МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»
Кафедра IT-инжиниринг

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой
PhD, доцент
_____ Т.С. Картбаев
«____» _____ 2019 г.

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Разработка системы управления взаимоотношениями с клиентами на платформе Node.js

Специальность: 5В070400 – «Вычислительная техника и программное обеспечение»

Выполнил: Берик С.Е. Группа: ВТ-15-2
Научный руководитель: доц. Аманбаев А.А.

Консультанты:
по экономической части: к.э.н., профессор _____ Ж.Г. Аренбаева
«27» мая 2019 г.

по безопасности
жизнедеятельности: д.т.н., ст. преп. _____ Ш.Ш. Бекбасаров
«14» мая 2019 г.

по применению
вычислительной техники: ст. преп. _____ М.Н. Майкотов
«14» мая 2019 г.

Нормоконтролер: ст. преп. _____ А.А. Айтказина
«15» 05 2019 г.

Рецензент: асс. проф. каф _____ Н.К. Мукажанов
«____» _____ 2019 г.

Алматы 2019

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»

Институт систем управления и информационных технологий

Кафедра IT–инжиниринг

Специальность 5В070400 – «Вычислительная техника и программное обеспечение»

ЗАДАНИЕ
на выполнение дипломного проекта

Студенту Берік Султанбек Ерікұлы

Тема проекта: Разработка системы управления взаимоотношениями с клиентами на платформе Node.js

Утверждена приказом по университету № 33 от «01» марта 2019 г.

Срок сдачи законченного проекта «24» мая 2019 г.

Исходные данные к проекту (требуемые параметры результатов исследования (проектирования) и исходные данные объекта): Руководство системы менеджмента качества на предприятии; международные стандарты ИСО–9001, данные преддипломной практики.

Перечень вопросов, подлежащих разработке в дипломном проекте, или краткое содержание дипломного проекта:

- аналитическая часть;
- проектная часть;
- экспериментальная часть;
- экономическая часть;
- безопасность жизнедеятельности;
- приложение А. Техническое задание;
- приложение Б. Листинг программы;
- приложение В. Акт внедрения.

Перечень графического материала (с точным указанием обязательных чертежей): представлены 11 таблиц, 53 иллюстрации.

Основная рекомендуемая литература:

1 В. Васвани. MySQL: использование и администрирование = MySQL Database Usage & Administration. – М.: «Питер», 2011. – 368 с.

2 Итан Браун. Веб–разработка с применением Node и Express. Полноценное использование стека – Санкт–Петербург: Питер, 2017. – 336 с.

3 Пятая редакция спецификации ECMA–262

4 Джо Брокмайер, Ди–Анн Лебланк, Рональд Маккарти, мл. Маршрутизация – М.: «Вильямс», 2002. – С. 240.

Консультации по проекту с указанием относящихся к ним разделов проекта

Раздел	Консультант	Сроки	Подпись
Экономическая часть	Аренбаева Ж.Г.	01.03.2019 – 27.05.2019	
Безопасность жизнедеятельности	Бекбасаров Ш.Ш.	05.03.2019 – 14.05.2019	
Программное обеспечение	Майкотов М.Н.	02.05 – 14.05	
Нормоконтролер	Айтказина А.А.	02.04 – 15.05.19	

ГРАФИК
подготовки дипломной проекта

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Аналитическая часть	01.02 – 14.02.2019	Выполнил
Проектная часть	13.03 – 30.03.2019	Выполнил
Экспериментальная часть	02.04 – 14.05.2019	Выполнил

Дата выдачи задания «25» октябрь 2018 г.

Заведующий кафедрой _____ Т.С. Картбаев

Научный руководитель проекта _____ А.А. Аманбаев

Задание принял к
исполнению студент _____ С.Е. Берік

Аңдатпа

CRM клиенттік қарым–қатынасты басқаруды қамтиды және бағдарламалық жасақтама болып табылады, ол клиенттің компаниямен қарым–қатынасын автоматтандыратын және ол туралы ақпаратты жинауға, сақтауға және өңдеуге мүмкіндік беретін қосымшалардың жиынтығы болып табылады. Бұл ақпарат түрлі көздерден (маркетинг, сату, қызмет көрсету және т.б.) алынған. Болашақта бұл компания қызметкерлеріне тұтынушылардың қажеттіліктерін жақсы түсіну және тұтынушылармен және серіктестермен қарым–қатынастарды тиімді түрде қалыптастыруға қажетті ақпаратпен қамтамасыз етеді. Сондай–ақ, бұл жүйе Сізге бірнеше ақпараттық арналарды (Интернет, телефон және факсимильді байланыс).

Аннотация

CRM предполагает управление взаимоотношениями с клиентами и представляет собой программное обеспечение, набор приложений, автоматизирующих процессы взаимоотношений компании с клиентом и позволяющих собирать, хранить и обрабатывать информацию о нем. Эту информацию получают из различных источников (отделы маркетинга, продаж, сервисного обслуживания и др.). В дальнейшем это обеспечивает сотрудников компании сведениями, необходимыми для лучшего понимания запросов потребителей и для эффективного построения взаимоотношений со своими покупателями и партнерами. Также данная система позволяет «связать» покупателей и работников организации путем использования многочисленных информационных каналов (Интернет, телефонная и факсимильная связи).

Abstract

CRM involves customer relationship management and is a software, a set of applications that automate the company's relationship with the customer and allow you to collect, store and process information about it. This information is obtained from various sources (marketing, sales, service, etc.). In the future, this provides company employees with the information necessary to better understand consumer needs and effectively build relationships with their customers and partners. Also, this system allows you to "connect" customers and employees of the organization by using multiple information channels (Internet, telephone and facsimile communications).

Содержание

Введение

Все чаще можно слышать от владельцев компаний, директоров и руководителей среднего звена о снижении количества заказов, спаде продаж, замораживании проектов и инвестиций. Для многих компаний актуальной стала тема сокращения персонала. «Легкие деньги» и достаточное количество клиентов, желающих купить, к которым за годы бурного роста экономики привыкли многие казахстанские компании, закончились.

В новых условиях компаниям продавать свои товары и услуги и удерживать клиентов стало намного сложнее. Клиенты стали тщательнее «считать деньги», торговаться и экономить. Покупательский спрос смещается в сторону более дешевых предложений, клиенты отказываются от всего дополнительного и необязательного, сопутствующих товаров. Усиливается конкуренция на рынках внутри страны и с иностранными компаниями.

В таких внешних условиях казахстанским компаниям необходимо изменение тактики работы. Главными проблемами, с которыми могут или уже столкнулись предприятия реального сектора казахстанской экономики, являются спад спроса на товары и услуги, отток клиентов, снижение прибыли. Поэтому одними из ключевых задач руководителя бизнеса становятся удержание клиентов, снижение затрат и повышение эффективности бизнес–процессов компании. Тактику активного роста и быстрого освоения рынка должны заменить экономия (срезание затрат) и повышение эффективности каждой операции основных бизнес–процессов компании.

Слово «инновации» сейчас у всех на слуху, в Казахсане строится инновационная экономика. Однако по оценкам правительства производительность труда в казахстанских компаниях ниже, чем в аналогичных западных компаниях до 5 раз, а товары и услуги имеют большую себестоимость по сравнению с импортными. При таких показателях эффективности казахстанские компании, скорее всего, не смогут конкурировать с представителями западного бизнеса.

Основной же стратегией успешного существования и дальнейшего развития современных компаний постепенно становится эффективное управление взаимоотношениями с клиентами. Ориентация компаний на усовершенствование отношений с клиентами обусловлена рядом тенденций, в частности усилением конкуренции, повышением требований покупателей к качеству предлагаемых продуктов и уровню сервиса, снижением эффективности традиционных маркетинговых средств, а также появлением новых технологий взаимодействия с клиентами и функционирования подразделений компании. Знание своих клиентов и удовлетворение запросов и потребностей каждого из них могут позволить компании получить новые возможности для сбыта товаров и услуг и стать ключевым фактором устойчивого развития и источником долгосрочного конкурентного преимущества компании на рынке.

Западный опыт показывает, что высокая эффективность работы с

покупателями обеспечивается за счет принятия компанией концепции по управлению взаимоотношениями с клиентами, получившей название концепции CRM (Customer Relationship Management).концепция позволяет «интегрировать» клиента в сферу организации – фирма получает максимально возможную информацию о своих клиентах и их потребностях и, исходя из этих данных, строит свою организационную стратегию, которая касается всех аспектов ее деятельности: производства, маркетинга, продаж, обслуживания и прочего.

Целью данного дипломного проекта является разработка и внедрение CRM–системы в компанию ТОО «Start IT». Данная система будет автоматизировать работу персонала компании, упрощать сбор данных о клиентах и их потребностях от компании.

1 Аналитический обзор современного состояния исследуемой проблемы

Любой бизнес, будь он малым или крупным, требует ведения учётного контроля. Как правило, выделяют: бухгалтерский, складской, кадровый, управленческий и клиентский виды учёта. И если первые 4 вида можно организовать, внедрив какую–либо ERP–систему (например, 1С), то с клиентскими дела обстоят несколько сложнее ввиду необходимости сочетания множества факторов.

Вместо табличек Excel, мессенджеров, множества документов и беготни по кабинетам остается один–единственный сервис. В него входят программы для сбора данных о клиентах, управления сделками, контроля за менеджерами, аналитики и прогнозирования. Он упрощает рутину, ускоряет принятие правильных решений и исключает ошибки.

Самый простой способ, существующий уже несколько тысяч лет, но до сих пор актуальный – письменный. Многие начинающие бизнесы ведут свой учёт в виде ручных записей на бумагу (например, блокнот, тетрадь или специальный журнал). Недостатки подобного подхода очевидны: записи можно потерять и испортить (залить кофе, испачкать, помять), а функциональные возможности желают лучшего.

Конечно же, можно воспользоваться стандартным Excel–документом и совершать записи всех взаимодействий с людьми в простой электронной таблице. А разместив его на каком–либо облачном сервисе, можно реализоваться и многопользовательский формат работы всей командой. При этом будет ряд проблем с узким функционалом и высокой вероятностью возникновения ошибок. Как показывает практика, чем больше людей имеет доступ к общему документу и регулярно с ним работает, тем больше проблем периодически возникает в связи с человеческим фактором (удалил не ту строчку, вписал запись не туда, куда нужно, некорректно сгруппировала данные и т.п.).

Чтобы повысить базовые возможности Excel–документа и получить действительно рабочие таблицы, позволяющие контролировать все взаимоотношения с клиентами, придётся поработать с макросами и визуальным офисным программированием, что изначально доставляет сложность и требует участия профессионального профильного программиста.

Гораздо проще задействовать CRM–систему. Разработка системы учета клиентов с нуля позволяет «заточить» её под свои нужды, реализовав только тот функционал, который необходим. Однако, создание подобной CRM трудоёмко и требует значительных финансовых вложений. Позволить такое «удовольствие» могут только крупные действующие бизнесы, уверенно себя чувствующие на рынке.

Для малого бизнеса, менее проблематично и финансово выгоднее обходится внедрение уже готовой CRM–системы, изначально имеющей весь

необходимый базовый функционал и опции. Остаётся лишь только её грамотно настроить и обучить персонал.

1.1 История CRM – систем

CRM (Customer Relationship Management – система управления взаимоотношениями с клиентами) довольно популярная тема на протяжении последних 10–15 лет. Сегодня трудно представить эффективно работающий конкурентный бизнес без отлаженной CRM системы.

1970 – начинают появляются первые компьютеры с кремниевыми чипами. Ранее производство товаров и предоставление услуг было очень однотипным, клиенты особо ничего не выбирали и покупали то, что есть. До 80–х годов и появления CRM мало какие компании пытались удерживать своих клиентов, т.к. с их ресурсами и возможностями они всегда могли изменить рынок сбыта, если это нужно было.

Количество специализированных CRM (ориентированных на управления продажами) к началу 80–х можно было пересчитать по пальцам. Но проходило время, и развитие компьютерной техники позволило значительно снизить стоимость вычислительных операций.

В 1987 году, основатель Contact Software International и SalesLogix Corporation, Пэт Салливан (Pat Sullivan), выпустил первый коммерческий софт для управления контактами, который получил название «ACT!».

ACT – до перевоплощения переводилась как Automated Control Technology. В последствии система получила другой перевод – Automated Contact Tracking (система управления контактами). Основное предназначение системы было – организация и управление процессами продаж и взаимодействия с клиентами. Программа позволяла отслеживать развитие взаимоотношений с клиентами, анализировать информацию для более эффективных продаж и развития лояльности клиентов.

SFA – Автоматизация процесса продаж.

В 1990 году на рынке появляются инновационные системы Управление продажами объединенные с системами менеджмент контактов такие как Saratoga Systems и Brock Systems Эти системы автоматизировали стандартный процесс продаж (sales force automation –SFA), что позволяло контролировать этапы сделок и следить за их эффективностью.

В 1993 году Томас Зибель (Thomas Siebel) учредил Siebel Systems. Зибель тратил почти все свое время на коммуникацию с клиентами и понимание того, что им нужно в результате кропотливой исследовательской работы появилась программа для автоматизации торговли и продаж, а чуть позже для автоматизации маркетинга и поддержки – Siebel Sales Enterprise – Siebel CRM (“Мать всех CRM”).

В 1995 году появился сам термина CRM (Customer Relationship Management) – управление взаимоотношениями с клиентами. Он впервые был

использован в Siebel Systems, для того чтобы отразить специфику этого типа программных продуктов.

CRM – система позволяла собирать и хранить информацию о клиентах, анализировать ее и делать определенные выводы или просто предоставлять эту информацию сотрудникам в удобном виде.

Основная задача CRM: получать на базе собранных данных информацию, которую можно использовать непосредственно для повышения доходности и эффективности ведения бизнеса, формируя на базе этих данных новые и дополнительные услуги для различных групп клиентов.

Использование CRM позволяет продавать клиенту больше товаров и услуг, основываясь на знании того, чего он на самом деле хочет. В CRM–системах учитывается не только личная информация о клиенте, но и сведения, относящиеся к взаимодействию клиента с компанией.

В отличие от предыдущих систем работы с клиентами, новая начала бороться за лояльность клиента. Теперь о клиенте не только собирали информацию, но и использовали её. Именно в это время стали практиковаться различные акции: вручение подарков клиентам, дисконтные скидки и прочее. Бизнес начал понимать, что лояльность клиента – это самое главное.

Первые отечественные разработки CRM массово начали использоваться и развиваться в 2002 году, изначально такие решения претендовали на рынок СМБ компаний (сегмент среднего и малого бизнеса). Примерно в это же время отечественные вендоры программного обеспечения бухгалтерии и ERP включили CRM–модули в свои пакеты.

Первая версия системы Terrasoft CRM была разработана в 2001–2002 годах. Помимо разработки программных продуктов компания занимается ИТ–консалтингом и реализует комплексные проекты по внедрению CRM–стратегии и автоматизации предприятий.

В 2004 году на рынок СНГ приходит Siebel, успех системы был достаточно стремительным, чему способствовали довольно удачная маркетинговая стратегия и тщательный отбор партнеров.

В 2004: произошло еще одно важное событие на рынке появляется CRM система с открытым кодом SugarCRM. SugarCRM была основана как коммерческий стартап на Sourceforge.org. Позднее появилась облачная версия этого продукта. Бесплатный открытый код требовал больших инвестиций в поддержку и сервис. Зарплаты программистов способных качественно поддерживать “бесплатный” софт – были и остаются стабильно высокой.

В 2005 году рынок CRM СНГ был все еще довольно фрагментарным: 14% принадлежало 1С, 11% – SAP CRM, 9% – Oracle/Siebel, 7% – IBM, 6% – SalesLogix, 4% – Microsoft CRM и еще 49% приходилось на более мелких игроков. Однако, из приведенных данных видно, что лидирующие позиции на местном рынке в основном продолжали удерживать западные бренды.

В наши дни к трендам CRM можно отнести – тесную интеграцию CRM систем с социальными сетями для сбора полной информации о клиентах,

работа CRM с Big Data (большими данными), IoT (интернет вещей) и искусственным интеллектом построенном на машинном обучении.

Подводя итоги хочется отметить, один неоспоримый факт – В современном деловом мире правила игры меняются очень динамично. Каждые пять лет у пользователей CRM происходит новое восприятие данного программного продукта и с каждым годом CRM становиться все нужнее и пользуется все большим спросом. Чем больше компания и выше ее активность, тем больше у нее клиентов и задач – соответственно появляется потребность в обработке больших клиентских баз и большого объема данных, а это основная задача CRM. Использование CRM увеличивает объем продаж в разы, поднимает уровень дохода и прибыль организации.

1.2 Анализ существующих систем

В основном у большинства CRM – систем схожий функционал и предназначен для упрощения механизмов взаимодействия с клиентами и позволяет:

– систематизировать сведения о звонках, заявках и прочих видов обращения к компании;

– в автоматизированном режиме выставлять счета своим клиентам;

– планировать дальнейшие шаги;

– оповещать о запланированных шагах (необходимости перезвонить, сбросить коммерческое предложение, встретиться с заказчиком и т.п.);

– записывать телефонные переговоры и все входящие и исходящие звонки;

– управлять бизнес–процессами внутри команды, устанавливая задачи каждого сотруднику, и контролировать их выполнение в режиме реального времени, просматривая статусы и процент выполнения плана;

– формировать лояльность клиента, привязывая его к себе.

1.2.1 CRM – система «Мега План»

Представитель современных CRM, активно развивающийся и занимающий уверенные позиции на рынке СНГ систем учёта клиентов.

