

СОДЕРЖАНИЕ

Введение	7
1 Анализ литературы	8
1.1 Понятие CRM системы	8
1.2 Классификация CRM систем.....	8
1.3 Функции CRM систем.....	9
1.4 Архитектура CRM системы.....	11
1.5 Обзор CRM систем	14
1.6 Постановка задачи	20
2 Моделирование предметной области	22
2.1 Анализ требований к программному средству.....	22
2.2 Выбор технологий проектирования.....	22
2.3 Спецификация требований к программному средству.....	25
3 Проектирование программного средства	27
3.1 Общая архитектура программного средства	27
3.2 Разработка модели базы данных	28
3.3 Разработка алгоритма аутентификации клиентских приложений	31
3.4 Разработка алгоритма аутентификации пользователей.....	33
3.5 Разработка функциональности для работы с базой данных CRM	35
4 Тестирование программного средства	36
5 Методика работы с программным средством.....	38
5.1 Размещение проекта на сервере	38
5.2 Взаимодействие с клиентским приложением.....	38
6 Техничко-экономическое обоснование	42
6.1 Характеристика программного продукта.....	42
6.2 Расчет сметы затрат и цены программного средства	42
6.3 Расчет заработной платы исполнителей	46
6.4 Оценка экономической эффективности применения программного средства у пользователя	49
6.5 Расчет капитальных затрат	51
6.6 Расчет экономического эффекта	52
6.7 Выводы по технико-экономическому обоснованию	53
Заключение	55

Список использованных источников.....	56
Приложение А Фрагмент кода программы.....	57

ОПРЕДЕЛЕНИЯ И СОКРАЩЕНИЯ

CRM – система управления отношениями с клиентами

БД – база данных.

СУБД – система управления базой данных.

ПС – программное средство.

ОС – операционная система.

ВВЕДЕНИЕ

В современном бизнесе необходимость автоматизация различных процессов стала уже привычным явлением. Уже становится сложно представить себе складской или бухгалтерский учет без применения специализированного программного обеспечения, торговые представители используют специальные приложения для оформления и отправки заказа в офис прямо с планшета или мобильного телефона, достаточно большая часть заказов приходит с сайта уже в виде готовых к обработке документов. Но при этом взаимоотношения с клиентами, по крайней мере, в среднем и малом бизнесе, почему-то очень часто ведутся без внедрения автоматизации и достаточного внимания к учету.

Что происходит, если работа отдела продаж ведется без системы учета? Каждый менеджер по продажам работает так, как ему удобнее, ведет фиксацию звонков, других видов взаимодействия с клиентами по собственному усмотрению: кто-то – на бумаге, кто-то – в Excel таблицах, а кто-то вообще не считает нужным фиксировать процесс своей работы.

Входящие звонки или заявки с сайта от новых заказчиков также не фиксируются, зачастую даже сложно понять, кто из менеджеров занимается входящей заявкой. В результате реальный учет ведется только на уровне оплаченных заказов и отгрузки товара. А насколько эффективно работает отдел продаж, отрабатываются ли все входящие ЛИДы, проводится ли какая-то работа с уже имеющимися контактами, определить оказывается невозможно.

Кроме того, в случае увольнения или болезни сотрудника, все его неоконченные переговоры и необработанные контакты компания может потерять, что также крайне нежелательно для эффективной работы отдела продаж.

Выход из этой ситуации – автоматизация и стандартизация управления отношений с клиентами, т.е. внедрение CRM-системы.

CRM-система - это прикладное программное обеспечение для организаций, предназначенное для автоматизации стратегий взаимодействия с заказчиками (клиентами), в частности, для повышения уровня продаж, оптимизации маркетинга и улучшения обслуживания клиентов путем сохранения информации о клиентах и истории взаимоотношений с ними, установления и улучшения бизнес-процессов и последующего анализа результатов.

Целью данного дипломного проекта является разработка серверной части CRM системы для малого бизнеса.

1 АНАЛИЗ ЛИТЕРАТУРЫ

1.1 Понятие CRM системы

CRM-система (Customer Relationship Management или Управление отношениями с клиентами) - это - прикладное программное обеспечение для организаций, предназначенное для автоматизации стратегий взаимодействия с заказчиками (клиентами), в частности, для повышения уровня продаж, оптимизации маркетинга и улучшения обслуживания клиентов путем сохранения информации о клиентах и истории взаимоотношений с ними, установления и улучшения бизнес-процессов и последующего анализа результатов.

1.2 Классификация CRM систем

Оперативные — в основе их работы лежит получение оперативного доступа к информации о любом клиенте, вне зависимости от того, каким способом и из какого источника была она получена. Одна из самых положительных черт этих CRM систем заключается в обеспечении надежного взаимодействия между отдельными подсистемами. На сегодняшний день оперативные CRM системы являются наиболее распространенными.

Аналитические — не только обеспечивают доступ к информации, но и позволяют проведение аналитической деятельности компании относительно контактов с клиентами с целью получения информации для разработки стратегии бизнеса. CRM системы этой группы обязательно должны обладать возможностью интеграции с различными инструментами, предназначенными для проведения анализа данных. Следует также иметь в виду, что компания должна обладать большой статистической базой. Основная сфера применения данного класса CRM — электронная коммерция. Основной пользователь — маркетинговые отделы компании, а также клиенты (по запросу).

Коллаборационные — предназначены для достижения более тесного взаимодействия с покупателем, в результате которого он получает возможность принимать участие в деятельности компании, влиять на некоторые процессы, например, в плане разработки дизайна, улучшения обслуживания и т.п. Для того, чтобы это было возможно, создаются оптимальные условия для того, чтобы клиент мог беспрепятственно получать доступ к внутренним процессам компании. Основная сфера применения данных CRM — электронная коммерция.

Sales Intelligence CRM — разрабатываются на основе аналитических систем, имеют возможность для проведения различных видов продаж: подменяющих, перекрестных и т.п. Помимо этого данные системы предоставляют возможность для получения самой различной аналитической информации, касающейся как непосредственно продаж, так и клиентов.

Системы для управления компаниями — представляют собой синтез оперативных и аналитических систем. Помимо прочего, позволяют выделение целевых групп, осуществление двусторонней связи с клиентами. С помощью этих систем создается статистическая база данных компании.

1.3 Функции CRM систем

1.3.1 Автоматизация бизнес-процессов

Разложить все рабочие процессы по полочкам, формализовать их - нетривиальная задача, решаемая бизнес-аналитиками. Если все сотрудники действуют согласно регламентированным процессам, уменьшается количество ошибок, работа компании ускоряется, а результаты труда становятся более прогнозируемыми. Если выполнение процессов прозрачно для руководителей, им гораздо легче выявлять слабые места в работе и направлять усилия сотрудников в нужное русло. Эти задачи могут быть решены с помощью автоматизации процессов, с использованием CRM-системы. Обеспечивая автоматизацию и оперативный контроль хода выполнения бизнес-процессов компании, CRM-система повышает вероятность их своевременного и качественного исполнения. Новые сотрудники быстрее входят в курс дела, улучшается коммуникация между отделами. А средства оценки эффективности бизнес-процессов способствуют оптимизации деятельности компании в целом.

1.3.2 Управление информацией о клиентах

«Сердцем» любой CRM-системы является база данных как физических, так и юридических лиц, которые взаимодействуют с Вашей компанией в рамках деятельности предприятия. Это не только клиенты, но и филиалы компании, партнеры, поставщики, конкуренты. База данных клиентов сама по себе ценный актив, а грамотное управление данными в CRM-системе позволяет использовать информацию в работе с максимальной эффективностью. Клиентская база консолидирована, организация получает полную информацию о своих клиентах и их предпочтениях и, основываясь на этих сведениях, строит стратегию взаимодействия. Единая база данных клиентов и полная история взаимоотношения с ними в совокупности с мощными аналитическими инструментами CRM позволяет удерживать и развивать существующих клиентов, выявляя наиболее ценных, а также привлекать новых клиентов.

1.3.3 Управление продажами

Главная функция CRM-системы - помогать менеджерам планировать продажи, организовывать прозрачное управление сделками и оптимизировать каналы продаж. Система хранит полную историю общения с клиентами, что помогает департаментам продаж анализировать поведение клиен-

тов, формировать подходящие им предложения, завоевывать лояльность. Планирование продаж в CRM-системе организовано в различных срезах.

Менеджер составляет план на основе данных по своим клиентам с учетом вероятности, а руководитель, проанализировав объем подтвержденных платежей, может составить для менеджера стимулирующий план. С помощью инструментов CRM-системы руководители могут контролировать качественные показатели работы менеджеров (воронка продаж), выполнение планов продаж, соблюдение сроков оплаты и поставки. Система позволяет оценивать объем и вероятность сделок, управлять бизнес-процессами продаж, следить за состоянием сделки и анализировать действия конкурентов. Одна из важнейших задач, которую помогает решить CRM-система, - организация cross-sales, up-sales.

Система позволяет формировать матрицу кросс-продаж и продуктово-сегментную матрицу, группировать клиентов по различным параметрам и выявлять их потенциальные интересы. Предлагая инструменты прогнозирования и анализа, автоматизируя взаимодействие сотрудников с клиентами и между собой, CRM-система формирует предпосылки для оптимизации существующих каналов сбыта и увеличения прибыли компании.

1.3.4 Управление маркетингом

CRM-система позволяет оптимальным образом организовать управление маркетингом компании: планировать и проводить маркетинговые мероприятия, управлять ресурсами и бюджетами на маркетинг, координировать все маркетинговые воздействия.

Сотрудники отдела маркетинга получают единую библиотеку маркетинговых материалов, инструменты для сегментации клиентов, автоматизации персонализированных рассылок для целевой аудитории. А для измерения прибыльности проводимых кампаний и эффективности деятельности отдела маркетинга CRM-система предлагает специальные инструменты анализа.

Среди базовых функций CRM-системы для автоматизации маркетинга присутствуют: управление прямыми маркетинговыми акциями (электронная рассылка, прямая рассылка), организация исследований, опросов клиентов. В итоге, такая автоматизация помогает усовершенствовать работу департамента маркетинга и повысить степень удовлетворенности клиентов.

1.3.5 Автоматизация документооборота

CRM-система предусматривает все необходимые инструменты для управления как внешним, так и внутренним документооборотом компании. Эти инструменты предоставляют средства автоматического формирования документов по шаблонам, подготовки печатных форм документов, поддержки версионности документов, быстрого поиска документов в системе, создание электронного хранилища документов и многое другое.

При ведении в CRM документации, можно организовать коллективную работу с документами при гибком разграничении прав доступа, электронное визирование, а также учет взаимосвязей между документами.

1.3.6 Оптимизация коммуникаций внутри компании

Низкий уровень развития коммуникаций между сотрудниками и подразделениями делает работу компании малоэффективной и приводит к сбоям основных бизнес-процессов. Как следствие — снижается прибыльность бизнеса.

CRM-система позволяет организовать эффективное взаимодействие и обмен информацией внутри компании, препятствуя возникновению «информационных провалов» и потере важной информации. Использование CRM-системы в компании поможет синхронизировать действия персонала, контролировать выполнение функциональных ролей команды в сделках, организовать автоматическое распределение задач между сотрудниками различных отделов на основании логики действующих бизнес-процессов.

Благодаря применению единых корпоративных стандартов и лучших практик ведения бизнеса, CRM-система обеспечит быстрое обучение новых сотрудников.

1.3.7 Аналитические возможности CRM-системы

Невозможно повысить рентабельность предприятия без глубокого анализа информации о клиентах, их ценности и доходности, выявления «узких мест» в бизнес-процессах компании, анализа системы продаж. CRM-система позволяет компании получить статистическую информацию, провести сложный анализ данных, что необходимо для принятия стратегически важных бизнес-решений.

Более 100 стандартных отчетов системы дают возможность анализировать и контролировать все типичные задачи бизнеса. С помощью встроенного построителя отчетов можно создать аналитические формы, отвечающие специфическим задачам каждого предприятия.

Кроме того, на панели итогов CRM-системы можно отслеживать KPI (ключевые показатели деятельности), анализ которых позволит руководству оценивать эффективность работы каждого сотрудника.

1.4 Архитектура CRM системы

Основные принципы CRM системы:

- 1) наличие единого хранилища информации, откуда в любой момент доступны все сведения обо всех случаях взаимодействия с клиентами;
- 2) синхронизированность управления множественными каналами взаимодействия;
- 3) постоянный анализ собранной информации о клиентах и принятии соответствующих организационных решений.

Таким образом, этот подход подразумевает, что при любом взаимодействии с клиентом по любому каналу, сотруднику организации доступна полная информация обо всех взаимоотношениях с клиентами и решение принимается на её основе, информация о котором, в свою очередь, тоже сохраняется и доступна при всех последующих взаимодействиях.

Классифицируют возможности (модули) CRM по функциональности и уровням обработки информации. По функциональности выделяют блоки:

- а) продажи;
- б) маркетинг;
- в) сервисное обслуживание.

