



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(национальный исследовательский университет)»

Институт (Филиал) № 8 «Компьютерные науки и прикладная математика» Кафедра 806
Группа М8О-408Б-20 Направление подготовки 01.03.02 Прикладная математика и
информатика

Профиль Информатика и компьютерные науки

Квалификация: бакалавр

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

на тему: «Разработка клиентской части CRM-системы»

Автор ВКРБ: Борисов Ян Артурович (_____)

Руководитель: Лукин Владимир Николаевич (_____)

Консультант: (_____)

Консультант: (_____)

Рецензент: (_____)

К защите допустить

Заведующий кафедрой № 806 «Вычислительная математика
и программирование» Крылов Сергей Сергеевич (_____)

___ 2024 года

Москва 2024

РЕФЕРАТ

Выпускная квалификационная работа бакалавра состоит из 50 страниц, 27 рисунков, 12 использованных источников, 2 приложений.

CRM, СЕРВИС, СИСТЕМА, ПРИЛОЖЕНИЕ, КЛИЕНТ, СЕРВЕР, КОМПАНИЯ, СОТРУДНИК, ОБЪЯВЛЕНИЕ, СДЕЛКА

Объектом разработки в данной работе является CRM-система, предоставляющая функции торговой площадки и рассчитанная на использование в сфере B2B.

Цель работы – создание CRM-системы с удобным функционалом для взаимодействия компаний.

Для достижения поставленной цели были проведены исследования функционала существующих CRM-систем. Основное содержание работы состояло в разработке CRM-системы, рассчитанной на использование в B2B-сфере.

Основной результат работы — клиентская часть CRM-системы, реализованная как одностраничное приложение, предоставляющее пользователю весь необходимый функционал и открытое для расширения и совершенствования.

Результаты разработки предназначены для использования компаниями крупного, среднего и малого бизнесов, которые продают свои товары и услуги другим компаниям. Результаты данной работы позволяют компаниям упростить и сделать более удобным взаимодействие с заказчиками.

СОДЕРЖАНИЕ

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ.....	5
ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ	6
ВВЕДЕНИЕ.....	7
1 CRM-СИСТЕМА ДЛЯ B2B-СЕГМЕНТА	10
1.1 Компания	10
1.2 Сотрудник.....	11
1.3 Контакт.....	12
1.4 Объявление	12
1.5 Сделка	12
1.6 Уведомление.....	13
2 АРХИТЕКТУРА ПРОЕКТА	14
2.1 Одностраничные и многостраничные приложения	14
2.1.1 Многостраничные приложения.....	14
2.1.2 Одностраничные приложения	15
2.2 Взаимодействие с сервером	15
2.2.1 HTTP.....	15
2.2.2 REST	16
2.3 Используемые технологии	18
2.3.1 TypeScript.....	19
2.3.2 Angular.....	19
2.3.3 RxJs.....	21
2.3.4 Angular Material.....	22
2.3.5 Docker.....	23

2.4 Авторизация и аутентификация. JWT-токены	24
3 КЛИЕНТСКАЯ ЧАСТЬ CRM-СИСТЕМЫ	26
3.1 Страницы регистрации и авторизации	26
3.2 Главная страница компании.....	30
3.3 Информация о компании.....	31
3.4 Контакты	33
3.5 Сотрудники	35
3.6 Объявления	37
3.7 Сделки	40
3.8 Уведомления.....	42
ЗАКЛЮЧЕНИЕ	44
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	45
ПРИЛОЖЕНИЕ А QR-код репозитория с исходным кодом.....	46
ПРИЛОЖЕНИЕ Б Методы HTTP-запросов	47

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

В настоящей выпускной квалификационной работе бакалавра применяются следующие термины с соответствующими определениями:

Веб-сервис — программная система, имеющая стандартизированный интерфейс и веб-адрес

Веб-сервер — программное обеспечение, принимающее запросы от клиентов и выдающее ответы на них

Фронтенд — публичная часть веб-сервисов, с которой пользователь взаимодействует напрямую

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей выпускной квалификационной работе бакалавра применяются следующие сокращения и обозначения:

CRM — Customer Relationship Management, или управление взаимоотношениями с клиентами

B2B — Business-to-business, или «бизнес для бизнеса»

HTTP — HyperText Transfer Protocol, или протокол передачи гипертекста

ОС — операционная система

VM — Virtual Machine, или виртуальная машина

JSON — Javascript Object Notation, или текстовый формат обмена данными, основанный на JavaScript

REST — Representational State Transfer, или передача состояния представления

API — Application Programming Interface, или программный интерфейс приложения

JWT — JSON Web Token

UI — пользовательский интерфейс

ВВЕДЕНИЕ

Одной из важнейших целей бизнеса с давних времен было упрощение взаимодействия с клиентами и ведение статистики уровня удовлетворенности клиента. Чтобы решить данную задачу были созданы CRM-системы.