Функционал включает в себя:

– управление задачами (постановка, корректировка, отслеживание, напоминание);

– собирает и отображает аналитическую информацию по каждому отделу и специалисту в целом;

– формирует вспомогательную отчётность, помогающую управлять бизнесом эффективнее и принимать верные управленческие решения;

– позволяет контролировать каждый этап сделки (переговоры, переписку, статусы и т.п.);

– имеется мобильное приложение;
– быстрая интеграция с 1С и сервисами IP–телефонии.

1.2.2 Mango CRM

Ещё одно интересное решение в плане ведения учёта взаимоотношений с клиентами. Изначально компания «Mango» специализировалась на облачных АТС, предоставляя услуги виртуальное телефонии. Но появление совершенно нового рынка CRM способствовало их бурному развитию в сторону клиентского учёта.

Основной функционал системы учета клиентов базируется на ведении полного цикла сделки, начиная от обработки звонка, интеграции с сайтом для получения заявок, заканчивая статистикой по завершенным процессам и выполненным задачам.

Если сравнивать все функциональные возможности «Mango CRM» с уже известным нам «МегаПлан», складывается ощущение её явного проигрыша. Более того, кроме как интеграции со своей IP–телефонией и сайтом в ней больше ничего сильно интересного не предусмотрено, что в условиях ведения современного бизнеса, сильно «сковывает руки».

1.2.3 MegaCRM

«MegaCRM» имеет следующие функциональные возможности:

– автоматический сбор и обработка по заранее настроенным правилам обращений со всех возможных источников (звонки, форма связи на сайте, онлайн–чат, заявки, электронные письма и т.д.);

– надёжная защита персональных данных и собранной клиентской базы;

– расширенная аналитика, позволяющая принимать верные управленческие решения;

– практически вся работа ведётся в одном окне, что упрощает адаптацию сотрудников к продукту, делая его интуитивно–понятным;

– имеет планировщик задач, своевременно напоминающий менеджерам про необходимость совершения того или иного действия;

– выставление счетов в несколько кликов;

– управление правами доступа.

1.2.4 Битрикс 24

«Битрикс 24» – это не просто CRM–система. В первую очередь, это огромный портал, охватывающий управление проектами, работу с бизнес–процессами и задачами, взаимодействие с клиентами и ведение продаж.

Минимальный функционал и стартовые возможности можно получить бесплатно. При этом у вас будет:

– доступ к системе на 12 пользователей;
– 5 Гб облачного пространства;
– управление задачами и проектами;
– возможность интеграции с сайтами и интернет–магазинами;
– онлайн–чат;
– планировщик работ.

В платных тарифах добавляются следующие возможности:

– запись телефонных разговоров;
– общение с другими пользователями «Битрикс 24» (внутренний чат);
– построение диаграммы Ганта;
– видеосообщения;
– создание опросов;
– онлайн работа с офисными документами;
– автоматизация бизнес–процессов;
– интеграция с 1С;
– контроль рабочего времени;
– анализ KPI;
– HR–отчёты;
– учёт рабочего времени;
– подключение шаблонов;
– импорт и экспорт данных;
– построение плана продаж;
– работа с социальными сетями;
– распознание «визиток»;
– автопроверка дубликатов и задвоения информации;
– быстрое выставление счетов и синхронизация с онлайн–кассами;
– отправка электронных писем и смс–сообщений из CRM;
– email–рассылка из системыю.

2 Описание Проекта

На данный момент автоматизация бизнес–решений очень важна для любого предприятия, продвигающего свой продукт на рынок. Особенно она важна для малого и среднего бизнеса, так как, конкуренция в данной части очень высока.

2.1 Выбор средств и технологий

В данном подразделе описываются какие технологии и аппаратные средства были использованы при разработке дипломного проекта, их ключевые достоинства и недостатки, почему были выбраны именно эти технологии, и за какую часть они отвечают.

2.1.1 Node.js

Node или Node.js – программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения.

Node.js наделяет JavaScript функцией взаимодействовать с устройствами ввода–вывода через свой API (написанный на C++), подключать другие внешние библиотеки, созданные на разных языках, обеспечивая вызовы к ним из JavaScript–кода. Node.js используестя чаще всего на сервере.

Скачать установочные файлы Node.js можно на официальном ресурсе nodejs.org. Установка и настройка ПО не занимает много времени, и довольно таки не сложная для начинающих пользователей.

Асинхронное программирование за последнее время стало не менее развитым направлением, чем классическое параллельное программирование, а в мире JavaSript, как в браузерах, так и в Node.js, понимание его приемов заняло одно из центральных мест в формировании мировоззрения разработчиков.

Среди преимуществ Node.js стоит выделить то, что он дает возможность пользоваться JavaScript на стороне клиента и на стороне сервера. Разработчики могут подключать специальное ПО JavaScript: JQuery, V8, JSON и управляемое событиями программирования. Также выделяют особые экосистемы, которые работают “поверх” Node.js, среди них веб–инфраструктура Express (быстрый, гибкий, минималистичный веб–фреймворк для приложений Node.js).

К недостаткам можно отнести следующий пункт: пользоваться неблокируемым вводом/выводом при ограниченных процессорных ресурсах Вы не сможете. Для этого используйте специальные архитектурные премы, например, разветвление пула процессов.

Node.js – довольно молодой, но уже популярный проект. Начал разрабатываться только в 2009 году, на данный момент используется во многих известных ресурсах.

2.1.2 Vue.js

Vue.js – JavaScript–фреймворк с открытым исходным кодом для создания пользовательских интерфейсов. Легко интегрируется в проекты с использованием других JavaScript–библиотек. Может функционировать как веб – фреймворк для разработки одностраничных приложений в реактивном стиле. Реализация web UI сталкивается со все более сложными задачами, требующими использования все более сложных инструментов. Добавим сюда тренд ухода от MVC архитектуры приложений и получим довольно интересную тенденцию.

Преимущества Vue.js:

– усиленный HTML. Это означает, что Vue.js имеет много схожих с Angular характеристик. Это может помочь оптимизировать обработку HTML–блоков с использованием разных компонентов;

– подробная документация. Vue.js имеет очень хорошую документацию, которая может увеличить скорость обучения разработчиков и сэкономить много времени на разработку приложения с использованием базовых знаний HTML и JavaScript;

– адаптивность. Vue.js обеспечивает быстрый период перехода от других фреймворков к Vue.js в виду его сходства с Angular и React с точки зрения дизайна и архитектуры;

– восхитительная интеграция. Vue.js можно использовать как для создания одностраничных приложений, так и для более сложных веб–интерфейсов приложений. Самое важное, что небольшие интерактивные части можно легко интегрировать в существующую инфраструктуру, не оказывая при этом отрицательного влияния на всю систему;

– большое масштабирование. Vue.js помогает разрабатывать довольно крупные шаблоны для многократного использования, которые можно разработать без траты огромного количества времени в виду простой структуры;

– крошечный размер. Vue.js может весить около 20 КБ и при этом сохранять свою свою скорость и гибкость, что позволяет достичь гораздо более высокой производительности, по сравнению с другими фреймворками.

Недостатки Vue.js:

– нехватка ресурсов. Vue.js по–прежнему имеет довольно небольшую долю рынка по сравнению с React или Angular. Это значит, что обмен знаниями в рамках фреймворка все еще формируется;

– риск чрезмерной гибкости. Иногда у Vue.js могут возникать проблемы при интеграции в огромные проекты, а опыта о возможных решениях до сих пор нет. Но они обязательно появятся в ближайшее время;

– отсутствие полной англоязычной документации. Это приводит к некоторым сложностям на различных этапах разработки. Тем не менее, все больше и больше материалов переводятся на английский язык.

2.1.3 Express.js

Express.js, или просто Express, фреймворк web–приложений для Node.js, реализованный как свободное и открытое программное обеспечение под лицензией MIT. Он спроектирован для создания веб–приложений и API. Де–факто является стандартным каркасом для Node.js.

Основная особенность этого фреймворка заключается в том, что для Express характерен небольшой объем базового функционала. Все остальные нужные вам функции нужно будет добирать за счет внешних модулей. По сути, Express в чистом виде – это сервер и у него может не быть ни одного модуля.

Благодаря такому минимализму разработчик изначально получает в свое распоряжение легкий и быстрый инструмент, который он может расширять и развивать.

При этом немаловажно, что выбор модулей для Express не связан ни с какими ограничениями: ни с количественными, ни с функциональными.

В результате, этот фреймворк обеспечивает разработчику возможность решать любые задачи, не ограничивая его при этом в выборе средств.

С одной стороны, не может не радовать тот факт, что отсутствие готовых универсальных решений, фактически означает, что каждое создаваемое приложение будет уникальным.

С другой стороны, разработчику нужно самостоятельно отбирать и организовывать модули, а это предполагает большой объем работы и соответственно, требует от разработчика больше времени и усилий.

2.1.4 MySQL

MySQL – свободная реляционная система управления базами данных[7]. Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию MySQL AB.

MySQL отличатся хорошей скоростью работы, надежностью, гибкостью. Работа с ней, как правило, не вызывает больших трудностей.

Немаловажным фактором является ее бесплатность. MySQL распространяется на условиях общей лицензии GNU (GPL, GNU Public License).

2.1.5 dbForge Studio

dbForge Studio for MySQL – универсальное решение для разработки, администрирования и управления базами данных MySQL и MariaDB. Данный продукт позволяет создавать и выполнять запросы, разрабатывать и отлаживать процедуры и функции, а также автоматизировать управление объектами баз данных MySQL с помощью удобного пользовательского интерфейса. Клиент

MySQL содержит инструменты для сравнения и синхронизации данных и схем, создания резервных копий баз данных по графику, а также для анализа и создания отчетов по данным таблиц MySQL.

2.1.6 Postman

Postman – удобный HTTP–клиент для тестирования веб–сайтов.

С помощью данного программного обеспечения можно составлять и редактировать простые или сложные HTTP–запросы. Составленные запросы автоматически сохраняются на будущее для повторного применения. Ответы от сервера можно тоже сохраняются как файлы на жестком диске. Таким образом, Postman экономит кучу времени веб–разработчикам.

2.2 Постановка цели и задачи

Целью данного дипломного проекта является автоматизация бизнес процессов предприятия, таких как сбор информации о клиентах, их требовании от продукта, предоставляемым компанией.

Главными задачами данной дипломной работы являются упрощение работы персонала предприятия, гибкость и универсальность системы, простой и интуитивно понятный интерфейс. Помимо этих задач, так же имеется возможность сохранения контактов и истории общения с клиентами при увольнении сотрудника. Его преемник получит все необходимые данные о контрагентах и сможет без промедлений приступить к работе. Возможность развивать отношения с потенциальными заказчиками и напоминать о себе действующим клиентам. Повышение эффективности продаж. Составление воронки продаж – на этом этапе можно выявить, на каком участке клиент отказывается от покупки или услуги. Анализ эффективности рекламной кампании: количество переходов, совершенных продаж, отказов и т.д. Анализ эффективности продаж. Благодаря понятным диаграммам и дэшбордам можно определить конверсию, например на 200 обзвонов и рассылок можно получить 10 реальных продаж (данные приблизительные, все зависит от вида товара, спроса, цены и других факторов). Создание, настройка и наладка бизнес–процессов поможет выявить, какой этап не эффективен для продаж и решить, что необходимо изменить. Сведение к минимуму ошибок человеческого фактора

В данной системе будет иметься база данных компании, их клиентов и сделок, созданных ими. Так же будет постоянный учет количества сделок в той, или иной воронке или этапе, которые будут заноситься в базу данных.

Для реализации данного проекта необходимо осуществить следующие требования:

- анализ существующего рынка;
- определение основных требовании от системы;
- проектирование базы данных;

– разработка программной и аппаратной части;

– выбор технологии разработки;

– обоснование экономической целесообразности разрабатываемого продукта;

– предложение мероприятий по улучшению условий труда в рамках реализуемого проекта.

3 Проектирование

В данной главе описывается процесс проектирования базы данных, программной и аппаратной части проекта.

3.1 Проектирование программной части

Одним из самых важных этапов разработки программного обеспечения – это написание программного кода. Необходимо продумать последовательность действий, которые могут выполняться в программе. Логику программы можно часто расписать на листке бумаги, либо построить блок–схему.

На рисунке 3.1.1 изображена блок–схема алгоритма создания новой сделки в воронку, в начальный этап.

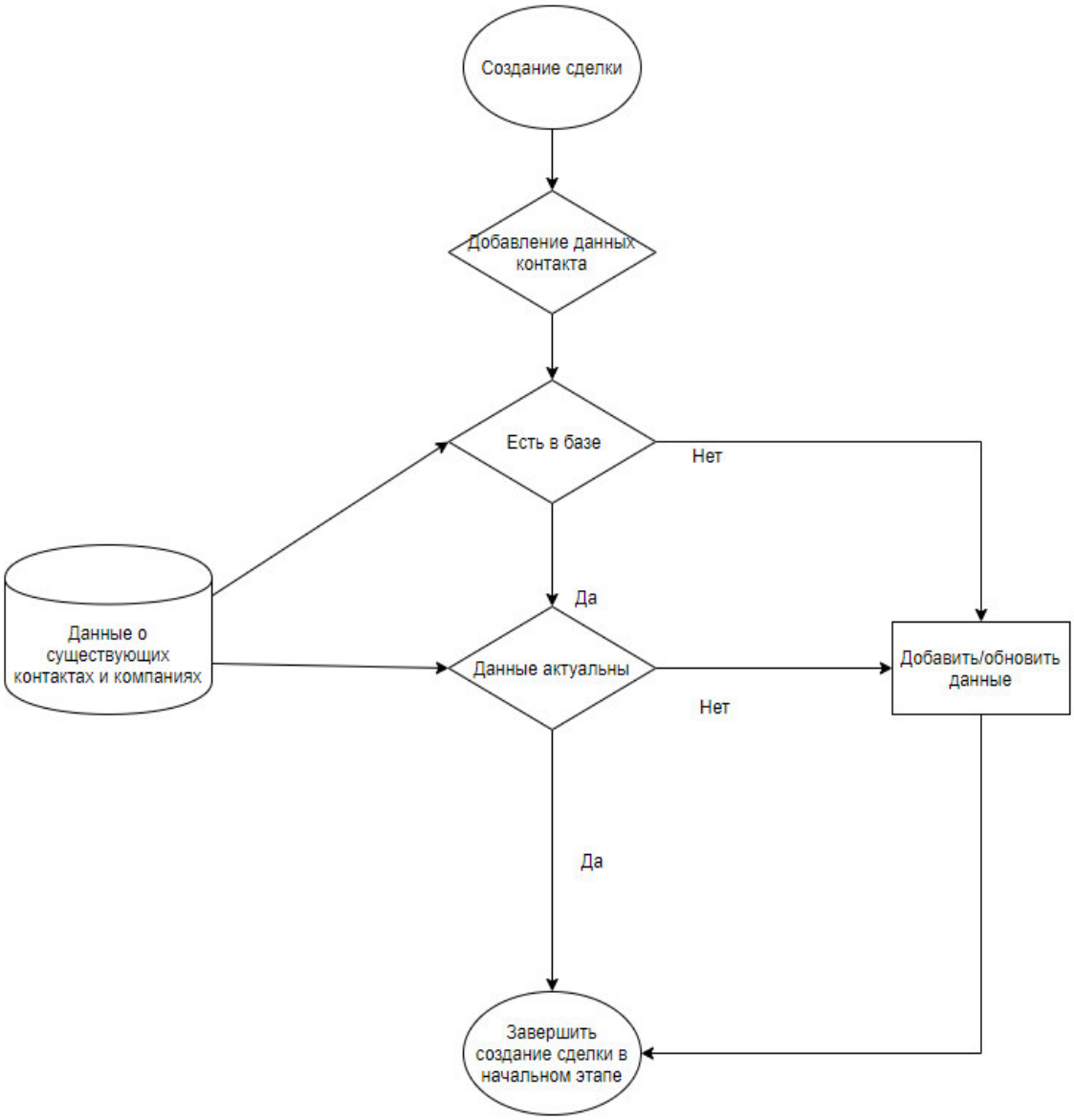

Рисунок 3.1.1 – Блок схема создания новой сделки

На рисунке 3.1.2 изображена блок–схема алгоритма переноса сделки в следующий этап в воронке. Для переноса сделки в следующий требуются дополнительные данные, которые компания получает при положительной работе с клиентом. В это входит такие данные как, номер счета клиента, способ оплаты, способ доставки продукта, если он имеется, номер договора, юридический/физический адрес компании клиента, если таковая имеется и тому подобное.

Так же, имеются дополнительные данные для переноса сделки с одной воронки в другую, которые настраивает для себя пользователь.

Рисунок 3.1.2 – Блок–схема алгоритма переноса сделки в следующий этап в воронке.

3.2 Проектирование базы данных

Каждый владелец сайта знает, что для правильного функционирования сайта нужны не только файлы с кодом страниц, но и базы данных. Для

взаимодействия с базами данных используются системы управления базами данных (СУБД).

База данных представляет собой определенный набор данных, которые, как правило, связаны объединяющим признаком либо свойством (или несколькими). Эти данные упорядочены, например, по алфавиту. Обилие различных данных, которые могут быть помещены в единую базу, ведет к множеству вариаций того, что может быть записано: личные данные пользователей, записи, даты, заказы и так далее. К примеру, если у вас интернет–магазин, то база данных вашего сайта может содержать прайс–листы, каталог товаров или услуг, отчеты, статистику и информацию о клиентах.

