ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА МАТЕМАТИЧЕСКОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ CRM ПРИЛОЖЕНИЯ ДЛЯ ЮРИДИЧЕСКОГО ЦЕНТРА

Выпускная квалификационная работа
обучающегося по направлению подготовки
02.03.02 Фундаментальная информатика и информационные технологии очной формы обучения, 4 курса группы 07001301
Карасева Дмитрия Антоновича

Научный руководитель
к.т.н., доцент
Бурданова Е.В.

БЕЛГОРОД 2017

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В современном мире надобность в автоматизации разных процессов стала обычным явлением. На данный момент трудно представить себе складской или бухгалтерский учет без использования специализированных программ, продавцы пользуются специальными приложениями для обработки и отправки заказа прямо с мобильного телефона или планшета, довольно много заказов приходит уже в готовом к обработке виде, но при этом работа с клиентами, почему-то часто ведется без использования.

Что же получается, если отдел продаж не использует систему учета? Все менеджеры работают наиболее удобным для себя образом, фиксируют звонки, другие виды взаимоотношений с клиентами так же на свое усмотрение.

Канал продаж, откуда пришел клиент также не фиксируется, так же становиться сложно понять кто из менеджеров занимается заявкой. В итоге получается, что ведется учет уже оплаченных заказов, а эффективность работы отдела продаж ставиться под сомнение, так же становиться не понятно, учитываются ли абсолютно все входящие звонки, производится ли работа с уже имеющейся базой контактов, обычно определить невозможно.

Также, если сотрудник увольняется либо уходит на больничный, его необработанные Лиды, незаконченные переговоры с клиентами компания может потерять, что крайне невыгодно в целом для компании. Единственный верный выход из данной ситуации — внедрение CRM-системы для автоматизации и стандартизации управления отношений с клиентами.

Данное решение поможет:

- Стандартизировать базу клиентов и контрагентов;
- С высокой эффективностью контролировать качество работы;
- Получить аналитику и полную статистику эффективности работы с клиентами;

- Планировать дальнейшую работу и разрабатывать высокоэффективную стратегию для развития предприятия.

Актуальность и значимость использования подобных систем в своей деятельности объясняется высокой конкуренцией, для России это особенно важно в связи с тем, что эпоха дефицита прошла, у потенциального потребителя появился выбор, в какую компанию обратиться, где приобретать товары и услуги. Если организация стремится преуспеть, необходимо становиться клиент ориентированной.

Выпускная квалификационная работа состоит из следующих частей:

В первой главе описывается анализ предметной области, анализ деятельности предприятия, выбор инструментальных средств, программного обеспечения;

В второй главе описывается реализация базы данных, бизнес логики;

В третьей главе описывается создание ПО CRM приложения, интерфейса и производиться тестирование разработанной системы.

Выпускная квалификационная работа состоит из введения, трех глав, включающих 14 параграфов, заключения, списка литературы из 14 наименований и 9 приложений. В тексте дипломной работы содержится 23 рисунка. Общий объем работы 59 листов.

ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Довольно часто руководители компаний, предприятий говорят о снижении продаж, спаде количества заказов, так же многие компании сокращают персонал. Время когда желающих купить товар было больше чем товара подошло к концу, бурный рост экономики закончился.

Современным компаниям адаптироваться к новым условиям которые диктует рынок стало сложнее. Покупатели стали требовать более качественного предоставления услуги за меньшую плату. Спрос среди покупателей стал смещаться в сторону более качественных и в то же время дешевых предложений. Так же конкуренция ростет за счет прихода на Российский рынок зарубежных компаний.

Для адаптации к данным событиям компаниям необходимо изменить принцип своей работы с клиентами. На первый план для руководителя предприятия выходят ключевые задачи удержания клиентов, повышение эффективности и снижения затрат предприятия.

Главной стратегией для успешного развития и дальнейшего существования предприятия становится эффективное управление взаимоотношениями с клиентами. Предприятия ориентируются на усовершенствование взаимоотношений с покупателями в частности из-за усиления конкуренции на рынке, повышения требований к представляемым товарам и услугам, сервису, сильным ослаблением эффективности обычных маркетинговых средств, новыми технологиями для выстраивания взаимоотношений между клиентами и исполнителями. Понимание и грамотное использование потребностей, удовлетворение запросов клиентов, открывает для компании множество новых возможностей продаж своих услуг и позволяет компании постоянно оставаться игроком в долгосрочном периоде даже в условиях высококонкурентой среды.

Зарубежный опыт доказывает высокую эффективность концепции CRM. Данная концепция позволяет «интегрировать» покупателя в

организацию - фирма получает возможность собрать и использовать множество информации о клиентах максимально эффективно. На основе этих данных компания может построить даленьнейшую стратегию развития, которая будет взаимодействовать с всеми аспектами её деятельности включая маркетинг, продажи, производство, обслуживание и прочего.

На данный момент CRM системы являются инструментом №1 для успешного развития и поддержания бизнеса. В 2000-х годах CRM-системамы использовало около 35% руководителей, за прошедшие годы данная цифра выросла более чем в 2 раза. Сейчас CRM систему используют больше 80% руководителей.

Рынок CRM решений растет из года в год, по разным оценкам на 50% в год. Использование в бизнесе CRM системы экономит средства компании, открывает новые возможности для монетизации клиентов, позволяет автоматизировать часть бизнес процессов, делает бизнес более эффективным и прибыльным.

1.1 ПОСТАНОВКА ЗАДАЧИ

Невзирая на кризис, рынок программных решений для бизнеса продолжает свое развитие. Текущая экономическая обстановка вносит масштабные коррективы в развитие ИТ отрасли. Основной задачей для предприятий становится оптимизация бизнеса. В связи с данным событием бизнесу требуется оптимизация услуг, внутренних процессов, поэтому рационально использовать систему управления взаимоотношениями с клиентами, которую часто называют CRM. Она предназначена для выполнения следующих задач:

- оптимальное использование ресурсов;
- улучшенное обслуживание клиентов;
- эффективный учет бизнес процессов;
- отчеты о результатах;

- упрощение взаимодействия менеджеров с клиентами.

Выгода от использования CRM системы очевидна. Первое, хранение информации в одном месте позволяет оперативно и эффективно предоставить информацию о взаимодействии как с прошлыми, так и с текущими клиентами. Второе, анализ данных о клиентах, которые накопились по мере использования, позволяет повысить эффективность производства, снизить затраты. Учитывая данные факторы сделан вывод о том, что создаваемая информационная система позволит увеличить производительность компании, персонала, позволит эффективно наблюдать за работой сотрудников, формировать отчеты, делать обдуманные решения.

1.2 АКТУАЛЬНОСТЬ ПОСТАВЛЕННОЙ ЗАДАЧИ

Для постоянного роста уровня работы бизнеса и его коммерческой составляющей необходимо использовать новые техники управления процессами.

На данный момент актуальной остается задача автоматизации процессов в компании, по обслуживанию клиентов, продаже товаров и услуг.

Учитывать и постоянно контролировать работу менеджеров, исполнителей, заказчиков, которые обрабатывают, исполняют, заказывают, множество услуг ежедневно, возможно только с помощью компьютеров. Ручной учет услуг исполнителей, заказов заказчиков не обеспечивает должного уровня эффективности работы. Система ручного учета выполнения договоров, как правило, не позволяет оперативно принимать решения, воздействовать на исполнителей, которые в свою очередь нарушают условия сотрудничества, что в свою очередь приводит к срывам сроков, недовольствам с стороны клиентов, перебоям с представлениями услуг заинтересованным лицам, потерям клиентов. Именно для этих целей существуют CRM системы, которые позволяют оперативно обрабатывать информацию, управлять и автоматизировать процессы компании, что обеспечивает должный уровень сервиса, освобождает время для работы с

исполнителями, улучшает производительность труда сотрудников, повышает лояльность клиентов за счет более сжатых сроков представления услуг.

Таким образом, в дипломной работе необходимо разработать web-приложение об организационной структуре, в котором должны быть описаны все организационные звенья с указанием направления их деятельности, функции, которые они выполняют, показано распределение видов продуктов, затрат и прибыли, закупки и продажи товаров. Также должна быть обеспечена возможность ведения всей учетной информации, загрузки и хранения документов предприятия.

1.3 АНАЛИЗ ДЕЯТЕЛЬНОСТИ ПРЕДПРИЯТИЯ

Юридический центр, занимается предоставлением юридической помощи, сопровождением, обработкой документов в вопросах регистрации, временной регистрации, прописки в Москве и Московской области, оформлении загранпаспортов, паспортов РФ, снятия жилья.

Однако для успешного функционирования субъектов даже малого и среднего бизнеса, помощь квалифицированных специалистов просто необходима. Хорошо организованная работа юридического центра может оказать неоценимую помощь при решении спорных ситуаций, урегулировании конфликтов.

Одним из ключевых моментов при формировании кадровой политики является принятие решения, что именно выбрать: юридический консалтинг, или же формирование отдела на предприятии? В данном случае придется учесть не только специфику хозяйственной деятельности организации, но и множество других нюансов. Сегодня все большую популярность обретает такое понятие, как корпоративный юрист. Что под ним подразумевается?

“Корпорация” означает объединение лиц по профессиональному признаку, которые зарегистрировали форму собственности и функционируют в качестве юридического лица. Как правило, в данном случае речь идет про акционерное общество. Таким образом, получается, что далеко не каждое

предприятие можно отнести к корпорации, а значит, более грамотно будет использовать такое понятие, как “юрист предприятия”.

Для субъектов малого и среднего бизнеса контракта с юридической организацией будет вполне достаточно. В любом случае ясно одно, что без помощи квалифицированных специалистов про нормальное функционирование организации не может быть и речи.

А теперь попробуем разобраться, что такое юридический отдел на предприятии и какую функцию он выполняет. Юридический отдел –это самостоятельная структура, которая находится в подчинении руководителя компании. При этом его основной функцией является соблюдение деятельности организации в соответствии с существующим законодательством, а также защита ее правовых интересов и информирование обо всех правках и новые законопроектах, напрямую связанных с ее деятельностью. Структура юридического центра разрабатывается руководителем с учетом объема работы и особенностей хозяйственной деятельности.

Основными задачами юридического центра являются:

- Проверка приказов, трудовых договоров и прочих документов на предмет соответствия законодательству РФ, а также подготовка заключений по правовым вопросам;
- Контроль документации на соответствие нормативным актам не только на государственном уровне, но и на уровне должностных инструкций самого предприятия;
- Подготовка исковых заявлений и отстаивание интересов субъектов в суде;
- Участие в подготовке и подписании коллективных договоров;
- Проведение анализа результатов рассмотрения судебных дел после вынесения постановления;

- Участие в разработке и подготовке нормативных актов для внутреннего пользования на предприятии.

1.4 ВЫБОР ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ ДЛЯ СОЗДАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Существует множество средств для создания CRM-приложения, рассмотрим некоторые из них.

Язык гипертекстовой разметки HTML позволяет создавать веб страницы. Веб страница, которую клиент видит на экране монитора в браузере, может включать в себя огромное множество различных файлов, например, сценариев Javascript, элементов Flash, изображений, музыкальных элементов и так далее, но основополагающей страницы достаточно часто является HTML документ**.** Язык HTML использует свою собственную вариацию символов благодаря которой современные веб браузеры отображают пользователю страницу. Данные символы называются дескрипторами и включают в себя так называемые элементы для создания гиперссылок.

PHP – язык который позволяет создавать сценарии, уже давным-давно он перерос данную возможность. Все дело в том, первая версия PHP была создана в далеком 1994 году Расмусом Лердорфом и представляла из себя набор инструментов, которые позволяли отслеживать посетителей веб страниц. По прошествии времени достаточно быстро PHP обрел широкую известность и популярность преобразившись в полноценный язык программирования.

PHP является серверным языком создания сценариев, конструкции, которые вставляются посредством изменения содержимого исполняемых файлов в HTML текст исполняются каждый раз при запросе страницы пользователям, а результат работы передается обычным HTML текстом браузеру, который в свою очередь отображает результат пользователю.

Язык программирования JavaScript изначально был создан для того, чтобы оживить веб страницы. Программное обеспечение, которое создается на данном языке программирования называется скриптами. В браузере данные скрипты подключаются напрямую к HTML и после загрузки страницы сразу исполняются.

В основном Javascript применяется в следующих категориях задач:

- необходимость динамического создания документа с помощью заранее прописанного сценария;
- динамические HTML страницы;
- взаимодействие с веб браузером пользователя;
- осуществление быстрой проверки правдивости вводимых пользователями данных в поля форм HTML до отправки их на сервер.

Язык программирования Python — это мощный инструмент для создания программ самого разнообразного назначения, доступный даже для новичков. С его помощью можно решать задачи различных типов. Python — универсальный язык программирования, применимый в том числе и в вебе. С технической точки зрения веб приложение на Python — полноценное приложение, загруженное в память, обладающее своим внутренним состоянием, сохраняемым от запроса к запросу.

Среди всех рассмотренных языков программирования, PHP является наиболее подходящим и удобным для создания backend части CRM-приложения, а Javascript для создания frontend части CRM-приложения. Рассмотрим более подробно.

JavaScript – язык программирования, являющийся прототип-ориентированным. Он отражает язык ECMAScript, чьим прототипом изначально и являлся. Первая вариация появилась ещё в 1995 году и с тех пор постоянно совершенствовалась, пока не пришла к нынешнему виду.

Чаще этот язык используется в разработке приложений и браузерах с целью придания им интерактивности и «живости».

Частой ошибкой встречается путаница между Java и JavaScript – это два разных языка, несмотря на схожие названия.

Базовой особенностью этого языка отмечается то, что на него повлияли другие (Python, Java и др.) языки программирования с целью придания максимального комфорта JavaScript и лёгкости в понимании его теми пользователями, которые не имеют соответствующего образования и глубинных знаний – не программистами. JavaScript – официально зарегистрированная торговая марка компании Oracle.

С помощью Javascript доступны к исполнению следующие функции:

- возможность изменять страницы браузеров;
- добавление или удаление тегов;
- изменение стилей страницы;
- информация о действиях пользователя на странице;
- запрос доступа к случайной части исходного кода страницы;
- внесение изменений в этот код;
- выполнение действия с cookie-файлами.

Область применения этого языка удивительно обширна и ничем не ограничена: среди программ, которые используют JS, присутствуют и тестовые редакторы, и приложения (как для компьютеров, так и мобильные и даже серверные), и прикладное ПО.