CRM-система (customer relationship management) — это прикладное программное обеспечение для автоматизации и контроля взаимодействия компании с клиентами [1]. Она хранит и структурирует информацию о заказах и покупателях, помогает оптимизировать маркетинг, повысить продажи и качество обслуживания. Первые прототипы современных CRM появились в далеком 1947 году и представляли структурированные блокноты. Этими блокнотами пользовались менеджеры по продажам, которые вели записи о взаимодействиях со своими заказчиками и партнерами. Конечно, такой подход был неудобен, ведь фактически эти блокноты были несколько несвязанных между собой CRM.

Однако с массовым распространением компьютеров положение дел изменилось. В 1995 году появилось понятие CRM-систем и началось их развитие. В 2007-2008 году произошел серьезный прорыв, CRM стали облачными, значительно улучшились надежность и безопасность хранения информации [2].

CRM-системы позволяют компаниям удобно взаимодействовать со своими клиентами, чтобы предоставлять им различные услуги. Перечислим основные задачи современной CRM-системы:

1) Сохранение истории взаимоотношения с клиентом. CRM сохраняет всю историю общения с клиентом, его заказы, оплаты, просмотренные и отложенные товары. Это позволяет менеджерам адаптироваться под каждого клиента и предлагать ему самые подходящие услуги.

2) Повышение эффективности работы менеджеров. CRM системы позволяют планировать работу менеджера, отслеживать выполнение им своих обязанностей, анализировать эффективность труда.

3) Автоматизация бизнес-процессов. CRM система позволяет проводить расчет заработной платы, запускать рассылки, вести всю необходимую отчетность, разрабатывать и внедрять программу лояльности.

4) Получение прозрачной и наглядной аналитики. Отчеты в CRM формируются быстро и предоставляются руководителю в доступном виде. С помощью показателей из CRM можно анализировать эффективность рекламы, статистику звонков, точки роста, динамику оплат, забытые сделки и другие показатели. CRM-аналитика имеет гибкую настройку и наглядную форму.

Однако мы не можем забывать и о том, что на рынке есть компании, основные клиенты которых тоже компании. Они не могут размещать объявления в социальных сетях или на популярных торговых площадках: целевая аудитория этих сервисов не заинтересована в оптовой закупке товаров для ведения бизнеса. Именно по этой причине появляется необходимость в CRM-системе, которая позволит компаниям сотрудничать друг с другом, вести так называемый B2B. Торговая площадка, встроенная в данную CRM-систему сможет агрегировать все актуальные услуги и товары в одном месте, что позволит компаниям-производителям услуг размещать объявления, назначать ответственных за них, а компаниям-потребителям — выбирать самое подходящее для них предложение, проводить сделки прямо в CRM-системе с контролем над всеми этапами.

Таким образом, поставленная задача и выполнима, и актуальна с практической точки зрения.

Цель работы – реализация клиентской части для CRM системы, главным преимуществом которой будет возможность публиковать услуги и товары на площадке, откликаться на объявления и проводить сделки. Клиентская часть реализована на языке Typescript с применением фреймворка Angular, использована библиотека компонентов Angular Material.

В работе решены следующие задачи:

1) разработана главная страница со статистикой компании;

- 2) разработан интерфейс и функционал площадки для объявлений;
- 3) разработана возможность просмотра контактов и создания новых.

Основные результаты:

- 1) создана клиентская часть CRM-системы;
- 2) создана страница площадки для объявлений, где пользователь может создать объявление о продаже товара, необходимого для деятельности компаний из разных сегментов бизнеса;
- 3) создана страница контактов, где пользователь может редактировать заметки о сотруднике.

Использование результатов работы некоторыми компаниями позволит им более эффективно находить и обслуживать клиентов, которым необходимы оптовые поставки либо поставки специфических товаров.

1 CRM-СИСТЕМА ДЛЯ B2B-СЕКМЕНТА

Необходимо разработать CRM-систему для B2B-сегмента, предлагающую функционал торговой площадки, под названием BRM.