По уровням обработки информации:

- 1) оперативный — регистрация и оперативный доступ к первичной информации по контактам, компаниям, проектам, документам;
- 2) аналитический — отчетность по первичным данным и самое главное более глубокий анализ информации в различных;
- 3) коллаборационный - уровень организации тесного взаимодействия с конечными потребителями, клиентами, вплоть до влияния клиента на внутренние процессы компании.

Существует два типа CRM-систем, созданных на основе разных технологий:

а) *saas* или система как сервис. При этом варианте все программное обеспечение и данные находится на сервере поставщика услуг. Вы получаете *online*- доступ к системе через браузер, программу-клиент или мобильное приложение. Все процессы происходят на стороне поставщика услуг. (рисунок 1.1).

б) *standalone* — лицензия на установку и использование программного продукта. Вы получаете решение, которое устанавливаете на собственный сервер, при желании, дорабатываете под свои потребности, в зависимости от тех возможностей, которые предоставляет поставщик CRM-системы.

Для реализации инфраструктуры CRM-системы существует ряд широко используемых технологий и серверов приложений.

К наиболее распространенным серверам приложений можно отнести такие программные продукты как Microsoft IIS, Apache HTTP Server, Oracle Application Server.

В качестве хранилищ данных используются реляционные базы данных, доступ к которым осуществляется с использованием технологий OLEDB, ODBC, JDBC и т.п.



Рисунок 1.1 - Обобщенная архитектура SaaS приложения

В качестве технологий программирования используются ASP, ASP.NET, PHP, JSP и прочие.

При реализации функциональных модулей чаще всего используются следующие решения:

а) Интеграция данных основывается на использовании программных посредников. Примером реализации данного подхода является программный интерфейс Unified Content API от компании IBM.

б) Интеграция приложений и сервисов основывается на компонентных технологиях, таких как CORBA, DCOM, веб-сервисы, .NET Remoting.

в) Для индексирования и поиска в корпусе полнотекстовых документов используется метод обратных индексов и различные статистические методы ранжирования результатов.

г) Оповещение пользователей выполняется по заранее определенным событиям с использованием таких push-технологий как рассылка электронной почты, RSS-рассылка и т.д.

д) В области обеспечения безопасности применяется правила распределения привилегий, разделение пользователей на группы и шифрование данных. Идентификация пользователей осуществляется по протоколам SSL, NTLM, Kerberos и т.п. Учетные записи пользователей могут храниться как в базе данных портала, так и в специализированных хранилищах, доступ к которым выполняется по протоколу LDAP.

Хотя подходы к разработке и реализации CRM-систем можно считать достаточно проработанными с точки зрения методов и используемых технологий, существует объективная необходимость их развития.

1.5 Обзор CRM систем

1.5.1 AmoCRM

Работа с API происходит на языке PHP. Всё общение с API проходит в зашифрованном виде по протоколу SSL. Это значит, что все ссылки к API должны содержать протокол HTTPS. Особенно важно помнить это при обращении через JS, если инициируется обращение к сторонним ресурсам. Внутри системы пользователь всегда находится в защищённом соединении и попытка обратиться к HTTP будет заблокирована или браузер пользователя выдаст ему предупреждение.

Сторонние разработчики могут создавать виджеты для amoCRM как на чистом JavaScript, так и с использованием PHP-библиотек. Виджеты в amoCRM могут отображать данные приложения в интерфейсах CRM, подключать JS-скрипты и прочее. Дополнения amoCRM также могут попасть в каталог готовых виджетов. Однако при использовании виджетов сторонних разработчиков пользователю не следует забывать, что отношения переходят в три плоскости: amo, хостер облака и владелец виджета. Это может потребовать дополнительных расходов и большего внимания к безопасности системы и потенциальным уязвимостям.

Ещё в amoCRM можно настроить уведомления, которые будут сообщать сторонним приложениям о событиях, произошедших в CRM. Для уведомлений создаются правила, по которым они обрабатываются.

По умолчанию таблица сделок состоит из 5-и полей:

- 1) название сделки;
- 2) основной контакт;
- 3) компания;
- 4) статус сделки;
- 5) бюджет.

Поля возможно добавлять/удалять из существующих и создавать произвольные в настройках аккаунта. Сделки теггируются, что значительно упрощает работу.

Таблица контактов состоит из 5-и полей:

- 1) контакт;
- 2) компания;
- 3) e-mail;
- 4) телефон.

Поля возможно добавлять/удалять из существующих. Возможность добавления произвольных полей данных не обнаружена, большой минус для нас. Контакты теггируются. Форма отдельного контакта содержит те же возможности, что и окно сделки, кроме возможности добавления произвольных полей данных.

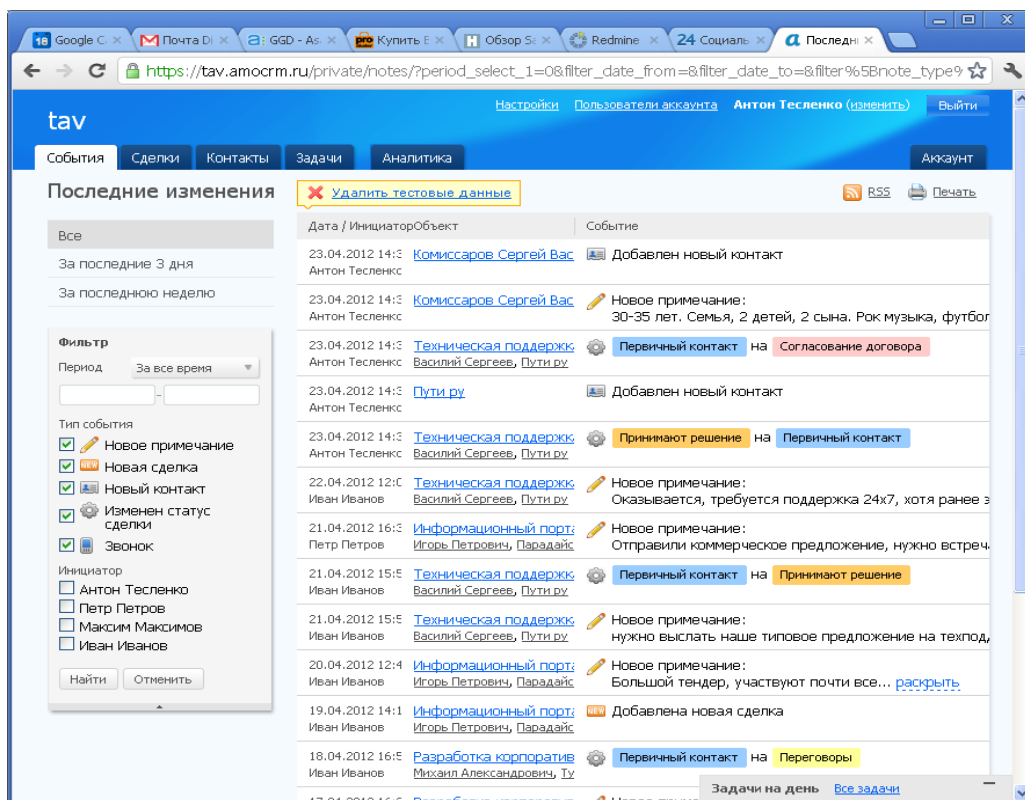


Рисунок 1.2 – Главная страница CRM системы «amoCRM»

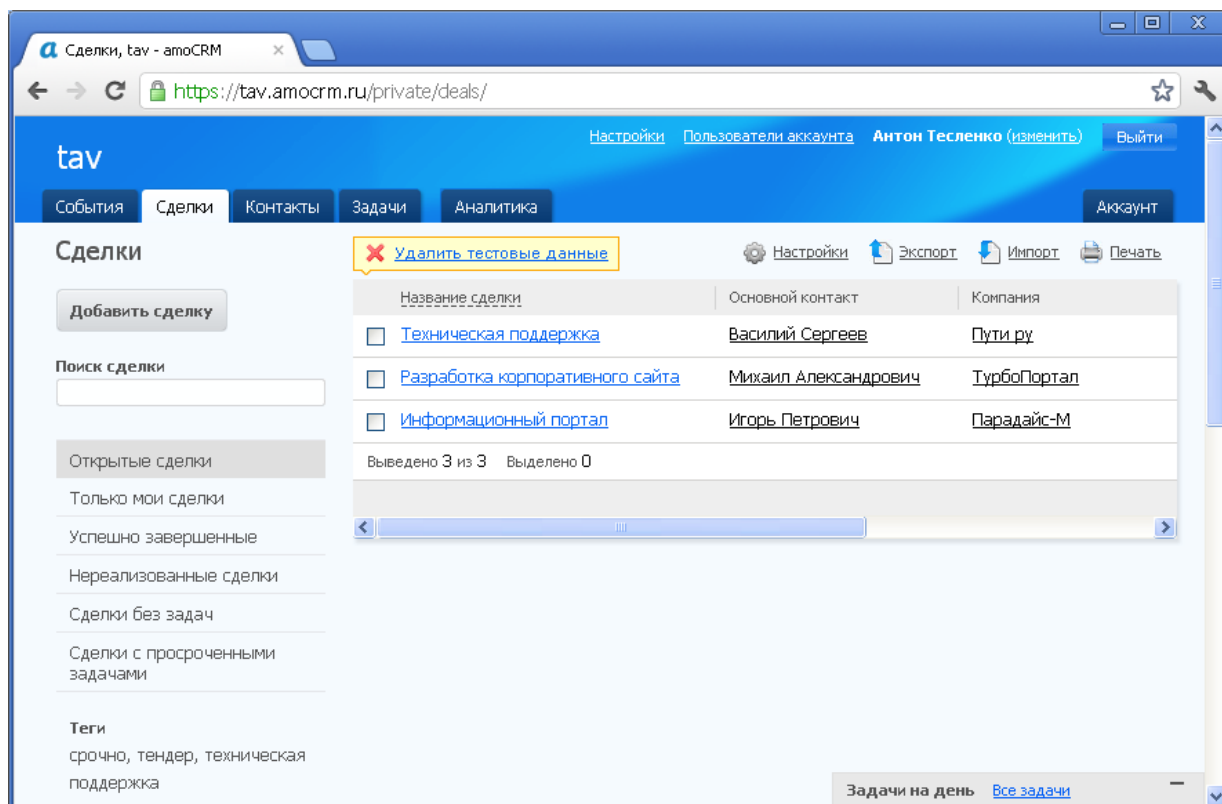


Рисунок 1.3 – Страница сделок CRM системы «amoCRM»

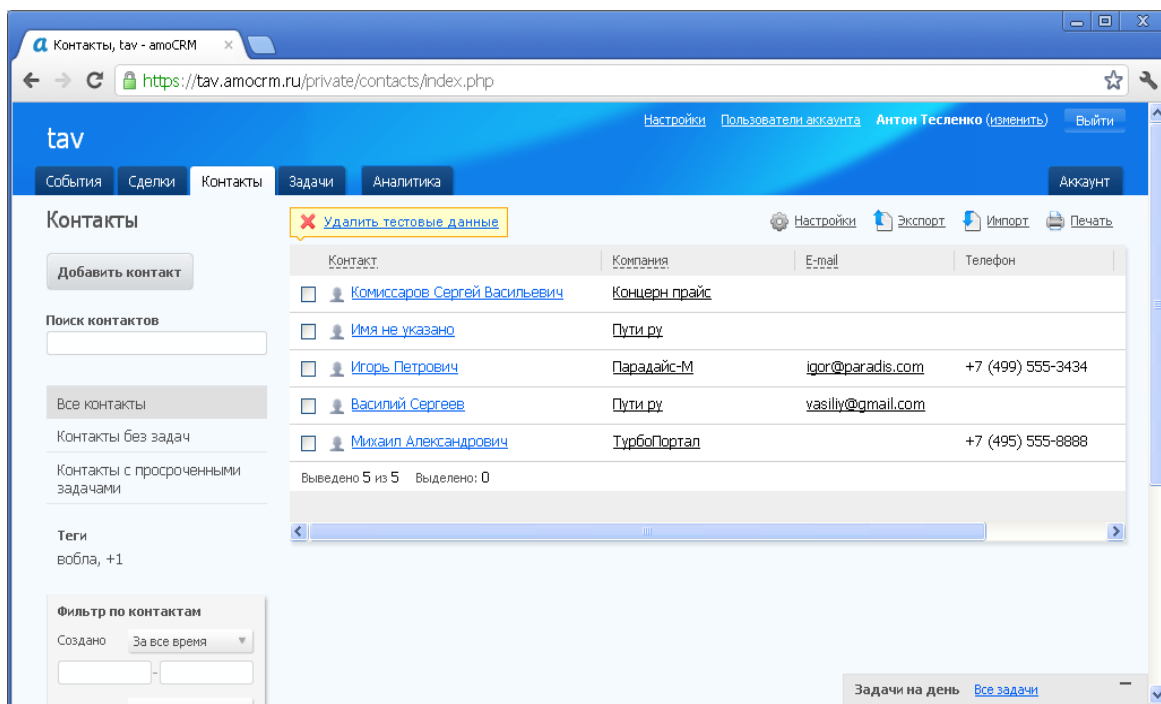


Рисунок 1.4 – Страница сделок CRM системы «amoCRM»

AmoCRM содержит в себе 2 вида отчётов:

- 1) сводный;
- 2) воронка продаж.