Преимущество перед другими языками (Perl, C) заключается в том, что можно создать HTML документ с внедренными командами PHP. Этот язык легок в освоении, но за то он имеет широкие возможности в применении. Он позволяет создавать web-приложения высокого качества за очень короткое время. Сейчас PHP постоянно совершенствуется, и он явно доминирует среди других языков web-программирования, и доминировать он будет еще долго. Основополагающий фактор языка PHP практичность. Он должен дать возможность программисту эффективно и быстро решать задачи. Основные характеристики PHP:

- простота;
- традиционность;
- безопасность;
- эффективность;
- гибкость.

Есть ещё одна важная характеристика языка PHP, он распространяется абсолютно бесплатно!

Традиционность. Язык программирования PHP кажется знакомым множеству программистов, работающих в самых различных областях. Большинство конструкций, которые были использованы в языке заимствованы из Perl, C.

Код, который получается во время работы очень сильно напоминает C. Данный фактор значительно снижает усилия, которые необходимы для его освоения. Данный язык, сочетает в себе достоинства Perl и C, он специально был создан и нацелен на эффективную работу в интернете, с понятным и обратно совместимым синтаксисом.

PHP обрел большую популярность среди веб-программистов. Настолько большую что является чуть ли не самым популярным языком для создания веб приложений.

Простота. Сценарий PHP файла может содержать более 10 тыс. строк или же всего лишь из 1 строки. Это зависит от задачи. В PHP не нужно подгружать библиотеки, устанавливать параметры для компиляции. PHP исполняет код после 1 экранирующей последовательности (<?) и выполняет его до того момента пока не встретит себе парную экранирующую последовательность (?>). Если программист при написании кода использовал верный синтаксис, то код исполняется в точности так, как и должен, если же программист допустил ошибку PHP уведомит его об этом.

PHP настолько универсальный язык что может с легкостью быть встроен в код HTML страниц, которые будут обрабатываться

интерпретатором языка PHP. Так же PHP может быть использован для написания CGI-сценариев. PHP может быть использован для формирования различных документов например PDF, DOC, EXCEL. PHP обладает огромным разнообразием функций.

Эффективность бесспорно очень важный фактор, на который смотрят программисты при выборе среды разработки.

Одно из самых важных преимуществ языка PHP заключается в его «движке», PHP не является ни интерпретатором, ни компилятором, а транслирующим интерпретатором. Именно такое устройство самого «движка» языка PHP позволяет ему исполнять сценарии в файлах с большой скоростью.

По различным оценкам, большая часть сценариев, написанных на PHP обрабатываются заметно быстрее аналогичных сценариев на других языках программирования, однако, чтобы не придумали разработчики языка программирования PHP, откомпилированные файлы будут работать быстрее в огромное количество раз, для этого в PHP существуют расширения, которые кешируют исполняемые части кода и таким образом заметно увеличивают производительность. Но даже без их использования производительность языка программирования PHP находится на высоком уровне позволяя создавать масштабные, серьезные веб приложения.

Безопасность. Неотъемлемый фактор любого приложения. PHP предоставляет возможности разработчикам для эффективного и гибкого управления безопасностью всего приложения, они в свою очередь делятся на несколько категорий, средств системного уровня и самого приложения.

В PHP реализованы средства безопасности системного уровня, которыми манипулирует администратор, при грамотной и верной настройке они обеспечивают большую свободу действий и эффективно обеспечивают безопасность. Язык программирования PHP может работать в режиме, называемом Safe mode. Данный режим ограничивает возможности использования языка программирования PHP по нескольким важным

пунктам. Регулируя данную настройку можно изменить максимально допустимое время исполнения скрипта и расхода памяти, ведь неконтролируемый расход памяти может негативно отразиться на быстродействии сервера. Так же администратор в праве установить определенные ограничения на каталоги, в которых можно исполнять прописанные PHP сценарии, а также использовать их для просмотра защищенной информации о сервере, к которой относится, например, файл passwd.

Средства безопасности уровня приложения. В PHP включены несколько надежных механизмов шифрования данных которые позволяют решать множество задач таких как шифрование паролей, генерация случайных названий файлов и прочие задачи. Также язык программирования PHP совместим с независимыми приложениями различных фирм, это открывает многочисленные возможности для интеграции с различными технологиями электронной коммерции. Совсем другое его преимущество заключается в том, что исходный текст PHP сценариев невозможно просмотреть в браузере, ведь сценарий компилируется на стороне сервера перед тем как отправить запрос клиенту. Языка программирования PHP реализован на стороне сервера, что в свою очередь позволяет предотвратить хищение сценариев пользователями.

Гибкость. Поскольку язык программирования PHP является встраиваемым языком программирования, PHP отличается необыкновенной гибкостью относительно к нуждам разработчика. Традиционно язык программирования PHP рекомендуется применять совместно с HTML, он так же не плохо интегрируется и в Javascript, XML, WML и прочие языки программирования. Кроме этого, правильно структурированные приложения на языке программирования PHP достаточно просто расширяются по мере необходимости.

У языка программирования PHP нет проблем с браузерами, ведь весь сценарий языка программирования PHP перед отправкой пользователю

исполняется и компилируется на сервере. По сути данные сценарии языка программирования PHP могут передаваться любым устройствам с браузерами. Программисты так же занимающиеся вспомогательным программным обеспечением, могут исполнять скрипты написанные на языке программирования PHP из-под командной строки.

Так как язык программирования PHP не содержит в себе никакого ориентированного под конкретный веб сервер кода, администратор в праве сам выбирать где и как PHP будет исполняться. PHP по сути является совсем платформенно независимым языком программирования и существует практически на всех платформах.

Бесплатное распространение. Именно стратегия Open Source позволяет программам, движкам, компонентам получать широкое распространение в массах, что в свою очередь оказало несомненно позитивное влияние на множество популярных проектов проекты, в 1 очередь — проекта Linux. Данное относится и к самой истории создания языка программирования PHP, ведь его поддержка пользователями со всего мира оказалась кстати и сыграла огромную роль в развитии проекта PHP.

Принятие данной стратегии и бесплатное распространение исходных текстов языка программирования PHP несомненно оказало большое влияние на рынок программ, сформировало большое отзывчивое сообщество программистов, использующий язык программирования PHP, на данный момент в интернете можно найти ответ практически на любой вопрос связанный с разработкой на данном языке веб приложений, даже на самые сложные вопросы.

ГЛАВА 2. ПРОЕКТИРОВАНИЕ СОЗДАВАЕМОГО ПРОГРАММНОГО ПРИЛОЖЕНИЯ

При проектировании любого веб приложения очень важным его аспектом является архитектура, именно она показывает концепт работы структуры технологий и функциональных процессов на уровне системы и во взаимосвязи. Как правило веб приложения создаются в виде композиции компонентов, которые взаимодействуют между собой на высоком уровне, они же могут выступать в роли системы. Грамотно выбранная архитектура веб приложения облегчает понимание всей системы, именно она определяет структуру и функциональность способом, который дает возможность заказчику задать вопросы об удовлетворении его проектных требований при этом раскрывая проектировочные решения, реализуя компоненты и распределяя функционал.

Проектируя современное веб приложение для организации, архитектура приложения обязательно должна создаваться с учетом множества потребностей заказчика, созданное веб приложение должно быть понятным, удобным для пользователей, позволять разработчикам программного обеспечения построить план работы, графики, которые могут отображать эффективность системы. Архитектура должна позволять построить графики эффективности системы, позволить задать функции, технологии, определить основные интерфейсы, также позволить оценить бюджет и время затраченное на разработку проекта. Учитывая данные факторы от разработчиков архитектуры требуется наличие навыков, опыта, для построения концепции приложения удовлетворяющей требованиям приложения на начальном этапе разработки и поддержании цельности данной концепции во время разработки системы. Получившаяся система должна быть пригодна для использования клиентами. Создание архитектуры веб приложения – достаточно трудоемкий процесс, при котором описывается архитектура информационной системы в мельчайших деталях. В настоящее

время используются следующие программные архитектурные решения: многоуровневая, клиент-серверная, файл-серверная.

Клиент-сервер. Основа данной концепции состоит в том, что кроме файлов базы данных, сервер исполняющий функции основного обязан выполнить большую часть обработки поступающих данных. Пользователи системы обращаются к основному серверу с помощью специального языка программирования структурированных запросов SQL. На нем описывают определенный список задач, который требуется выполнить серверу. SQL запросы поступающие от пользователей сервер принимает и обрабатывает. В качестве ответа пользователь, отправивший запрос получает набор данных, который уже был обработан сервером. Клиент и сервер передают между друг другом не полный набор данных в отличии от концепции файл-сервер, а только тот который был необходим клиенту. Пользователь, вызвав запрос всего лишь в несколько строк, может вызвать процесс, который затронет миллионы строк, большую часть таблиц, и получить в ответе всего лишь несколько строк.

Избежать передачи по сети интернет масштабных объемов трафика позволяет технология клиент-сервер, она перекладывает всю обработку данных на основной сервер. Помимо этого, такой подход дает возможность избегать конфликтов при одновременном изменении данных, которых обычно характерны для технологии клиент-сервер множеством пользователей. Данная технология позволяет согласовать изменение данных между множеством пользователей, таким образом сохраняя целостность данных в таблицах. Данные и некоторые другие преимущества сделали данную технологию достаточно популярной. Из недостатков технологии можно выделить высокие требования к железу основного сервера, ведь чем больше пользователей нагружают сервер, тем больше данных требуется ему обработать и тем больше мощностей ему требуется.

Файл-сервер. Данная архитектура предполагает наличие одного компьютера в сети, который будет хранить файлы базы данных. Архитектура

соответствует запросам пользователей и передает файлы с сервера на их рабочие станции, где осуществляется основная работа над данными. В данном случае основной сервер исполняет лишь только роль хранения файлов, при этом не взаимодействуя с обработкой данных. По завершению работы клиенты производят копирование файлов с данными на основной сервер, откуда позже могут их взять и обработать другие клиенты. Таким образом организация хранения данных таким образом обладает рядом недостатков, к примеру, при одновременном взаимодействии с участком данных множества клиентов производительность такой системы сильно падает, поскольку им необходимо дождаться пока предыдущий пользователь, который работал с данными закончит работу. В ином случае, возможны потери данных, потеря изменений, внесенных другими клиентами.

2.1 ВЫБОР СУБД

Базы данных представляют из себя совокупности структурированных данных. В СУБД можно хранить любые варианты данных, от картин, находящихся в галерее, до большого количества информации о клиентах, менеджерах, исполнителях. Для управления данными находящимися в базе данных требуется система для управления СУБД. На данный момент компьютеры прекрасно справляются с обработкой огромных объемов данных. Управление СУБД реализуется различными способами, такими как отдельные утилиты, код входящий в другие приложения.

СУБД веб-приложения нашего проекта электронной кафедры должна удовлетворять следующим критериям:

- Распространенность программной платформы;
- Максимальная кроссплатформенность для подключения сторонних веб-приложения с целью обобщения данных;
- Переносимость;

На основе этих критериев выбор реляционной базы данных является обоснованным решением.

Базы данных реляционного типа были созданы для быстрого сохранения, обработки и хранения огромных объемов информации. Характеристики использования реляционной модели данных и реляционных СУБД.

Использование ключей. Каждая строка данных в таблице идентифицируется уникальным “ключом”, который называется первичным ключом. Зачастую, первичный ключ это автоматически увеличиваемое (автоинкрементное) число (1,2,3,4 и т.д). Данные в различных таблицах могут быть связаны вместе при использовании ключей. Значения первичного ключа одной таблицы могут быть добавлены в строки (записи) другой таблицы, тем самым, связывая эти записи вместе.

Отсутствие избыточности данных. В проекте базы данных, которая создана с учетом правил реляционной модели данных, каждый кусочек информации, например, имя пользователя, хранится только в одном месте. Это позволяет устранить необходимость работы с данными в нескольких местах. Дублирование данных называется избыточностью данных и этого следует избегать в хорошем проекте базы данных.

Ограничение ввода. Используя реляционную базу данных можно определить какой вид данных позволено сохранять в столбце. Можно создать поле, которое содержит целые числа, десятичные числа, небольшие фрагменты текста, большие фрагменты текста, даты и т.д.

Назначение прав. Большинство РСУБД предлагают настройку прав доступа, которая позволяет назначать определенные права определенным пользователям. Некоторые действия, которые могут быть позволены или запрещены пользователю: SELECT (выборка), INSERT (вставка), DELETE (удаление), ALTER (изменение), CREATE (создание) и т.д. Это операции, которые могут быть выполнены с помощью структурированного языка запросов (SQL).

Переносимость. Реляционная модель данных стандартна. Следуя правилам реляционной модели данных можно быть уверенным, что данные могут быть перенесены в другую реляционную СУБД относительно просто [1].

Вследствие вышеперечисленных преимуществ СУБД MySQL, выбор этого хранилища данных позволяет рационально использовать ресурсы вычислительной системы, обслуживающей электронную кафедру компьютерных технологий.

Взаимодействие базы данных и веб сервера возможно организовать на основании двух принципиально разных сценариях:

- Бизнес логика находится в базе данных;
- Бизнес логика находится в коде веб сервера.

В первом случае база данных хранит данные и предоставляет интерфейс доступа к данным:

- Выборка данных — решается через представления;
- Модификация данных — решается через хранимые процедуры.

Программа для веба сервера является драйвером для доступа к бизнес-логике, то есть она просто связывает Браузер с бизнес логикой, которая реализована в базе данных.

Во втором случае база данных хранит данные, и предоставляет прямой доступ к данным. Бизнес-логика реализована в коде web-сервера. В этом случае база данных предоставляет транзакции для проведения атомарных операций.

База данных необходимая для реализации CRM приложения должна иметь реляционную модель работы. Данная модель позволяет создавать связи межу самыми различными сущностями из различных таблиц. Для хранения данных база должна обладать особой структурой таблиц. Записи в базе данных хранят в себе множество полей с данными. Записи имеют

уникальные ключи. Каждый из столбцов содержит различные данные. Таким образом данные хранящиеся в одной таблице взаимосвязаны.

Взаимосвязи, которые хранятся в базах данных могут быть рассмотрены в виде математического множества, которое включает в себя атрибуты и информацию, которая хранится в базе данных. При задании структуры таблицы базы данных используется заранее созданный и описанный самой базой данных тип, например, строка, целочисленное значение, логическое значение и так далее. Каждый тип данных предназначен для решения определенных задач и не является взаимозаменяемым.