Основными сущностями приложения будут:

- 1) компания;
- 2) сотрудник;
- 3) контакт;
- 4) объявление;
- 5) сделка.

В компании может быть несколько сотрудников. Владелец компании имеет возможность создавать учетные записи для сотрудников. Каждый сотрудник может размещать объявления от лица своей компании, просматривать объявления других компаний и откликаться на них. Когда сотрудник откликается на объявление — создается сделка, доступ к ней открывается сотрудникам компании, на чье объявление откликнулись.

Сделка — процесс согласования деталей предоставления услуги между потребителем и поставщиком, состоящий из нескольких этапов. Успешное закрытие сделки означает, что обе компании пришли к соглашению, и поставщик обеспечил потребителя услугой в соответствии со всеми договоренностями. Сотрудник может добавлять сотрудников своей или чужих компаний в контакты для упрощения дальнейшей коммуникации.

Рассмотрим данные сущности подробнее.

1.1 Компания

Владелец компании может зарегистрировать свою компанию в системе.

Для регистрации компании необходимо заполнить поля “Название”, “Описание” и “Отрасль”.

Для выбора доступны следующие отрасли:

- 1) информационные технологии;
- 2) юридические услуги;

- 3) транспорт;
- 4) промышленность;
- 5) еда и напитки;
- 6) одежда и обувь;
- 7) развлечения;
- 8) туризм;
- 9) медицина;
- 10) рестораны и бары;
- 11) гостиницы.

Для владельца компании необходимо заполнить поля “Имя”, “Фамилия”, “Почта”, “Пароль”, “Должность” и “Департамент”. Система не позволяет зарегистрировать владельца без компании или компанию без владельца, для предотвращения существования этих сущностей в разрыве друг от друга.

1.2 Сотрудник

Сотрудник — это основной пользователь нашей системы. Он имеет следующие характеристики:

- 1) компания;
- 2) фамилия;
- 3) имя;
- 4) почтовый адрес;
- 5) занимаемая должность;
- 6) отдел компании;
- 7) дата регистрации.

Сотрудники имеют возможность создавать объявление от лица своей компании и откликаться на объявления других компаний, тем самым создавая сделку. Ответственный за сделку сотрудник может изменять её статус и редактировать основную информацию. Ответственный за объявление — имеет возможность в любой момент редактировать параметры объявления.

1.3 Контакт

Пользователи могут добавлять в контакты сотрудников своей компании или компании-партнера, для упрощения дальнейшей коммуникации. Контактная запись состоит из фотографии сотрудника, компании, департамента, заметок, рабочего номера телефона и почты. Заметки о контакте можно редактировать, дополняя необходимой информацией.

1.4 Объявление

В CRM-системе разработана полноценная площадка для размещения объявлений. Каждое объявление имеет следующие характеристики:

- 1) компания;
- 2) название;
- 3) текст;
- 4) отрасль;
- 5) стоимость;
- 6) сотрудник, создавший объявление;
- 7) сотрудник, ответственный за объявление;
- 8) дата создания.

При отклике на объявление создаётся сделка, откликнувшийся сотрудник назначается ответственным за проведение этой сделки.

1.5 Сделка

Сделка — одна из ключевых сущностей любой современной CRM. Имеет следующие характеристики:

- 1) объявление, на основе которого она создана;
- 2) название;
- 3) описание;
- 4) стоимость;
- 5) статус;
- 6) ответственный;
- 7) компания-поставщик;

- 8) сотрудник-клиент;
- 9) компания сотрудника-клиента;
- 10) дата создания.

1.6 Уведомление

В BRM существует два типа уведомлений:

- 1) уведомление о новой сделке;
- 2) уведомление о закрытой сделке.

В BRM все уведомления общие на всю компанию. Их может просматривать любой сотрудник, свежие уведомления имеют статус «Не просмотрено», при просмотре уведомления сотрудником статус меняется на противоположный.

Уведомление о новой сделке приходит в компанию вместе с новой сделкой, когда на объявление компании откликается клиент. Задача этого уведомления — обратить внимание сотрудников компании на то, что появилась новая сделка. Помимо общей информации, это уведомление содержит новую сделку и компанию, откликнувшуюся на объявление.