Сводный, представлены графики, которые позволяют быстро оценить эффективность работы как всего отдела продаж, так и каждого менеджера в частности. Со страницы отчета можно легко перейти к списку сделок того или иного менеджера, а также просмотреть список сделок на каждом этапе продаж. Обратите внимание, что на круговых диаграммах учитываются только открытые сделки, а на графике новых сделок в разрезе времени — все сделки. Позволяет оценить работоспособность сейлзов и общий уровень продаж. Возможно настроить отчет с использованием тегов.

Воронка продаж, по описанию на сайте «позволяют оценивать эффективность работы на каждой стадии продажи, посмотреть данные в разрезе определенного менеджера, типа клиента или же продукта (для этого используйте теги). Воронка иллюстрирует до какого этапа дошла сделка, на каком этапе она была потеряна. Обратите внимание, что на графике учитываются только закрытые сделки, текущие открытые сделки в нем не отражены».

Достоинства:

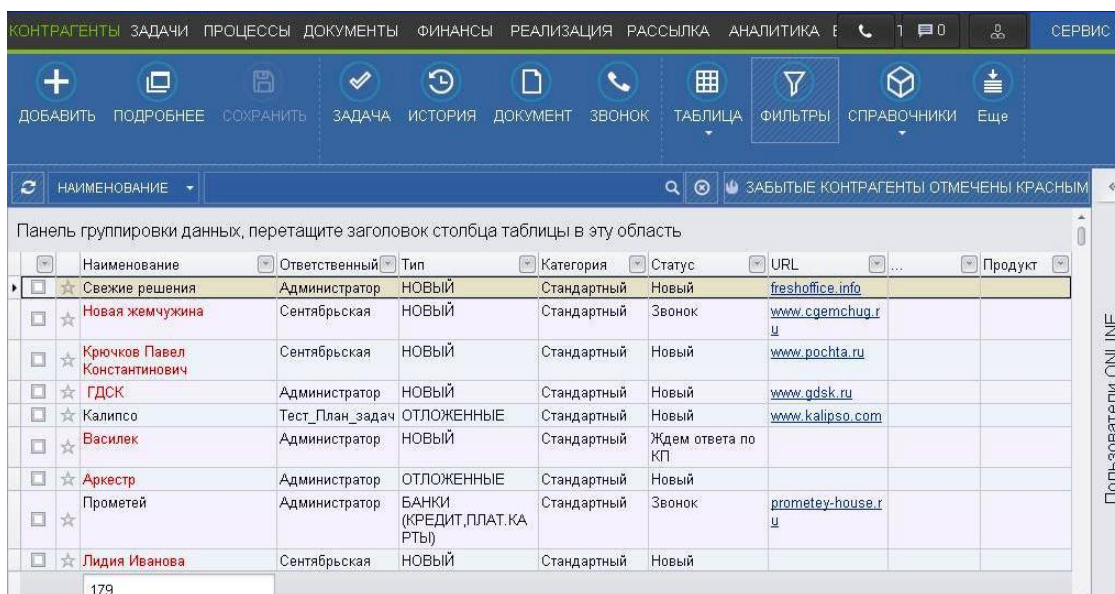
- 1) система очень проста и интуитивно понятна;
- 2) теггирование задач, контактов, сделок – очень удачное решение, подобного функционала не было обнаружено ни в одной другой системе;
- 3) интеграция со сторонними сервисами рассылок – MailChimp, Un-iSender.

Недостатки:

- 1) отсутствует возможность добавлять произвольные поля данных клиентов;
- 2) отсутствуют справочники: компании, продукты.

1.5.2 FreshOffice CRM

Задачи у FreshOffice организованы в табличной и календарной форме. Визуальный планировщик реализован в форме сетки календаря, задачи можно назначать через карточку, открывающуюся по клику по дате. Карточка задачи продумана по целям, процессам, результатам. Задачи допускается переносить и поручать.



Панель группировки данных, перетащите заголовок столбца таблицы в эту область

	Наименование	Ответственный	Тип	Категория	Статус	URL	Продукт
<input type="checkbox"/>	Свежие решения	Администратор	НОВЫЙ	Стандартный	Новый	freshoffice.info	
<input type="checkbox"/>	Новая жемчужина	Сентябрьская	НОВЫЙ	Стандартный	Звонок	www.cgemchug.ru	
<input type="checkbox"/>	Крючков Павел Константинович	Сентябрьская	НОВЫЙ	Стандартный	Новый	www.pochta.ru	
<input type="checkbox"/>	ГДСК	Администратор	НОВЫЙ	Стандартный	Новый	www.gdsk.ru	
<input type="checkbox"/>	Калипсо	Тест_План_задач	ОТЛОЖЕННЫЕ	Стандартный	Новый	www.kalipso.com	
<input type="checkbox"/>	Василек	Администратор	НОВЫЙ	Стандартный	Ждем ответа по КП		
<input type="checkbox"/>	Аркестр	Администратор	ОТЛОЖЕННЫЕ	Стандартный	Новый		
<input type="checkbox"/>	Прометей	Администратор	БАНКИ (КРЕДИТ, ПЛАТ.КАРТЫ)	Стандартный	Звонок	prometey-house.ru	
<input type="checkbox"/>	Лидия Иванова	Сентябрьская	НОВЫЙ	Стандартный	Новый		

179

Пользователи ONLINE

Рисунок 1.5 – Главная страница «FreshOffice CRM»

Процессы — фактически карточка продажи с товарами, задачами, документами, прикрепленными файлами, заметками. Продажу в CRM очень легко заносить, если качественно и грамотно заполнены остальные карточки и справочники. В CRM возможен простой учет финансов по трем типам операций: приход, расход, перевод средств. Для удобства пользователя и наглядности процессы и задачи на разных этапах выполнения имеют разные цвета. При работе в CRM контрагента можно передавать другому сотруднику, назначать куратора, менять отдел.

Аналитика у FreshOffice — красивая панель с возможностью построения отчетов по задачам, результативности, финансам, реализации, процессам, звонкам и документообороту. Отчеты имеют табличную форму, некоторые — графическую реализацию. В печатных формах отчетов колонн-тулы и заголовки автоматически не подгружаются — они могут быть прописаны в настройках отчета вручную, не самое удобное решение с точки

зрения подготовки отчета. Кроме того, в системе предусмотрен несложный OLAP — стандартные отчеты с настраиваемыми полями данных.

Неплохое преимущество этой CRM — свой соффон с возможностью логирования звонков и установкой звука на удержании. Что касается интерактива, то в CRM справа находится панель с пользователями онлайн и возможность обмениваться сообщениями лично и делать рассылку выбранным пользователям. Однако, как мне кажется, в таком интерактиве может крыться одна из причин медлительности системы в целом.

Примечательно, что во FreshOffice свои справочники для каждого раздела, в разделе и отображаемые. База знаний с разделами и подразделами реализована как дерево, включает в себя очень простой редактор, можно прикреплять файлы. Настраиваемые фильтры с выбором условий помогают строить любые выборки в существующих таблицах, пользовательские фильтры могут быть сохранены.

Из недостатков можно отметить отсутствие модуля работы с персоналом (хотя в процессах есть оценка выполнения задачи по сотруднику), непродуманные печатные формы, отсутствие некоторых важных для любой российской CRM функций (производства, заказов). Кроме того, рассылка e-mail из этой CRM возможна только через Outlook.

1.5.3 Битрикс24

«Живая лента» в Битрикс24. Самое слабое место в системе. Аккумулирует, как лента социальной сети, все происходящие события, но из-за их количества и однотипности такой массив данных сложно воспринимать. Есть ощущение, что первое время лента будет читаться, но по мере нарастания количества событий, количество прочтений будет пропорционально снижаться. Впрочем, есть фильтр, позволяющий сортировать события.

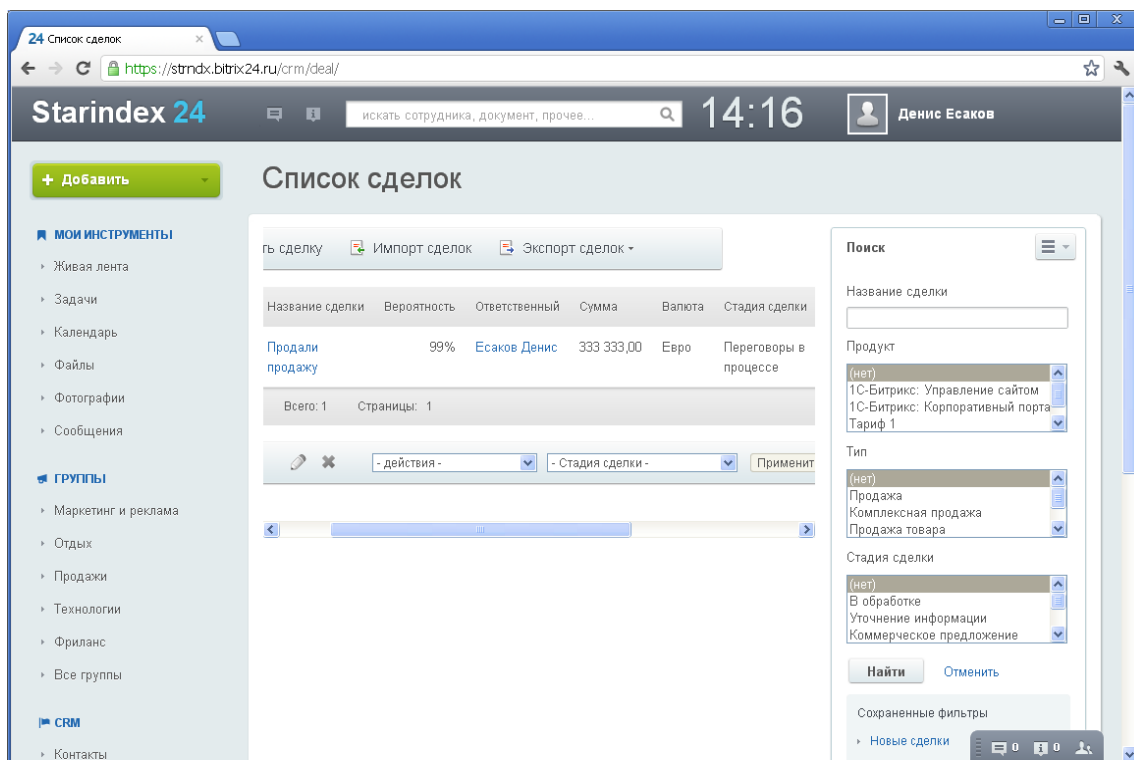


Рисунок 1.6 – Главная страница «Битрикс24»

Настраиваемые поля сделки, подробный фильтр (Поиск), назначаемые действия. Функционал более изощренный, чем в AmoCRM, но менее лаконичный. Пользоваться горизонтальными бегунком очень не удобно.

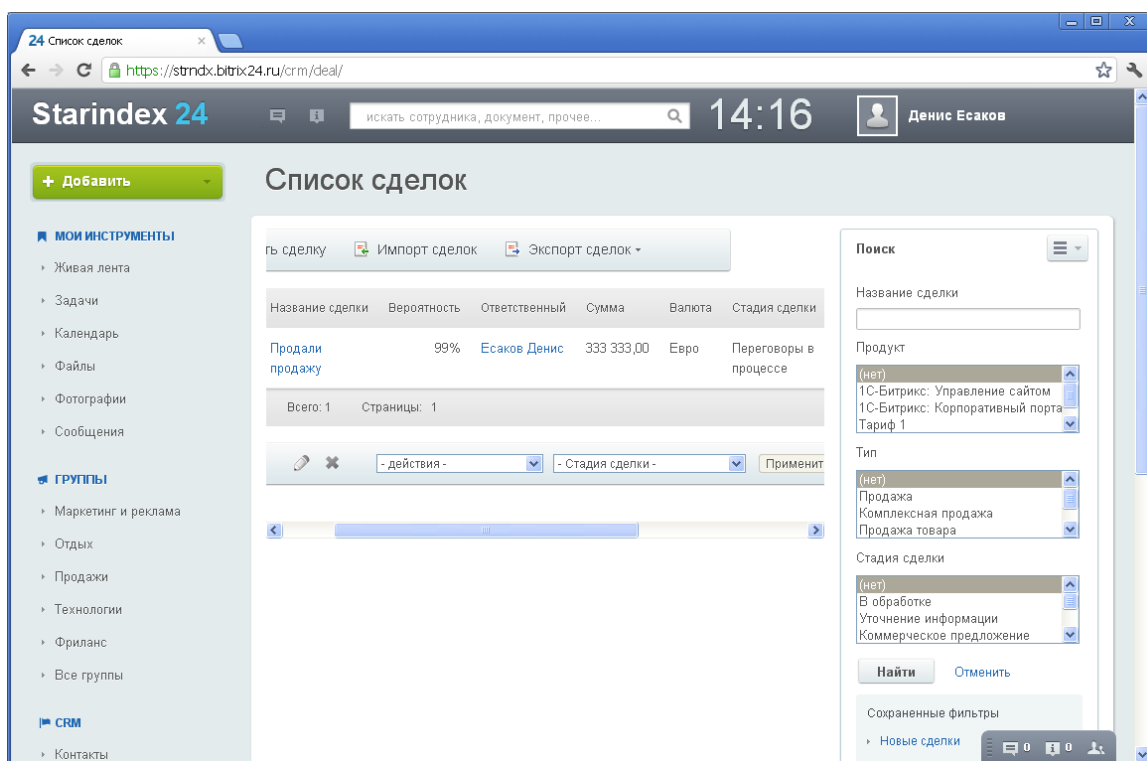


Рисунок 1.7 – Страница сделок «Битрикс24»

Свойства контактов настраиваются под требования пользователя. Гибко настраиваемый фильтр (Поиск). Отдельный пункт – справочник Компании. При экспорте контактов формат полей скачет – числовые форматы (например, телефон) отображаются не в читабельном формате. Надеюсь, это издержки бета-версии, которые будут поправлены в ближайшее время разработчиками.