2.1.1 ОБОСНОВАНИЕ ВЫБОРА СУБД MYSQL

MYSQL является полнофункциональной, свободной системой которая позволяет управлять реляционными СУБД. Первое использование СУБД MYSQL было в 1990 году. Авторы ядра СУБД MYSQL пытались решить проблему постоянного роста компьютерной информации с помощью маленькой на тот момент MYSQL. В тот момент, когда создатели поняли, что она не справляется с задачами, которые они возлагали на неё, они решили создать более мощную базу, которая в последствии превратилась в всеми известную MYSQL. Реляционной СУБД MYSQL поддерживаются различные механизмы хранения данных (движки). Данные механизмы определяют взаимодействие базы с данными, то как она извлекает и обрабатывает данные. Следствием из этого является то, что механизм хранения MYSQL имеет свой собственный набор преимуществ и возможностей. С течением времени данные механизмы только укрепляют свои позиции. Основные преимущества:

- MYSQL запускается на любой OS (не требует платной лицензии Windows);
- Полнофункциональная MYSQL бесплатна;

- Легковесна, не требует мощного железа;
- Множество хостинг провайдеров, поддерживающих MYSQL;
- Выбор движков под каждую таблицу и каждый случай;
- Полнотекстовый поиск.

2.2 ВЫБОР ФРЕЙМВОРКА

При разработке сайта есть вероятность изменений изначальной бизнес логики, дизайна, интерфейса и функционала. Большие изменения могут так же появится в будущем, допустим у предприятия появится новый отдел, которому нужен свой собственный интерфейс. Таким образом изначально необходимо с особой осторожностью подойти к выбору фреймворка на котором в дальнейшем будет построен основной функционал системы. Если этого не сделать, в будущем может возникнуть множество ошибок, проблем при работе с системой. Их можно будет решить, но это потребует затрат множества ресурсов.

По большей части это связанно с большими, сложными проектами, но даже на небольшом сайте может настать момент, когда данная проблема будет актуальна. Выбранная система должна иметь легко расширяемую, гибкую, быструю и безопасную структуру.

Для того чтобы обезопасить себя от проблем, с которыми возможно придется столкнуться необходимо выбрать уже проверенное программное решение, которое позволит с легкостью подключать дополнительные функциональные компоненты, модули, разрабатываемые сторонними разработчиками. В данном программном решении будет жесткая структура, которая позволит нам писать код только в определенных предназначенных для этого каталогах. Таким образом это позволит не сломать работу системы, и даст возможность реализовать текущие задачи.

Описанный выше каркас в мире программирования называется фреймворк. Он не просто состоит из ряда библиотек, это целая архитектура,

которая гарантирует общую структуру у программ и обеспечивает их поведение по умолчанию.

Рассмотрим несколько популярных фреймворков и выберем один из них.

Zend – данный фреймворк распространяется свободно с открытым исходным кодом и используется для разработки сервисов и веб приложений. В нем используется объектно-ориентированный подход и большая часть самых современных функций. Он является надежным решением, которое обеспечивает множество вариантов для конфигураций. Если проект является крупным, то данный фреймворк отлично подходит, но в данном случае проект не является столь масштабным чтобы его использовать.

Code Igniter - достаточно старый фреймворк который зарекомендовал себя как удобное и быстрое решение для старта. Данный фреймворк поддерживает обратную совместимость, это позволяет избегать конфликта при миграции с одной версии PHP на другую. Фреймворк не требует много ресурсов и работает практически на всех платформах хостинга, но также обладает рядом минусов, таких как небольшое русскоязычное сообщество, относительно небольшая распространенность, разрабатывается уже довольно долго.

Yii2 – современный и распространённый фреймворк на котором написана не одна тысяча проектов. Данный фреймворк позволяет писать аккуратный код без лишней сложности, без повторений. Таким образом позволяя писать быстро, качественно и удобно. Так же обладает рядом минусов, нет встроенного удобного шаблонизатора, сложный для старта, нужно изучить большую документацию, старая архитектура, не позволяющая применять все современные возможности PHP.

Laravel качественно написанный, быстрый, удобный в использовании, функциональный фреймворк. За несколько лет он собрал большое сообщество разработчиков. Огромное количество информации доступно на просторах интернет о данном фреймворке. Большое сообщество создает

множество модулей для фреймворка. Даже хостинг компании начали предлагать свои решения для приложений построенные на фреймворке Laravel. Он является наиболее быстро растущим, часто используемым, поддерживаемым в течении нескольких лет фреймворком. В данном случае Laravel подходит больше всего.

Среди всех рассмотренных выше фреймворков, Laravel является самым подходящим для разработки CRM приложения. Далее рассмотрим его более подробно.

2.2.1 ОБОСНОВАНИЕ ВЫБОРА ФРЕЙМВОРКА LARAVEL

Laravel бесплатный фреймворк с открытым исходным кодом. Распространяется под лицензией MIT и есть в свободном доступе на GitHub.

- Среди главных особенностей фреймворка следует выделить:
- наличие системы Eloquent ORM, которая реализовывает шаблоны проектирования ActiveRecord;
- обратная маршрутизация;
- для упрощения генерации страниц используется страничный вывод;
- наличие модульного тестирования, который предотвращает возникновение ошибок;
- система миграций, которая позволяет управлять версиями для баз данных;
- автозагрузка классов, с помощью которой происходит автоматическая загрузка PHP классов;
- поддержка контроллеров.

Мощный фреймворк обладает особой гибкостью и хорошей функциональностью для решения многих задач в сфере веб-разработки.

Фреймворк — это некий скелет структуры проекта, предоставляющий реализованный функционал "из коробки". Данный функционал позволяет

решать наиболее типичные задачи при разработке. Архитектура фреймворка может как реализовывать некий собственный набор паттернов, так и представлять собой реализацию отдельных общеизвестных паттернов.

Архитектура Laravel строится вокруг популярного в последнее время принципа Inversion of Control >> Dependency Injection >> Service Container. Этот принцип хорошо знаком разработчикам Symfony, т.к. является краеугольным камнем всей Symfony разработки.

Service Container Laravel очень напоминает Symfony с незначительными отличиями. Это неудивительно, ведь он реализует тот же паттерн, хоть и немного иначе. В целом же надо отметить, что именно Service Container обделен вниманием в документации Laravel. Вероятно, разработчики фреймворка не считают данный компонент центральным в его построении. Хотя, возможно, подробное описание опущено, чтобы не усложнять документацию и предоставить разработчику возможность ориентироваться в работе с компонентом по ситуации.

Laravel использует часть компонентов Symfony. Это не означает, что компоненты внедряются копипастом и, соответственно, работают по принципу «достаточно только прочитать документацию оригинала и можно использовать». Скорее компоненты удачно вплетены в структуру фреймворка, где-то немного изменены (обернуты), где-то дополнены и подстроены для соответствия "экосистеме" фреймворка. На самом деле, это достаточно логично, т.к. многие компоненты Symfony поставляются как готовый к использованию код и хорошо решают задачи, которые ставятся перед фреймворками на общем уровне.

2.3 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

Этап физического проектирования БД направлен на разработку методов взаимодействия с БД, которые позволяют обеспечить эффективность запросов, которые идут к базе данных, таким образом учитывая особенности

конкретной СУБД может быть необходимо создание дополнительных структур таких как индексы и так далее.

Что же такое Индекс, это объект СУБД, который создает пользователь с целью увеличения производительности в выполнении запросов. Базы данных могут хранить огромное количество строк в таблицах, они хранятся в СУБД в случайном порядке, их поиск может занимать много времени за счет полного сканирования базы данных. Индексы формируются, используя значения нескольких или одного из столбцов таблицы, указателей на определенные строки таблицы позволяя таким образом найти нужную строку по заданному клиентом значению. Ускорение работы получается за счет того, что индексы имеют определенную структуру, которая оптимизирована под поиск, например, сбалансированного дерева. В таблицах баз данных используется распространенный движок (система хранения) MyISAM. Таблицы MyISAM хорошо подходят для использования в WWW и других средах, где преобладают запросы на чтение. Таблицы типа MyISAM показывают высокую скорость работы при выборках SELECT. Во многом это связано с отсутствием поддержки транзакций и внешних ключей. Эта система хранения поддерживается многими СУБД и является оптимальной для разрабатываемой системы.

СУБД создаются для того чтобы хранить в них определенную информацию и позже получать её при необходимости. Это значит, что база данных должна иметь возможность делать выборку по записям, а также иметь возможность помещать данные, вставлять их в таблицы. Язык запросов к СУБД под названием SQL был придуман специально для данных операций. Операции вставки и выборки являются частью языка SQL.

Связь является ассоциацией между двумя сущностями. Одна из сущностей может быть связанна с другой сущностью либо же сама с собой.

Связи предоставляют возможность найти другую сущность (сущности) которые связанны с нею.

Например, связи между сущностями могут выражаться следующими фразами – "ГРУППА может иметь несколько СТУДЕНТОВ", "каждый СТУДЕНТ обязан числиться ровно в одной ГРУППЕ".

Графически связь изображается линией, соединяющей две сущности:

Рис. 2.1. Логическая модель связи между группой и студентами

Каждая связь имеет два конца и одно или два наименования. Наименование обычно выражается в неопределенной глагольной форме: "иметь", "принадлежать" и т.п. Каждое из наименований относится к своему концу связи. Иногда наименования не пишутся ввиду их очевидности.

Каждая связь может иметь один из следующих типов связи:

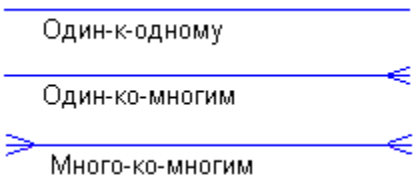

Рис. 2.2. Типы связи реляционных баз данных

Связь типа один-к-одному означает, что один экземпляр первой сущности (левой) связан с одним экземпляром второй сущности (правой). Связь один-к-одному чаще всего свидетельствует о том, что на самом деле мы имеем всего одну сущность, неправильно разделенную на две.

Связь типа один-ко-многим означает, что один экземпляр первой сущности (левой) связан с несколькими экземплярами второй сущности

(правой). Это наиболее часто используемый тип связи. Левая сущность (со стороны "один") называется родительской, правая (со стороны "много") – дочерней.

Связь типа много-ко-многим означает, что каждый экземпляр первой сущности может быть связан с несколькими экземплярами второй сущности, и каждый экземпляр второй сущности может быть связан с несколькими экземплярами первой сущности. Тип связи много-ко-многим является *временным* типом связи, допустимым на ранних этапах разработки модели. В дальнейшем этот тип связи должен быть заменен двумя связями типа один-ко-многим путем создания промежуточной сущности.

Модальность "может" означает, что экземпляр одной сущности может быть связан с одним или несколькими экземплярами другой сущности, а может быть и не связан ни с одним экземпляром.

Модальность "должен" означает, что экземпляр одной сущности обязан быть связан не менее чем с одним экземпляром другой сущности.

Проектирование базы данных – это вопрос идентификации данных, их связи и помещение результатов решения данного вопроса на бумагу (или в компьютерную программу). Проектирование базы данных независимо от РСУБД, которую можно использовать для ее создания.

Процесс проектирования модели базы данных включает подготовку схемы модели базы данных.

Имеются три основных типа моделей данных на основе записей:

- иерархическая;
- сетевая;
- реляционная.

Сетевая модель содержит в себе данные и связи. Данные в ней представляются в виде коллекций записей, а связи в виде наборов. В отличие от реляционной модели, связи здесь явным образом моделируются наборами, которые реализуются с помощью указателей. Сетевую модель можно

представить, как граф с записями в виде узлов графа и наборами в виде его ребер.

Иерархическая модель является ограниченным подтипом сетевой модели. В ней данные также представлены как коллекции записей, а связи - как наборы. Однако в иерархической модели узел может иметь только одного родителя. Иерархическая модель может быть представлена как древовидный граф с записями в виде узлов (которые также называются сегментами) и множествами в виде ребер.

Сущность — это реальный или виртуальный объект, имеющий существенное значение для рассматриваемой предметной области, информация о котором подлежит хранению. Если не вдаваться в подробности, то можно считать, что сущности соответствуют таблицам реляционной модели.

Под информационным объектом понимается некоторая сущность фрагмента действительности, например, организация, документ, сотрудник, место, событие и т. д.

Связи в моделях выступают в виде средства, которое представляет отношения между несколькими сущностями.

В предметной области выделяются различные типы объектов, представляемые в информационной системе в каждый момент времени конечным набором экземпляров данного типа. Каждый тип объекта включает (идентифицируется) присущий ему набор атрибутов (свойств, характерных признаков, параметров).

Связь — это ассоциация между сущностями, включают по одной сущности из каждого участвующего в связи типа сущности.

Атрибут представляет логически неделимый элемент структуры информации, характеризующийся множеством атомарных значений.

Один или некоторая группа атрибутов объекта данного типа могут исполнять роль ключевого атрибута, по которому различаются конкретные

экземпляры объектов (объект «Лицо» - ключ совокупность атрибутов «Фамилия», «Имя», «Отчество» или один атрибут «Номер паспорта»).

Получается, что логическая модель является графическим представлением структуры данных принимая в учет иерархическую, сетевую, реляционную модель данных, независимо от финальной реализации аппаратной платформы и базы данных. На основании логической модели строится физическая модель.

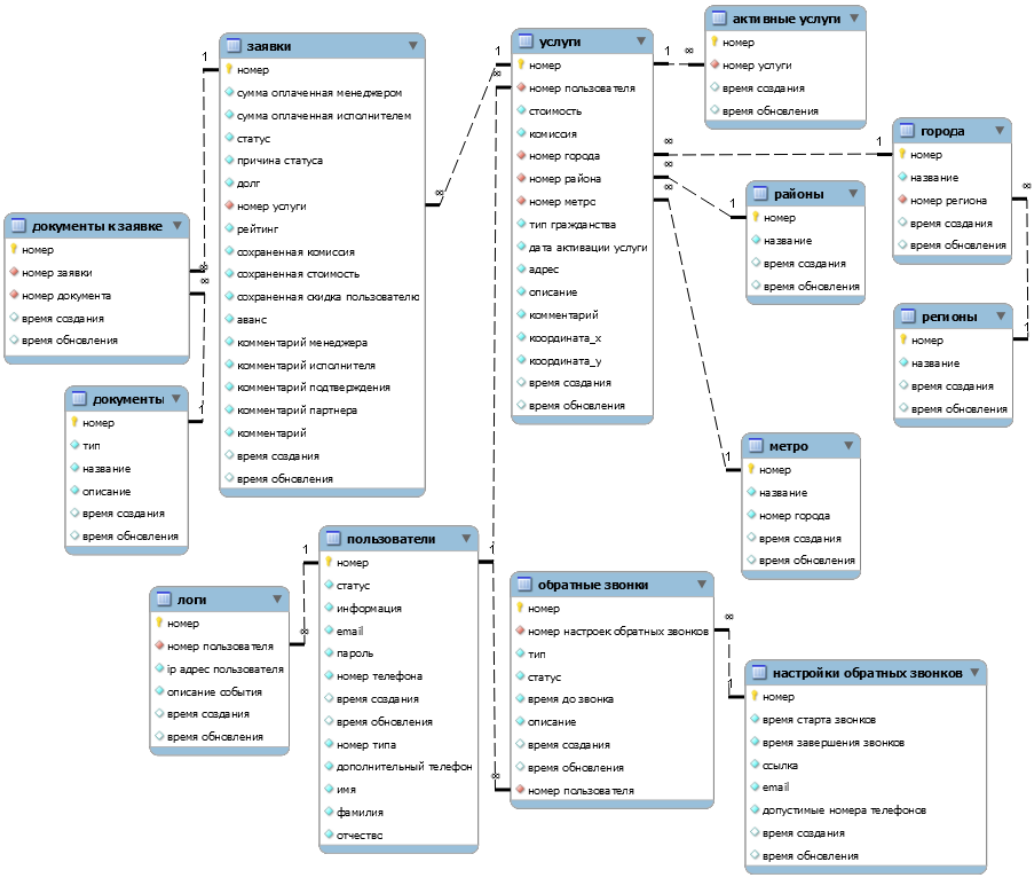

Рис. 2.3. Логическая модель БД

Физическое проектирование базы данных – процесс подготовки описания реализации базы данных на вторичных запоминающих устройствах; на этом этапе рассматриваются основные отношения, организация файлов и индексов, предназначенных для обеспечения

эффективного доступа к данным, а также все связанные с этим ограничения целостности и средства защиты.