Уведомление о закрытой сделке имеет более важное значение. Как было отмечено, рейтинг компании растёт, если она закрывает сделки со своими клиентами. Но тогда компания может сразу же переводить все новые сделки по своим объявлениям в статус «Завершено», даже не связываясь с клиентом, тем самым искусственно увеличивать свой рейтинг. Уведомление о закрытой сделке существует, чтобы предотвратить такие случаи. Оно приходит компании-клиенту, когда компания-поставщик переводит сделку с этим клиентом в состояние «Завершено».

2 АРХИТЕКТУРА ПРОЕКТА

2.1 Одностраничные и многостраничные приложения

Выделяют два основных подхода к разработке веб-приложений: многостраничные (multi-page applications или МРА) и одностраничные (single-page applications или SPA). Разберем каждый из этих двух подходов подробнее.

2.1.1 Многостраничные приложения

Многостраничные приложения — это набор статичных веб-страниц, которые связаны между собой с помощью ссылок [6]. При переходе между страницами, клиент запрашивает новую страницу у сервера, что ведёт к полному обновлению страницы в браузере. Существуют два подвида многостраничных приложений:

1) Набор готовых свёрстанных страниц, которые лежат на сервере и сервер отдаёт их по настроенным путям. Такие приложения подходят для того, чтобы собрать несколько связанных страниц, которые содержат статичную информацию, например лендинги — веб-сайты, которые содержат информацию о компании или продукте.

2) Приложения с динамической генерацией HTML на сервере, при каждом запросе сервер запускает скрипт по генерации HTML-страницы, которую затем отдаёт клиенту. Этот подход позволяет реализовать сложную бизнес-логику на сервере, например обращение к базе данных для извлечения какой-либо информации, генерируя персонализированные страницы для каждого пользователя.

При всех плюсах многостраничных приложений, полное обновление страницы при каждом запросе не позволяет создать полноценный опыт взаимодействия клиента с приложением. В стремлении решить эти проблемы веб-разработчики изобрели одностраничные приложения.

2.1.2 Одностраничные приложения

Одностраничные приложения работают так, что при открытии страницы браузер загружает сразу всё приложение, а затем с помощью JavaScript обращается к серверу для загрузки лишь малого количества данных и их отображения на конкретной странице приложения [7]. Такие приложения работают быстро и создают меньшую нагрузку на сервер, что очень важно для поддержания производительности всей системы в целом. Сейчас SPA набирают огромную популярность, разрабатываются множество фреймворков и библиотек для создания интерфейсов, появляется все больше подходов к разработке и архитектурных паттернов. Самыми популярными инструментами для разработки SPA можно назвать: React, Angular и Vue.js. Однако у SPA есть и значимый недостаток - проблемы с SEO. Поисковые машины индексируют отдельные страницы, у каждой из которых есть свой заголовок, описание и прочие метатеги. В случае с SPA для корректного сбора аналитики и оптимизации поиска разработчикам необходимо тщательно настраивать обработку различных событий самостоятельно. К тому же разработка одностраничных приложений обычно более трудоёмка и требует особого тщательного тестирования.

2.2 Взаимодействие с сервером

Для обмена информацией с сервером используется протокол HTTP, взаимодействие клиента с сервером в нашей CRM-системе осуществляется согласно архитектурному подходу REST.

2.2.1 HTTP

HTTP — это протокол прикладного уровня, позволяющий системам обмениваться данными (изначально предназначенный для передачи гипертекстовых документов, то есть документов, которые могут содержать ссылки, позволяющие организовать переход к другим документам). Протокол

HTTP лежит в основе обмена данными в Интернете и является протоколом клиент-серверного взаимодействия, что означает инициирование запросов к серверу самим получателем, обычно веб-браузером. Полученный итоговый документ может состоять из различных поддокументов, являющихся частью итогового документа: например, из отдельно полученного текста, описания структуры документа, изображений, видеофайлов, скриптов и многого другого. Структура HTTP-запроса хорошо видна на рисунке 1.