Большой выбор опций по настройке задач. Есть календарь, в котором они отображаются. Широкие возможности по сортировке задач, два состояния фильтра – обычный и расширенный. Обычный состоит из стандартного набора свойств, расширенный позволяет вводить свои свойства и строить свою собственную выдачу в рамках возможностей Битрикса24.

Доступен только один отчет «Воронка продаж». Но есть гибкий фильтр позволяющий получить и другие виды отчетов – эффективность менеджера, продукты-лидеры/аутсайдеры и пр. Нет отчета по активности контрагентов, существенный минус.

Достоинства:

1) CRM - это лишь часть системы Битрикс24, которая является корпоративным интернет порталом, выполняющим также роль внутренней социальной сети.

2) практически все свойства функционала настраиваются и изменяются.

3) есть справочники, облегчающие работу с данными и унифицирующие их.

4) также есть проектная связка Задачи — Календарь и файловое хранилище.

Недостатки:

1) отсутствуют теги;

2) громоздкий, сложно-понятный интерфейс.

1.6 Постановка задачи

Проанализировав основные функции, а так же все достоинства и недостатки существующих CRM систем на территории СНГ, можно выдвинуть следующие требования к разрабатываемой CRM системе:

- 1) регистрация и авторизация пользователя;
- 2) добавление нового контакта;
- 3) добавление новой сделки;
- 4) добавление новой задачи;
- 5) добавление комментария к сделке;
- 6) добавление комментария к задаче;
- 7) добавление комментария к контакту;
- 8) изменение пользовательских настроек;
- 9) изменение контакта;
- 10) изменение сделки;

11) получение статистики по сделкам за период.

Выходные данные: http-ответ с запрашиваемой информацией в теле сообщения.

Входные данные: http-запрос с заголовками и телом сообщения.

Среда эксплуатации: веб-сервер, поддерживающий следующие технологии:

- а) cent OS;
- б) mysql;
- в) nginx web server.

2 МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

2.1 Анализ требований к программному средству

В реализуемом программном средстве пользователь обращаясь по запросу из клиентского приложения будет обладать списком определенных возможностей. Полный список возможностей для пользователей, поддерживаемых программным средством будет следующим:

- 1) регистрация и авторизация пользователя;
- 2) добавление нового контакта;
- 3) добавление новой сделки;
- 4) добавление новой задачи;
- 5) добавление комментария к сделке;
- 6) добавление комментария к задаче;
- 7) добавление комментария к контакту;
- 8) изменение пользовательских настроек;
- 9) изменение контакта;
- 10) изменение сделки
- 11) удаление контакта
- 12) удаление сделки
- 13) удаление задачи
- 14) удаление комментария к сделке
- 15) получение статистики по сделкам за период

Пользователь может обращаться к данным функциям по http – запросу из клиентского приложения. Каждому http - запросу с набором передаваемых параметров соответствует своя функция.

2.2 Выбор технологий проектирования

В результате анализа выдвинутых требований, а также в целях упрощения разработки, увеличения эффективности, масштабируемости при реализации программного-средства будут задействованы:

- 2) php 5.6 (язык программирования);
- 3) mysql 5.6 (система управления базой данных);
- 4) laravel 5.2 (основной фреймворк для всей архитектуры).

2.2.1 Выбор языка программирования

В качестве языка разработки был выбран язык PHP версии 5.6.

PHP (Hypertext Preprocessor - Препроцессор Гипертекста)– это широко используемый язык сценариев общего назначения с открытым исходным кодом.

PHP - язык программирования, специально разработанный для написания web-приложений (скриптов, сценариев), исполняющихся на Web-сервере. Синтаксис языка во многом основывается на синтаксисе C, Java и

Perl. Он очень похож на C и на Perl, поэтому для профессионального программиста не составит труда его изучить. С другой стороны, язык PHP проще, чем C, и его может освоить веб-мастер, не знающий пока других языков программирования.

Огромным плюсом PHP, в отличие от, например, JavaScript, является то, что PHP-скрипты выполняются на стороне сервера. PHP не зависит от скорости компьютера пользователя или его браузера, он полностью работает на сервере. Пользователь даже может не знать, получает ли он обычный HTML-файл или результат выполнения скрипта.

Сценарии на языке PHP могут исполняться на сервере в виде отдельных файлов, а могут интегрироваться в html страницы.

PHP способен генерировать и преобразовывать не только HTML документы, но и изображения разных форматов - JPEG, GIF, PNG, файлы PDF и FLASH. PHP способен формировать данные в любом текстовом формате, включая XHTML и XML.

Язык программирования PHP, особенно в связке с популярнейшей базой данных MySQL - оптимальный вариант для создания веб-приложений различной сложности.

Язык PHP постоянно совершенствуется, и ему наверняка обеспечено долгое доминирование в области языков web-программирования.

Я использовал язык PHP, т.к. в моём проекте была необходимость работы с базой данных MySQL, с чем данный язык отлично справляется.

2.2.2 Выбор базы данных

В соответствии с требованиями, предъявляемыми к разрабатываемой системе было решено использовать СУБД MySQL версии 5.6.

MySQL – свободная реляционная система управления базами данных.

Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей.

В MySQL 5.6 основные усилия были направлены на повышение производительности, масштабируемости и гибкости. Масштабным по значимости изменениям подвергся движок InnoDB.

К ключевым улучшениям можно отнести: поддержка средств полнотекстового поиска, возможность доступа к данным через memcached API, увеличена производительность работы при интенсивной записи данных, а также увеличена масштабируемость при обработке большого числа одновременных запросов.

Еще одним новшеством версии 5.6 является возможность исполнения DDL-операций (Data Definition Language) без перевода СУБД в офлайн и прерывания доступа к таблицам. Администраторы получают возможность производить операции связанные с сбросом схемы, добавлением или удалением столбцов данных или переименованием столбцов без отключения

СУБД.

Реализован интерфейс для прямого доступа к таблицам InnoDB в стиле NoSQL-систем с использованием API, манипулирующего парами ключ/значение и совместимого с memcached.

Появилась возможность создания в InnoDB полнотекстовых индексов для организации быстрого поиска по словоформам среди текстового контента, хранимого в таблицах InnoDB. Ранее полнотекстовый поиск был доступен только для таблиц MyISAM.

Повышение эффективности оптимизатора запросов, оптимизация процесса выбора результирующего набора значений, сортировки и выполнения запроса. Новые оптимизации Index Condition Pushdown (ICP) и Batch Key Access (BKA) позволяют до 280 раз увеличить пропускную способность выполнения некоторых запросов, увеличена эффективность выполнения запросов вида «SELECT... FROM single_table... ORDER BY non_index_column [DESC] LIMIT [M,]N;», увеличена производительность запросов «SELECT... LIMIT N» выводящих только часть строк из большой выборки.

Расширены средства диагностики работы оптимизатора, добавлена поддержка EXPLAIN для операций INSERT, UPDATE и DELETE. Результаты работы EXPLAIN теперь могут быть выведены в формате JSON. Новый режим трассировки оптимизатора позволяет проследить за каждым принятым решением в процессе оптимизации запроса.

Дополнительные оптимизации выполнения подзапросов, при которых вложенные запросы вида «SELECT... FROM table1 WHERE... IN (SELECT... FROM table2 ...)» транслируются в более оптимальное представление на стадии до непосредственного выполнения запроса, например, заменяются на более эффективный JOIN.

Расширение реализации системы диагностики PERFORMANCE_SCHEMA, предоставляющей низкоуровневые средства для мониторинга за выполнением запросов и различными событиями при работе СУБД. PERFORMANCE_SCHEMA позволяет детально оценить узкие места при выполнении длительных запросов, а также представить сводную статистику, сгруппированную по запросам, нитям, пользователям, хостам и объектам.

Улучшена реализация движка InnoDB, отмечается рост производительности при выполнении транзакций и при активности с преобладанием операций чтения данных — в некоторых ситуациях ускорение достигает 230%.

Режим отложенной репликации, позволяющий реплицировать данные не сразу, а с определённой задержкой, что позволяет обеспечить защиту от ошибок оператора (например, случайное удаление содержимого таблиц).

Увеличение максимального размера файлов с логами изменений (InnoDB Redo Log) с 4 Гб до 2 Тб.

Улучшение безопасности: поддержка указания параметров аутентификации в файле `.mylogin.cnf` в зашифрованном виде; добавление плагина `sha256_password` для хранения хэшей паролей с использованием алгоритма SHA-256; добавление в таблицу `mysql.user` поля со временем истечения действия пароля; новая SQL-функция `VALIDATE_PASSWORD_STRENGTH()` для оценки надёжности пароля.

2.2.3 Выбор фреймворка для разработки

Разрабатываемое приложение является Веб-ориентированным. В связи с этим, возникает необходимость в выборе технологии разработки на языке PHP, которая соответствовала бы гибкости, расширяемости.

Наиболее подходящим для наших целей является фреймворк Laravel.

Laravel – это фреймворк для web приложений с выразительным и элегантным синтаксисом. Он позволит упростить решение основных набравших задач, таких как аутентификация, маршрутизация, сессии и кэширование. Laravel – это попытка объединить всё самое лучшее, что есть в других PHP фреймворках, а также Ruby on Rails, ASP.NET MVC и Sinatra.

Laravel – доступный, но мощный. Располагает множеством отличных инструментов для крупных, надёжных приложений:

- 1) превосходная IoC (Инверсия управления);
- 2) удобная система миграций;
- 3) интегрированная система модульного тестирования.

2.3 Спецификация требований к программному средству

В результате анализа выдвинутых требований и выбора технологий проектирования разработана следующая спецификация требований к программному средству:

- 1) авторизация пользователя через логин и пароль с поддержкой автоматической авторизации без повторного ввода данных;
- 2) регистрация пользователя;
- 3) добавление нового контакта;
- 4) создание новой сделки;
- 5) создание новой задачи;
- 6) изменение настроек приложения;
- 7) просмотр статистики продаж за период;
- 8) удаление контакта;
- 9) удаление сделки;
- 10) удаление задачи.

Входные данные: http-запрос с заголовками и телом сообщения.

Выходные данные: http-ответ с запрашиваемой информацией в теле сообщения.

Среда эксплуатации: веб-сервер, поддерживающий следующие технологии:

- 1) cent OS;
- 2) mySql 5.6;
- 3) php 5.6.

3 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

3.1 Общая архитектура программного средства

Архитектура программного средства построена по стилю REST.

REST — это стиль архитектуры программного обеспечения для построения распределенных масштабируемых веб-сервисов. Принцип взаимодействия клиент-сервер по принципу REST представлен на рисунке 3.1

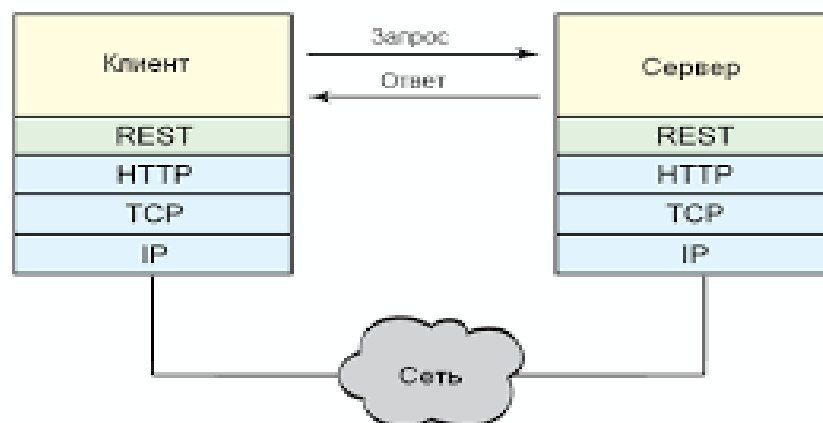


Рисунок 3.1 – Взаимодействие клиента и сервера.

Сервер может считаться RESTful если он соответствует принципам REST. Когда вы разрабатываете API, который будет в основном использоваться мобильными устройствами, понимание и следование трем наиважнейшим принципам может быть весьма полезным. Причем не только при разработке API, но и при его поддержке и развитии в дальнейшем.