В первую очередь это удобно тем, что информацию можно быстро заносить в базу данных и так же быстро ее извлекать при необходимости. Если на заре развития web–разработки все необходимые данные нужно было прописывать в коде страницы, то теперь такая необходимость отсутствует – нужная информация может быть запрошена из базы данных при помощи скриптов. Специальные алгоритмы хранения и поиска информации, которые используются в базах данных, позволяют находить нужные сведения буквально за доли секунд – а при работе в виртуальном пространстве скорость работы ресурса важна как ничто другое.

Немаловажной является и взаимосвязь информации в базе данных: изменение одной строчки может привести к значительным изменениям других строк. Работать с данными таким образом гораздо проще и быстрее, чем если бы изменения касались только одного места в базе данных.

При разработке CRM – системы необходимо создать базу данных для хранения информации о компании которая пользуется данной системы, данных о пользователях – работников компании, данных о сделках, контактах и компаниях, данных о пользовательских полях карточки сделки/контакта/компании, данных о воронке и этапе, данных о тэгах, задачах, данные чата и почты.

Огромным плюсом использования в качестве системы хранения данных MySQL, то что она является очень быстрой в плане обработки запросов. А это является очень важным критерием, так как, в системе хранится очень большой объем данных. Помимо этого, так же она является надежной, безопасной, мало затратной в плане ресурсов сервера.

Первое с чем взаимодействует пользователь – это регистрация в системе,

в качестве аккаунта и добавление пользователей, т.е. персонала, для дальнейшей работы в ней.

В системе по умолчанию имеется только одна воронка, с несколькими этапами, в карточке находятся только стандартные поля, такие как бюджет, ответственный за создание сделки. Все воронки, этапы и карточка являются изменяемыми под нужды пользователя, благодаря чему система получается гибкой, и настраивается под нужды пользователя.

Каждая созданная сделка, в последующем закрепляется в базе данных за определенной компанией. В свою очередь за сделкой закрепляется контакт, и компания, если таковая имеется.

В карточке сделки, контакта или компании так же имеются поля для заполнения, которые являются изменяемыми и полностью настраиваемыми.

3.2.1 ER – диаграмма базы данных

Собрав все проанализированные данные, можно построить диаграмму «сущность–связь» (ER–диаграмма) (см. рисунок 3.2.1).

ER–диаграмма – это тип блок–схемы, которая отображает как сущности, такие как автомобили, объекты или концепции, относится друг с другом в пределах определенной системы. ER–диаграммы используется для проектирования или отладки реляционных баз данных в области разработки программного обеспечения, информационных систем для бизнеса, образования и исследований.

Так как база данных является обширной, далее я поделю разбор диаграммы по пунктам (Рисунок 3.2.2).

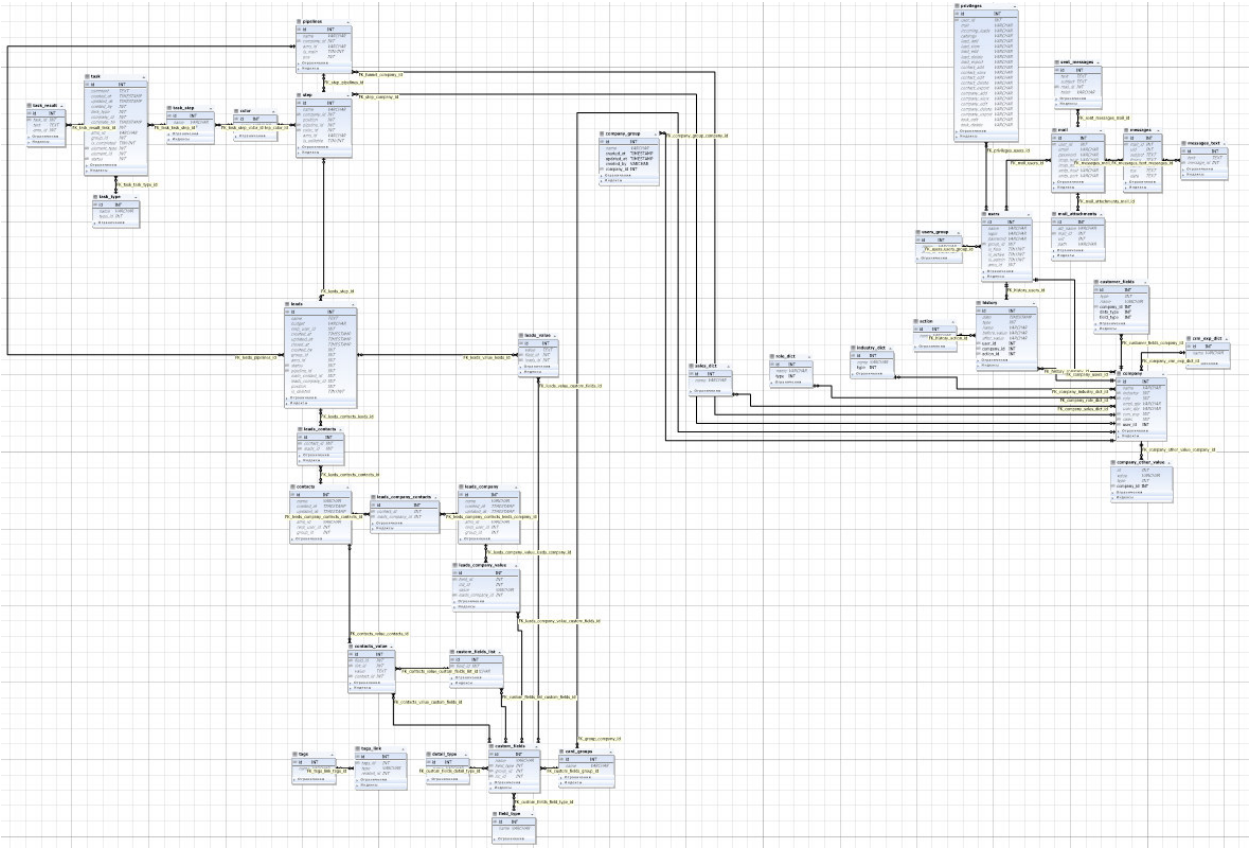

Рисунок 3.2.1 – ER – диаграмма базы данных

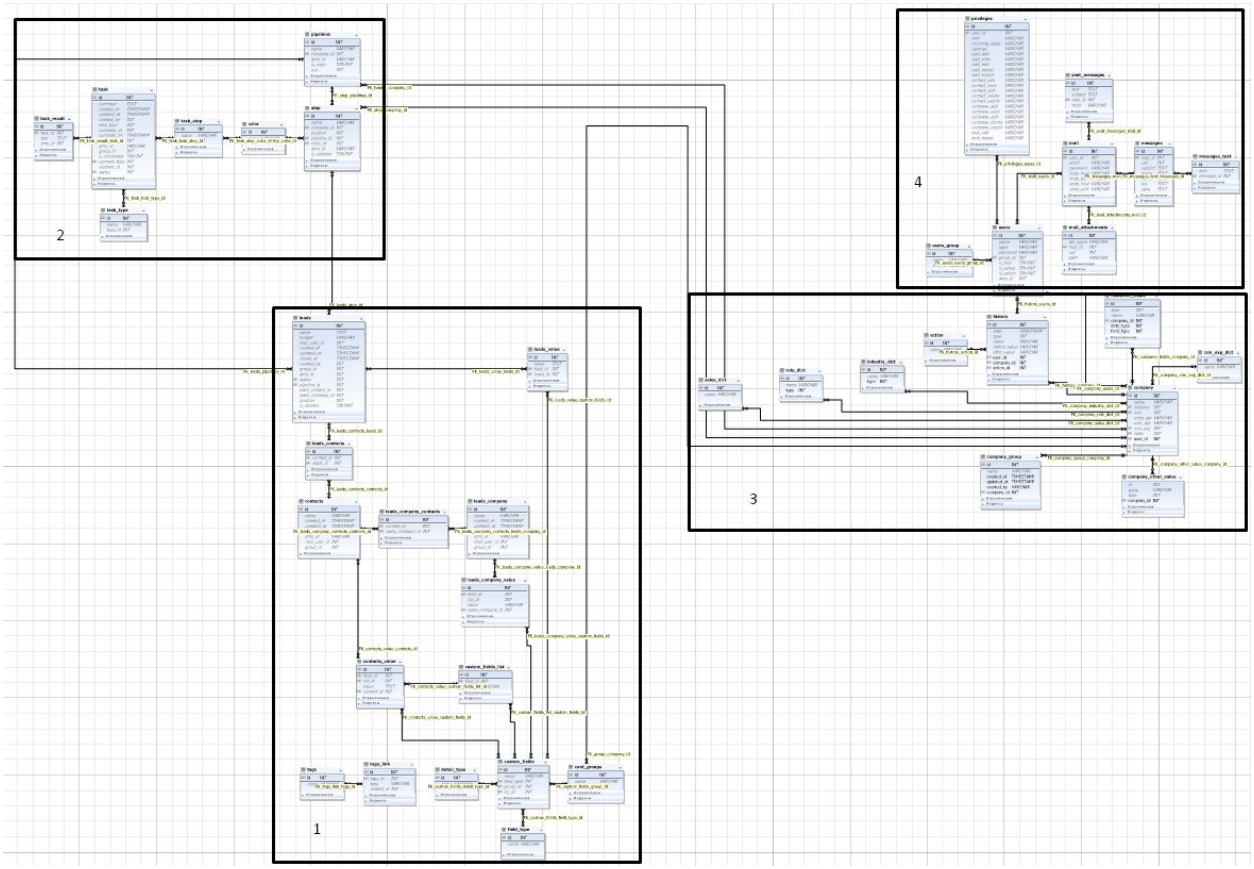

Рисунок 3.2.2 – Пункты ER – диаграммы

3.2.1.1 Сделки и пользовательские поля

В данной части диаграммы показаны таблицы где хранятся данные о сделках, контактах, компаниях, добавленных полях и т.д. (Рисунок 3.2.3).

3.2.1.2 Воронки, этапы и задачи

В данной части диаграммы показаны таблицы где хранятся данные о воронках, этапах и задачах (Рисунок 3.2.4).

3.2.1.3 Компания

В данной части диаграммы показаны таблицы где хранятся о компании (Рисунок 3.2.5).

3.2.1.4 Пользователи, доступ и почта

В данной части диаграммы показаны таблицы где хранятся данные о пользователях, их правах доступа и почты, (Рисунок 3.2.6).

