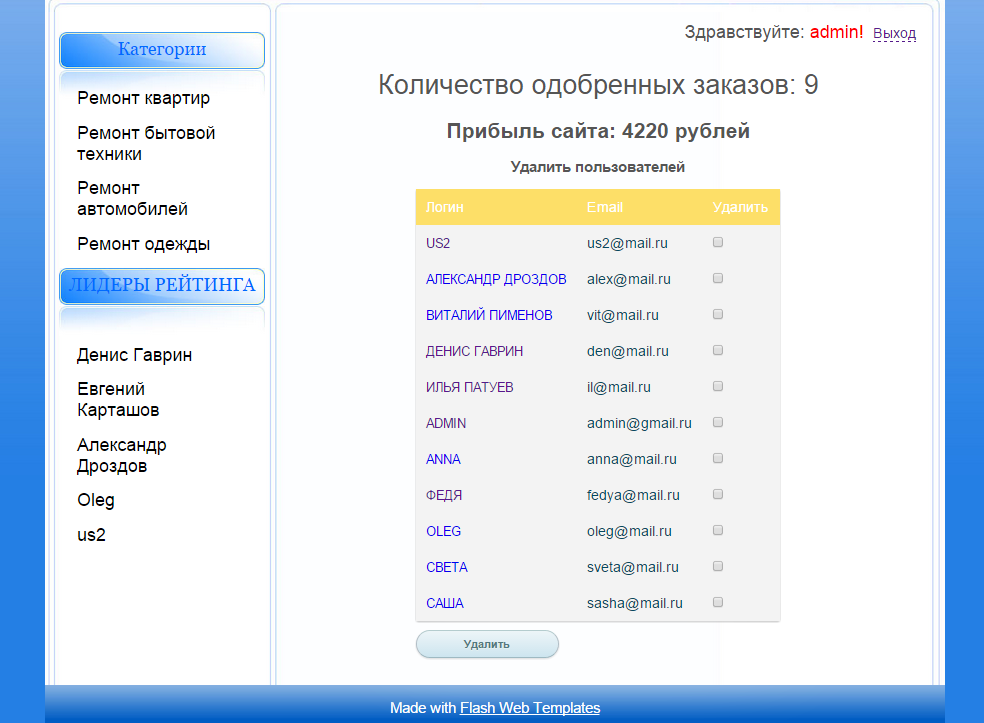


рис. 2.19. Личный кабинет администратора

12) Форум поддержки

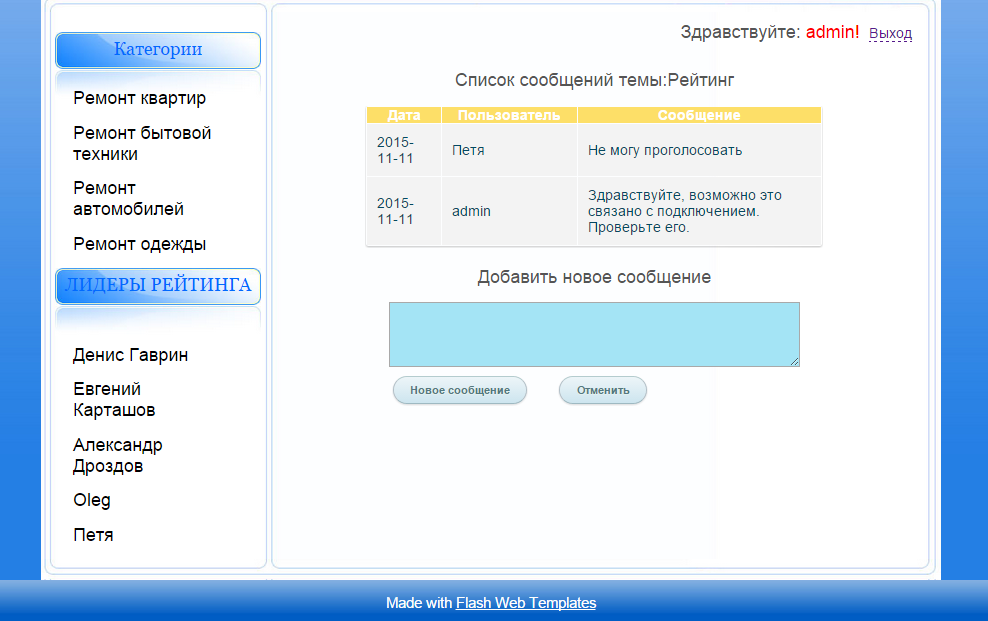


рис. 2.20. Форум поддержки.

# ЗАКЛЮЧЕНИЕ

В ходе данного курсового проекта было разработано Web-приложение по предоставлению коммерческих услуг в городе Апатиты.

В результате, пользуясь данным Web-приложением, заказчик имеет возможность:

* регистрироваться на сайте;
* оформлять заказ;
* выставлять рейтинг исполнителю заказа;
* отправлять сообщения исполнителю;
* создавать форумы поддержки, и получать там консультации.

Исполнитель имеет возможность:

* регистрироваться на сайте;
* рекламировать свои услуги;
* одобрять или отклонять заказы;
* отправлять сообщения заказчику;
* давать и получать консультации на форуме поддержки.

Администратор имеет возможность:

* просматривать личные кабинеты всех пользователей Web - приложения;
* вести учет одобренных сделок, и прибыли сайта;
* удалять пользователей Web – приложения;
* создавать форумы поддержки;
* давать консультации пользователям сайта.

Посетитель имеет возможность:

* просматривать категории услуг и исполнителй;
* просматривать форумы поддержки.

Для реализации такого Web-приложения были решены следующие задачи:

1) изучена предметная область и существующие аналоги;

3) изучена архитектура Web-приложения и методы его разработки;

4) изучены требования к разрабатываемому Web-приложению;

5) разработаны модели UML и структура базы данных;

6) разработано и протестировано Web-приложение по предоставлению ремонтных услуг;

Данный проект имеет большой потенциал, в дальнейшем планируется разработка дополнительных функций, таких как:

* интернет магазин;
* интеллектуальный помощник пользователя по навигации сайта;
* оплата предоставляемых услуг;
* расширение сбора статистики по Web-приложению.

##### СПИСОК ЛИТЕРАТУРЫ

1. Дмитрий Котеров, Алексей Костарев PHP. В подлиннике. — СПб.: «БХВ-Петербург», 2005. — С. 1120. — ISBN 5-94157-245-Х

2. Костарев А. Ф. PHP 5. — СПб.: «БХВ-Петербург», 2008. — С. 1104. — ISBN 978-5-9775-0315-0

3. Соколов Сергей Александрович. HTML и CSS в примерах, типовых решениях и задачах. Профессиональная работа. — М.: Вильямс, 2007. — С. 416. — ISBN 978-5-8459-1192-6

4. Фримен Эрик, Фримен Элизабет. Изучаем HTML, XHTML и CSS = Head First HTML with CSS & XHTML. — 1-е изд. — М.: «Питер», 2010. — С. 656. — ISBN 978-5-49807-113-8

5. Эд Титтел, Мэри Бурмейстер. HTML 4 для «чайников» = HTML 4 For Dummies. — 5-е изд. — М.: «Диалектика», 2006. — С. 368. — ISBN 0-7645-8917-2

6. Якоб Нильсен Веб-дизайн. — СПб: Символ-Плюс, 2003. — 512 с. — ISBN 5-93286-004-9

7. Brian Lozier. Template engines.

8. Fabien Potencier. Templating Engines in PHP, Templating engines in PHP - Follow-Up.

9. http://ru.wikipedia.org