Принципы REST:

- 1) независимость от состояния. RESTful сервер не должен отслеживать, хранить и использовать в работе текущую контекстную информацию о клиенте.
- 2) кэширование на определенный период времени и использование повторно без новых запросов к серверу.
- 3) RESTful сервер должен прятать от клиента как можно больше деталей своей реализации. Клиенту не следует знать о том, какая СУБД используется на сервере или сколько серверов в данный момент обрабатывают запросы и прочие подобные вещи.

Ограничения на унифицированный интерфейс являются фундаментальными в дизайне REST-сервисов. Каждый из сервисов функционирует и развивается независимо.

Ограничения для унификации интерфейса:

а) идентификация ресурсов. Индивидуальные ресурсы идентифицированы в запросах, например, с использованием URI в интернет-системах. Ресурсы сами по себе отделены от представлений, которые возвращаются клиентам. Например, сервер может отсылать данные из базы данных в виде HTML, XML или JSON, ни один из которых не является типом хранения внутри хранилища сервера.

б) манипуляция ресурсами через представление. В момент, когда клиенты хранят представление ресурса, включая метаданные, они имеют достаточно данных для модификации или удаления ресурса.

в) «самодостаточные» сообщения. Каждое сообщение достаточно информативно для того, чтобы описать каким образом его обрабатывать. К примеру, какой парсер необходимо применить для извлечения данных из сообщения согласно Internet медиа-типу.

г) гипермедиа, как средство изменения состояния сервера. Клиенты могут изменить состояние системы только через действия, которые динамически идентифицируются на сервере посредством гипермедиа (к примеру, гиперссылки в гипертексте, формы связи, флажки, радиокнопки и прочее). До того момента, пока сервер в явном виде не сообщит обратное, клиенты могут полагаться на то, что любое из предоставленных действий доступно для выполнения на сервере.

3.2 Разработка модели базы данных

Неотъемлемой частью конечного программного средства является база данных, используемая системой в процессе работы. Информационная модель предметной области данных представлена на рисунке 3.2, назначение таблиц описано в таблице 3.1.

База данных — представленная в объективной форме совокупность самостоятельных материалов, систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины. Основными объектами любой базы данных являются таблицы. Таблицы базы данных создаются таким образом, чтобы каждая из них содержала информацию об одном информационном объекте.

Таблица 3.1 – Описание таблиц модели базы данных

Название таблицы	Описание
users	Таблица хранит пользователей зарегистрированных в системе
applications	Список авторизованных приложений, которым открыт доступ в системе
application_user	Таблица связей пользователей и их приложений
tokens_cemetery	Таблица ключей приложений,

	котрым закрыт доступ к системе
contacts	Таблица контактов
deals	Таблица сделок с контактами
tasks	Таблица задач для менеджера(пользователя)
company	Таблица с компаниями
sales_funnels	Таблица воронок продаж
comments	Таблица для хранения комментариев к сделкам(контактам, задачам)

Далее описаны ключевые поля таблиц системы:

Поля таблицы users:

- 1) name - это краткое имя пользователя системы;
- 2) email - электронная почта пользователя, по электронной почте пользователь авторизуется в системе;
- 3) password - зашифрованный пароль пользователя;
- 4) remember_token – токен пользователя;
- 5) created_at – дата регистрации пользователя;
- 6) updated_at – дата внесения изменений в профиль пользователя.

Поля таблицы applications:

- 1) name - имя пользовательского приложения;
- 2) key - ключ пользовательского приложения;
- 3) secret - пароль пользовательского приложения;
- 4) is_active – индикатор активности пользовательского приложения;
- 5) created_at – дата регистрации приложения;
- 6) updated_at – дата внесения изменений в настройки приложения.

Поля таблицы application_user:

- 1) application_id – id приложения в таблице applications ;
- 2) user_id – id пользователя в таблице users;
- 3) authorization_code - код авторизации пользовательского приложения.

Поля таблицы tokens_cemetery:

- 1) token_id – id токенов неактивных приложений.

Поля таблицы contacts:

- 1) name - имя контакта в CRM;
- 2) email - email контакты;
- 3) phone - телефон контакта;
- 4) skype – skype контакта;
- 5) info – дополнительная информация о контакте;
- 6) created_at – дата создания контакта;
- 7) updated_at – дата внесения изменений в профиль контакта.

Поля таблицы deals:

- 1) title – название сделки в CRM;

2) contact_id – id контакта в таблице contacts , который связан со сделкой;

3) sales_funnel_id – id воронки продаж из таблицы sales_funnels к которой принадлежит сделка ;

4) amount – сумма сделки;

5) company_id – id компании, из таблицы company;

6) created_at – дата создания сделки;

7) updated_at – дата внесения изменений в сделку.

Поля таблицы tasks:

1) title – название задачи в CRM;

2) description – описание задачи;

3) element_type – указывает к какому типу элемента принадлежит задача(сделка, контакт);

4) element_id – id элемента(сделка, контакт) из таблицы deals либо contacts;

5) complete_till – дата до которой нужно выполнить задачу;

6) created_at – дата создания задачи;

7) updated_at – дата внесения изменений в задачу.

Поля таблицы company:

1) title - имя компании в CRM;

2) created_at – дата создания компании;

3) updated_at – дата внесения изменений в компанию.

Поля таблицы comments:

1) text – текст комментария в CRM;

2) element_type – указывает к какому типу элемента принадлежит комментарий(сделка, контакт);

3) element_id – id элемента(сделка, контакт) из таблицы deals либо contacts;

4) created_at – дата создания комментария;

5) updated_at – дата внесения изменений в комментарий.

Поля таблицы sales_funnels:

1) title – название воронки продаж комментария в CRM;

2) created_at – дата создания комментария;

3) updated_at – дата внесения изменений в комментарий.

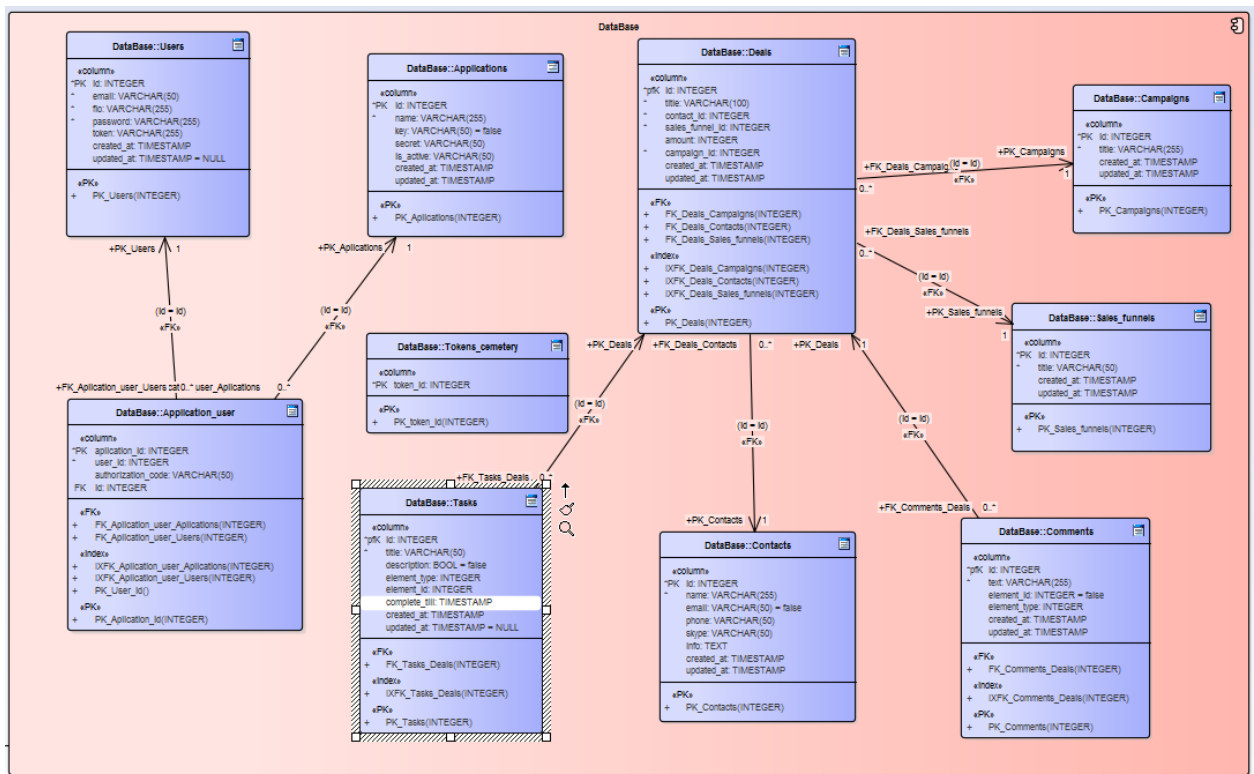


Рисунок 3.2 – Модель базы данных.

3.3 Разработка алгоритма аутентификации клиентских приложений

Разрабатываемое приложение должно разрешать сторонним приложениям возможность получать и записывать данные в БД, мы должны уметь ограничивать доступ только к приложениям которым открыт доступ, также мы должны иметь возможность включать и отключать доступ любому приложению в любое время.

Ниже представлен список функций для реализации этих задач:

- 1) выдать приложению аутентификационный токен;
- 2) деактивировать приложение, и не обрабатывать более его запросы;
- 3) разрешить пользователям авторизовываться через сторонние приложения;
- 4) разрешить пользователям прекращать работу через сторонние приложения.

Чтобы иметь возможность идентифицировать и аутентифицировать приложения, которые пытаются обращаться к API, каждому приложению нужно присвоить ключ и секретное слово, с их помощью клиентское приложение создает хэш, с помощью которого обращается к серверному приложению за аутентификацией. Для этих целей хорошо подойдёт веб токен (JWT), считаю, что это идеальный подход, отличная комбинация безопасности и простоты.

Если время жизни аутентификационного токена истекло, ответ будет `token_expired` с кодом 401, приложение, которое обращается по API должно предусматривать такие ситуации и произвести получение нового аутентификационного токена в таком случае.

Если в какой-то момент времени нужно деактивировать конкретное приложение, с целью закрыть ему доступ к API, нужно сменить значение поля `is_active` на 0, данная функциональность реализована в пользовательском интерфейсе на стороне сервера. Если запрос делает неактивное приложение - оно получит ответ `app_inactive` с кодом ответа 403.

3.4 Разработка алгоритма аутентификации пользователей

Для того, чтобы дать сторонним приложениям делать запросы, связанные с пользователями и их данными, нужно разрешить пользователям входить в систему через приложения и разрешать приложению делать определенные действия от имени пользователей.

Ниже представлен алгоритм реализации данного функционала:

- 1) пользователь запускает стороннее приложение;
- 2) кликает "Войти";
- 3) идет переход на определенный url серверного приложения ;
- 4) пользователь входит с логином и паролем;
- 5) происходит редирект на специальный URI с кодом;
- 6) приложение использует этот код для запроса токена аутентификации пользователя;
- 7) приложение использует полученный токен для будущих запросов.

В методе `HomeController@showAuthorizationForm` идет проверка параметров авторизации и отображение, при необходимости формы для логина:

```
public function authorizeApp(Request $request)
{
    $validator = Validator::make($request->all(), [
        'app_key' => 'required|exists:applications,key,is_active,1',
        'redirect_uri' => 'required:active_url',
    ]);
    if (!$validator->passes()) {
        return view('authorize-app')->withInvalid('true');
    }
    $app = Application::whereKey($request->app_key)->first();
    return view('authorize-app', compact('app'));
}
```

Если параметры запроса не валидны, будет показан шаблон с переменной `$invalid`, если проверка пройдена успешно будет показан шаблон с

состоянием приложения, которое пытается получить доступ к аутентификационному токenu. При условии что пользователь ввел верные данные, произойдет переход по заданному url с кодом `redirect_uri?code=6bc4`, приложение должно обработать этот момент и забрать код из url-адреса, чтобы выписать аутентификационный токен для доступа к личным данным.

Чтобы сделать запрос, который требует чтобы пользователь был аутентифицирован, необходимо послать пользовательский аутентификационный токен в заголовке `Authorization`:

```
Authorization: Bearer  
eyJ0eXAiOi~~~.eyJpc3MiOi~~~.MSZBIGimZWrc9DIZZduh~~~
```

Теперь пользователь может использовать стороннее приложение для доступа к личным данным. Для того чтобы пользователь имел возможность выйти из системы, создадим таблицу `tokens_cemetery` в которой будут храниться уничтоженные пользовательские аутентификационные токены:

```
Schema::create('tokens_cemetery', function (Blueprint $table) {  
    $table->string('token_id');  
});
```

Реализацией функционала выхода из приложения служит метод `logoutUser()`:

```
public function logoutUser(Request $request)  
{  
    $token = $request->bearerToken();  
    DB::table('tokens_cemetery')->insert(['token_id' => $token]);  
    return response('token_deceased');  
}
```

Далее нужно обновить посредника `ApiUserAuth` добавив в него проверку уничтоженных аутентификационных токенов:

```
if (DB::table('tokens_cemetery')->whereTokenId($payload['jti'])->first()) {  
    return response('token_deceased', 403);  
}  
$request->merge(['__authTokenId' => $payload['jti']]);
```

Теперь, если пользователь выйдет из приложения, запросы по API с уничтоженным аутентификационным токеном получают код 403 и ответ `token_deceased`, приложение должно попросить пользователя заново войти в систему и разрешить далее использовать свои личные данные и осуществлять действия от имени пользователя.