Физическое проектирование является последним этапом создания проекта базы данных, при выполнении которого проектировщик принимает решения о способах реализации разрабатываемой базы данных. Во время предыдущего этапа проектирования была определена логическая структура базы данных. Хотя эта структура не зависит от конкретной целевой СУБД, она создается с учетом выбранной модели хранения данных. Однако, приступая к физическому проектированию базы данных, прежде всего необходимо выбрать конкретную целевую СУБД. Поэтому физическое проектирование неразрывно связано с конкретной СУБД. Между логическим и физическим проектированием существует постоянная обратная связь, так как решения, принимаемые на этапе физического проектирования с целью повышения производительности системы, способны повлиять на структуру логической модели данных.

Как правило, основной целью физического проектирования базы данных является описание способа физической реализации логического проекта базы данных.

Проектирование базы данных — это итерационный процесс, который имеет свое начало, но не имеет конца и состоит из бесконечного ряда уточнений. Его следует рассматривать прежде всего, как процесс познания. Как только проектировщик приходит к пониманию работы предприятия и смысла обрабатываемых данных, а также выражает это понимание средствами выбранной модели данных, приобретенные знания могут показать, что требуется уточнение и в других частях проекта [2].

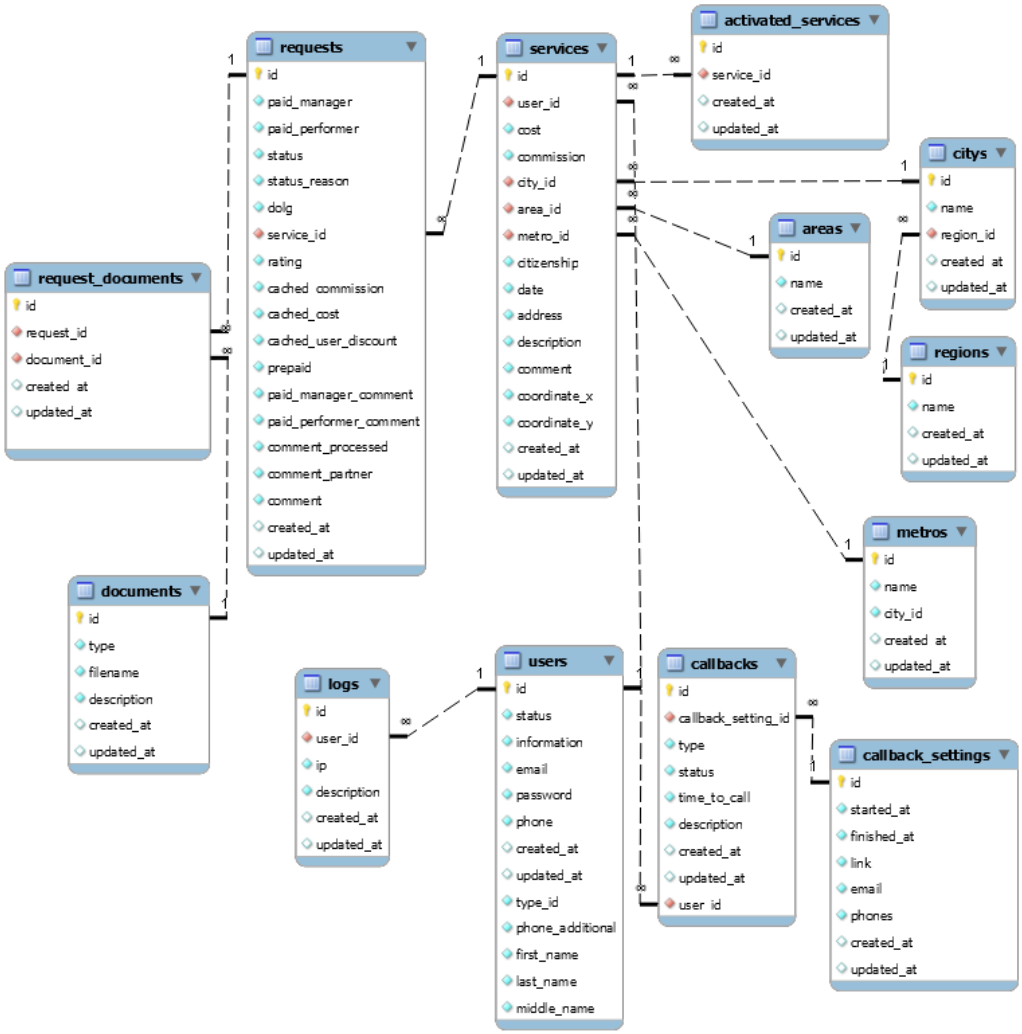

Рис. 2.4. Физическая модель БД.

2.4 СОЗДАНИЕ БАЗЫ ДАННЫХ

Для разработки физической модели базы данных было использовано программное средство phpMyAdmin, предоставляющее дружественный интерфейс для управления базами данных СУБД MYSQL.

Для создания новой таблицы необходимо выполнить следующую последовательность действий:

- Выбрать базу данных;
- Нажать кнопку «Создать таблицу»;

- После загрузки формы создания таблицы, необходимо ввести имя для создаваемой таблицы;
- Заполнить все необходимые поля, задать имена полей таблицы, указать типы данных;
- Создать первичные и внешние ключи;
- Нажать кнопку «Сохранить».

Данные этапы представлены на рисунках 5-13.

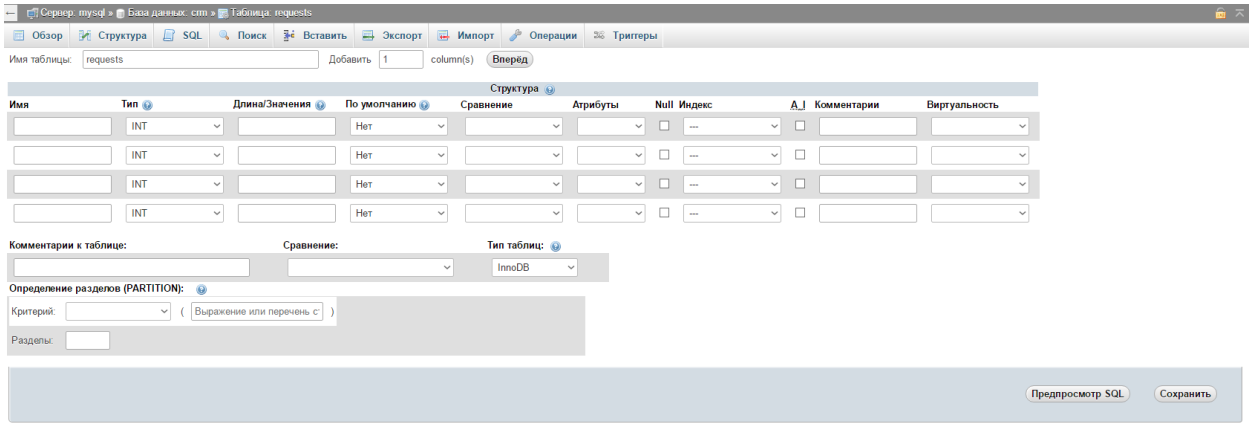

Рис. 2.5. Создание таблицы «Заявки».

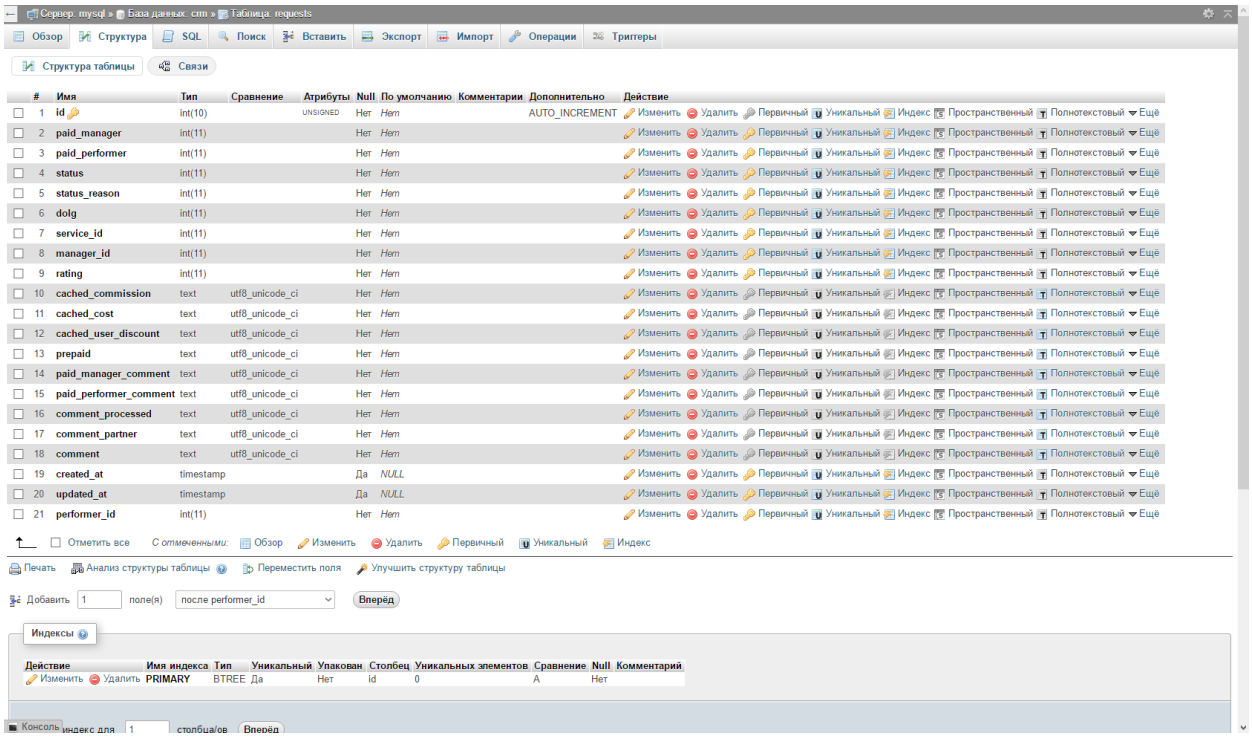

Рис. 2.6. Колонки таблицы «Заявки».

Первичный ключ – это атрибут или группа атрибутов, которые единственным образом идентифицируют каждую строку в таблице.

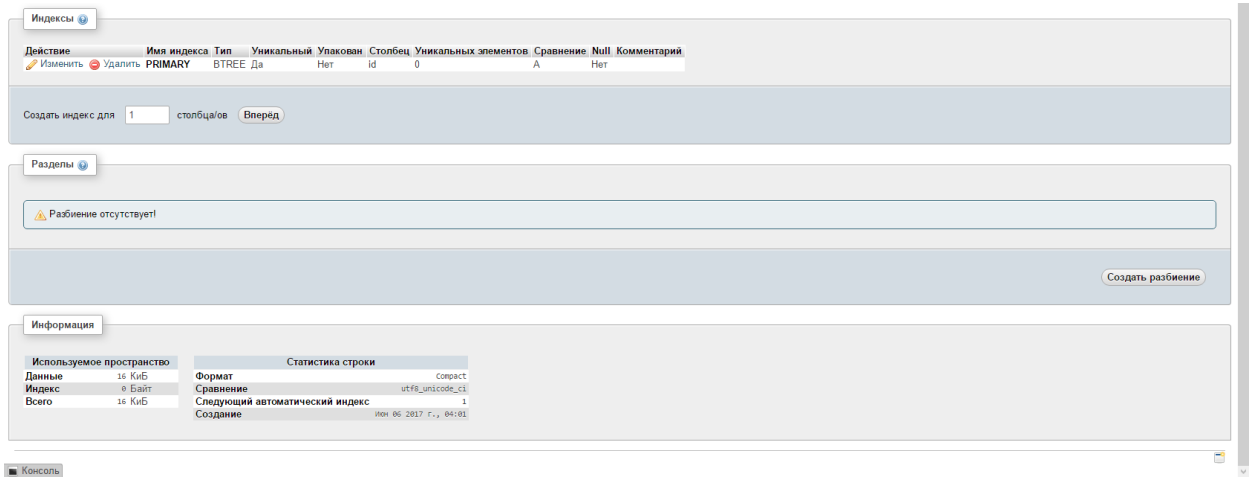

Рис. 2.7. Первичный ключ таблицы «Заявки».

Мы создали таблицу средствами phpMyAdmin, для того чтобы посмотреть SQL код необходимый для создания таблицы мы можем экспортировать таблицу в формате SQL.

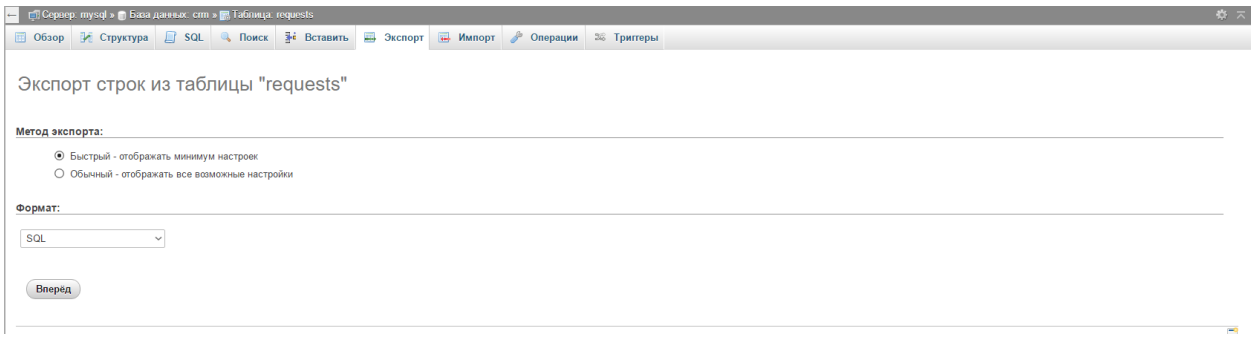

Рис. 2.8. Экспорт таблицы в формате SQL.