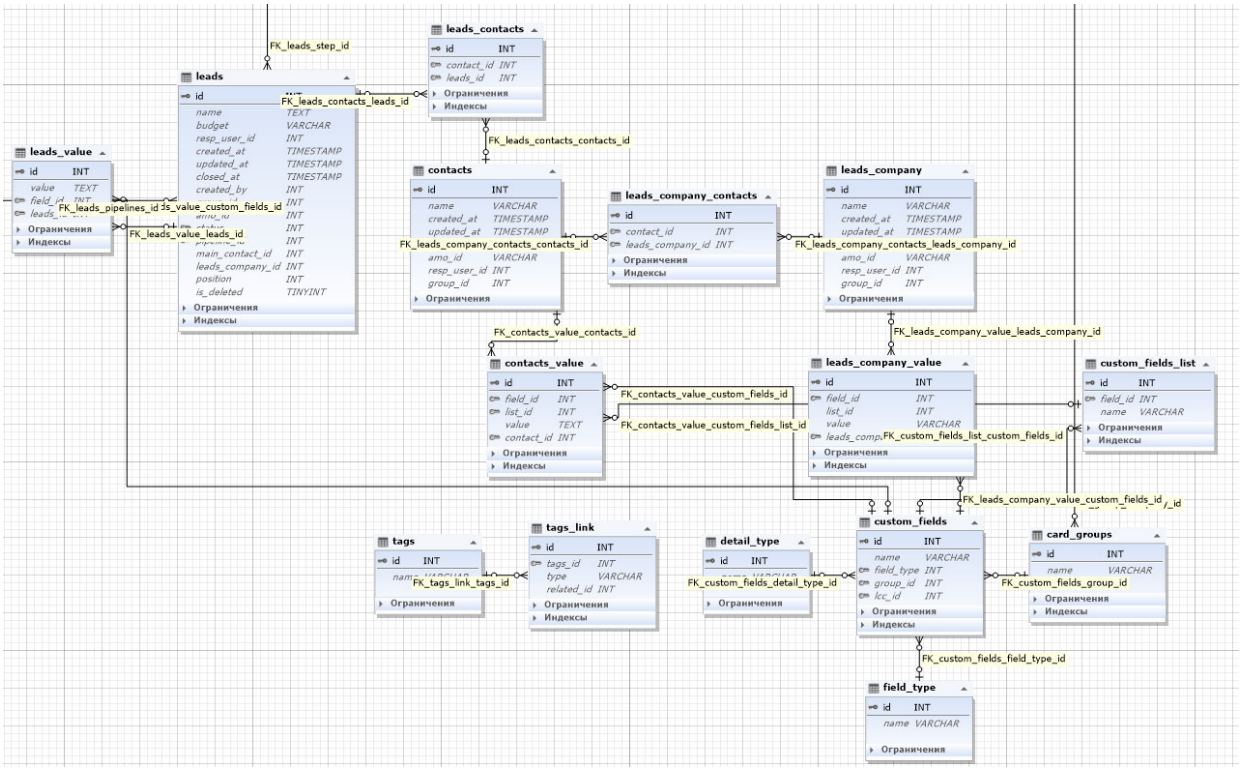

Рисунок 3.2.3 – Сделки и пользовательские поля

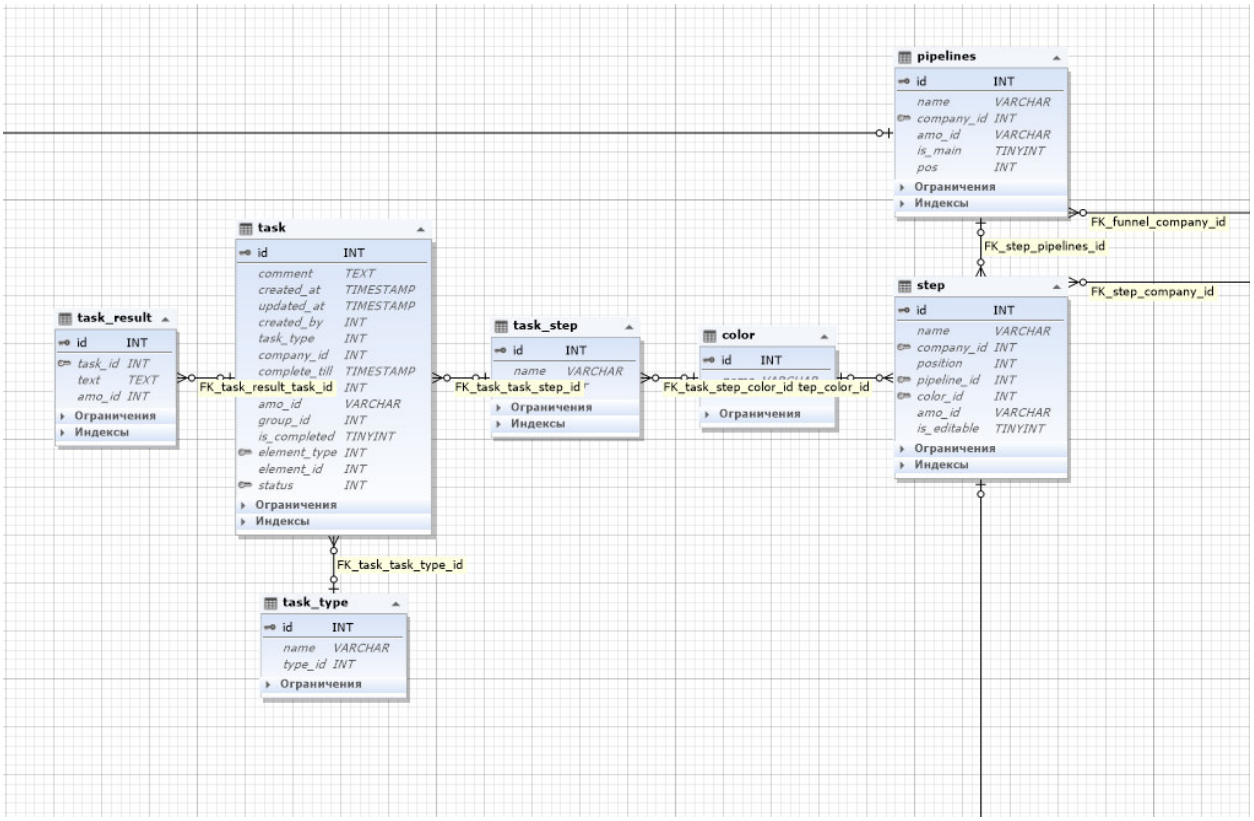

Рисунок 3.2.4 – Воронки, этапы и задачи

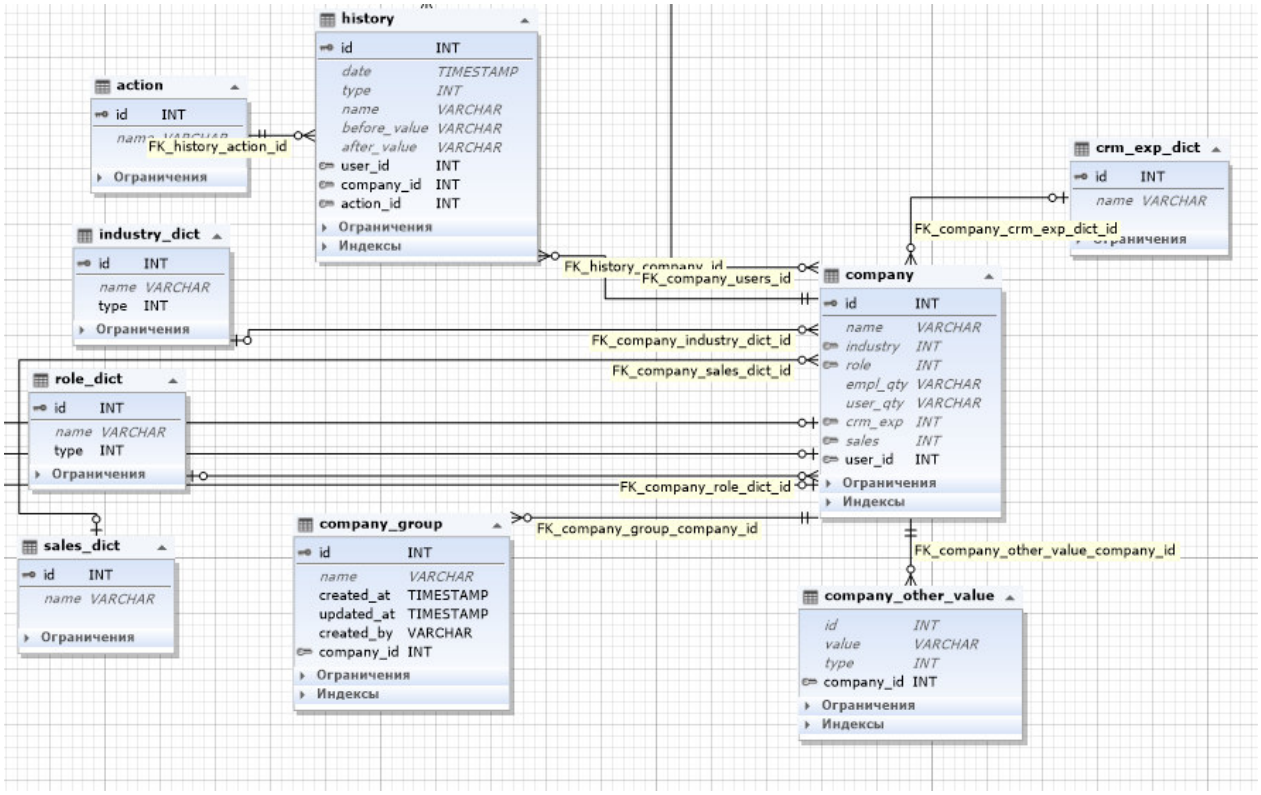

Рисунок 3.2.5 – Компания

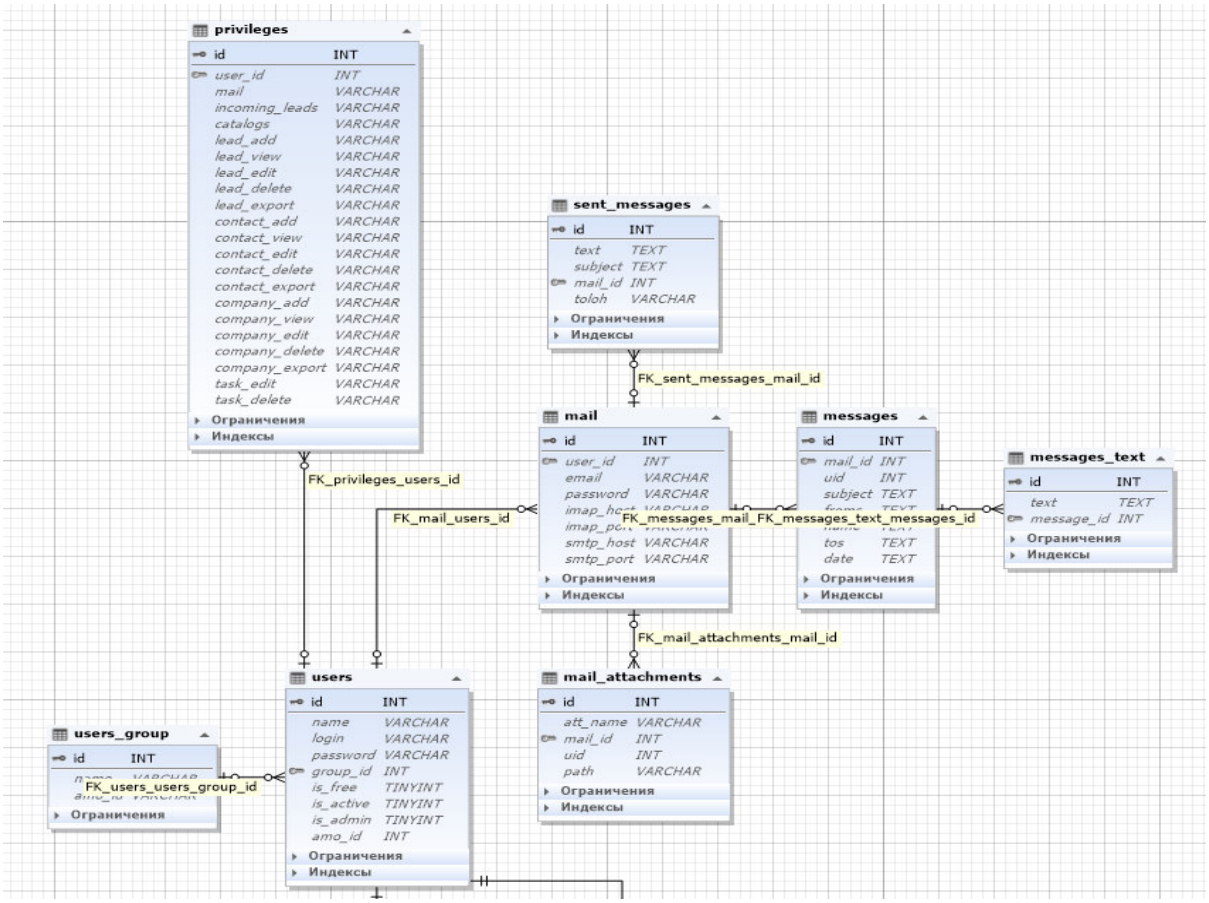

Рисунок 3.2.6 – Пользователи, доступ и почта

3.3 Создание базы данных

В данном подразделе будут приведены запросы, которые применялись для создания таблиц базы данных, а также краткое пояснение к каждой таблице.

3.3.1 Разбор пункта 3.2.1.1

Таблица leads. Данная таблица хранит в себе данные о сделке, т.е. ее название, бюджет, дату создания, изменения и закрытия, ответственный, кем она была создана, воронку, этап, контакт и компанию, если таковые имеются на момент создания (Рисунок 3.3.1)

```
CREATE TABLE amocrm.leads (
  id int(11) NOT NULL AUTO_INCREMENT,
  name text CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  budget varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  resp_user_id int(11) DEFAULT NULL,
  created_at timestamp NULL DEFAULT NULL,
  updated_at timestamp NULL DEFAULT NULL,
  closed_at timestamp NULL DEFAULT NULL,
  created_by int(11) DEFAULT NULL,
  group_id int(11) DEFAULT NULL,
  amo_id int(11) DEFAULT NULL,
  status int(11) DEFAULT NULL,
  pipeline_id int(11) DEFAULT NULL,
  main_contact_id int(11) DEFAULT NULL,
  leads_company_id int(11) DEFAULT NULL,
  `position` int(11) DEFAULT NULL,
  is_deleted tinyint(1) DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.1 – leads

Таблица leads_value хранит в себе данные о полях сделки (Рисунок 3.3.2)

```
CREATE TABLE amocrm.leads_value (
  id int(11) NOT NULL AUTO_INCREMENT,
  value text CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  field_id int(11) DEFAULT NULL,
  leads_id int(11) DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.2 – leads_value

Таблица contacts хранит в себе данные о контакте (Рисунок 3.3.3)

```
CREATE TABLE amocrm.contacts (
  id int(11) NOT NULL AUTO_INCREMENT,
  name varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  created_at timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  updated_at timestamp NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  created_by int(11) DEFAULT NULL,
  amo_id varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  resp_user_id int(11) DEFAULT NULL,
  group_id int(11) DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.3 – contacts

Таблица leads_company хранит в себе данные о компании (Рисунок 3.3.4)

```
CREATE TABLE amocrm.leads_company (
  id int(11) NOT NULL AUTO_INCREMENT,
  name varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  created_at timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  updated_at timestamp NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  created_by int(11) DEFAULT NULL,
  amo_id varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  resp_user_id int(11) DEFAULT NULL,
  group_id int(11) DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.4 – leads_company

Таблицы contacts_value и leads_company_value хранят в себе данные о полях контактов и компании (Рисунок 3.3.5 и 3.3.6)

```
CREATE TABLE amocrm.contacts_value (
  id int(11) NOT NULL AUTO_INCREMENT,
  field_id int(11) DEFAULT NULL,
  list_id int(11) DEFAULT NULL,
  value text DEFAULT NULL,
  contact_id int(11) DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.5 – contacts_value

```
CREATE TABLE amocrm.leads_company_value (
  id int(11) NOT NULL AUTO_INCREMENT,
  field_id int(11) DEFAULT NULL,
  list_id int(11) DEFAULT NULL,
  value varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  leads_company_id int(11) DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.6 – leads_company_value

Таблица custom_fields хранит в себе данные о наименовании полей сделок, контактов и компании (Рисунок 3.3.7)

```
CREATE TABLE amocrm.leads_company_value (
  id int(11) NOT NULL AUTO_INCREMENT,
  field_id int(11) DEFAULT NULL,
  list_id int(11) DEFAULT NULL,
  value varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  leads_company_id int(11) DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.7 – custom_fields

Таблицы detail_type и field_type хранят в себе типы карточки и поля в нем соответственно (Рисунок 3.3.8 и 3.3.9)

```
CREATE TABLE amocrm.detail_type (
  id int(11) NOT NULL AUTO_INCREMENT,
  name varchar(50) DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.8 – detail_type

```
CREATE TABLE amocrm.field_type (
  id int(11) NOT NULL AUTO_INCREMENT,
  name varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.9 – field_type

Таблицы leads_contacts и leads_company_contacts являются связующими, и хранят в себе только id сделки и контакта, и компании и контакта соответственно (Рисунок 3.3.10 и 3.3.11)

```
CREATE TABLE amocrm.leads_contacts (
  id int(11) NOT NULL AUTO_INCREMENT,
  contact_id int(11) DEFAULT NULL,
  leads_id int(11) DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.10 – leads_contacts

```
CREATE TABLE amocrm.leads_company_contacts (
  id int(11) NOT NULL AUTO_INCREMENT,
  contact_id int(11) DEFAULT NULL,
  leads_company_id int(11) DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.11 – leads_company_contacts

3.3.2 Разбор пункта 3.2.1.2

Таблица pipelines хранит в себе данные о воронках системы (Рисунок 3.3.12)

```
CREATE TABLE amocrm.pipelines (
  id int(11) NOT NULL AUTO_INCREMENT,
  name varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  company_id int(11) DEFAULT NULL,
  amo_id varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  is_main tinyint(1) DEFAULT NULL,
  pos int(11) DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.12 – pipelines

Таблицы step и color хранят в себе данные об этапах и их цвете в системе соответственно (Рисунок 3.3.13 и 3.3.14)

```
CREATE TABLE amocrm.step (
  id int(11) NOT NULL AUTO_INCREMENT,
  name varchar(50) DEFAULT NULL,
  company_id int(11) DEFAULT NULL,
  `position` int(11) DEFAULT NULL,
  pipeline_id int(11) DEFAULT NULL,
  color_id int(11) DEFAULT NULL,
  amo_id varchar(255) DEFAULT NULL,
  is_editable tinyint(1) DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.13 – step

```
CREATE TABLE amocrm.color (
  id int(11) NOT NULL AUTO_INCREMENT,
  name varchar(50) DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.14 – color

Таблицы task_type, task_result, task_step и task хранят в себе данные о задачах, их типах, результате задачи и на каком этапе выполнения находится задача соответственно (Рисунок 3.3.15, 3.3.16, 3.3.17 и 3.3.18)

```
CREATE TABLE amocrm.task_type (
  id int(11) NOT NULL AUTO_INCREMENT,
  name varchar(255) DEFAULT NULL,
  type_id int(11) DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.16 – task_type

```
CREATE TABLE amocrm.task_result (
  id int(11) NOT NULL AUTO_INCREMENT,
  task_id int(11) DEFAULT NULL,
  text text DEFAULT NULL,
  amo_id int(11) DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.17 – task_result

```
CREATE TABLE amocrm.task_step (
  id int(11) NOT NULL AUTO_INCREMENT,
  name varchar(50) DEFAULT NULL,
  color_id int(11) DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.15 – task_step

```
CREATE TABLE amocrm.task (
  id int(11) NOT NULL AUTO_INCREMENT,
  comment text CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  created_at timestamp NULL DEFAULT CURRENT_TIMESTAMP,
  updated_at timestamp NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  created_by int(11) DEFAULT NULL,
  task_type int(11) DEFAULT NULL,
  company_id int(11) DEFAULT NULL,
  complete_till timestamp NULL DEFAULT NULL,
  resp_user_id int(11) DEFAULT NULL,
  amo_id varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  group_id int(11) DEFAULT NULL,
  is_completed tinyint(1) DEFAULT NULL,
  element_type int(11) DEFAULT NULL,
  element_id int(11) DEFAULT NULL,
  status int(11) DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.18 – task

3.3.3 Разбор пункта 3.2.1.3

Таблица company хранит в себе данные компании – пользователя которые он заполняет при регистрации (Рисунок 3.3.19)

```
CREATE TABLE amocrm.company (
  id int(11) NOT NULL AUTO_INCREMENT,
  name varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  industry int(11) DEFAULT NULL COMMENT 'чем занимается',
  role int(11) DEFAULT NULL COMMENT 'кем является',
  empl_qty varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL COMMENT 'кол-во сотрудников',
  user_qty varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL COMMENT 'кол-во юзеров',
  crm_exp int(11) DEFAULT NULL COMMENT 'опыт в работе с срм',
  sales int(11) DEFAULT NULL COMMENT 'наличие отдела продаж',
  user_id int(11) NOT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.19 – company

Таблицы crm_exp_dict, industry_dict, role_dict, sales_dict, company_group и company_other_value являются словарями данных, и хранят в себе статическую информацию о компании – пользователе. Опыт работы с CRM–системами, направление компании, должность создателя аккаунта, количество персонала, который будет пользоваться системой соответственно (Рисунок 3.3.20, 3.3.21, 3.3.22, 3.3.23, 3.3.24 и 3.3.25).

```
CREATE TABLE amocrm.crm_exp_dict (
  id int(11) NOT NULL AUTO_INCREMENT,
  name varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.20 – crm_exp_dict

```
CREATE TABLE amocrm.industry_dict (
  id int(11) NOT NULL AUTO_INCREMENT,
  name varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  type int(11) NOT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.21 – industry_dict

```
CREATE TABLE amocrm.role_dict (
  id int(11) NOT NULL AUTO_INCREMENT,
  name varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  type int(11) NOT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.22 – role_dict

```
CREATE TABLE amocrm.sales_dict (
  id int(11) NOT NULL AUTO_INCREMENT,
  name varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.23 – sales_dict

```
CREATE TABLE amocrm.company_group (
  id int(11) NOT NULL AUTO_INCREMENT,
  name varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  created_at timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  updated_at timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  created_by varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL,
  company_id int(11) NOT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.24 – company_group

```
CREATE TABLE amocrm.company_other_value (
  id int(11) DEFAULT NULL,
  value varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  type int(11) DEFAULT NULL,
  company_id int(11) NOT NULL
)
```

Рисунок 3.3.25 – company_other_value

Таблицы history хранит в себе все изменения, т.е. перенос сделки из этапа в этап, изменение сделки, контакта или компании, и т.д. (Рисунок 3.3.26).

```
CREATE TABLE amocrm.history (
  id int(11) NOT NULL AUTO_INCREMENT,
  date timestamp NULL DEFAULT '0000-00-00 00:00:00' ON UPDATE CURRENT_TIMESTAMP,
  type int(11) DEFAULT NULL,
  name varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  before_value varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL COMMENT 'значение до',
  after_value varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL COMMENT 'значение после',
  user_id int(11) NOT NULL,
  company_id int(11) NOT NULL,
  action_id int(11) NOT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.26 – history

Таблица action является словарем данных и хранит в себе наименования всех возможных изменений в системе (Рисунок 3.3.27).