3.5 Разработка функциональности для работы с базой данных CRM

Основная задача разрабатываемого приложения – это обмен информацией с клиентскими приложениями. Обмен информацией будет происходить по технологии REST.

Ниже приведён пример запросов REST API:

- GET /api/v1/contacts/list;
- POST /api/v1/contacts/add;

Клиентское приложение будет получать ответ от сервера в формате JSON.

Ниже приведён пример ответа от сервера в формате JSON:

```
{
  "status":
  {
    "code":"ok",
    "message":null
  },
  {
    "contact_id": 274
  }
}
```

Проектируемое приложение состоит из трех основных сущностей – это контакты, сделки и задачи. Каждая сущность имеет свой набор ресурсных URL и соответствующие им передаваемые параметры. Пример параметров для ресурсного URL представлен в таблице 3.2.

Таблица 3.2 – Пример параметров для ресурса /api/v1/contacts/add

Название таблицы	Тип	Описание
name	строка	имя контакта
email	строка	email контакта
phone	строка	телефонный номер контакта
skype	строка	skype контакта
info	текст	дополнительная информация относящаяся к контакту
company_id	число	id компании к которой будет добавлен контакт

4 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

Завершающим этапом разработки программного средства является тестирование, которое представляет собой процесс исследования приложения с целью получения информации о качестве программного продукта.

Тестирование функциональности приложения осуществляется с помощью библиотеки PHPUnit, которая входит в состав фреймворка Laravel. При тестировании, запросы к серверу осуществляются с имитацией авторизованного приложения.

Ниже приведены таблицы с тестами модулей программного средства:

Таблица 4.1 – Пример параметров для ресурса /api/v1/contacts/add

Условие	Предполагаемый результат	Реальный результат	Результат теста
Отправка POST запроса по адресу <code>http://example.com/api/v1/contacts/add</code> вместе с передаваемыми параметрами о контакте	Получение массива в формате JSON, с данными об успешном выполнении запроса	Получение массива в формате JSON, с данными об успешном выполнении запроса	Пройдено
Отправка POST запроса по адресу <code>http://example.com/api/v1/deals/add</code> вместе с передаваемыми параметрами о сделке	Получение массива в формате JSON, с данными об успешном выполнении запроса	Получение массива в формате JSON, с данными об успешном выполнении запроса	Пройдено
Отправка POST запроса по адресу <code>http://example.com/api/v1/companies/add</code> вместе с передаваемыми параметрами о компании	Получение массива в формате JSON, с данными об успешном выполнении запроса	Получение массива в формате JSON, с данными об успешном выполнении запроса	Пройдено

Продолжение таблицы 4.1

Условие	Предполагаемый результат	Реальный результат	Результат теста
Отправка POST запроса по адресу http://example.com/api/v1/comments/add вместе с передаваемыми параметрами о комментарии	Получение массива в формате JSON, с данными об успешном выполнении запроса	Получение массива в формате JSON, с данными об успешном выполнении запроса	Пройдено

5 МЕТОДИКА РАБОТЫ С ПРОГРАММНЫМ СРЕДСТВОМ

5.1 Размещение проекта на сервере

Для размещения сервиса в интернете необходимо виртуальный либо выделенный сервер с поддержкой PHP версии 5.6 и выше, и сервером баз данных MySQL версии 5.6 и выше. Файлы из папки easycrm нужно скопировать в папку /www/example.com, где example.com имя вашего домена прикреплённого к серверу. Так же нужно в СУБД MySQL создать базу данных с именем crm. После проделанных операций, по средствам ssh нужно переместиться в папку с проектом на сервере и выполнить команду `php artisan migrate`, для применения миграций к базе данных на сервере.

5.2 Взаимодействие с клиентским приложением

Для доступа к данным сервиса по средствам API, необходима авторизация через клиентское приложение. Все методы могут быть использованы только после авторизации.

Ниже представлен список функций сервиса и требуемые параметры:

Добавление нового контакта:

1) функция - POST `http://example.com/api/v1/contacts/add`

2) передаваемые параметры:

- name - имя контакта в CRM;
- email - email контакты;
- phone - телефон контакта;
- skype – skype контакта;
- info – дополнительная информация о контакте.

Получение списка контактов:

1) функция: GET `http://example.com/api/v1/contacts/list`

Получение подробной информации о контакте:

1) функция: GET `http://example.com/api/v1/contacts/{id}`, где id – уникальный номер контакта в базе данных.

Обновление информации о контакте:

1) функция: POST `http://example.com/api/v1/contacts/update`

2) передаваемые параметры:

- id – уникальный номер контакта в базе данных;
- name - имя контакта в CRM(не обязательно);
- email - email контакты(не обязательно);
- phone - телефон контакта(не обязательно);
- skype – skype контакта(не обязательно);
- info – дополнительная информация о контакте(не обязательно).

Удаление контакта:

1) функция: GET `http://example.com/api/v1/contacts/delete/{id}`, где id – уникальный номер контакта в базе данных.

Добавление новой сделки:

1) функция: POST <http://example.com/api/v1/deals/add>

2) передаваемые параметры:

- title – название сделки в CRM;
- contact_id – id контакта в таблице contacts , который связан со сделкой;
- sales_funnel_id – id воронки продаж из таблицы sales_funnels к которой принадлежит сделка;
- amount – сумма сделки(не обязательно);
- company_id – id компании, из таблицы company.

Получение списка сделок:

1) функция: GET <http://example.com/api/v1/deals/list>

Получение подробной информации о сделке:

1) функция: GET <http://example.com/api/v1/deals/{id}>, где id – уникальный номер сделки в базе данных.

Обновление информации о сделке:

1) функция: POST <http://example.com/api/v1/deals/update>

2) передаваемые параметры:

- id – уникальный номер контакта в базе данных;
- name - имя контакта в CRM(не обязательно);
- email - email контакты(не обязательно);
- phone - телефон контакта(не обязательно);
- skype – skype контакта(не обязательно);
- info – дополнительная информация о контакте(не обязательно).

Удаление сделки:

1) функция: GET <http://example.com/api/v1/deals/delete/{id}>, где id – уникальный номер сделки в базе данных.

Добавление новой задачи:

1) функция: POST <http://example.com/api/v1/tasks/add>

2) передаваемые параметры:

- title – название задачи в CRM;
- description – описание задачи(не обязательно);
- element_type – указывает к какому типу элемента принадлежит задача (сделка, контакт);
- element_id – id элемента(сделка, контакт) из таблицы deals либо contacts;
- complete_till – дата до которой нужно выполнить задачу.

Получение списка задач:

1) функция: GET <http://example.com/api/v1/tasks/list>

Получение подробной информации о задаче:

1) функция: GET <http://example.com/api/v1/tasks/{id}>, где id – уникальный номер задачи в базе данных.

Обновление информации о задаче:

1) функция: POST [http://example.com/api/v1/tasks /update](http://example.com/api/v1/tasks/update)

2) передаваемые параметры:

- id – уникальный номер задачи в базе данных;
- title – название задачи(не обязательно);
- description – описание задачи;
- element_type – указывает к какому типу элемента принадлежит задача(сделка, контакт);
- element_id – id элемента(сделка, контакт) из таблицы deals либо contacts;
- complete_till – дата до которой нужно выполнить задачу.

Удаление задачи:

1) функция: GET [http://example.com/api/v1/tasks /delete/{id}](http://example.com/api/v1/tasks/delete/{id}), где id – уникальный номер задачи в базе данных.

Добавление новой компании:

1) функция: POST <http://example.com/api/v1/companies/add>

2) передаваемые параметры:

- title - имя компании в CRM.

Получение списка компаний:

1) функция: GET [http://example.com/api/v1/companies /list](http://example.com/api/v1/companies/list)

Обновление информации о компании:

1) функция: POST <http://example.com/api/v1/companies/update>

2) передаваемые параметры:

- id – уникальный номер компании в базе данных;
- title – название компании.

Удаление компании:

1) функция: GET <http://example.com/api/v1/companies/delete/{id}>, где id – уникальный номер компании в базе данных.

Добавление нового комментария:

1) функция: POST <http://example.com/api/v1/comments/add>

2) передаваемые параметры:

- text – текст комментария в CRM;
- element_type – указывает к какому типу элемента принадлежит комментарий(сделка, контакт);
- element_id – id элемента(сделка, контакт) из таблицы deals либо contacts.

Обновление комментария:

1) функция: POST [http://example.com/api/v1/ comments /update](http://example.com/api/v1/comments/update)

2) передаваемые параметры:

- id – уникальный номер комментария в базе данных;
- text – текст комментария(не обязательно).

Удаление комментария:

1) функция: GET <http://example.com/api/v1/comments/delete/{id}>, где id – уникальный номер комментария в базе данных.

Добавление новой воронки продаж:

1) функция: POST <http://example.com/api/v1/funnels/add>

2) передаваемые параметры:

- title – название воронки продаж в CRM.

Получение списка воронок продаж:

1) функция: GET <http://example.com/api/v1/funnels/list>

Обновление информации о воронке продаж:

1) функция: POST <http://example.com/api/v1/funnels/update>

2) передаваемые параметры:

- id – уникальный номер воронки продаж в базе данных;

- title – название воронки продаж.

Удаление компании:

1) функция: GET <http://example.com/api/v1/funnels/delete/{id}>, где id – уникальный номер воронки продаж в базе данных.

6 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ

6.1 Характеристика программного продукта

Целью дипломного проекта является разработка серверной части программного средства для автоматизации работы с клиентами.

Разработка серверной части программного средства для работы с клиентами позволит создать программное средство, которое будет предоставлять доступ к информации о клиентах компании. Программное средство предоставит простое добавление и удобный доступ к информации о клиентах.

Целью технико-экономического обоснования ПС является определение экономической выгоды создания данного продукта и его дальнейшего применения.

Использование программного средства для автоматизации работы с клиентами обеспечит быстрый доступ к нужной информации, тем самым увеличит производительность труда менеджеров.

Для расчёта экономических характеристик программного средства, необходимо определить смету затрат, цену и прибыль.

6.2 Расчет сметы затрат и цены программного средства

Таблица 6.1 – Исходные данные

Наименование показателей	Буквенные обозначения	Единицы измерения	Количество
Коэффициент новизны	Кн	единиц	0,7
Группа сложности		единиц	2
Дополнительный коэффициент сложности	Кс	единиц	0,39
Поправочный коэффициент, учитывающий использование типовых программ	Кт	единиц	0,7
Установленная плановая продолжительность разработки	Тр	лет	0,33
Продолжительность рабочего дня	Тч	ч	8
Тарифная ставка 1-ого разряда	Тм1	руб.	600 000
Коэффициент премирования	Кп	Единиц	1,4

Продолжение таблицы 6.1

Норматив дополнительной заработной платы исполнителей	Нд	%	20
Отчисление в фонд социальной защиты населения	Нсз	%	34
Отчисления в Белгосстах		%	0,6
Расходы на научные командировки		%	30
Прочие прямые расходы		%	20

Объем ПС определяется путем подбора аналогов на основании классификации типов ПС, каталога функций ПС и каталога аналогов ПС в разрезе функций, которые постоянно обновляются и утверждаются в установленном порядке.

Таблица 6.2 – Характеристика функций и их объем

Номер функции	Наименование (содержание) функций	Объем функций
101	Организация ввода информации	140
102	Контроль, предварительная обработка и ввод информации	460
109	Организация ввода / вывода информации в интерактивном режиме	370
111	Управление вводом / выводом	1670
203	Формирование баз данных	2540
204	Обработка наборов записей базы данных	2560
207	Манипулирование данными	7600
208	Организация поиска и поиск в базе данных	5920
506	Обработка ошибочных и сбойных ситуаций	1540
703	Расчет показателей	520
Итого		23320

Объем ПС определяется на основе нормативных данных, приведенных в таблице 6.2.

Общий объем программного продукта определяется исходя из количества и объема функций, реализованных в программе:

$$V_o = \sum_{i=0}^n V_i, \quad (6.1)$$

где

V_i – объем отдельной функции ПС;

V_o – общий объем ПС;

n – общее число функций.

$$V_o = 23320(\text{строк кода})$$

На основании общего объема ПС определяется нормативная трудоемкость (T_n) с учетом сложности ПС. Для ПС 2-ой группы, к которой относится разрабатываемый программный продукт, нормативная трудоемкость составит 646 человеко-дней.