После чего открываем скачанный файл в текстовом редакторе и видим код для создания таблицы.

```
-- --------------------------------------------------------

--
-- Структура таблицы `requests`
--

CREATE TABLE `requests` (
  `id` int(10) UNSIGNED NOT NULL,
  `paid_manager` int(11) NOT NULL,
  `paid_performer` int(11) NOT NULL,
  `status` int(11) NOT NULL,
  `status_reason` int(11) NOT NULL,
  `dolg` int(11) NOT NULL,
  `service_id` int(11) NOT NULL,
  `manager_id` int(11) NOT NULL,
  `rating` int(11) NOT NULL,
  `cached_commission` text COLLATE utf8_unicode_ci NOT NULL,
  `cached_cost` text COLLATE utf8_unicode_ci NOT NULL,
  `cached_user_discount` text COLLATE utf8_unicode_ci NOT NULL,
  `prepaid` text COLLATE utf8_unicode_ci NOT NULL,
  `paid_manager_comment` text COLLATE utf8_unicode_ci NOT NULL,
  `paid_performer_comment` text COLLATE utf8_unicode_ci NOT NULL,
  `comment_processed` text COLLATE utf8_unicode_ci NOT NULL,
  `comment_partner` text COLLATE utf8_unicode_ci NOT NULL,
  `comment` text COLLATE utf8_unicode_ci NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  `performer_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

--
-- Дамп данных таблицы `requests`
--

INSERT INTO `requests` (`id`, `paid_manager`, `paid_performer`, `status`, `status_reason`, `dolg`, `service_id`, `manager_id`, `rating`, `cached_commission`, `cached_cost`, `cached_user_discount`, `prepaid`, `paid_manager_comment`,
`paid_performer_comment`, `comment_processed`, `comment_partner`, `comment`, `created_at`, `updated_at`, `performer_id`) VALUES
(1, 1, 1, 1, 1000, 1, 1, 1, '5', '1000', '5', '1000', 'test', 'test', 'test', 'test', 'test', NULL, NULL, 1),
(2, 2, 2, 2, 5000, 2, 2, 2, '5', '2000', '5', '1000', 'test 2', 'test 2', 'test 2', 'test 2', 'test 2', NULL, NULL, 2),
(3, 3, 3, 3, 15000, 3, 3, 3, '5', '5000', '5', '2000', 'test 3', 'test 3', 'test 3', 'test 3', 'test 3', NULL, NULL, 3),
(4, 4, 4, 4, 7500, 4, 4, 4, '5', '2500', '5', '2000', 'test 4', 'test 4', 'test 4', 'test 4', 'test 4', NULL, NULL, 4);

--
-- Индексы сохранённых таблиц
--

--
-- Индексы таблицы `requests`
--
ALTER TABLE `requests`
  ADD PRIMARY KEY (`id`);
```

Рис. 2.9. Код для создания таблицы

Далее нажимаем «Обзор», после чего открывается веб страница, в которой можно посмотреть на список заявок.

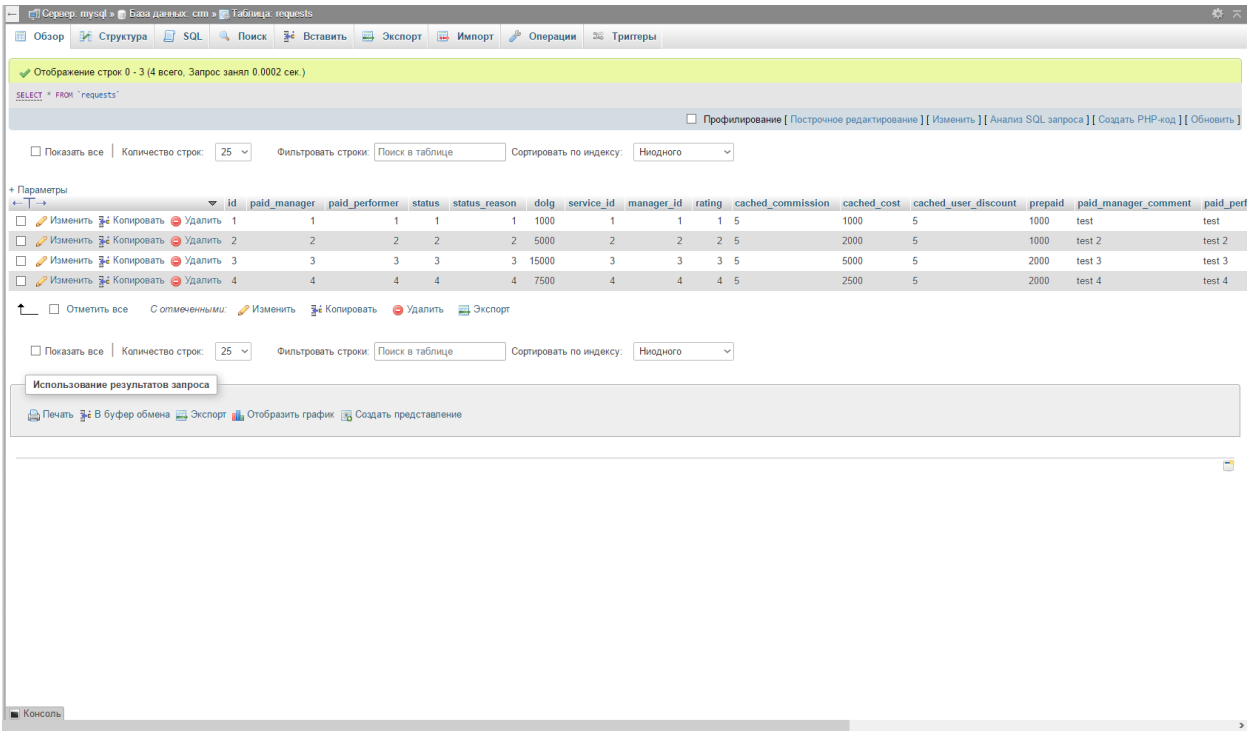

Рис. 2.10. Список заявок

Используя такой же метод создаются остальные необходимые для базы таблицы. Между несколькими таблицами могут быть созданы отношения

(связи), которые определяют, что для каждой записи из одной таблицы может существовать запись в подчиненной таблице.

Всего существует три разных разновидности данных связей между таблицами:

- один-ко-многим;
- один-к-одному;
- многие-ко-многим.

Один-ко-многим является самым распространенным и используется, когда к одной записи в главной таблице соответствует несколько записей в подчиненной (дочерней).

Создаем связи между таблицами с помощью вторичных ключей, как, например, в таблице «Документы к заявкам»:

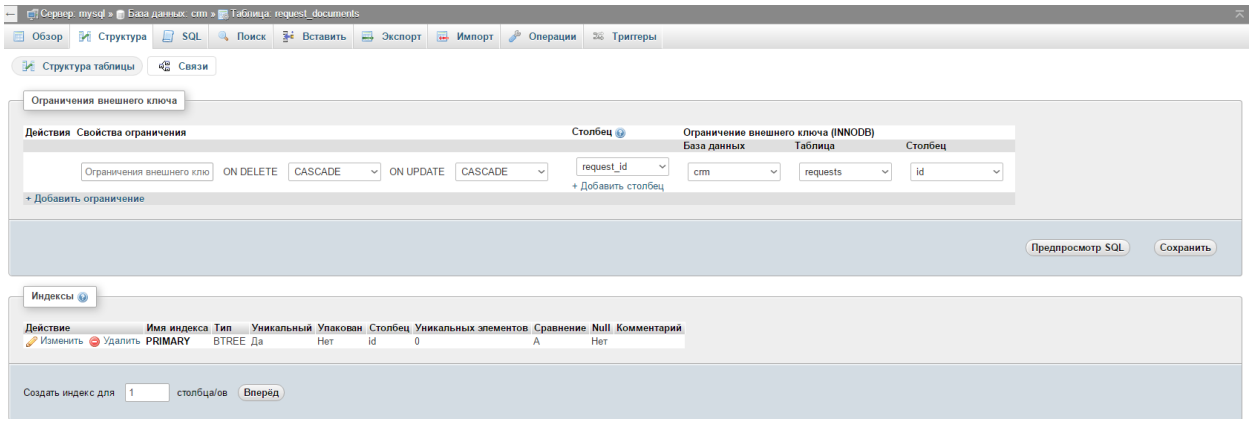

Рис. 2.11. Создание связи между таблицами.

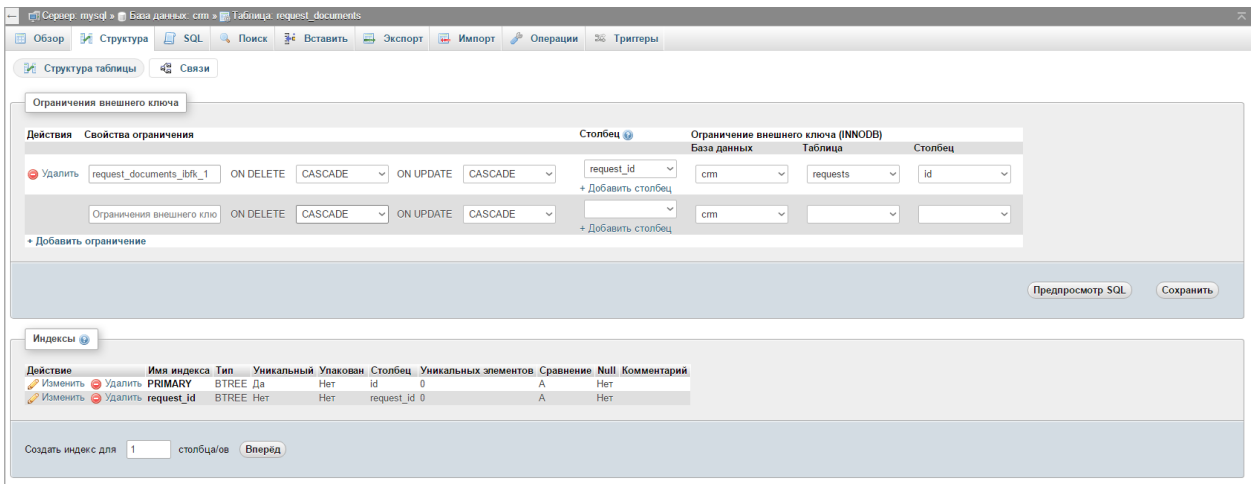

Рис. 2.12. Связь между таблицами.

В итоге на рисунке 2.13 мы имеем полноценную базу данных.

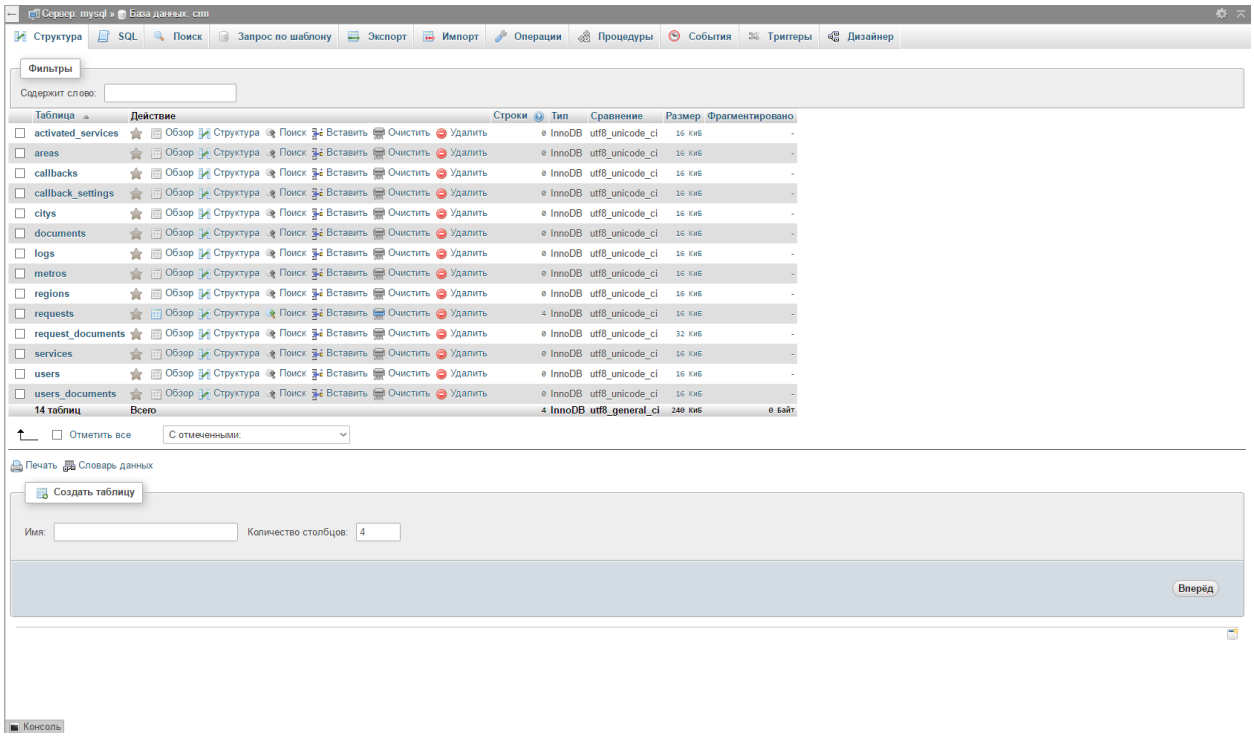

Рис. 2.13. База данных

2.5 СТРУКТУРА ПРИЛОЖЕНИЯ LARAVEL

Стандартная структура приложения, построенного на фреймворке Laravel спроектирована таким образом, что она позволяет стать комфортной отправной точкой для маленьких и крупных проектов. Конечно же, имеется возможность поменять стандартную структуру приложения, фреймворк Laravel не устанавливает никаких ограничений на то в каком месте должен находиться класс, самое важное нужно чтобы Composer мог его найти и загрузить.

После того как Laravel был успешно установлен, настроен веб сервер, Laravel выводит стандартную страницу приветствия.