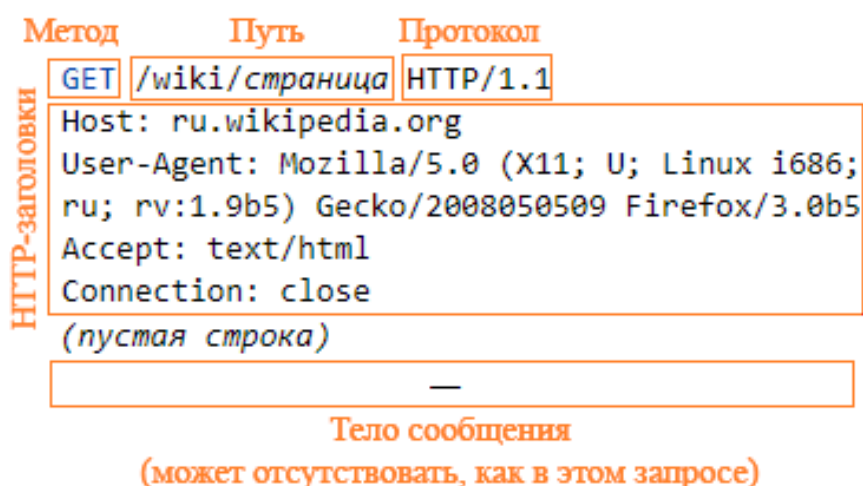


Рисунок 1 – Структура HTTP-запроса

В Приложении Б перечислены методы взаимодействия с ресурсом по протоколу HTTP.

2.2.2 REST

“REST” — это архитектурный стиль взаимодействия компонентов распределенного приложения в сети. Данный термин был введен Роем Филдингом, одним из создателей протокола “HTTP”. В своей диссертации «Архитектурные стили и дизайн сетевых программных архитектур» он подвёл теоретическую основу под способ взаимодействия клиентов и серверов во Всемирной паутине, абстрагировав его и назвав «передачей представительного состояния» («Representational State Transfer»). Главная идея REST заключается в том, что каждый запрос клиента к серверу должен

содержать в себе исчерпывающую информацию о желаемом ответе сервера, при этом сервер не обязан сохранять информацию о состоянии клиента. REST архитектура обладает преимуществами:

- 1) производительность: взаимодействие компонентов системы может быть доминирующим фактором производительности и эффективности сети с точки зрения пользователя;
- 2) масштабируемость для обеспечения большого числа компонентов и взаимодействий компонентов;
- 3) простота интерфейса взаимодействия с сервисами;
- 4) открытость компонентов к возможным изменениям и расширениям функционала для удовлетворения изменяющихся потребностей;
- 5) прозрачность связей между компонентами системы для сервисных служб;
- 6) надежность, выражающаяся в устойчивости к отказам на уровне системы при наличии отказов отдельных компонентов, соединений или данных.

Для того, чтобы эти преимущества были достигнуты, Филдинг сформулировал пять обязательных требований для построения распределенных REST-приложений:

1) Клиент-серверная модель взаимодействия

Разграничение потребностей является принципом, лежащим в основе данного накладываемого ограничения. Отделение потребности интерфейса клиента от потребностей сервера, хранящего данные, повышает переносимость кода клиентского интерфейса на другие платформы, а упрощение серверной части улучшает масштабируемость.

2) Отсутствие состояния

Протокол взаимодействия между клиентом и сервером требует соблюдения следующего условия: в период между запросами клиента никакая информация о состоянии клиента на сервере не хранится. Все запросы от

клиента должны быть составлены так, чтобы сервер получил всю необходимую информацию для выполнения запроса. Состояние сессии при этом сохраняется на стороне клиента. Информация о состоянии сессии может быть передана сервером какому-либо другому сервису (например, в службу базы данных) для поддержания устойчивого состояния, например, на период установления аутентификации. Клиент инициирует отправку запросов, когда возникает необходимость перейти в новое состояние.

3) Единообразие интерфейса

Наличие унифицированного интерфейса является фундаментальным требованием дизайна REST-сервисов. Унифицированные интерфейсы позволяют каждому из сервисов развиваться независимо.

4) Кэширование

Правильное использование кэширования способно частично или полностью устранить некоторые проблемы клиент-серверного взаимодействия, ещё больше повышая производительность и масштабируемость системы.

5) Слои

Клиент обычно не способен точно определить, взаимодействует ли он напрямую с сервером или же с промежуточным узлом, в связи с иерархической структурой сетей. Применение промежуточных серверов способно повысить масштабируемость за счёт балансировки нагрузки и распределённого кэширования. Промежуточные узлы могут подчиняться политике безопасности с целью обеспечения конфиденциальности информации.

2.3 Используемые технологии

Фронтенд реализован на языке программирования Typescript с использованием фреймворка Angular, для разработки элементов страниц используется библиотека компонентов Angular Material.