Нормативная трудоемкость служит основой для оценки общей трудоемкости T_o . Общая трудоемкость определяется по формуле:

$$T_o = T_n * K_c * K_t * K_n, \quad (6.2)$$

где

K_c – коэффициент, учитывающий сложность ПС;

K_t – поправочный коэффициент, учитывающий степень использования при разработки стандартных модулей;

K_n – коэффициент, учитывающий степень новизны ПС.

Дополнительные затраты труда на разработку ПС учитываются через коэффициент сложности, который вычисляется по формуле:

$$K_c = 1 + \sum_{i=1}^n K_i, \quad (6.3)$$

где

K_i – коэффициент, соответствующий степени повышения сложности ПО за счет конкретной характеристики;

n – количество учитываемых характеристик.

$$K_c = 1 + 0,06 + 0,07 + 0,26 = 1,39$$

Поправочный коэффициент, учитывающий степень использования при разработки стандартных модулей:

$$K_t = 0,7$$

Поправочный коэффициент, учитывающий степень использования при разработки стандартных модулей:

$$K_n = 0,7$$

С учетом дополнительного коэффициента сложности $K_{сл}$ рассчитывается общая трудоемкость ПС:

$$T_o = 646 * 1,39 * 0,7 * 0,7 \approx 440 \text{ чел./дн}$$

На основе общей трудоемкости и требуемых сроков реализации проекта вычисляется плановое количество исполнителей. Численной исполнителей проекта рассчитывается по формуле:

$$Ч_p = \frac{T_o}{T_p * \Phi_{эф}} \quad (6.4)$$

где

T_o – общая трудоемкость разработки проекта, чел/дн.;

$\Phi_{эф}$ – эффективный фонд времени работы одного работника в течение года, дн.;

T_p – срок разработки проекта, лет.

Эффективный фонд времени работы одного разработчика вычисляется по формуле:

$$\Phi_{эф} = D_r - D_{п} - D_{в} - D_o, \quad (6.5)$$

где

D_r – количество дней в году, дн.;

$D_{п}$ – количество праздничных дней в году, не совпадающих с выходными днями, дн.;

$D_{в}$ – количество выходных дней в году, дн.;

D_o – количество дней отпуска, дн.

$$\Phi_{эф} = 366 - 9 - 102 - 24 = 231 \text{ (дня в год)}$$

Учитывая срок разработки проекта $T_p = 4 \text{ мес.} = 0,33 \text{ года}$, общую трудоемкость и фонд эффективного времени одного работника, вычисленные ранее, можно рассчитать численность исполнителей проекта:

$$Ч_p = \frac{440}{0,33 * 231} \approx 6 \text{ человека}$$

6.3 Расчет заработной платы исполнителей

Расчет основной заработной платы осуществляется в следующей последовательности.

Определим месячные (T_m) и часовые ($T_{\text{ч}}$) тарифные ставки ведущего инженера-программиста (тарифный разряд – 15; тарифный коэффициент – 3,48), двух инженеров-программистов 1-й категории (тарифный разряд – 14; тарифный коэффициент – 3,25) и двух инженеров-программистов 2-ой категории (тарифный разряд – 13; тарифный коэффициент – 3,04).

Месячная тарифная ставка каждого исполнителя (T_m) определяется путем умножения действующей месячной тарифной ставки 1-ого разряда (T_{m1}) на тарифный коэффициент (T_k), соответствующий установленному тарифному разряду:

$$T_m = T_{m1} * T_k, \quad (6.6)$$

Часовая тарифная ставка рассчитывается путем деления месячной тарифной ставки на установленный при сорокачасовой рабочей недели и восьмичасовом рабочем дне фонд рабочего времени – 170 часов.

$$T_{\text{ч}} = \frac{T_m}{170}, \quad (6.7)$$

где

$T_{\text{ч}}$ – часовая тарифная ставка (тыс. руб.);

T_m – месячная тарифная ставка (тыс. руб.);

Определим месячную и часовую тарифные ставки ведущего инженера-программиста:

$$T_m = 600\,000 * 3,48 = 2\,088\,000 \text{ руб.}$$

$$T_{\text{ч}} = 2\,088\,000 / 170 = 12\,282 \text{ руб.}$$

Определим месячную и часовую тарифные ставки инженера-программиста 1-й категории:

$$T_m = 600\,000 * 3,25 = 1\,950\,000 \text{ руб.}$$

$$T_{\text{ч}} = 1\,950\,000 / 170 = 11\,470 \text{ руб.}$$

Определим месячную и часовую тарифные ставки инженера-программиста 2-й категории:

$$T_m = 600\,000 * 3,04 = 1\,824\,000 \text{ руб.}$$

$$T_{\text{ч}} = 1\,824\,000 / 170 = 10\,729 \text{ руб.}$$

Расчет месячных и почасовых тарифных ставок сведен в таблицу 6.3.

Таблица 6.3 – Исполнители и трудоемкость проекта

Должность	Чел/дн занято- сти	Тариф- ный раз- ряд	Тарифный коэффици- ент	Месячная тарифная ставка (руб.)	Часо- вая та- рифная ставка (руб.)
Ведущий инженер-программист	76	15	3,48	2 088 000	12 282
Инженер-программист 1-й категории	76	14	3,25	1 950 000	11 470
Инженер-программист 1-й категории	76	14	3,25	1 950 000	11 470
Инженер-программист 2-й категории	76	13	3,04	1 824 000	10 729
Инженер-программист 2-й категории	76	13	3,04	1 824 000	10 729

Основная заработная плата исполнителей рассчитывается по формуле:

$$З_0 = \sum_{i=1}^n T_{\text{ч}}^i * T_{\text{ч}} * \Phi_{\text{п}} * K_{\text{п}}, \quad (6.8)$$

где

$T_{\text{ч}}^i$ – часовая тарифная ставка i -ого исполнителя, руб/час;

$T_{\text{ч}}$ – количество часов работы в день, час;

$\Phi_{\text{п}}$ – плановый фонд рабочего времени i -ого исполнителя, дн.;

$K_{\text{п}}$ – коэффициент премирования.

Следовательно, сумма основной заработной платы составляет:

$$З_0 = 12\,282 * 8 * 76 * 1,4 + 11\,470 * 8 * 76 * 1,4 * 2 + 10\,729 * 8 * 76 * 1,4 * 2 = 48\,246\,016 \text{ руб.}$$

Дополнительная заработная плата ($З_{\text{д}}$) определяется по нормативу в процентах к основной заработной плате по формуле:

$$З_д = \frac{З_о * Н_о}{100\%}, \quad (6.9)$$

где

$Н_д$ – норматив дополнительной заработной платы в целом по организации, равный 20%.

Дополнительная заработная плата ($З_д$) составляет:

$$З_д = \frac{48\,246\,016 * 20\%}{100\%} \approx 9\,649\,203 \text{ руб.}$$

Таблица 6.4 – Расчет себестоимости и отпускной цены ПС

Наименование статей	Норматив	Методика расчета	Значение, руб.
Отчисление в фонд социальной защиты	$Н_{сз} = 34\%$	$З_{сз} = \frac{(З_о + З_д) * Н_{сз}}{100\%}$	19 684 374
Отчисления в Белгосстрах	$Н_{не} = 0,6\%$	$Н_e = \frac{(З_о + З_д) * Н_{не}}{100\%}$	347 371
Материалы и комплектующие	—	$M = \frac{H_m * V_o}{100}$ $H_m = 380 \text{ руб./100 строк}$	88 620
Машинное время	—	$P_m = \frac{Ц_m * V_o}{100 * H_{мв}}$ $Ц_m = 6\,000 \text{ руб}$ $H_{мв} = 12 \text{ машино-часов}$	116 600
Расходы на научные командировки	$Н_к = 30\%$	$P_k = \frac{З_о * Н_к}{100\%}$	14 473 805
Прочие прямые расходы	$Н_{пз} = 20\%$	$П_з = \frac{З_о * Н_{пз}}{100\%}$	9 649 203
Накладные расходы	$Н_{рн} = 100\%$	$P_n = \frac{З_о * Н_{рн}}{100\%}$	48 246 016

Продолжение таблицы 6.4

Полная себестоимость	—	$C_{\pi} = Z_o + Z_d + Z_{cz} + H_e + M + P_M + P_{HK} + P_3 + P_H$	150 497 208
Прогнозируемая прибыль	$Y_p = 30\%$	$P_{\pi c} = \frac{C_{\pi} * Y_p}{100\%}$	45 149 162
Прогнозируемая цена без налогов	—	$\Pi_{\pi} = C_{\pi} + P_{\pi c}$	195 646 370
Налог на добавленную стоимость	$H_{dc} = 20\%$	$\begin{aligned} & \text{НДС} \\ & = \frac{(\Pi_{\pi} + O_{mp}) * H_{dc}}{100\%} \end{aligned}$	40 717 246
Прогнозируемая отпускная цена	—	$\Pi_o = \Pi_{\pi} + O_{mp} + \text{НДС}$	244 303 480
Освоение ПС	$H_o = 10\%$	$P_o = \frac{C_{\pi} * H_o}{100\%}$	15 049 720
Сопровождение ПС	$H_c = 20\%$	$P_c = \frac{C_{\pi} * H_c}{100\%}$	30 099 441

6.4 Оценка экономической эффективности применения программного средства у пользователя

Для расчета экономического эффекта применения нового ПС необходимы данные имеющегося внедренного на производстве аналога (базового варианта). Некоторые показатели базового варианта не могут быть получены (составляют коммерческую тайну либо защищены авторскими правами разработчиков). Поэтому расчет экономического эффекта будем проводить, опираясь на известные данные показателей базового и нового варианта ПС. Показатели обоих вариантов приведены в таблице 6.5.

Таблица 6.5 – Исходные данные для расчета экономического эффекта

Наименование показателей	Обозначение	Единицы измерения	Значение показателей	
			Базовый вариант	Новый вариант
1	2	3	4	5

Продолжение таблицы 6.5

Капитальные вложения, включая стоимость услуг по сопровождению и адаптации ПС	$K_{\text{пр}}$	руб.	–	244 424 820
Затраты на освоение ПС	$K_{\text{ос}}$	руб.	–	15 049 720
Затраты на сопровождение ПС	$K_{\text{с}}$	руб.	–	30 099 441
Всего затрат:				289 573 981
Время простоя сервиса, обусловленное ПО, в день	Π_1, Π_2	мин	20	14
Стоимость одного часа простоя	$C_{\text{п}}$	руб.	80 000	80 000
Среднемесячная зарплата одного программиста	$З_{\text{см}}$	руб.	10 000 000	10 000 000
Коэффициент начислений на зарплату	$K_{\text{н}}$	–	1,5	1,5
Среднемесячное количество рабочих дней	$D_{\text{р}}$	день	22	22
Количество типовых задач, решаемых за год	$З_{\text{т1}}, З_{\text{т2}}$	задача	4 500	4 500
Объем выполняемых работ	A_1, A_2	задача	4 500	4 500
Средняя трудоемкость работ в расчете на задачу	$T_{\text{с1}}, T_{\text{с2}}$	Человеко-часов на задачу	1	0,3
Количество часов работы в день	$T_{\text{ч}}$	час	8	8
Ставка налога на прибыль	$H_{\text{п}}$	%	24	24

6.5 Расчет капитальных затрат

Общие капитальные вложения (K_0) заказчика (потребителя), связанные с приобретением, внедрением и использованием ПС, включают в себя затраты на приобретение, освоение ПС.

Таблица 6.6 – Расчет капитальных затрат.

Наименование	Методика расчета	Значение (руб.)
Затраты пользователя на приобретение ПС по отпускной цене разработчика	$K_{пр}$	244 303 480
Затраты на освоение ПС	$K_{ос}$	15 049 720
Затраты на сопровождение ПС	K_c	30 099 441
Общие капитальные вложения	$K_0 = K_{пр} + K_{ос} + K_c$	289 573 981
Экономия затрат на заработную плату на 1 задачу	$C_{зе} = \frac{3_{см} * (T_{с1} - T_{с2})}{T_ч * D_p}$	36 460
Экономия на заработную плату при использовании нового ПС	$C_3 = C_{зе} * A_2$	164 070 000
Экономия с учетом начисления на заработную плату	$C_n = C_3 * K_{нз}$	246 105 000
Экономия за счет сокращения простоя сервиса	$C_c = \frac{(П_1 - П_2) * C_{п} * D_{рг}}{60}$	176 000
Общая годовая экономия текущих затрат, связанных с использованием нового ПС	$C_0 = C_n + C_c$	268 466 085

Внедрение нового ПС позволит пользователю сэкономить на текущих затратах 268 466 085 руб.

6.6 Расчет экономического эффекта

Для пользователя в качестве экономического эффекта выступает лишь чистая прибыль – дополнительная прибыль, остающаяся в его распоряжении ($\Delta\P_{\text{ч}}$), которая определяется по формуле:

$$\Delta\P_{\text{ч}} = C_o - C_o * \frac{H_{\text{п}}}{100}, \quad (6.10)$$

где

$H_{\text{п}}$ – ставка налога на прибыль, равная 24%.