В корневой директории только что установленного фреймворка Laravel можно наблюдать следующие каталоги:

app – содержит разрабатываемое приложение, основная директория с которой чаще всего ведется работа

bootstrap – данная директория содержит файлы необходимые для автозагрузки файлов в фреймворк Laravel, также внутри имеется папка cache которая содержит скомпилированный код фреймворка Laravel для увеличения производительности.

config – в данном каталоге находятся все основные конфигурационные файлы, такие как настройка приложения и базы данных.

database – в данном каталоге хранятся «миграции» и «сиды» для заполнения таблиц данными.

public – основная папка, корневая домена приложения, содержит файлы css, js а также изображения и прочий контент.

resources – в данной директории находятся файлы локализации, шаблоны, рабочие SASS, LESS, JS файлы приложения, которые могут собираться с помощью Laravel Elixir для фреймворков по типу AngularJS, VueJS, React, которые в последствии обрабатываются внешним инструментом public.

storage – данный каталог хранит в себе кеши файлов, любые пользовательские файлы, файлы сесии и автоматически сгенерированные файлы.

test – данный каталог служит для проведения Unit тестов

vendor – данная директория содержит в себе компоненты, которые устанавливает Composer из файла composer.json.

Каталог app включает в себя все основные классы приложения. По стандарту данный каталог имеет namespace App, но в Laravel имеется возможность — это изменить с помощью Artisan команды app:name, классы из данного каталога подгружаются по стандарту PSR-4.

В директории App находятся несколько подкаталогов которые играют важную роль, такие как Providers, Http, Console. Каталоги Console и Http предоставляют API доступ к приложению, Console – командная строка и соответственно HTTP – работа через браузер. В директории Console содержатся классы команд Artisan. В директории Http содержаться контроллеры, фильтры и запрос, классы в которых происходит валидация ввода пользователя. Таким образом подобный подход подталкивает новичков отходить от общепринятого вредного для всех подхода написания кода в контроллерах и отделять логику приложения.

Каталог Commands находящийся в папке app, включает в себя классы команд приложения. Данные команды выполняют задания, которые есть возможно поставить в очередь, также задачи, которые синхронно запускаются с процессом обработки запроса.

Каталог Events находящийся в папке app включает в себя классы событий. Их использование не обязательно, но в случае если они требуются в приложении, можно создать их через командный интерфейс Artisan CLI.

Каталог Handlers находящийся в папке app включает в себя классы обработчиков для событий и команд. Такие обработчики получают событие либо команду, после чего выполняют требуемые действия в ответ на инициализацию события или команды.

Каталог Services находящийся в папке app включает в себя различные механизмы приложения помощники и сервисы. К примеру сервис регистрации, который по умолчанию включен в Laravel и отвечает за валидацию данных пользователей и их регистрацию.

Каталог Exceptions находящийся в папке app включает в себя обработчики исключений, хорошей практикой считается размещение в данном каталоге исключений, которые могут появляться при работе приложения.

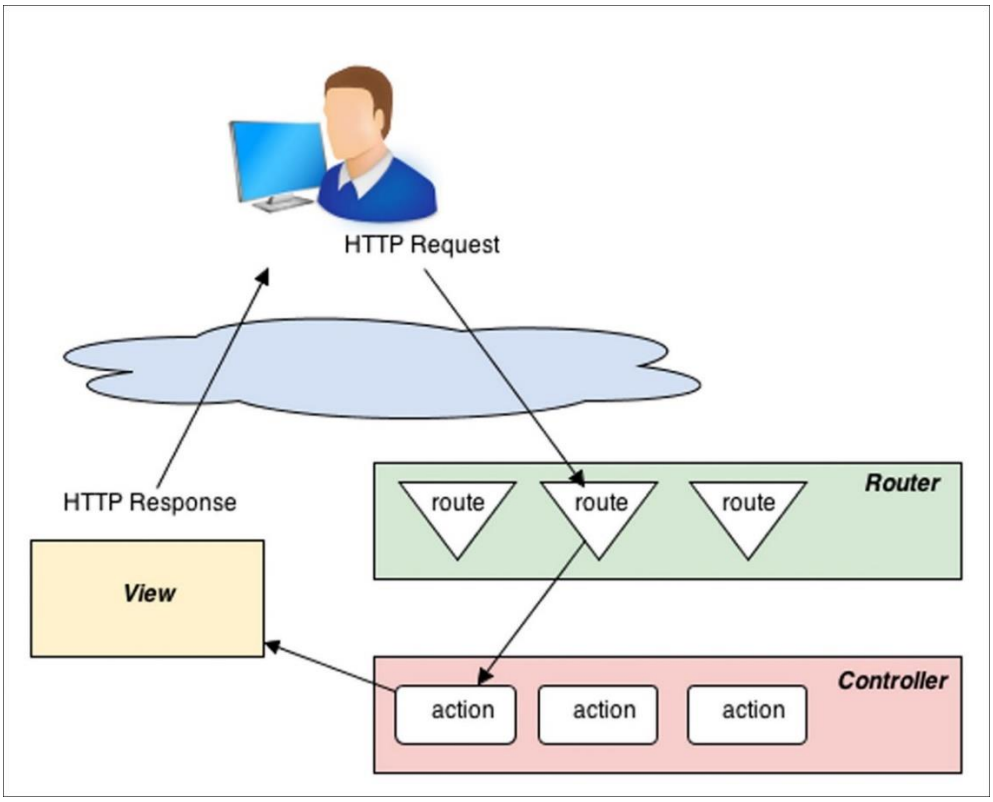

Рис. 2.14. Структура приложения Laravel

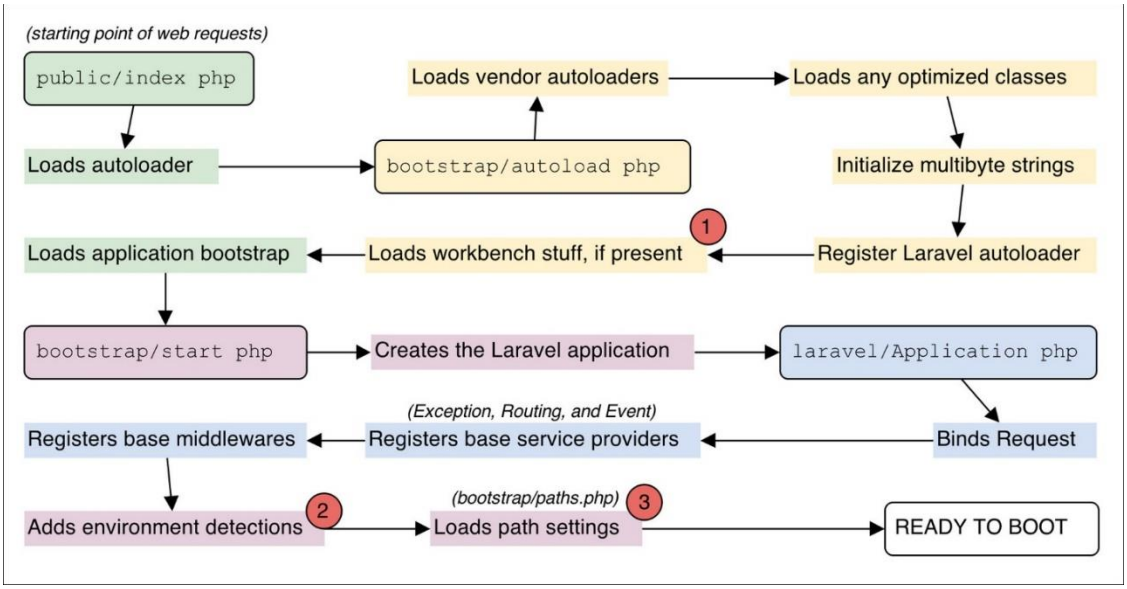

Рис. 2.15. Схема обработки запроса

Рассмотрим пример жизненного цикла работы приложения. Каждому шагу соответствует действие.

- Пользователь посылает запрос к сценарию входа index.php;

- Сценарий загружает компоненты от поставщиков из папки vendor, оптимизированные классы и создает экземпляр приложения, для последующей обработки;
- Приложение инициализирует запрос, регистрирует сервисы, миддлвары, добавляет переменные окружения, загружает настройки;
- Приложение ищет подходящий маршрут с помощью инициализированного роутера, которые берет параметры из запроса, если такой найден приложение идет далее, если же нет, то выбрасывает исключение;
- Установленный в роутинге контроллер проверяет правильность введенных данных и если все данные введены верно сценарий выполняется далее, если же нет, то валидатор выбрасывает исключение;
- Выполняются действия в контроллере и возвращается ответ от сервера, он может быть разных типов, с разными статусами;
- Полученный ответ отправляет обработанный результат в браузер пользователя.

Таким образом, мы рассмотрели структуру базового приложения Laravel и теперь, на его основе, можем приступить к созданию своего спроектированного ранее приложения.

ГЛАВА 3. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Разрабатываемое ПО состоит из совокупности правил и процедур, программ, соединенных в одну архитектуру, которая обладает возможностью обработки информации для предоставления пользователю результата.

При разработке логической структуры сайта нужно учитывать доступность важных разделов для пользователей. Посетители не должны искать необходимые разделы, они должны находится под рукой и быть на виду. Пользователь не должен совершать более двух-трех кликов мышкой, чтобы попасть в нужный раздел.

Разработка структуры веб-приложения предполагает использование отношения тех или иных моделей и категорий данных к тем или иным категориями данных, а те к категории к вышележащим категориям.

Структуру представляют в виде чертежа (блок-схемы), на котором каждая страница отображается отдельным прямоугольником. Связи между ними показывают схему переходов по страницам.

Также создают каркасы главной и основных типовых страниц, показывающие расположение текста и графики на странице, а также то, как пользователи будут работать с этими элементами. Каркас страницы должен предполагать возможности последующего расширения.

Разработка структуры включает всё, что касается его содержания и информационной стратегии, определяющей, как должна быть организована подача информации, чтобы пользователи могли быстро и легко её найти. Первоочередной задачей на данном этапе является создание структуры, отражающей взаимосвязи типовых страниц и их наиболее значимые функциональные возможности.

Более эффективную работу сайта обеспечивает соблюдение принципа: «пользователю удобнее добираться до нужной ему страницы максимум за два щелчка мыши». Поэтому обычно рекомендуется использовать не более двух уровней вложенности в пределах каждой группы элементов.

Завершив формирование облика сайта, компоновку страниц и определив размещение содержимого, переходят к следующему этапу web-разработки — к визуальному оформлению [3].

Начало работы пользователей программной системы следует начинать со страницы авторизации.

После авторизации администратору должны быть доступны управления моделями и структурами данных. Наилучшей реализацией такой идеи будет группировка всех категорий в одном месте кабинета, поэтому добавление главной страницы обосновывается именно этим предположением.

Каждая сущность должна иметь отдельную категорию для редактирования данных, что позволит гибко и независимо управлять ими.

3.1 ПОДКЛЮЧЕНИЕ К БАЗЕ ДАННЫХ

Для работы приложения в первую очередь требуется установить соединение с созданной ранее базой данных.

PDO — является расширением для PHP. Оно предоставляет разработчику возможность просто и универсально получать доступ к различным СУБД.

PDO предлагает единые методы для работы с различными базами данных, хотя текст запросов может немного отличаться. Так как многие СУБД реализуют свой диалект SQL, который в той или иной мере поддерживает стандарты ANSI и ISO, то при использовании простых запросов можно добиться совместимости между различными языками. На практике это означает, что можно достаточно легко перейти на другую СУБД, при этом, не меняя или частично изменяя код программы.

Скорость работы и масштабируемость: PDO не использует абстрактных слоёв для подключения к БД, наподобие ODBC, а использует для разных БД их «родные» драйверы, что позволяет добиться высокой производительности. В настоящее время для PDO существуют драйверы

практически ко всем общеизвестным СУБД и интерфейсам. Так же позволяет работать сразу с несколькими базами данных одновременно [4].

3.2 СОЗДАНИЕ МОДЕЛЕЙ ДАННЫХ

Модель представляет собой отдельный объект данных. Модель представляет из себя запись таблицы базы данных. Каждое поле объекта данных представляется атрибутом модели. Каждый атрибут имеет текстовую метку и может быть проверен на корректность, используя набор правил.

Модель должна содержать всю бизнес-логику вашего приложения. Или, другими словами, то, как приложение взаимодействует с базой данных.

Бизнес-логика — это:

- Объекты реального мира, которые используются в вашем приложении;
- То, как эти объекты взаимодействуют друг с другом;
- Набор правил для доступа к этим объектам и для их обновления.

Мы должны хранить информацию о наших пользователях, и поэтому нам нужна модель User и таблица User в базе данных.

Пользователям надо будут вводить имя, адрес электронной почты и пароль, а также другие детали профиля. Чтобы удостовериться, что они вводят правильно отформатированные данные, должна быть проверка их ввода.

Пользователи смогут создавать сообщения. Пользователь может иметь много сообщений, и каждое сообщение должно принадлежать пользователю.

Это основные особенности работы моделей в приложениях MVC. По существу, для каждой важной вещи в приложении, вероятно, потребуется модель. Вам, возможно, понадобится проверять данные, используемые в вашей модели, а также здесь должна быть вся логика, отвечающая за взаимодействие моделей друг с другом.

Система объектно-реляционного отображения (ORM) Eloquent — красивая и простая реализация шаблона ActiveRecord в Laravel для работы с базами данных. Каждая таблица имеет соответствующий класс-модель, который используется для работы с этой таблицей. Модели позволяют запрашивать данные из таблиц, а также вставлять в них новые записи.

Рассмотрим пример создания модели на основе таблицы «Пользователи» см. приложение 1.

В каждой модели, мы указываем поля типа данных Date, поля, которые мы можем массово заполнять, а также поля, которые необходимо скрыть от пользователей.

Теперь, когда готова модель, можно приступать к написанию кода для работы с ней. Всю логику обработки мы помещаем в действие контроллера.

3.3 СОЗДАНИЕ КОНТРОЛЛЕРОВ

Вместо того, чтобы определять всю логику обработки запросов в одном файле routes.php, вы можете организовать её с помощью классов контроллеров. Контроллеры могут группировать связанную с обработкой HTTP-запросов логику в отдельный класс. Контроллеры хранятся в папке app/Http/Controllers.

Важно помнить, что при определении маршрута контроллера нам не надо указывать полное пространство имён контроллера, а только ту часть имени класса, которая следует за «корнем» пространства имён — App\Http\Controllers. По умолчанию RouteServiceProvider загрузит файл routes.php вместе с группой маршрутизации, содержащей корневое пространство имён контроллера.

Если вы решите разместить свои контроллеры, используя пространства имён PHP, в поддиректориях App\Http\Controllers, то просто используйте конкретное имя класса относительно корня пространства имён App\Http\Controllers [5].