2.3.1 TypeScript

TypeScript — язык программирования, представленный Microsoft в 2012 году и позиционируемый как средство разработки веб-приложений, расширяющее возможности JavaScript. TypeScript обратно совместим с JavaScript и компилируется в него. По сути, браузер исполняет код, написанный на TypeScript так, как будто он написан на JavaScript. Основной причиной появления TypeScript является то, что компании, работающие над крупными проектами, часто сталкивались с ошибками, допущенными во время написания программ на JavaScript [10].

Главным преимуществом Typescript является его строгая типизация данных, что позволяет обнаруживать многие ошибки уже на этапе написания программного кода. TypeScript расширяет JavaScript в плане ООП, вводя в язык интерфейсы, перечисляемые типы, модификаторы доступа и прочие достоинства данной парадигмы. В 2019 году TypeScript стал одним из самых любимых языков программирования и попал в десятку популярных [4].

Однако не для всех проектов использование данного языка можно назвать полностью оправданным. Разработка на TypeScript обычно более трудоёмкая и занимает больше времени, к тому же порог входа в TypeScript выше, чем в JavaScript.

Фактически, на данный момент TypeScript — это стандарт для Frontend-разработки, который используется во всех хоть сколько-нибудь крупных проектах, которые планируется поддерживать и расширять на протяжении долгих лет.

2.3.2 Angular

Angular — открытая и свободная платформа для разработки веб-приложений, написанная на языке TypeScript, разрабатываемая командой из компании Google, а также сообществом разработчиков из различных компаний. Данный фреймворк предоставляет множество полезных

инструментов для разработки одностраничных приложений. Отметим основные преимущества Angular.

MVC-архитектура, изображенная на рисунке 2, представляет собой цепочку элементов “модель-представление-контроллер”. Этот подход позволяет четко разделить логику каждого слоя приложения. Все запросы поступают непосредственно к контроллеру, который затем работает в режиме подготовки данных, необходимых представлению. Затем представление использует эти данные, для отображения финального презентабельного ответа. В качестве представления Angular вводит компоненты, которые содержат в себе всю логику пользовательского интерфейса, имеют свои собственные HTML и CSS файлы, а также TypeScript код. Контроллерами же в Angular выступают сервисы, которые используются для обработки данных и доставки их компонентам [3].

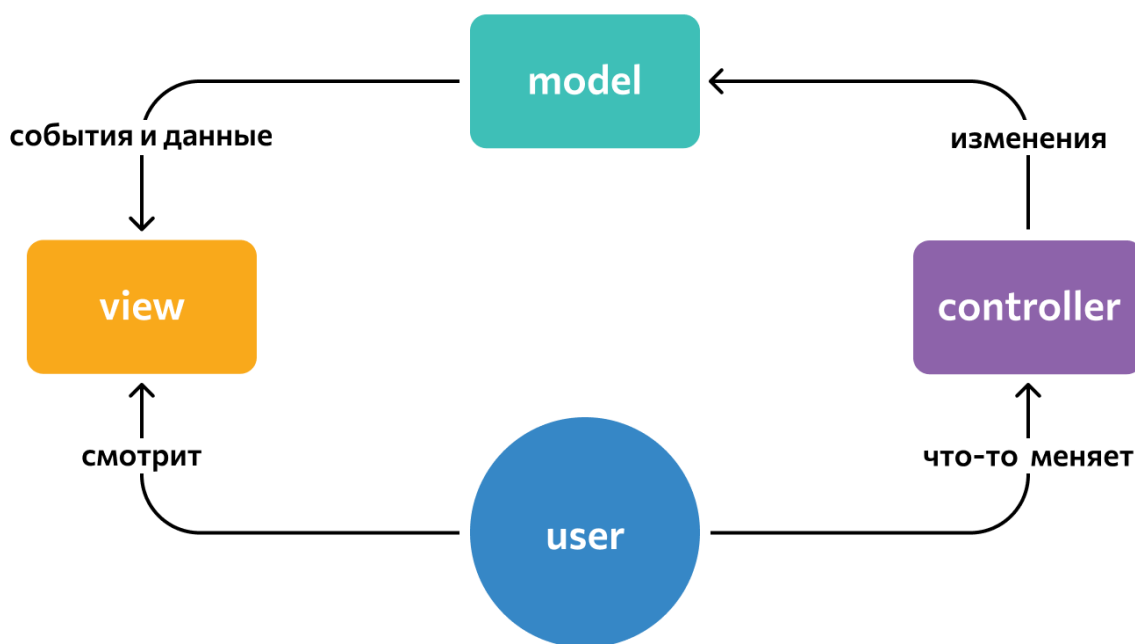


Рисунок 2 – Архитектура “модель-представление-контроллер”

Модулярность: разработчики Angular вводят понятие компонент, которые позволяют мыслить о разработке приложения как о сборке конструктора. Такая философия позволяет поделить крупное и сложное

приложение на маленькие составные части, что делает его поведение понятным и упрощает отладку, так как программист всегда знает, в каком месте он может начать поиск ошибки.