Тогда чистая прибыль будет равна:

$$\Delta\P_{\text{ч}} = 268\,466\,085 - 268\,466\,085 * 0,24 = 204\,034\,224 \text{ (руб.)}$$

В процессе использования нового ПС чистая прибыль в конечном итоге возмещает капитальные затраты. Однако полученные при этом суммы результатов (прибыль) и затрат (капитальных вложений) по годам приводят к единому времени – расчетному году (за расчетный принят 2016 год) путем умножения результатов и затрат за каждый год на коэффициент приведения (α_1), который рассчитывается по формуле:

$$\alpha_1 = (1 + E_i)^{t_p - t}, \quad (6.11)$$

где

E – норматив дисконтирования разновременных затрат и результатов, 0,4;

t_p – расчетный период, $t_p = 1$;

t – период, потоки которого приводятся к расчетному

Коэффициентам приведения (α_1) по годам будут соответствовать следующие значения:

$$\alpha_1 = (1 + 0,15)^{1-1} = 1 - \text{расчетный год};$$

$$\alpha_2 = (1 + 0,15)^{1-2} = 0.8696 - 2017 \text{ год};$$

$$\alpha_3 = (1 + 0,15)^{1-3} = 0.7561 - 2018 \text{ год};$$

$$\alpha_4 = (1 + 0,15)^{1-4} = 0.6575 - 2019 \text{ год};$$

В таблице 6.7 видно, что затраты разрабатываемое программное средства окупаются за два года.

Таблица 6.7 – Расчет экономического эффекта от использования нового программного средства

Показатели	Ед · из м.	2016	2017	2018	2019
Результаты:					
Прирост прибыли за счет экономии затрат (Пч)	ру б.	–	204 034 224	204 034 224	204 034 224
Прирост прибыли с учетом фактора времени	ру б.	–	177 428 161	154 270 277	134 152 502
Затраты:					
Приобретение ПС (Кпр)	ру б.	244 303 480	–	–	–
Освоение ПС (Кос)	ру б.	15 049 720	–	–	–
Сопровождение ПС	ру б.	30 099 441	–	–	–
Всего затрат	ру б.	289 573 981	–	–	–
То же с учетом фактора времени	ру б.	289 573 981	–	–	–
Экономический эффект:					
Превышение результата над затратами	ру б.	-289 573 981	177 428 161	154 270 277	134 152 502
То же с нарастающим итогом	ру б.	-289 573 981	- 112 145 820	42 124 457	176 276 959
Коэффициент приведения	ед ин иц	1	0,8696	0,7561	0,6575

6.7 Выводы по технико-экономическому обоснованию

Разработка и внедрение серверной части программного средства для автоматизации работы с клиентами является экономически выгодным.

Чистая прибыль от реализации ПС ($P_{\text{ч}} = 204\,034\,224$ рублей) остается организации-разработчику и представляет собой экономический эффект от создания нового программного средства. Положительный экономиче-

ский эффект главным образом достигается за счет уменьшения трудоемкости работ пользователей в расчете на одну задачу.

Продукт является экономически выгодным, так как он окупается за два года эксплуатации, что означает экономическую целесообразность данной разработки.

ЗАКЛЮЧЕНИЕ

Результатом дипломного проектирования является функционально законченное приложение, которое позволяет совершать операции с базой данных CRM системы, и обмениваться этой информацией с клиентскими приложениями. Клиентские приложения могут размещаться на различных платформах, единственное условие, это поддержка интерфейса взаимодействия с сервером. Данное приложение может найти своё применение в различного рода малом и среднем бизнесе.

Можно выделить следующие достоинства разработанного программного средства:

- простой и удобный интерфейс взаимодействия;
- малые затраты на изготовление и поддержку благодаря правильному подбору технологий реализации;
- быстрота работы и малые затраты системных ресурсов при работе приложения благодаря настройке кэширования и оптимизации базы данных;
- простота установки и настройки;

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] Занстра, М. PHP: объекты, шаблоны и методики программирования / М. Занстра, 2014.
- [2] Блэк, Р. Ключевые процессы тестирования / Р. Блэк, 2014. - 341 с.
- [3] Маклафлин, Б. PHP и MySQL. Исчерпывающее руководство / Б. Маклафлин, 2014. - 544 с.
- [4] Gilmore, W. Easy Laravel 5 / W. J. Gilmore, 2015. - 263 с.
- [5] Laravel Framework [Электронный ресурс]. – Режим доступа: <https://laravel.com/>.
- [6] PHPUnit tests [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://phpunit.de/>.
- [7] Bootstrap (фреймворк) [Электронный ресурс]. – Электронные данные. – Режим доступа: [https://ru.wikipedia.org/wiki/Bootstrap_\(фреймворк\)](https://ru.wikipedia.org/wiki/Bootstrap_(фреймворк)).
- [8] MySql 5.6 [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://habrahabr.ru/post/168441/>.

ПРИЛОЖЕНИЕ А

(обязательное)

Фрагмент кода программы

```
<?php

namespace App\Http\Controllers\Api;

use Illuminate\Http\Request;

use App\Http\Requests;
use App\Http\Controllers\Controller;
use Illuminate\Support\Str;
use App\Models\Application;
use \Firebase\JWT\JWT;

class AuthController extends Controller
{
    public function authenticateApp(Request $request)
    {
        $credentials = base64_decode(
            Str::substr($request->header('Authorization'), 6)
        );

        try {
            list($appKey, $appSecret) = explode(':', $credentials);

            $app = Application::whereKeyAndSecret($appKey, $appSecret)-
>firstOrFail();
        } catch (\Throwable $e) {
            return response('invalid_credentials', 400);
        }

        if (! $app->is_active) {
            return response('app_inactive', 403);
        }

        return response([
            'token_type' => 'Bearer',
            'access_token' => $app->generateAuthToken(),
        ]);
    }
}
```

```

public function authenticateUser(Request $request)
{
    $code = $request->json('code');

    $app = $request->__authenticatedApp;

    if (! $code || ! $user = $app->users()->wherePivot('Authorization_code',
$code)->first()) {
        return response('invalid_code', 400);
    }

    $app->users()->updateExistingPivot($user->id, ['Authorization_code' =>
null]);

    return response([
        'token_type' => 'Bearer',
        'access_token' => $user->generateAuthToken($app),
    ]);
}

public function logoutUser(Request $request)
{
    $token = $request->bearerToken();
    DB::table('tokens_cemetery')->insert(['token_id' => $token]);
    return response('token_deceased');
}
}

```

<?php

```
namespace App\Http\Middleware;
```

```
use Closure;
```

```
use \Firebase\JWT\JWT;
```

```
class ApiAppAuth
```

```

{
    /**
     * Handle an incoming request.
     *
     * @param \Illuminate\Http\Request $request
     * @param \Closure $next
     * @return mixed
     */
}

```

```

*/
public function handle($request, Closure $next)
{
    $authToken = $request->bearerToken();

    if (! $authToken) {
        return response('app_unauthorize', 403);
    }

    try {
        // проверка валидности токена
        $this->payloadIsValid(
            // JWT::decode принимает строку с токеном первым аргументом
            // затем ключ, которым закодирован токен
            // и список алгоритмов
            $payload = (array) JWT::decode($authToken,
                'w5yuCV2mQDVTGmn3', ['HS256'])
            );

        $app = Application::whereKey($payload['sub']->firstOrFail());
    } catch (\Firebase\JWT\ExpiredException $e) {
        return response('token_expired', 401);
    } catch (\Throwable $e) {
        return response('token_invalid', 401);
    }

    if (! $app->is_active) {
        return response('app_inactive', 403);
    }

    // Получив инстанс аутентифицированного приложения
    // передаем его в Request. Это позволит нам
    // иметь легкий доступ к инстансу приложения повсеместно.
    $request->merge(['__authenticatedApp' => $app]);

    return $next($request);
}

private function payloadIsValid($payload)
{
    $validator = Validator::make($payload, [
        'iss' => 'required|in:valhalla',
        'sub' => 'required',
    ]);
}

```



```

        if (! $validator->passes()) {
            throw new \InvalidArgumentException;
        }
    }
}

<?php

namespace App\Http\Middleware;

use Closure;

class ApiUserAuth
{
    /**
     * Handle an incoming request.
     *
     * @param \Illuminate\Http\Request $request
     * @param \Closure $next
     * @return mixed
     */
    public function handle($request, Closure $next)
    {
        $authToken = $request->bearerToken();

        try {
            $this->payloadIsValid(
                $payload = (array) JWT::decode($authToken,
                    'w5yuCV2mQDVTGmn3', ['HS256'])
            );

            $app = Application::whereKey($payload['iss']->firstOrFail());

            $user = User::whereEmail($payload['sub']->firstOrFail());
        } catch (\Throwable $e) {
            return response('token_invalid', 401);
        }

        if (! $app->is_active) {
            return response('app_inactive', 403);
        }

        if (DB::table('tokens_cemetery')->whereTokenId($payload['jti']->first()) {

```

```

        return response('token_deceased', 403);
    }

    $request->merge(['__authTokenId' => $payload['jti']]);

    $request->merge(['__authenticatedApp' => $app]);

    $request->merge(['__authenticatedUser' => $user]);

    return $next($request);
}

private function payloadIsValid($payload)
{
    $validator = Validator::make($payload, [
        'iss' => 'required',
        'sub' => 'required',
        'jti' => 'required',
    ]);

    if (!$validator->passes()) {
        throw new \InvalidArgumentException;
    }
}
}

<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Support\Facades\Auth;

class Authenticate
{
    /**
     * Handle an incoming request.
     *
     * @param \Illuminate\Http\Request $request
     * @param \Closure $next
     * @param string|null $guard
     * @return mixed
     */
    public function handle($request, Closure $next, $guard = null)

```

```

    {
        if (Auth::guard($guard)->guest()) {
            if ($request->ajax() || $request->wantsJson()) {
                return response('Unauthorized.', 401);
            } else {
                return redirect()->guest('login');
            }
        }
    }

    return $next($request);
}
}

```

<?php

```

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use \Firebase\JWT\JWT;

class Application extends Model
{
    public function generateAuthToken()
    {
        $jwt = JWT::encode([
            'iss' => 'easycrm',
            'sub' => $this->key,
            'iat' => time(),
            'exp' => time() + (5 * 60 * 60),
        ], 'w5yuCV2mQDVTGmn3');

        return $jwt;
    }
}

```

<?php

```

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Comment extends Model
{
    protected $table = 'comments';
}

```

```

        public $timestamps = true;
        protected $fillable = array(
            'text',
            'element_id',
            'element_type',
        );
    }
}

<?php

namespace App\Models;

use Illuminate\Foundation\Auth\User as Authenticatable;

class User extends Authenticatable
{
    public function generateAuthToken(Application $app)
    {
        $jwt = JWT::encode([
            'iss' => $app->key,
            'sub' => $this->email,
            'iat' => time(),
            'jti' => sha1($app->key.$this->email.time()),
        ], 'w5yuCV2mQDVTGmn3');

        return $jwt;
    }

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'name', 'email', 'password',
    ];

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password', 'remember_token',
    ];
}

```

```

    ];
}
<?php

namespace App\Providers;

use Illuminate\Support\ServiceProvider;
use Response;

class ResponseServiceProvider extends ServiceProvider
{
    /**
     * Bootstrap the application services.
     *
     * @return void
     */
    public function boot()
    {
        Response::macro('success', function ($data) {
            return Response::json([
                'errors' => false,
                'data' => $data,
            ]);
        });

        Response::macro('error', function ($message, $status = 400) {
            return Response::json([
                'errors' => true,
                'message' => $message,
            ], $status);
        });
    }

    /**
     * Register the application services.
     *
     * @return void
     */
    public function register()
    {
        //
    }
}

```

```

<?php

namespace App\Http\Controllers\Auth;

use App\User;
use Validator;
use App\Http\Controllers\Controller;
use Illuminate\Foundation\Auth\ThrottlesLogins;
use Illuminate\Foundation\Auth\AuthenticatesAndRegistersUsers;

class AuthController extends Controller
{
    /**
     |-----
     | Registration & Login Controller
     |-----
     |
     | This controller handles the registration of new users, as well as the
     | authentication of existing users. By default, this controller uses
     | a simple trait to add these behaviors. Why don't you explore it?
     |
     */

    use AuthenticatesAndRegistersUsers, ThrottlesLogins;

    /**
     * Where to redirect users after login / registration.
     *
     * @var string
     */
    protected $redirectTo = '/';

    /**
     * Create a new authentication controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware($this->guestMiddleware(), ['except' => 'logout']);
    }

    /**
     * Get a validator for an incoming registration request.

```

```

*
* @param array $data
* @return \Illuminate\Contracts\Validation\Validator
*/
protected function validator(array $data)
{
    return Validator::make($data, [
        'name' => 'required|max:255',
        'email' => 'required|email|max:255|unique:users',
        'password' => 'required|min:6|confirmed',
    ]);
}

/**
 * Create a new user instance after a valid registration.
 *
 * @param array $data
 * @return User
 */
protected function create(array $data)
{
    return User::create([
        'name' => $data['name'],
        'email' => $data['email'],
        'password' => bcrypt($data['password']),
    ]);
}
}

```