Контроллеры создаем для каждой сущности.

3.4 СОЗДАНИЕ ВИДОВ

Виды или как их называют представления являются частью MVC архитектуры, они отвечают за данные которые будут представлены конечному пользователю. Обычно они создаются в виде шаблонов, которые содержат HTML или PHP код, а также код обрабатываемый шаблонизатором. В Laravel вид представляет из себя Blade шаблон, состоящий из тегов шаблонизатора Blade, а также при желании из PHP и HTML кода. Blade шаблоны генерируют динамический контент, как, например, заголовок страницы, формы, различные поля, PHP код можно исполнять так же для генерации динамического контента, тогда как обработчики фреймворка Laravel организует полученные данные в готовую html страницу и отдают её пользователю.

3.5 ФУНКЦИОНАЛЬНОЕ ТЕСТИРОВАНИЕ

Создавая информационные системы, часто можно встретить дефекты и ошибки, которые возникают в условиях ограниченного времени, завышенных требований к продукту. Данное явление вполне ожидаемое, поэтому обязательный этап, который следует после разработки любого программного обеспечения это его тестирование. Тестирование программного обеспечения – это проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом.

В более широком смысле, тестирование – это одна из техник контроля качества, включающая в себя действия по планированию работ, проектированию тестов, выполнению тестирования и анализу полученных результатов.

Функциональное тестирование является одним из основных видов тестирования написанного ПО. Его задачи сводятся к установлению соответствия созданного ПО с исходными функциональными требованиями

от закачика. Проведение данного вида тестирования позволяет проверить разработанное программное обеспечение, его возможность решать задачи, поставленные пользователями.

Функциональное тестирование происходит по заранее подготовленным для него тестовым сценариям и все найденные ошибки заносятся в баг-трекинговую систему компании.

Как правило, функциональное тестирование производят на нескольких уровнях:

Интеграционное тестирование. Данное тестирование производят после того как выполнено компонентное тестирование. Оно направлено на поиск дефектов различных подсистем на уровне обмена и управления данными.

Компонентное тестирование. Данное тестирование направлено на отдельные компоненты продукта. Оно фокусируется на специфике, функциональных особенностях продукта, назначении.

Так как мы имеем дело с небольшим приложением, при нахождении ошибок можем сразу же исправить их.

На рис. 3.1. изображена страница с пользователями системы. В системе есть несколько типов пользователей, партнеры, менеджеры, посредники, исполнители, администраторы. Для администратора системы доступны возможности редактирования данных, удаления пользователей, просмотра их основных показателей, показаны их долги, статус, внесения прочей информации о пользователях. Так же имеется возможность создавать для каждой группы пользователей персонализированные сообщения, в которых можно указать правила пользования сервисом, условия поощрения и наказания за определенные успехи или проступки в системе.

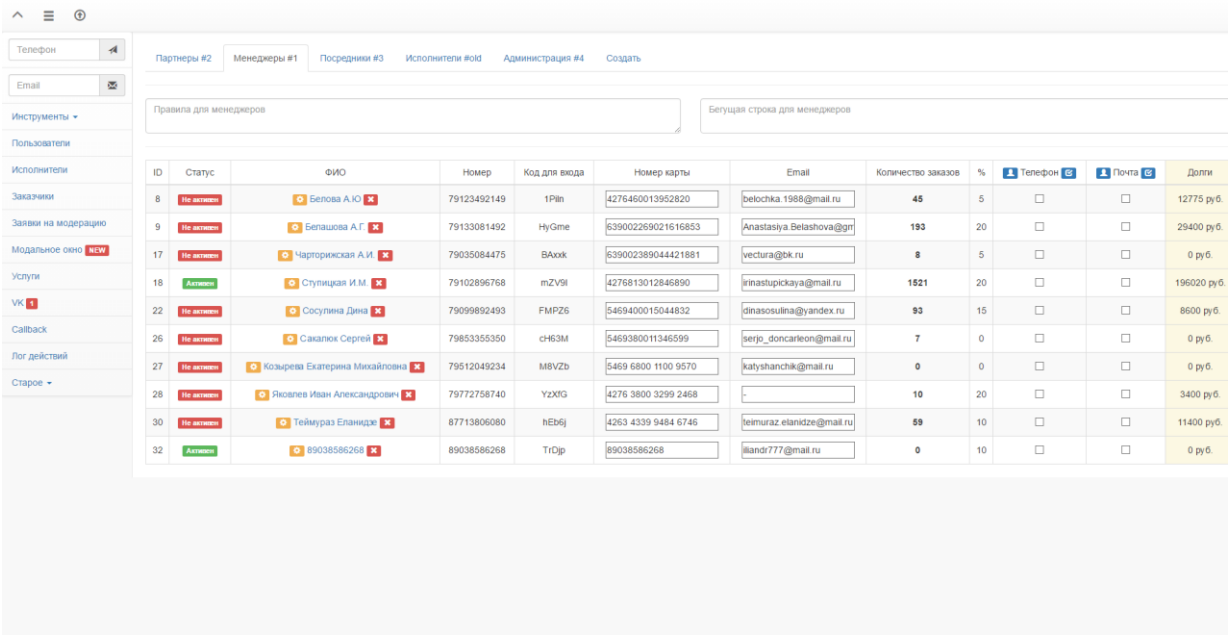

Рис. 3.1. Пользователи

На рис. 3.2 изображена страница исполнителей. На данной странице можно просматривать предлагаемые исполнителями услуги, удалять их, отключать. Так же имеется возможность отключать исполнителей, добавлять исполнителей, и экспортировать данные с этой страницы в excel.

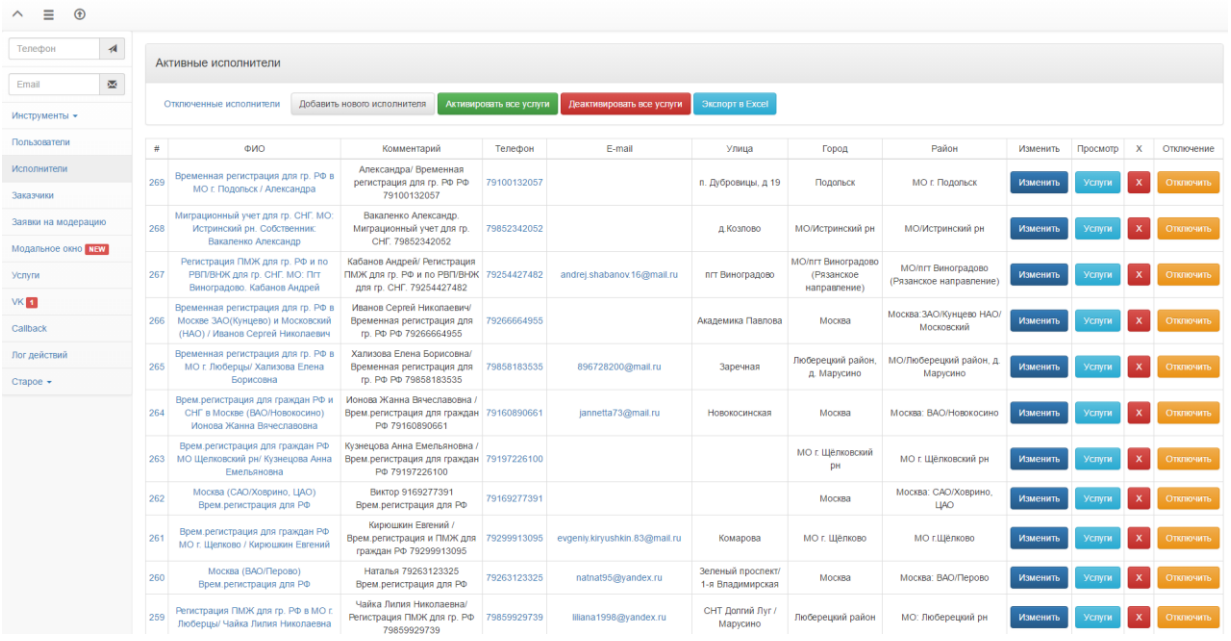

Рис. 3.2. Исполнители

На рис. 3.3 изображена страница с заявками на модерацию. Сюда попадают заявки, формируемые менеджерами. На данной странице можно редактировать данные заявок, выставлять им рейтинг, устанавливать статусы заявок, просматривать прикрепленные к ним документы.

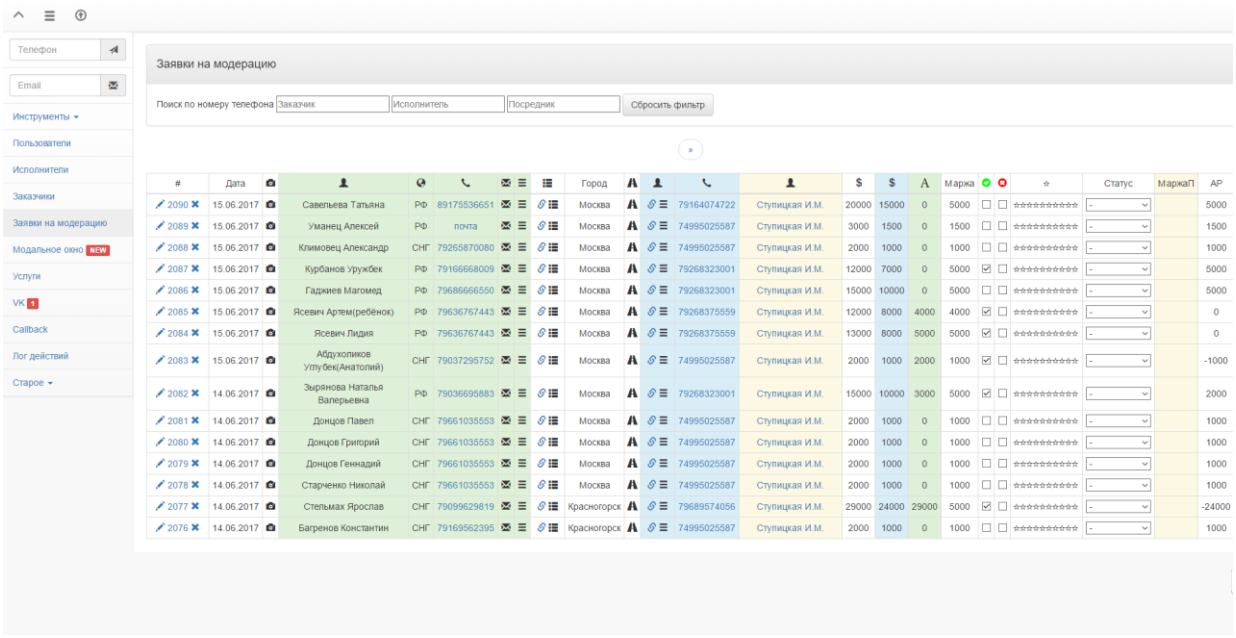

Рис. 3.3. Заявки на модерацию

На рис. 3.4. изображена страница с модальным окном. Данная страница позволяет собирать данные о новых клиентах и впоследствии работать с ними отвечая на вопросы клиентов в разных категориях. Так как есть возможность встраивания данной системы на любой сайт, есть возможность настройки данного окна, кастомизации цвета под любой сайт или лендинг, настройки размеров. На данной странице так же можно удалить вопросы пользователей.

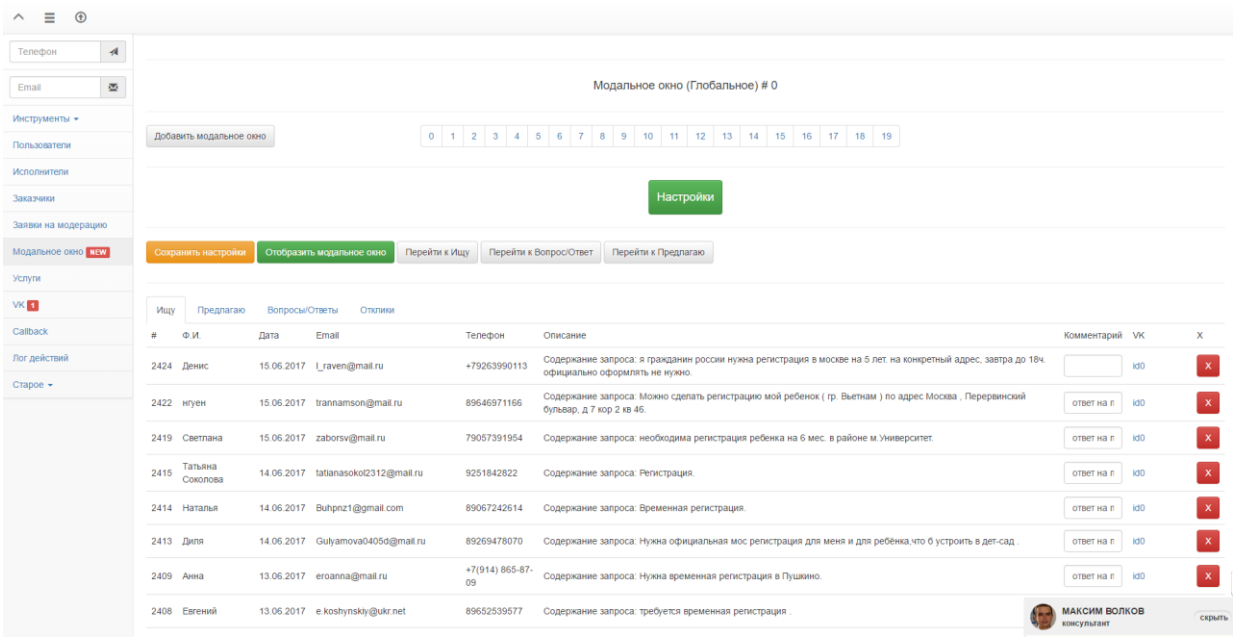

Рис. 3.4. Модальное окно

На рис. 3.5 изображена страница услуг. На данной странице работают менеджеры. Они могут создавать заявки разговаривая с клиентом по телефону. На выбор у них доступны фильтры, устанавливая которые они могут отбирать услуги для пользователей.