```
CREATE TABLE amocrm.action (
  id int(11) NOT NULL AUTO_INCREMENT,
  name varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.27 – action

3.3.4 Разбор пункта 3.2.1.4

Таблица users и users_group хранит в себе данные о пользователях и к какому отделу этот пользователь относится, соответственно (Рисунок 3.3.28 и 3.3.29)

```
CREATE TABLE amocrm.users (
  id int(11) NOT NULL AUTO_INCREMENT,
  name varchar(50) DEFAULT NULL,
  login varchar(50) DEFAULT NULL,
  password varchar(255) DEFAULT NULL,
  group_id int(11) DEFAULT NULL COMMENT 'наименование отдела',
  is_free tinyint(1) DEFAULT NULL,
  is_active tinyint(1) DEFAULT NULL,
  is_admin tinyint(1) DEFAULT NULL,
  amo_id int(11) DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.28 – users

```
CREATE TABLE amocrm.users_group (
  id int(11) NOT NULL AUTO_INCREMENT,
  name varchar(50) DEFAULT NULL,
  amo_id varchar(255) DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.29 – users_group

Таблица privileges хранит в себе данные о привилегиях пользователя, например, возможность создавать, изменять, удалять сделки, отправлять почту и тому подобное (Рисунок 3.3.30).

```
CREATE TABLE amocrm.privileges (
  id int(11) NOT NULL AUTO_INCREMENT,
  user_id int(11) DEFAULT NULL,
  mail varchar(255) DEFAULT NULL,
  incoming_leads varchar(255) DEFAULT NULL,
  catalogs varchar(255) DEFAULT NULL,
  lead_add varchar(255) DEFAULT NULL,
  lead_view varchar(255) DEFAULT NULL,
  lead_edit varchar(255) DEFAULT NULL,
  lead_delete varchar(255) DEFAULT NULL,
  lead_export varchar(255) DEFAULT NULL,
  contact_add varchar(255) DEFAULT NULL,
  contact_view varchar(255) DEFAULT NULL,
  contact_edit varchar(255) DEFAULT NULL,
  contact_delete varchar(255) DEFAULT NULL,
  contact_export varchar(255) DEFAULT NULL,
  company_add varchar(255) DEFAULT NULL,
  company_view varchar(255) DEFAULT NULL,
  company_edit varchar(255) DEFAULT NULL,
  company_delete varchar(255) DEFAULT NULL,
  company_export varchar(255) DEFAULT NULL,
  task_edit varchar(255) DEFAULT NULL,
  task_delete varchar(255) DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.30 – privileges

Таблицы mail, messages, messages_text, mail_attachments, sent_messages хранят в себе данные о подключении почты, заголовках сообщении, тексте сообщении, вложении в них, и отправленных об сообщениях соответсвенно (Рисунок 3.3.31, 3.3.32, 3.3.33, 3.3.34, 3.3.35).

```
CREATE TABLE amocrm.mail (
  id int(11) NOT NULL AUTO_INCREMENT,
  user_id int(11) DEFAULT NULL,
  email varchar(50) DEFAULT NULL,
  password varchar(255) DEFAULT NULL,
  imap_host varchar(255) DEFAULT NULL,
  imap_port varchar(255) DEFAULT NULL,
  smtp_host varchar(255) DEFAULT NULL,
  smtp_port varchar(255) DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.31 – mail

```
CREATE TABLE amocrm.messages (
  id int(11) NOT NULL AUTO_INCREMENT,
  mail_id int(11) DEFAULT NULL,
  uid int(11) DEFAULT NULL,
  subject text CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  froms text CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  name text CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  tos text CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  date text CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.32 – messages

```
CREATE TABLE amocrm.messages_text (
  id int(11) NOT NULL AUTO_INCREMENT,
  text text DEFAULT NULL,
  message_id int(11) DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.33 – messages_text

```
CREATE TABLE amocrm.mail_attachments (
  id int(11) NOT NULL AUTO_INCREMENT,
  att_name varchar(255) DEFAULT NULL,
  mail_id int(11) DEFAULT NULL,
  uid int(11) DEFAULT NULL,
  path varchar(255) DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.34 – mail_attachments

```
CREATE TABLE amocrm.sent_messages (
  id int(11) NOT NULL AUTO_INCREMENT,
  text text DEFAULT NULL,
  subject text DEFAULT NULL,
  mail_id int(11) DEFAULT NULL,
  toloh varchar(255) DEFAULT NULL,
  PRIMARY KEY (id)
)
```

Рисунок 3.3.35 – sent_messages

3.4 Инструменты для работы с базой данных

При разработке базы данных для данного дипломного проекта использовался инструмент под названием dbForge Studio for MySQL.

Данная программа полностью средой разработки базы данных из которой можно удаленно подключится к серверу с базой данных или же подключится к базе данных на локальном сервере. Так же из плюсов хотелось бы выделить то что, можно создавать таблицы, триггеры, процедуры, функции непосредственно в среде разработки, и интуитивно понятный интерфейс. На рисунке 3.4.1 изображено окно подключения к базе данных.

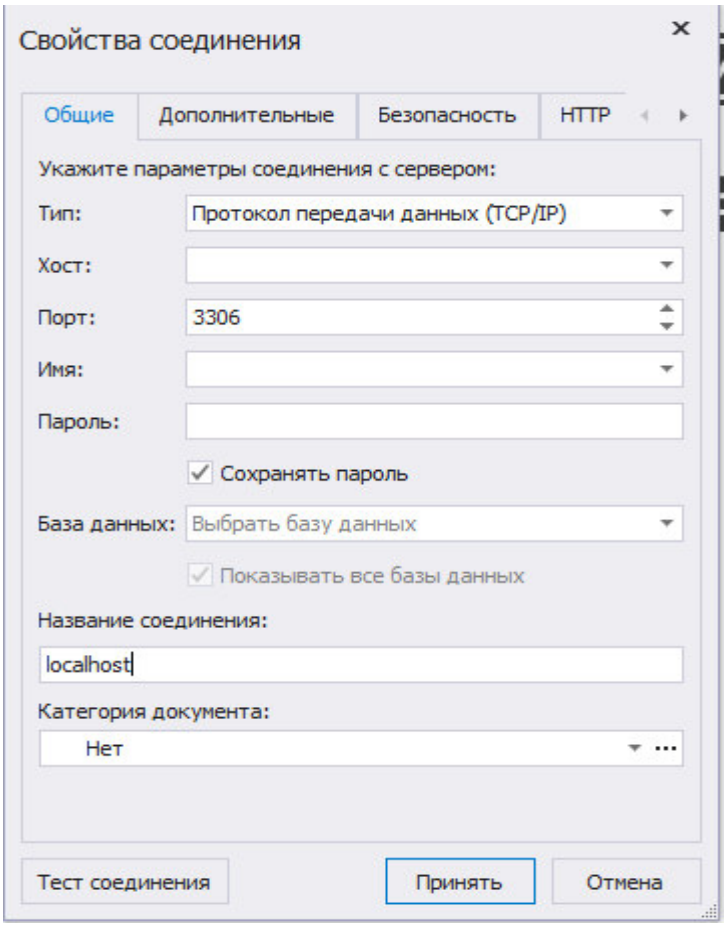

Рисунок 3.4.1 – Окно соединения

Далее после ввода хоста, порта, имени и пароля подключения сбоку появится список подключении и, хранящиеся на сервере или на локальном хранилище, базы данных, изображено на рисунке 3.4.2.

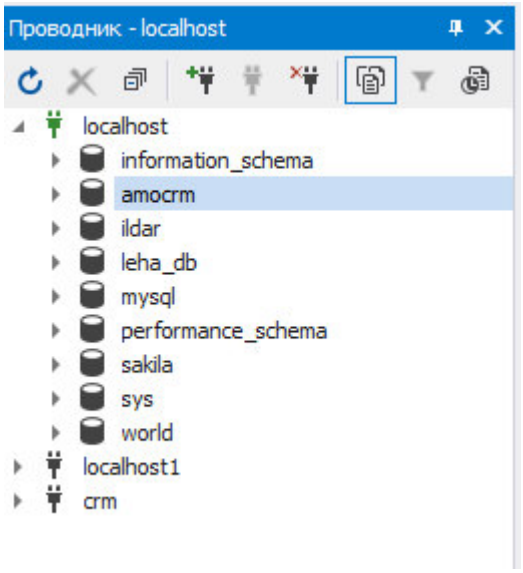

Рисунок 3.4.2 – Проводник

После этого нажав на соединения правой кнопкой мыши можно будет создать базу данных, рисунок 3.4.3.

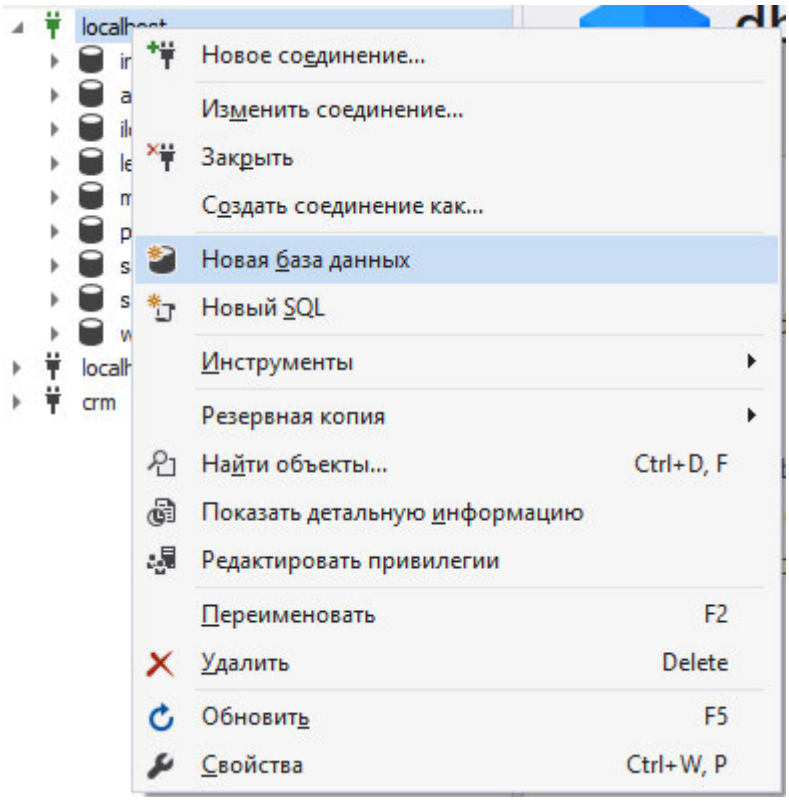

Рисунок 3.4.3 – Создание базы данных

Далее у нас появится окно настройки для именования базы и выбора ее кодировки.

Выбор кодировки требуется для того чтобы, база данных могла хранить в себе разные символы данных, в данном случае лучшим выбором будет utf8 (Рисунок 3.4.4).

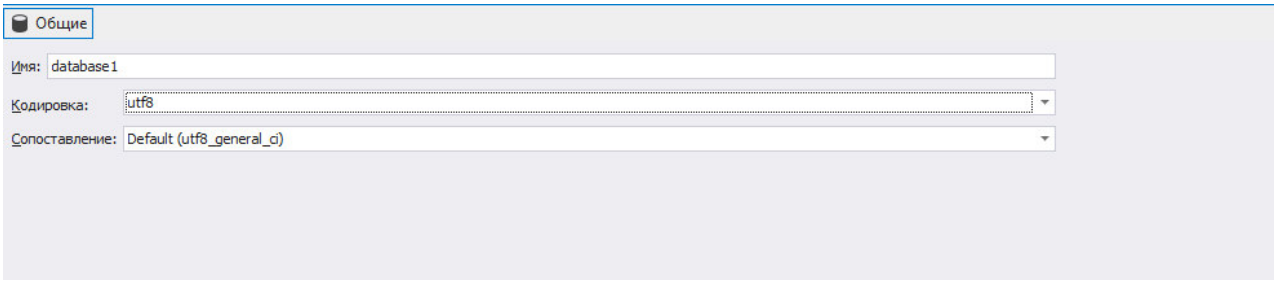

Рисунок 3.4.4 – Настройка базы данных

После этого можно создавать таблицы, триггеры, процедуры, все что потребуется для комфортной работы. Чтобы создать таблицу, нужно нажать ПКМ по базе, и выбрать пункт Новый объект, и в списке выбрать Таблица (Рисунок 3.4.5).

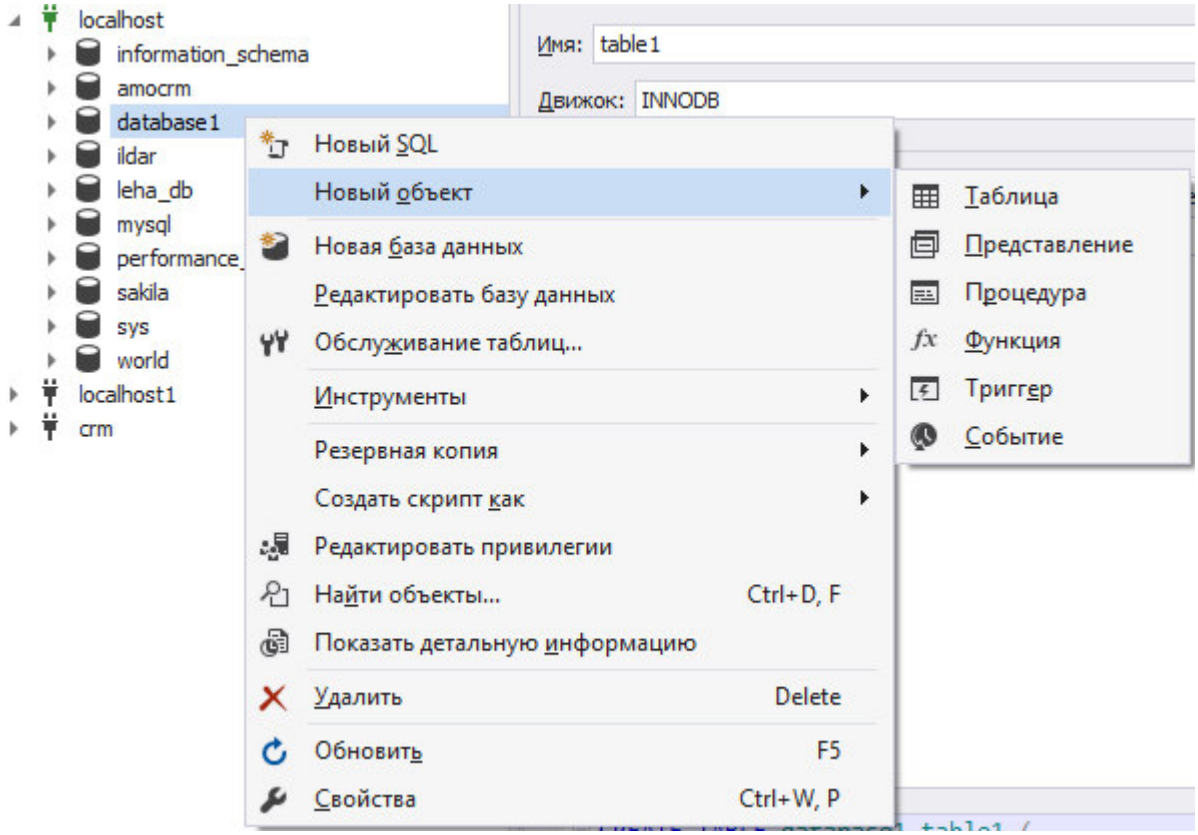

Рисунок 3.4.5 – Создание таблиц

Далее появится окно настройки таблицы, создания в ней полей и выбор типов для полей (Рисунок 3.4.6).

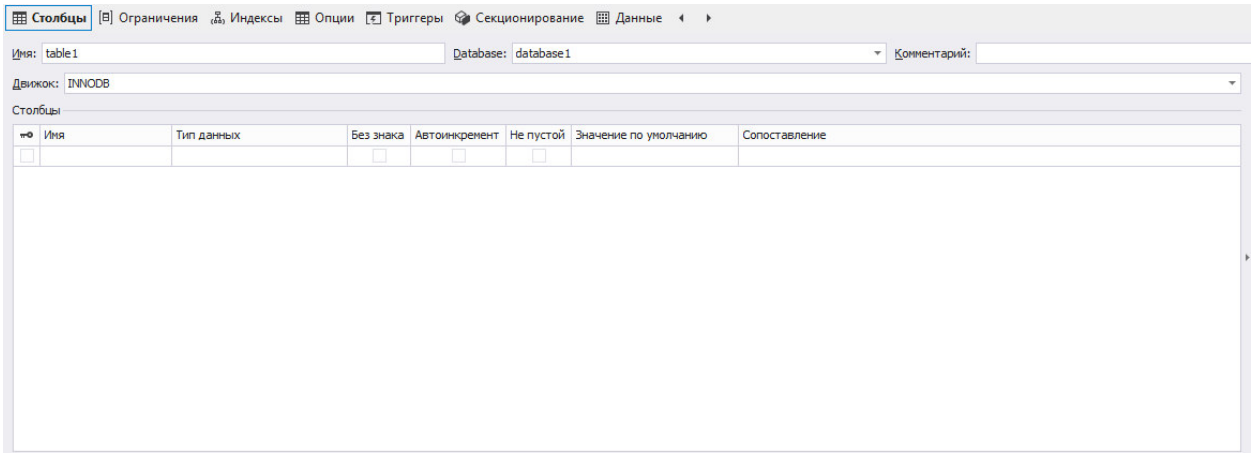

Рисунок 3.4.6 – Настройка таблиц

3.5 Инструменты для тестирования

Для проверки работоспособности и тестирования запросов использовался Postman, интерфейс программы показан на рисунке 3.4.7

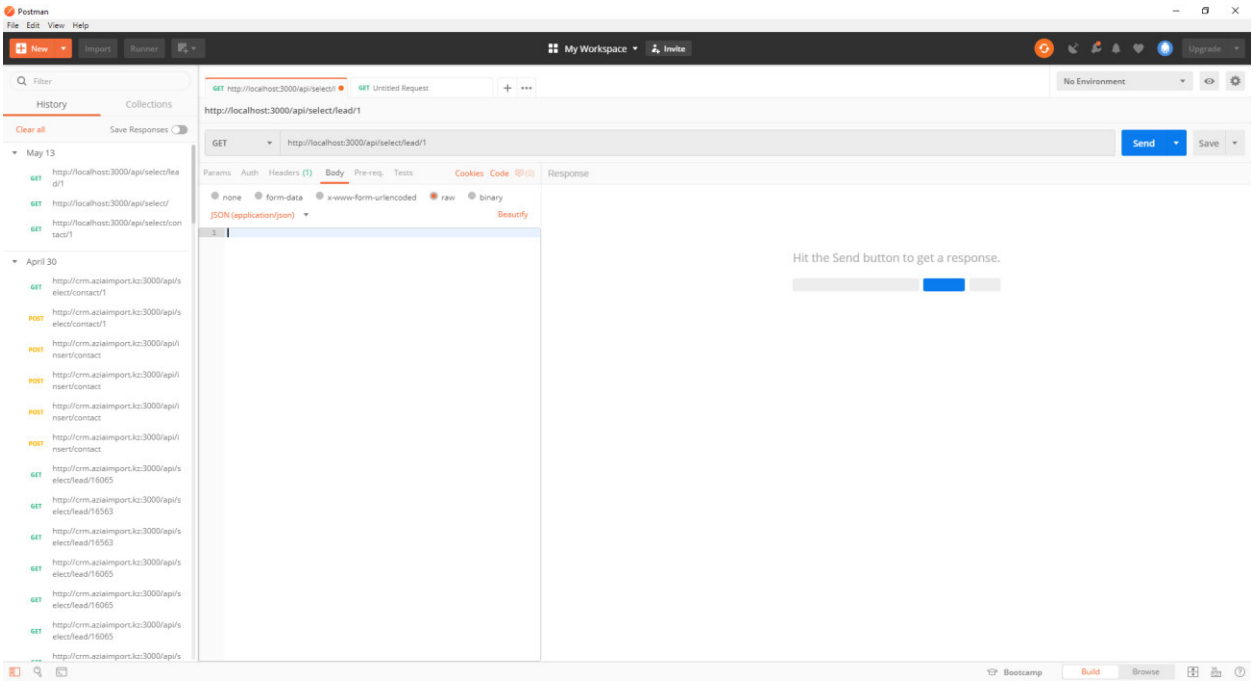

Рисунок 3.4.7 – Интерфейс Postman

Пример GET запроса приведен на рисунке 3.4.8

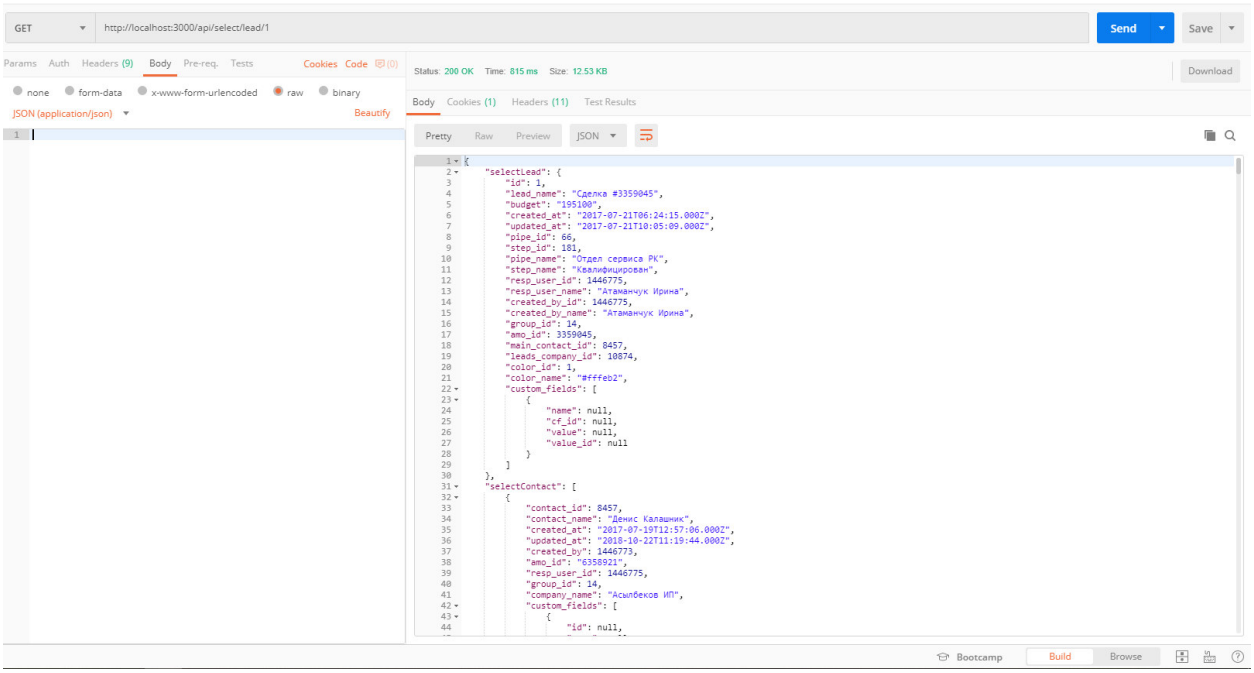

Рисунок 3.4.8 – GET запрос

Запросы POST чуть более сложны, но все равно понятны и логичны. Пример POST запроса приведен на рисунке 3.4.9. Теперь нам нужно тело POST–запроса. Нажмем на "Body" под URL запроса, изменим тип на "raw" и "Text" на "JSON". Теперь вставим этот запрос в редактор:

```
{
        "name": "test",
        "login": "test@test.kz",
        "password": "1234"
}
```

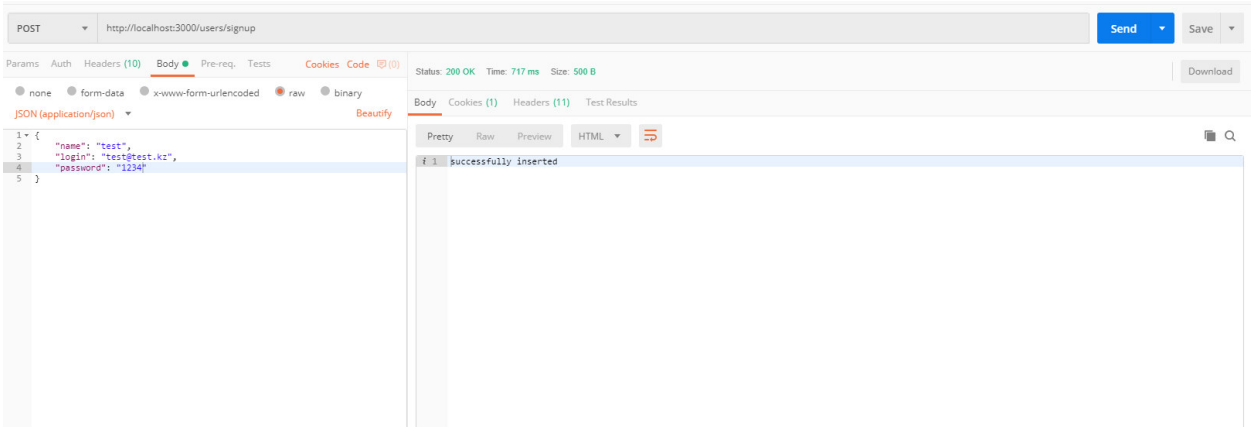

Рисунок 3.4.9 – POST запрос

4 Разработка

Разрабатываемая система представляет собой клиент–серверное приложение. Клиент серверное приложение – это приложение, которое имеет две основные части: клиент и сервер.

Клиент, это та часть приложения, которая отображается пользователю, выполняется в веб–браузере и взаимодействует визуально с пользователем. На этой стороне работают такие языки разметки, стилей и программирования как HTML, CSS и JavaScript.

Серверная часть приложения не имеет собственного визуального представления и взаимодействует с пользователем через веб–браузер. Название этой части вытекает из того, что все действия выполняются на сервере – специальном компьютере, который может быть расположен как за тысячи километров от браузера, так и в непосредственной близости, вплоть до одной машины. На сервере обычно располагается база данных и оперируют такие языки как Java, PHP, C# и т. д.

Перед тем как начать разработку нам потребуется платформа, на которой он будет реализован, в данном дипломном проекте им будет являться Node.js. Скачав установочный файл на официальном сайте nodejs.org, и установив его на компьютер, можно начинать разработку.

Помимо самой платформы понадобится фреймворк Express.js, на нем будет построена архитектура серверной части приложения. Он инкапсулирует обработку высокоуровневых объектов приложения, например, такие как объекты запроса и ответа, предоставляя их в распоряжение разработчику.

Маршрутизатор (Route) – в соответствии с маршрутом делегирует дальнейшую обработку запроса управляющим объектам нижележащего уровня: контроллеру или API.

Контроллер (Controller) – отвечает за обработку запроса. Содержит методы для реализации функциональных требований к приложению.

API – стандартизированный интерфейс для работы с данными.

Архитектура запросов к серверной части будет построена на RESTful API.

REST (сокращение от англ. Representational State Transfer – «передача состояния представления») – архитектурный стиль взаимодействия компонентов распределённого приложения в сети. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа–системы. В определённых случаях (интернет–магазины, поисковые системы, прочие системы, основанные на данных) это приводит к повышению производительности и упрощению архитектуры. В широком смысле[*уточнить*] компоненты в REST взаимодействуют наподобие взаимодействия клиентов и серверов во Всемирной паутине. REST является альтернативой RPC[1].

В сети Интернет вызов удалённой процедуры может представлять собой обычный HTTP–запрос (обычно «GET» или «POST»), а необходимые данные передаются в качестве параметров запроса.

Для веб–служб, построенных с учётом REST (то есть не нарушающих накладываемых им ограничений), применяют термин «RESTful».

В отличие от веб–сервисов (веб–служб) на основе SOAP, не существует «официального» стандарта для RESTful веб–API. Дело в том, что REST является архитектурным стилем, в то время как SOAP является протоколом. Несмотря на то, что REST не является стандартом сам по себе, большинство RESTful–реализаций используют стандарты, такие как HTTP, URL, JSON и XML.

Запрос к API определяется параметром “api” в строке URL. В соответствии с REST, на примере работы с документами, семантика запросов была построена следующим образом, таблица 4.1.

Таблица 4.1 – Семантика запросов

Ресурс/Метод	GET	POST	PUT	DELETE
/lead	Вернуть список всех сделок.	Создать сделку	–	–
/lead/:id	Вернуть сделку с указанным идентификатором	–	Обновить сделку	Удалить сделку

Для удовлетворения требований были реализованы следующие классы и компоненты, представленные в таблицах ниже.

Таблица 4.2 – Классы пакета app и api

Класс	Описание
app	Отвечает за делегирование обработки запроса к контроллерам.
api	Отвечает за делегирование обработки запроса к API.

Таблица 4.3 – Классы пакета auth.

Класс	Описание
auth	Обрабатывает запросы на авторизацию и аутентификацию пользователя

Таблица 4.4 – Классы пакета Routes.

Класс	Описание
index	Отвечает за обработку запроса, и выдачи ответа клиенту

Таблица 4.5 – Коды ошибок реализованные в проекте.

Код	Описание
200	Успешный запрос. Если клиентом были запрошены какие–либо данные, то они находятся в заголовке и/или теле сообщения.
301	Запрошенный документ был окончательно перенесен на новый URI
400	Сервер обнаружил в запросе клиента синтаксическую ошибку.
403	Сервер понял запрос, но он отказывается его выполнять из–за ограничений в доступе для клиента к указанному ресурсу.
404	Самая распространённая ошибка при пользовании, основная причина – ошибка в написании адреса Web–страницы.
500	Любая внутренняя ошибка сервера, которая не входит в рамки остальных ошибок класса.
502	Сервер, выступая в роли шлюза или прокси–сервера, получил недействительное ответное сообщение от вышестоящего сервера.

Ниже представлена часть кода пакета auth, отвечающая за регистрацию пользователя. В коде использовался модуль bcrypt, для шифрования пароля, jsonwebtoken для получения токена, для хранения его в cookie – файлах (Рисунок 4.1).