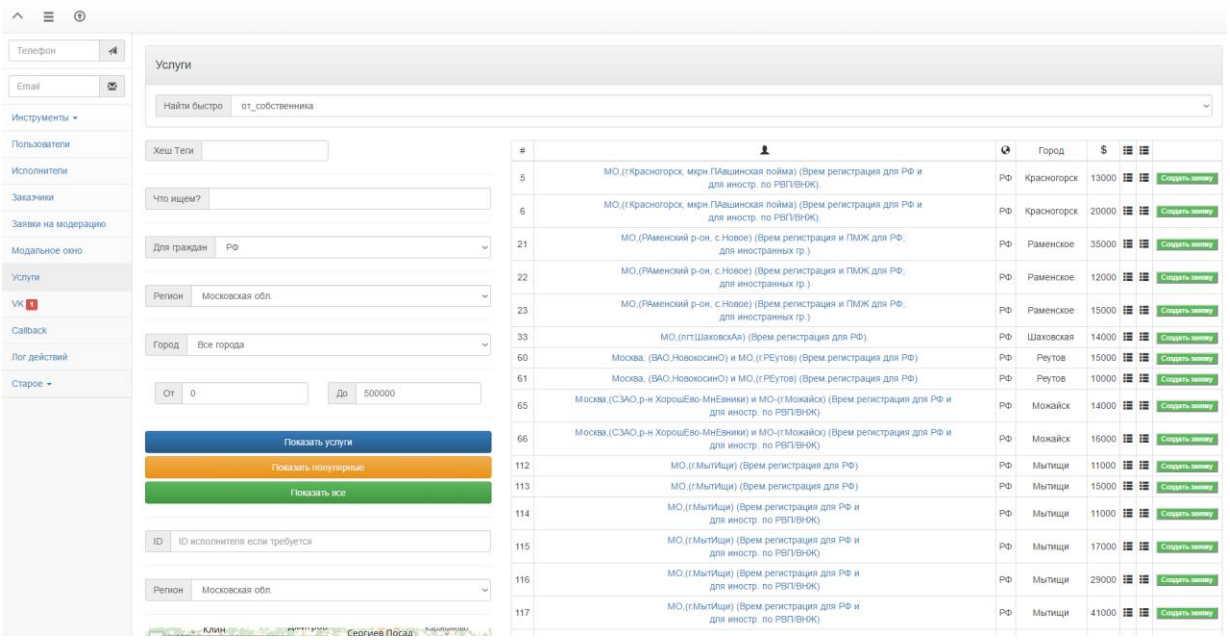

Рис. 3.5. Услуги

На рис. 3.6 изображены логи действий пользователей. Каждое действие в системе записывается и хранится месяц, чтобы не допустить халатности со стороны менеджеров и сотрудников. Если в системе была сделана ошибка в любой момент можно узнать кто её допустил.

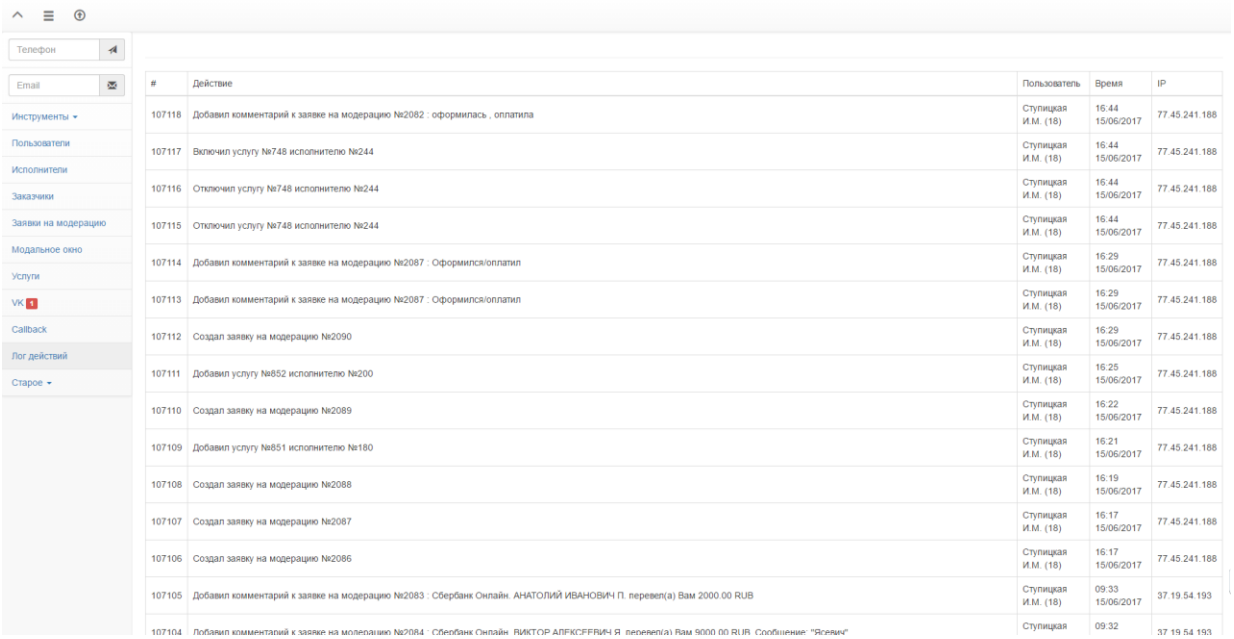

Рис. 3.6. Логи действий

На рис. 3.7. изображена панель управления некоторыми функциями CRM приложением. Они позволяют очищать кеш, импортировать и экспортировать объявления с различных интернет ресурсов.

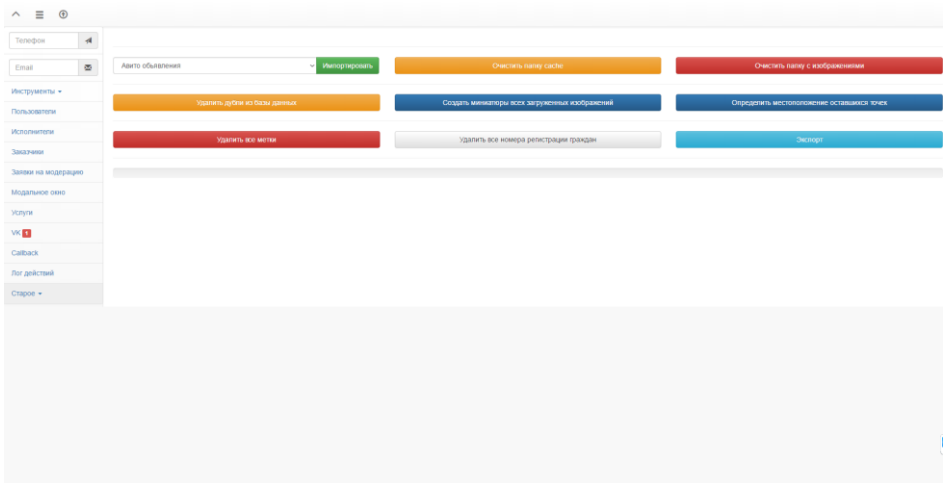

Рис. 3.7. Управление CRM приложением

На рис. 3.8 изображена страница с заказчиками, на данной странице собираются лиды с социальных сетей, с страниц на которых можно оставить заявку на консультацию. Так же для каждого лида можно сгенерировать определенный токен ключ активации, который он может использовать и получить скидку заполнив информацию о себе.

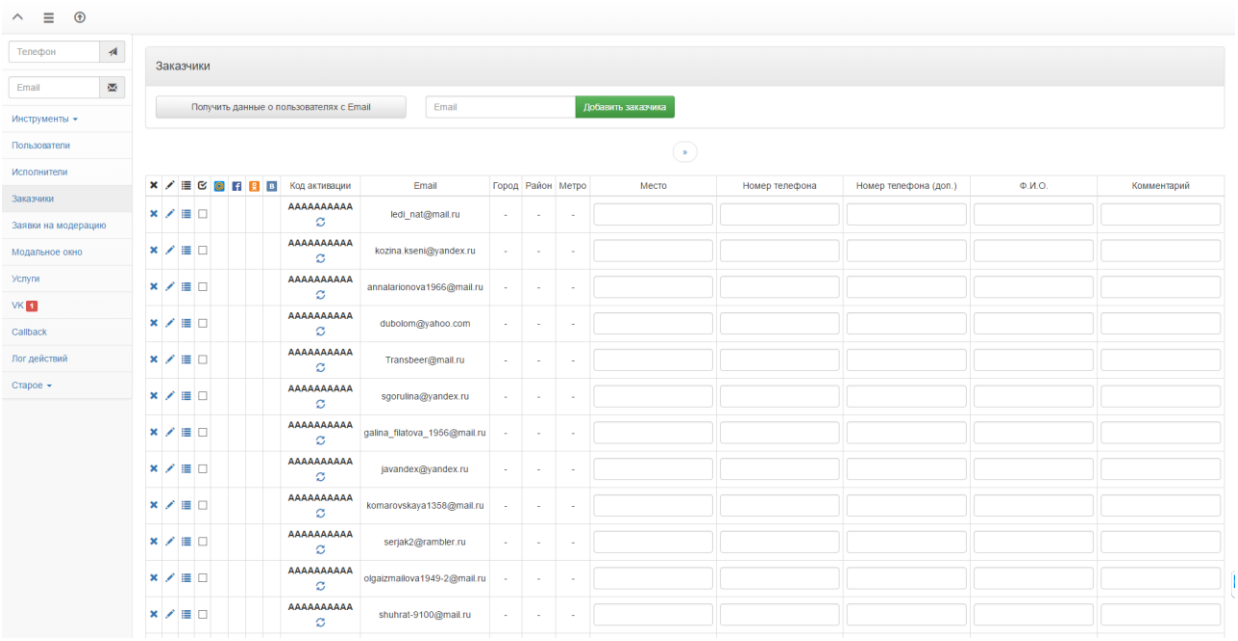

Рис. 3.8. Заказчики

Таким образом готовое CRM приложение было протестировано, мы убедились, что все работает правильным образом, дефектов не выявлено.

ЗАКЛЮЧЕНИЕ

В процессе работы над выпускной квалификационной работой было спроектировано и разработано CRM приложение которое осуществляет автоматизированный учет деятельности на предприятии. В данном приложении присутствует разделение пользователей на группы с различным уровнем доступа к элементам системы. В разработанном приложении у администратора имеются возможности регистрации пользователей в системе, назначение им уровня доступа, создания заявок, документов, просмотра логов действий пользователей.

В ходе данной работы были созданы представления, контроллеры, модели, были приобретены практические и теоретические знания и навыки в области создания баз данных, веб программировании, а также создания веб сервисов и приложений.

Предлагаемое CRM приложение отличается гибкостью, скоростью, безопасностью. Все использованные технологии являются бесплатными и свободно распространяемыми.

Спроектированная и разработанная система позволяет эффективно управлять учетной деятельностью предприятия без затрат лишних ресурсов.

Таким образом, задачи, поставленные в начале выпускной квалификационной работы, были выполнены, а цель – достигнута.

В ходе написания курсовой работы нами были изучены многие программные средства, такие как HTML, PHP, JavaScript и СУБД MySQL. Были разработаны логическая, физическая модели базы данных.

В конечном итоге было реализовано приложение, взаимодействующее с разработанной базой данных: выполняющее подключение к ней, позволяющее добавлять новые данные, совершать поиск среди уже содержащихся в БД данных, выводить сводные данные с учетом связей, а также выполнять удаление.

Был разработан удобный для пользователя интерфейс, позволяющий взаимодействовать с БД.

Работоспособность разработанного приложения проверена на тестовых примерах, приведены сведения об интерфейсе, показаны варианты обработки данных.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Руководство по проектированию реляционных баз данных (1-3 часть из 15) [перевод] https://habrahabr.ru/post/193136/
2. Проектирование структуры базы данных методом сущность-связь http://www.studfiles.ru/preview/6060415/
3. Основные этапы разработки web-приложений http://www.rusnauka.com/16_ADEN_2011/Informatica/3_85389.doc.htm
4. Базы и банки данных – методичка по лабораторным работам http://bib.convdocs.org/v1903/?download=file
5. Контроллеры | Laravel по-русски https://laravel.ru/docs/v5/controllers
6. Белл Чарльз Обеспечение высокой доступности систем на основе MySQL / Белл Чарльз – Русская Редакция – 2012 – 624 с.
7. Дебуа П. MySQL: Сборник рецептов / Дебуа П. Символ-Плюс – 2007 - 1056 с.
8. Электронный ресурс документация PHP http://www.php.net/manual/ru/.
9. Электронный ресурс документация Laravel https://laravel.com/docs/master.
10. «Web-дизайн. Удобство использования Web-сайтов», Якоб Нильсен и ХоаЛоранжер– СПб.: Эксмо, 2015. – 800 с.
11. «Разработка веб-приложений с помощью PHP и MySQL», Люк Веллинг, Лаура Томсон/ Пер. с англ. — 8-е изд. — М.: Вильямс, 2005, 1328 с
12. «Большая книга CSS», Пер с англ./Крис Джамса, Конрад Кинг, Энди Андерсон - М.: ООО "ДиаСофтЮП", 2005.- 672 с.
13. Дунаев В. Самоучитель JavaScript, 2-е изд. – СПб.: Питер, 2005. – 395 с.
14. Создание Web-страниц и Web-сайтов. Самоучитель: [учеб.пособие] / под ред. В. Н. Печникова. – М.: Изд-во Триумф, 2006.— 464 с.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ 1.

```
<?php
namespace App;
use Illuminate\Auth\Authenticatable;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Foundation\Auth\Access\Authorizable;
use Illuminate\Contracts\Auth\Authenticatable as AuthenticatableContract;
use Illuminate\Contracts\Auth\Access\Authorizable as AuthorizableContract;
use Illuminate\Auth\Passwords\CanResetPassword;
use Illuminate\Contracts\Auth\CanResetPassword as CanResetPasswordContract;
use Illuminate\Notifications\Notifiable;

class User extends Model implements AuthenticatableContract, AuthorizableContract,
CanResetPasswordContract
{
    use Authenticatable, Authorizable, CanResetPassword, Notifiable;

    /**
     * The attributes that should be mutated to dates.
     *
     * @var array
     */
    protected $dates = [
        'created_at',
        'updated_at',
    ];

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
```

```
protected $fillable = [
    'id',
    'status',
    'information',
    'email',
    'password',
    'phone',
    'type_id',
    'phone_additional',
    'first_name',
    'last_name',
    'middle_name',
];

/**
 * The attributes that should be hidden for arrays.
 *
 * @var array
 */
protected $hidden = [
    'id',
    // 'status',
    // 'information',
    // 'email',
    'password',
    // 'phone',
    // 'type_id',
    // 'phone_additional',
    // 'first_name',
    // 'last_name',
    // 'middle_name',

    'created_at',
    'updated_at',
];
```

}

ПРИЛОЖЕНИЕ 2.

ПРИЛОЖЕНИЕ 3.

ПРИЛОЖЕНИЕ 4.

ПРИЛОЖЕНИЕ 5.

ПРИЛОЖЕНИЕ 6.

ПРИЛОЖЕНИЕ 7.

ПРИЛОЖЕНИЕ 8.

ПРИЛОЖЕНИЕ 9.