```
exports.signup = async (req, res, next) => {
    try {
        let hash = bcrypt.hashSync(req.body.password);
        let insert = await conn.query(`INSERT INTO amocrm.users (name, login, password) VALUES (?, ?, ?)`, [req.body.name, req.body.login, hash]);
        res.status(200).send('successfully inserted');
    } catch (e) {
        console.log(e.message);
        next(e);
    }
}
```

Рисунок 4.1 – Часть программного кода.

5 Экономическая часть

Темой дипломного проекта является «Разработка системы управления взаимоотношениями с клиентами»

5.1 Трудовые ресурсы, используемые в работе

В работе над проектом задействованы:

- дизайнер – создание интерфейса приложения;
- инженер–разработчик – разработка алгоритмов, программирование, проектирование базы данных, сопровождение и инструктаж персонала сопровождения;
- программист – разработка алгоритмов и программирование.

Количество сотрудников, задействованных в проекте, и их месячная заработная плата представлены в таблице 5.1.2.

Модель распределения сложности разработки ПО и стадии разработки представлены в таблице 5.1.1

Таблица 5.1.1 – Этапы разработки ПО

Этапы разработки ПО	Вид работы	Трудоемкость, чел. час.
Этап 1	Постановка задач	16
Этап 2	Изучение литературы	32
Этап 3	Выбор среды разработки	8
Этап 4	Просмотр документации	16
Этап 5	Разработка интерфейса	44
Этап 6	Создание необходимых графических элементов	48
Этап 7	Разработка форм приложения	132
Этап 8	Тестирование приложения	16
Этап 9	Подведение итогов по разработанному ПО	8
Этап 10	Внедрение	24
Итого: трудоемкость выполнения		344

Продолжительность рабочего дня равна 8 часам. В результате для реализации программного обеспечения необходимо 43 рабочих дней.

Количество сотрудников, задействованных в проекте, и их месячная заработная плата представлены в таблице 5.1.2.

Таблица 5.1.2 – Данные о сотрудниках

Должность	Количество	Заработная плата в месяц, тенге
Дизайнер	1	100000
Инженер–разработчик	1	160000
Программист	1	150000
Всего	3	410000

5.2 Техническое оборудование, использованное при разработке

Характеристики оборудования, используемого в работе, а также его стоимость приведены в таблице 5.2.1

Таблица 5.2.1 – Перечень оборудования

Оборудование	Название оборудования	Характеристики	Количество, ед.	Стоимость за единицу , тг.
Ноутбук	HP Pavilion 15–bc006ur	Intel Core i5 5600, 8 Gb DDR3, 1024 Gb HDD, NVidia Ge Force 920M	1	380000
Сервер	–	8 Gb ROM	1	90000
Итого			2	470000

5.3 Расчет и определение затрат на создание и реализацию проекта

Разработка Web – приложения – сложный и трудоемкий процесс, требующий, наряду с интеллектуальными, техническими затратами, и финансовых затрат. Поэтому необходимым является произведение расчета стоимости разработки. Разработка Web – приложения требует большого количества интеллектуальных затрат сотрудников, выполняющих работу, а также необходимых технических средств для ее реализации. Все это требует финансовых вложений, на основе которых высчитывается конечная стоимость проекта.

Затраты на разработку данного приложения вычисляются по формуле:

$$С = ФОТ + Сн + А + Э + Спр + Н \quad (5.1)$$

где, ФОТ – фонд оплаты труда;
Сн – социальный налог;
А – амортизационные отчисления;
Э – затраты на электроэнергию;

Спр – прочие расходы;
Н – накладные расходы.

5.3.1 Расходы на оплату труда основным разработчикам

Основные затраты на реализацию программного продукта составляют на выплату заработной платы работникам, которые рассчитываются по формуле:

$$ФОТ = Зосн + Здоп \quad (5.2)$$

где, Зосн – основная заработная плата;
Здоп – дополнительная плата.

Для расчета затрат на основную заработную плату используются данные о среднедневном заработке и фактическом времени работы каждого сотрудника.

Среднедневной заработок:

$$D = \frac{ЗПм}{Др} \quad (5.3)$$

где, ЗПм – ежемесячный размер заработной платы;
Др – количество рабочих дней в месяце (21 день).

Дизайнер:

$$D = \frac{100000}{21} = 4761 \text{ тенге/день}$$

Инженер–разработчик:

$$D = \frac{160000}{21} = 7619 \text{ тенге/день}$$

Программист:

$$D = \frac{150000}{21} = 7142 \text{ тенге/день}$$

Заработная плата за один час работы сотрудника рассчитывается по формуле:

$$Н = \frac{D}{Чр} \quad (5.4)$$

где, D – средний дневной заработок работника;
Чр – количество часов рабочего дня (8 часов)

Дизайнер:

$$Н=\frac{4761}{8}=595,12 \text{ тенге/час}$$

Инженер–разработчик:

$$Н=\frac{7619}{8}=952,37 \text{ тенге/час}$$

Программист:

$$Н=\frac{7142}{8}=892,75 \text{ тенге/час}$$

Сводные результаты расчета затрат на основную заработную плату работников, задействованных в разработке представлены в таблице 4.4.

Таблица 4.4 – затраты на основную заработную плату работников

Наим. / №	Работник	Количество отработанных часов	Количество отработанных дней	Полученная заработная плата
1	Инженер–разработчик	206	26	123786,00
2	Дизайнер	108	13	99047,00
3	Программист	48	6	42852,00
Итого				265685,00

Таким образом, согласно произведенным расчетам, основная заработная плата составляет 265685,00 тг.

Дополнительная заработная плата рассчитывается по формуле:

$$Здоп = Зосн*0.1 \qquad (5.6)$$

и составит:

$$Здоп = 265685* 0.1 = 26568,00 \text{ тг}$$

Фонд оплаты труда, согласно произведенным расчетам и формуле 4.2 составляет:

$$ФОТ = 265685+ 26568 = 292253{,}00 \text{ тг}$$

Далее рассмотрим социальный налог

5.3.2 Расчет затрат по социальному налогу

Социальный налог составляет 11% от дохода сотрудника и рассчитывается по формуле:

$$Сн = (ФОТ – ПО)*0.11 \quad (5.7)$$

где, ПО – пенсионные отчисления, которые составляют 10% от ФОТ и социальным налогом не облагаются и рассчитываются по формуле:

$$ПО = ФОТ * 0{,}1 \quad (5.8)$$

$$ПО = 292253 * 0{,}1 = 29225{,}30 \text{ тенге.}$$

Таким образом, социальный налог составит:

$$Сн = (292253– 29225) * 0.11= 28933{,}08 \text{ тенге}$$

Далее вычислим амортизационные отчисления.

5.3.3 Расчет амортизационных отчислений

Амортизационные отчисления рассчитываются по формуле:

$$A_j = \frac{H_A * C_{ПЕР} * N}{100*12*n} \quad (5.9)$$

где, H_A–норма амортизации;

$C_{ПЕР}$ – первоначальная стоимость оборудования;

N – количество дней на выполнение работ;

n – количество рабочих дней в месяце.

Следовательно, амортизационные отчисления по используемому оборудованию, в соответствии с формулой 4.9 составят:

На оборудование:

$$A = \frac{25*470000*43}{100*12*21} = \frac{505250000}{25200} = 20049{,}60 \text{ тг}$$

5.3.4 Расчет затрат на электроэнергию

Так как в процессе производства используется электрооборудование необходимо рассчитать затраты на электроэнергию. Затраты на электроэнергию для производственных нужд включают в себя расходы электроэнергии на оборудование и дополнительные нужды, рассчитываются по формуле:

$$Э=Зэл.эн.обор. +Здоп. нужд \quad (5.11)$$

где, Зэл.эн.обор – затраты на электроэнергию оборудования;
Здоп.нуж – затраты электроэнергии на дополнительные нужды.
Расходы электроэнергии на оборудование рассчитываются по формуле:

$$Зэл.эн.обор. =W*T*S*Кисп \quad (5.12)$$

где, W – потребляемая мощность, Вт;
T – количество часов работы оборудования;
S – стоимость киловатт–часа электроэнергии (1кВтч = 16.65 тенге);
Кисп – коэффициент использования (Кисп = 0,9).
Потребляемая мощность HP Pavilion 15–bc006ur составляет 90 Вт.
Потребляемая мощность сервера составляет 65 Вт. Потребляемой мощностью адаптеров для зарядки смартфона и планшета пренебрегаем, т.к. они используются исключительно для тестирования и
отладки приложения, при этом постоянно подключены к ноутбуку.

Время высчитывается на основе количества рабочих дней и рабочих часов в день.

Затраты на электроэнергию основного оборудования составляют:

$$Зэл.эн.обор = (0.09 + 0.065)* (40 * 8) * 16.65 * 0.9 =7432,00 \text{ тенге}$$

Затраты на дополнительные нужды берутся по укрупненному показателю в размере 5% от затрат на оборудование:

$$Здоп.нуж = 0,05 * Зэл.эн.обор \quad (5.13)$$
$$Здоп.нуж = 0,05 * 7432= 371,60 \text{ тенге}$$

Суммарные затраты на электроэнергию составляют:
Э = 7432 + 371,60 = 7803,60 тенге
Рассчитаем накладные расходы.

5.3.5 Расчет накладных расходов

Накладные расходы рассчитываются как 20% от всех затрат и определяются по формуле:

$$Нр = (ФОТ + Сн + А + Э)*0.2 \quad (5.14)$$

Нр = (292253+ 28933 + 20049,60+ 7803) * 0,2 = 349038,6 * 0,2 = 69807,72 тенге

Сводные результаты расчета стоимости разработки Web – приложения с применением кроссплатформенных технологий представлены в таблице 5.8.1 и диаграмме 5.1.

Таблица 5.8.1 – Результирующая таблица себестоимости Web – приложения

Наименование статей затрат	Сумма, тенге	В процентах от общей суммы, %
ФОТ	292253,00	69
Затраты по социальному налогу	28933,08	7
Амортизационные отчисления	20049,60	5
Затраты на электроэнергию	7803,60	2
Накладные расходы	69807,72	17
ИТОГО	418847	100

5.3.6 Цена реализации проекта

Цена проекта складывается из себестоимости и желаемого чистого дохода. Минимальная цена программного продукта рассчитывается по следующей формуле:

$$Цп = С + П \quad (5.15)$$

где, С – себестоимость разработки приложения;
П – чистый доход от программного продукта.

Для определения начальной цены используется желаемый уровень рентабельности. Для данной отрасли он составляет 25%.

$$Ц_п = С*(1+\frac{Р}{100}) \quad (5.16)$$

где, Р – рентабельность.

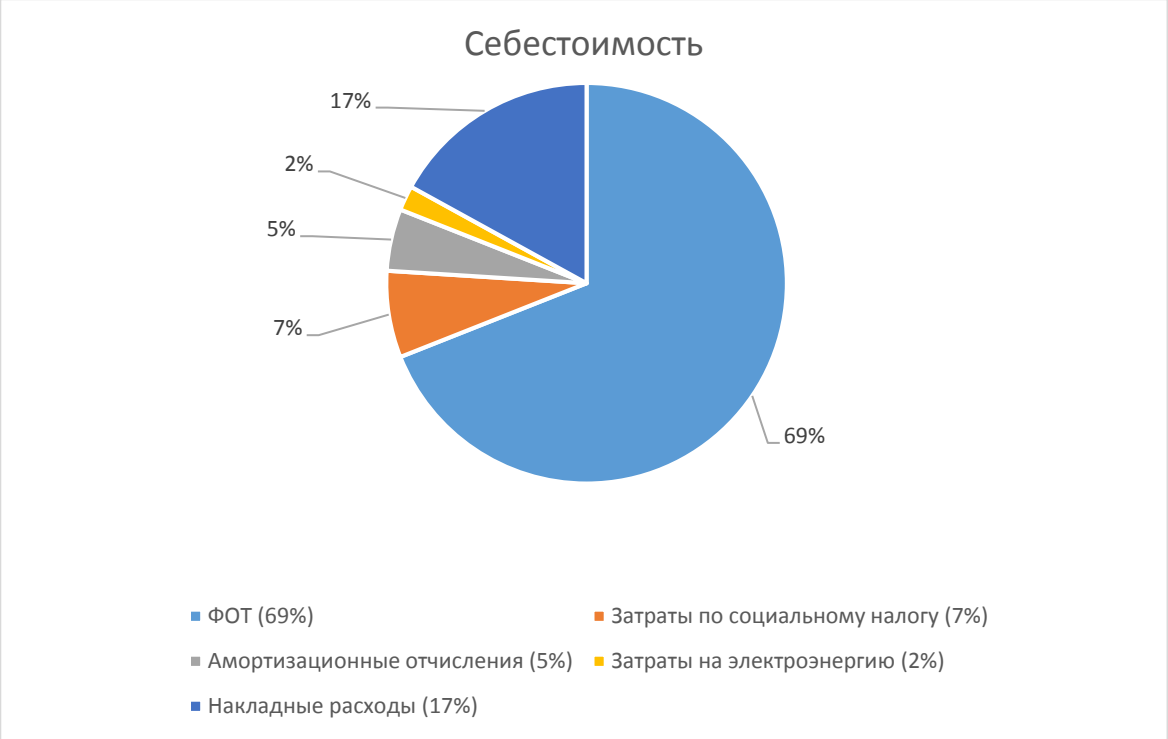

Диаграмма 5.1 – Себестоимость проекта

Тогда по формуле 5.16:

$$Цп=418847*\left(1+\frac{25}{100}\right)=418847*1,25=523558,75$$

Далее определяется цена реализации с учётом налога на добавленную стоимость (НДС):

$$Цр = Цп + НДС \qquad (5.17)$$

где, НДС – налог на добавленную стоимость.

Ставка НДС на 2019 год, согласно пункту 1 статье 268 «Ставки налога на добавленную стоимость» Налогового кодекса РК составляет 12 %, тогда:

$$Цр = 523558,75 + (523558,75 * 0,12) = 586385,80 \text{ тенге}$$

Стоимость разработки Web – приложения составила 586385,80 тенге, в эту стоимость заложены все возможные затраты при разработке программного продукта.

Прибыль от реализации программного продукта равна (5.13):

$$П= Ц_р-С \qquad (5.18)$$

Расчет прибыли приведен в формуле 5.14.

$$П=586385,80 - 418847 =167538,80$$

где: П – прибыль;

$Ц_р$ – цена реализации ПП;

С – себестоимость.

Таким образом, стоимость разработки с учетом НДС составляет 586385,80 тг, а прибыль от проекта составляет 167538,80

5.4 Вывод по технико–экономическому разделу

Стоимость разработки Web – приложения составила 586385,80 тенге, в которую заложены все возможные затраты при разработке программного продукта, прибыль составляет 167538,80.

По анализу расходов на реализацию проекта основной статьей расходов является затраты на оплату труда.

Разработка Web – приложения являет дорогим проектом, требующих больших интеллектуальных и финансовых затрат, а также обязательного использования компьютерной техники. Расчеты показали, что реализация проекта будет экономически целесообразна при условии высокого качества продукта и активной рекламной кампании, позволяя получить прибыль по истечению определенного времени, что позволяет сделать вывод, что производство и реализация данного программного продукта можно считать экономически выгодным, эффективным и рентабельным.

6 Охрана труда и безопасность жизнедеятельности

Темой дипломного проекта является «Разработка системы управления взаимоотношениями с клиентами»

Цель данного дипломного проекта заключается в разработке сайта, позволяющего предприятию заменить кадры сайтом.

Пожарная безопасность обеспечивается системой предотвращения пожара и системой пожарной защиты. Во всех служебных помещениях обязательно должен быть «План эвакуации людей при пожаре», регламентирующий действия персонала в случае возникновения очага возгорания и указывающий места расположения пожарной техники.

Как известно, пожар может возникнуть при взаимодействии горючих веществ, окислителя и источников зажигания. В помещениях с ПК присутствуют все три основные фактора, необходимые для возникновения пожара.

Горючими компонентами являются: строительные материалы для акустической и эстетической отделки помещений, перегородки, двери, полы, изоляция кабелей и др.

Источниками зажигания в ПК могут быть электрические схемы от ПК, приборы, применяемые для технического обслуживания, устройства электропитания, кондиционирования воздуха, где в результате различных нарушений образуются перегретые элементы, электрические искры и дуги, способные вызвать загорания горючих материалов.

6.1 План эвакуации, категорию и степень огнестойкости здания

Здание относится к категории огнестойкости Д – Пониженная пожароопасность, так как, в здании хранятся негорючие вещества и материалы в холодном состоянии. На рисунке 6.1 изображен план эвакуации

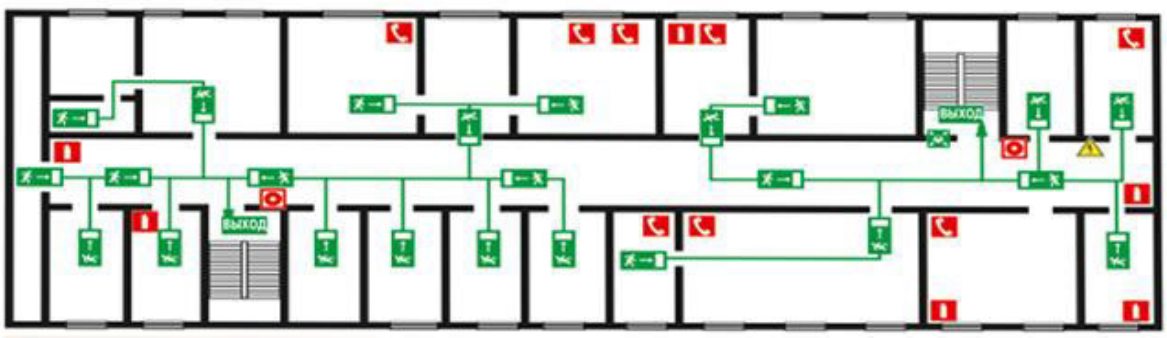

Рисунок 6.1 – План эвакуации

Необходимо определить время эвакуации из кабинета сотрудников предприятия при возникновении пожара в здании. Административное здание не оборудовано автоматической системой сигнализации и оповещения о

пожаре. Здание 2 этажное, имеет размеры в плане 15х45 м, в его коридорах шириной 3 м имеются схемы эвакуации людей при пожаре. Кабинет объемом 62,5 м^3 расположен на втором этаже в непосредственной близости от лестничной клетки, ведущей на первый этаж. Лестничные клетки имеют ширину 2 м и длину 30 м. В кабинете работает 3 человека. На этаже около 100 человек.

Таблица 6.1 – Исходные данные

Этажность	Размеры в плане	Ширина коридора	Объем, м^3	Ширина лестничной клетки, м	Длина пути, м
2	15х45	3	62,5	2	30

6.2 Расчетная часть

6.2.1 Расчет допустимой продолжительности при пожаре с учетом мебели в помещении

$$\tau = \sqrt[3]{\frac{W_{пом} * c * (t_{Kp} - t_H)}{(1-\varphi) * Q * f * n}} \qquad (6.1)$$

где Wпом – объем воздуха в рассматриваемом здании или помещении, м^3;

c – удельная изобарная теплоемкость газа, кДж/кг–град;

t_{Kp} – критическая для человека температура, равная 70°С;

t_H – начальная температура воздуха, °С;

φ – коэффициент, характеризующий потери тепла на нагрев конструкций и окружающих предметов принимается в среднем равным 0,5;

Q – теплота сгорания веществ, кДж/кг;

f – площадь поверхности горения, м^2;

n – весовая скорость горения, кг/м^2–мин.

Свободный объем помещения соответствует разности между геометрическим объемом и объемом оборудования или предметов, находящихся внутри. Если рассчитать свободный объем невозможно, допускается принимать его равным 80% геометрического объема. Удельная теплоемкость сухого воздуха при атмосферном давлении 760 мм рт. ст. составляет 1005 кДж/кг·град при температуре от 0 до 60°С и 1009 кДж/кг·град при температуре от 60 до 120°С.

$$\tau = \sqrt[3]{\frac{62,5 * 1009 * (70-25)}{(1-0,5) * 14 * 25 * 14}} = \sqrt[3]{\frac{2837812,5}{2450}} = \sqrt[3]{1\ 158,29} = 10,50 \text{ мин.}$$

Критическую продолжительность пожара по снижению концентрации кислорода:

$$\tau^0 = \sqrt[3]{\frac{(0{,}01)^{-1} * W\text{пом}}{\pi * n * W2 * V^2}} \qquad (6.2)$$

где W_2 – расход кислорода на сгорание 1 кг горючих веществ, м /кг, согласно теоретическому расчету составляет 4,76 об мин;

v – линейная скорость распространения огня по поверхности горючих веществ, м/мин, равная 0,36 м/мин.

$$\tau^0 = \sqrt[3]{\frac{(0{,}01)^{-1} * 62{,}5}{3{,}14*14*4{,}76*0{,}36^2}} = \sqrt[3]{\frac{6250}{27{,}11}} = 6{,}13 \text{ мин}$$

6.2.2 Плотность людского потока

$$D = \frac{N*f}{L*d} \qquad (6.3)$$

где N – число людей;

f – средняя площадь горизонтальной проекции человека;

d – ширина эвакуационного пути, м;

L – длина пути.

Плотность потоков измеряют как отношение части площади проходов, занятой людьми, к общей площади проходов. Эта величина характеризует степень заполнения эвакуационных путей эвакуирующимися. Часть площади проходов, занятую людьми, определяют как сумму площадей горизонтальных проекций каждого человека. Площадь горизонтальной проекции одного человека зависит от возраста, характера, одежды и колеблется в пределах от 0,04 до 0,126 м^2. В каждом отдельном случае площадь проекции одного человека может быть определена, как площадь эллипса (1):

$$f = \pi \frac{ac}{4} \qquad (6.4)$$

где a – ширина человека, м;

c – его толщина, м.

Я возьму усредненное значение проекции человека с шириной в плечах 0,45 м и толщиной 0,27 м.

Таким образом усредненное значение проекции человека получается:

$$f= \pi \frac{0,45* 0,27}{4} \approx 0,1 \text{ м}^2$$

$$D= \frac{100*0,1}{12* 3} = 0,27 \text{ чел/м}$$

6.2.3 Скорость движения

Обследования скоростей движения при предельных плотностях показали, что минимальные скорости на горизонтальных участках пути колеблются в пределах от 15 до 17 м/мин. Расчетная скорость движения, узаконенная нормами проектирования для помещений с массовым пребыванием людей, принимается равной 16 м/мин.

На участках эвакуационного пути или в зданиях, где заведомо плотности потоков при вынужденном движении будут меньше предельных значений, скорости движения будут соответственно больше. В этом случае при определении скорости вынужденного движения исходят из длины и частоты шага человека. Для практических расчетов можно скорость движения определять по формуле (6.2):

$$V=n*(D-0,1) \qquad (6.5)$$

где *n* – число шагов в мин., равное 100.

Таким образом скорость движения будет равна:

$$V=100* (0,33-1)=23 \text{ м/мин}$$

6.2.4 Время движения

Продолжительность эвакуации людей до выхода наружу из здания определяют по протяженности путей эвакуации и пропускной способности дверей и лестниц. Расчет ведется для условий, что на путях эвакуации плотности потоков равномерны и достигают максимальных значений.

Необходимость учета времени начала эвакуации впервые в нашей стране установлена ГОСТ 12.1.004–91 [6]. Исследования, проведенные в различных странах, показали, что при получении сигнала о пожаре, человек будет исследовать ситуацию, оповещать о пожаре, пытаться бороться с огнем, собирать вещи, оказывать помощь и т.п.

Ввиду того что продолжительность этого этапа существенно влияет на общее время эвакуации, очень важно знать, какие факторы определяют его величину (следует иметь в виду, что большинство этих факторов также будут влиять на протяжении всего процесса эвакуации). Опираясь на существующие работы в этой области, можно выделить следующие:

– состояние человека: устойчивые факторы (ограничение органов чувств, физические ограничения, временные факторы (сон/бодрствование), усталость, стресс, а также состояние опьянения);

– система оповещения;

– действия персонала;

– социальные и родственные связи человека;

– противопожарный тренинг и обучение;

– тип здания.

$$t = \frac{L}{V} \qquad (6.6)$$

Учитывая скорость движения и длину пути время движения составит:

$$t = \frac{30}{23} = 1{,}3 \text{ мин}$$

6.3 Вывод

Таким образом, сделав нужные расчеты, я выяснил что время эвакуации из кабинетов бизнес центра “LOTOS” меньше допустимого. Поэтому здание, в котором располагается предприятие, можно не оборудовать дополнительной системой оповещения о пожаре и средствами автоматической сигнализации.

Заключение

В ходе выполнения дипломного проекта была разработана система управления взаимоотношениями с клиентами.

При выполнении дипломного проекта были выполнены следующие задачи:

– был проведен анализ рынка CRM – систем;

– спроектирована база данных;

– спроектирована программная часть;

– стоимость разработки Web – приложения составила 586384,84 тенге, в которую заложены все возможные затраты при разработке программного продукта.

Также были исследованы условия труда в помещении, где велась разработка, и были предложены мероприятия по улучшению пожарной безопасности.

Список литературы

1 С.Архипенков. Лекции по управлению программными проектами. Москва, 2009.

2 А. Якобсон, Г.Буч, Дж.Рамбо. Унифицированный процесс разработки программного обеспечения. Санкт–Петербург: издательство “Питер”, 2002.

3 Мартин Фаулер. UML Основы, 3–е издание. Санкт–Петербург: издательство “СимволПлюс”, 2005.

4 Г.Буч, Дж.Рамбо, Ивар Якобсон. Язык UML. Руководство пользователя, 2–е издание. Санкт–Петербург: издательство “Питер”, 2007.

5 Mikowski M., Powell J. Single Page Web Applications. Manning, 2014.

6 Алекс Маккоу. Веб–приложения на JavaScript. Москва: издательство “Питер”, 2012

7 JavaScript – [Электронный ресурс] URL:https://developer.mozilla.org (дата обращения: 26.05.2015).

8 Mario Casciaro. Node.js Design Patterns. UK: Packt Publishing, 2014

9 Node.js – [Электронный ресурс] URL:https://nodejs.org/ (дата обращения: 04.05.2015)

10 Шэлли Пауэрс. Изучаем Node.js. Москва: издательство “Питер”, 2014

11 . Exressjs – [Электронный ресурс] URL:http://expressjs.com (дата обращения: 29.03.2015).

12 Ethan Brown. Web Development with Node and Express. US: O’Reilly Media, 2014.

13 В. Васвани. MySQL: использование и администрирование = MySQL Database Usage & Administration. – М.: «Питер», 2011. – 368 с.

14 Роберт Шелдон, Джоффрей Мойе. MySQL 5: базовый курс = Beginning MySQL. – М.: «Диалектика», 2007. – 880 с. – ISBN 978–5–8459–1167–4.

15 Кузнецов Максим, Симдянов Игорь. MySQL на примерах. – Спб.: «БХВ–Петербург», 2008. – С. 952. – ISBN 978–5–9775–0066–1.

16 Поль Дюбуа. MySQL, 3–е издание = MySQL, 3ed. – М.: «Вильямс», 2006. – 1168 с. – ISBN 5–8459–1119–2.

17 Кузнецов Максим, Симдянов Игорь. MySQL 5. В подлиннике. – Спб.: «БХВ–Петербург», 2006. – С. 1024. – ISBN 5–94157–928–4.

18 Кузнецов Максим, Симдянов Игорь. Самоучитель MySQL 5. – Спб.: «БХВ–Петербург», 2006. – С. 560. – ISBN 5–94157–754–0.

19 «MySQL: особенности и сферы применения» URL: https://www.bytemag.ru/articles/detail.php?ID=6547 (дата обращения: 12.02.2019)

20 «SQL или NoSQL – вот в чём вопрос» URL: https://habr.com/ru/company/ruvds/blog/324936/ (дата обращения: 08.02.2019)

Приложение А
(обязательное)

Техническое задание для CRM – системы

1 Наименование программного продукта (ПП):
CRM (Customer Relationship Management)

2 Цель разработки ПО:
Целью разработки является создание удобной и интуитивно понятной системы управления взаимоотношениями с клиентами. Так как на рынке наши пользователи в используют российские аналоги, то целью так же является разработка отечественного продукта ориентированного на наш рынок. Таким образом основной целью является разработка отечественного продукта и дальнейшая дистрибьюция его на рынок.

3 Идеология программного обеспечения:
Идеологией программного обеспечения является удобство и простота использования пользователем, т.е. максимальная клиентоориентированность итогового продукта.

4 Используемая технология создания ПО:
REST API.

5 Выбор модели ПО:
Структурная модель. Позволяет оформлять отдельные блоки программы (повторяющиеся и неповторяющиеся) в процедуры и функции, которые затем могут использоваться в других частях программы. Изменения в коде функций и процедур не влекут за собой изменение других частей кода программы.

Приложение Б
(обязательное)

Листинг программы

app.js

```
var createError = require('http–errors');
var express = require('express');
var path = require('path');
var cookieParser = require('cookie–parser');
var logger = require('morgan');
global.config = require('./config/config');
var bodyParser = require('body–parser');
var cors = require('cors')

var indexRouter = require('./routes/index');
var usersRouter = require('./routes/users');
var regRouter = require('./controller/registration');
var axios = require('./axios/axios');
var mail = require('./mail/mailer');
var excel = require('./controller/excel');
var word = require('./controller/word');
var sms = require('./controller/sms')


var app = express();

app.use('/', indexRouter);
app.use('/api/users', usersRouter);
app.use('/axios', axios);
app.use('/mail', mail);
app.use('/excel', excel);
app.use('/word', word);
app.use('/sms', sms)

app.use(function(req, res, next) {
  res.header('Access–Control–Allow–Credentials', true);
  res.header('Access–Control–Allow–Origin', '*');
  res.header('Access–Control–Allow–Methods',
'GET,PUT,POST,DELETE,UPDATE,OPTIONS');
  res.header("Access–Control–Expose–Headers", "*");
  res.header('Access–Control–Allow–Headers', 'X–Requested–With, X–
HTTP–Method–Override, Content–Type, Accept');
```

```
res.header("Access–Control–Allow–Headers", "Origin, Accept, Content–Type, Access–Control–Allow–Headers, Authorization, X–Requested–With, X–HTTP–Method–Override");
  next();
});
app.use(bodyParser.json())

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'pug');

app.use(logger('dev'));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));




// catch 404 and forward to error handler
app.use(function(req, res, next) {
    console.log(Error);
    next(createError(404));
});

// error handler
app.use(function(err, req, res, next) {
  // set locals, only providing error in development
  res.locals.message = err.message;
  res.locals.error = req.app.get('env') === 'development' ? err : {};

  // render the error page
  res.status(err.status || 500);
  res.render('error');
});

module.exports = app;
auth.js

const mysql = require('mysql');
const bcrypt = require('bcrypt–nodejs');
```

```
const jwt = require('jsonwebtoken');
var express = require('express');
var Query = require('node–mysql–ejq');

const conn = mysql.createConnection(config.db);
const secret = require('../config/config');
var query = new Query(conn);

var router = express.Router();

exports.signup = async (req, res, next) => {
	try {
		let hash = bcrypt.hashSync(req.body.password);
		let insert = await conn.query(`INSERT INTO amocrm.users
(name, login, password) VALUES (?, ?, ?)`, [req.body.name, req.body.login, hash]);
		res.status(200).send('successfully inserted');
	} catch (e) {
		console.log(e.message);
		next(e);
	}
}

exports.signin = async (req, res, next) => {
	try {
		let user = await conn.query(`SELECT * FROM amocrm.users
WHERE login = '${req.body.login}'`);
		if (!user[0]) {
			res.status(401).send('Not AUTH!');
			return;
		}

		user = user[0];
		if (!bcrypt.compareSync(req.body.password, user.password)) {
			res.status(401).send('Not AUTH!');
			return;
		}
		const payload = {
			id: user.id
		}
		let token = jwt.sign(payload, config.jwtSecret, {expiresIn: 144000
});
		res.status(200).send({token})
```

```
            //res.status(200).cookie('Authorization', token, { maxAge: 900000,
httpOnly: true }).send({token});
            } catch(e) {
                  console.log(e);
                  next(e);
            }
      }

      exports.compreg = async (req, res, next) => {
            console.log('asdf');
            var data = req.body;
            var table = req.params.table;
            try{
                  await jwt.verify(req.cookies.token, config.jwtSecret, function(err,
decoded){
                        if(err){
                              res.status(401).send('Unauthorized user');
                        } else {
                              data.user_id = decoded.id;
                        }
                  });
                  console.log(data)
                  var insert = await query.insert({table: 'company', data: data});
                  res.status(200).send();
            } catch(e){
                  console.log(e.message);
                  next(e);
            }
      }

      exports.user = async (req, res)=> {
            try{
                  res.status(200).send(req.user);
            }catch(e){
                  res.status(500).send(e);
            }
      }

      exports.mid = async (req, res, next) => {

            // console.log(req)
            const authHeader = req.get('Authorization');
```

```
            console.log('authHeader',authHeader)
            try{

                var {id} = await jwt.verify(authHeader, config.jwtSecret);
                var select = await conn.query(`SELECT u.id user_id, u.name
username, u.login login, u.group_id group_id, u.is_free is_free, u.is_active is_active,
u.is_admin is_admin, u.amo_id amo_id, p.id priv_id, p.mail, p.incoming_leads,
p.catalogs, p.lead_add, p.lead_view, p.lead_edit, p.lead_delete, p.lead_export,
p.contact_add, p.contact_view, p.contact_edit, p.contact_delete, p.contact_export,
p.company_add, p.company_view, p.company_edit, p.company_delete,
p.company_export, p.task_edit, p.task_delete
FROM
    users u
LEFT JOIN
    privileges p
ON
    p.user_id = u.id
WHERE
    u.id = ${id}`)
req.user = select[0];
            }catch(e){
                if(!id){
                    res.status(401).send('poshel naxuiy');
                    return;
                }
            }
            next();
    }
    var mysql = require('mysql');
    var util = require('util');
    var express = require('express');
    var config = require('../config/config');
    var {mid} = require('../controller/auth');
    var con = mysql.createConnection(config.db);
    var Query = require('node–mysql–ejq');
    var query = new Query(con);

    con.query = util.promisify(con.query)

    var router = express.Router();

    //select all leads by pipeline or pipeline with step limit 20
```

```
router.post('/api/select', async function(req, res){
	var p_id = req.body.p_id;
	var s_id = req.body.s_id;
	var list = req.body.list;
	try{
		if(list == 1 && !s_id){
			var select = await con.query(`SELECT L.id leads_id, L.name leads_name, L.budget budget, L.created_at leads_created_at, L.main_contact_id main_contact, L.leads_company_id leads_company, C.id main_contact_id, C.name contact_name, leads_company_id, LC.name company_name, P.id pipeline_id, P.name  pipeline_name, P.pos pipeline_position, S.id step_id, S.name step_name, S.position step_position, CL.id color_id, CL.name color_name
FROM
	leads L
LEFT JOIN
	contacts C
ON
	L.main_contact_id = C.id
LEFT JOIN
	leads_company LC
ON
	L.leads_company_id = LC.id
LEFT JOIN
	pipelines P
ON
	L.pipeline_id = P.id
LEFT JOIN
	step S
ON
	L.status = S.id
LEFT JOIN
	color CL
ON
	S.color_id = CL.id
WHERE
	P.id = ${p_id}
AND
	L.is_deleted = 0
ORDER BY
	L.updated_at DESC
LIMIT 50`)
```

```
        var selCount = await con.query(`SELECT COUNT(*) AS count
FROM
        leads
WHERE
        pipeline_id = ${p_id}
AND
        is_deleted = 0`)
var selSumm = await con.query(`SELECT SUM(budget) sumBudget
FROM
        leads
WHERE
        pipeline_id = ${p_id}
AND
        is_deleted = 0`)
        selCount = selCount[0].count;
        selSumm = selSumm[0].sumBudget;
        res.status(200).json({select, selCount, selSumm});

        }else if(!s_id && !list){
        var selectStep = await con.query(`SELECT s.id id, s.name name, s.position
position, cl.id color_id, cl.name color_name
FROM
        step s
JOIN
        color cl
ON
        s.color_id = cl.id
WHERE
        pipeline_id = ${p_id}`)
        for(var i=0; i<selectStep.length; i++){
        console.log(selectStep[i]);
        s_id = selectStep[i].id
        var selectLead = await con.query(`SELECT L.id leads_id, L.name
leads_name, L.budget budget, L.created_at leads_created_at, L.main_contact_id
main_contact, L.leads_company_id leads_company, C.id main_contact_id, C.name
contact_name, LC.id leads_company_id, LC.name company_name, P.id pipeline_id,
P.name  pipeline_name, P.pos pipeline_position, S.id step_id, S.name step_name,
S.position step_position, CL.id color_id, CL.name color_name
FROM
        leads L
LEFT JOIN
        contacts C
```

```
ON
	L.main_contact_id = C.id
LEFT JOIN
	leads_company LC
ON
	L.leads_company_id = LC.id
LEFT JOIN
	pipelines P
ON
	L.pipeline_id = P.id
LEFT JOIN
	step S
ON
	L.status = S.id
LEFT JOIN
	color CL
ON
	S.color_id = CL.id
WHERE
	P.id = ${p_id}
AND
	status = ${s_id}
AND
	L.is_deleted = 0
ORDER BY
	L.updated_at DESC
LIMIT 20`)
```

mailer.js

```
var iconv = require('iconv-lite');
var nodemailer = require('nodemailer');
var iconv = require('iconv');
var util = require('util');
var MailParser = require('mailparser').MailParser;
var bcrypt = require('bcrypt-nodejs');
var imaps = require('imap-simple');
var fs = require('fs');
var path = require('path');
var Imap = require('imap'),
	inspect = require('util').inspect;
process.env["NODE_TLS_REJECT_UNAUTHORIZED"] = 0;
```

```
var mysql = require('mysql');
var express = require('express');
var config = require('../config/config');
var con = mysql.createConnection(config.db);
con.query = util.promisify(con.query)

var router = express.Router();

//add new mail address
router.post('/add', async function(req, res){
try{
	var insert = await con.query(`INSERT INTO mail (user_id, email, password,
	imap_host, imap_port, smtp_host, smtp_port) VALUES (?, ?, ?, ?, ?, ?, ?)`,
	[req.body.usert_id, req.body.password, req.body.imap_port,
	req.body.imap_host, req.body.smtp_host, req.body.smtp_port]);
	res.status(200).send(insert);
} catch(e){
console.log(e.message);
res.send()
}
});

//send messages
router.post('/send', async function(req, res){
var mail_id = req.body.id;
console.log(req.body)
try{
	var select = await con.query(`SELECT * FROM mail WHERE id =
	${mail_id}`)
	console.log(select)
	var transporter = nodemailer.createTransport({
		host: select[0].smtp_host,
		port: select[0].smtp_port,
		secure: false,
		auth: {
		user: select[0].email,
		pass: select[0].password
	},
	tls: {
	// do not fail on invalid certs
	rejectUnauthorized: false
	}
```

```
 });
var mo = req.body;
        console.log(mo);
        var insertData = await con.query(`INSERT INTO messages(toloh, subject, text, mail_id) VALUES (?, ?, ?, ?)`,[mo.to, mo.subject, mo.text, select[0].id]);
const mailOptions = {
        from: `${select[0].email}`, // sender address
        to: req.body.to, // list of receivers
        subject: req.body.subject, // Subject line
        html: req.body.text// plain text body
}
console.log(mailOptions);
transporter.sendMail(mailOptions, function (err, info) {
        if(err){
                console.log(err)
        }else{
                console.log(info);
        }
});
res.status(200).send(select)
} catch(e){
        console.log(e.message);
        res.status(500).send();
};
});
```

Приложение В
(обязательное)

Акты внедрения

ТОО "Start IT"
Республика Казахстан,
040006 г.Талдыкорган,
ул.Ескельды би
д.212

START

Тел: +7 (7282) 40 20 22
+7 (727) 313 30 18
e-mail: info@StartIT.kz
Моб: +7 705 879 23 70
web: StartIT.kz

Акт внедрения мобильного приложения «CRM»

Настоящий Акт свидетельствует, что web - приложение «CRM», разработанное Бериком Султанбеком, внедрено в ТОО «Start IT».

Процесс внедрения проходил с 5 по 6 мая 2019 г.

Заявленные характеристики системы предполагали наличие следующих основных возможностей:

- Заполнение и редактирование сделок;
- Работа с почтой;
- Фильтрация сделок.

В ходе опытной эксплуатации web - приложения подтверждено, что оно обладает всеми заявленными возможностями.

На момент подписания настоящего Акта система установлена и эксплуатируется сотрудниками данной компании. Программой пользуются 7 человек.

Генеральный директор **Гончаренко Д.В.**