Внедрение зависимостей (Dependency Injection или DI): шаблон разработки программного обеспечения, который позволяет описать, каким способ компоненты получают информацию о своих зависимостях, что освобождает программиста от необходимости создавать все необходимые сущности вручную. В Angular сервисы — это объекты-одиночки (singleton), которые создаются в приложении только один раз и внедряются в необходимые для использования места через механизм Dependency Injection.

Связь данных с представлением: Angular позволяет связать представление данных с их моделями (two-way data binding), что даёт возможность разработчику не задумываться о поддержании хранящихся в приложении данных актуальными при изменении представления, или наоборот.

2.3.3 RxJs

Клиентская часть любого серьёзного веб-приложения подразумевает асинхронное выполнение каких-либо задач. Если бы все задачи выполнялись синхронно, то вместо плавного опыта взаимодействия с приложениями, пользователю приходилось бы постоянно ждать выполнения браузером каких-либо действий, при этом приложение зависало бы, и пользователю постоянно приходилось бы наблюдать перед собой колесо загрузки. Асинхронное программирование — одна из самых важных и в то же время сложных задач для фронтенд-разработчика.

Старый механизм асинхронного программирования в JavaScript — это Promise. Написать хороший и качественный Promise было трудной задачей для большинства разработчиков. На помощь им пришла библиотека RxJs. Она по умолчанию встроена в Angular и, более того, сам Angular реализован с

использованием RxJs. Таким образом, работая с Angular, необходимо понимать принцип её работы.

Асинхронные задачи в RxJs представлены как потоки определенных событий, на выполнение которых можно подписаться. Эта абстракция значительно упрощает написание асинхронного кода, ведь для работы с этими потоками сам RxJs вводит большое количество различных операторов, с помощью которых можно преобразовывать приходящие события необходимым образом и предоставлять подписчику уже в том виде, которого он сам требует.

Две основные сущности в RxJs — Observable и Subject. Основное их отличие заключается в том, что поток событий в Observable однонаправлен на чтение, невозможно поместить в него какие-либо новые объекты после его инициализации. Subject, наоборот, позволяет добавлять новые события в уже “просматриваемый” поток, что не только делает его более гибким, но и вынуждает работать с ним аккуратно. В силу своего функционала Subject не уничтожается самостоятельно, тем самым, забыв отписаться от его исполнения, мы можем столкнуться с утечками памяти, что негативно повлияет на работу всего приложения [8].

Ярким примером использования реактивного подхода RxJs в фреймворке Angular является выполнение HTTP-запросов к серверу. Все HTTP-методы, реализованные в сервисе HTTPClient, возвращают Observable.

2.3.4 Angular Material

Библиотеки компонент — одни из самых полезных инструментов, созданных для frontend-разработки. Если бы разработчикам приходилось разрабатывать компоненты с нуля, то создание простейшего приложения растягивалось бы на длительное время и требовало бы настолько большое количество затрат, что любая разработка была бы неоправданной и непривлекательной для заказчика программного продукта.

Для разработки BRM-системы использовалась библиотека компонент Angular Material, созданная разработчиками фреймворка Angular. В ней мы смогли найти все необходимые компоненты для качественного отображения данных и взаимодействия с ними. Например: диалоговые окна, которые используются нами для того, чтобы предоставить пользователю возможность заполнить данные в форме и отправить их на сервер, при этом не покидая основную страницу, с которой пользователь взаимодействовал до этого, переключатель страниц (paginator), который позволяет поделить все данные на страницы и не запрашивать большое количество данных с сервера одним запросом [12].

Angular Material позволяет придать приложению достаточно качественный дизайн “из коробки”, ведь при реализации всех компонентов Google следовала своей же дизайн-системе Material, в которой собрано множество законов и практик, чтобы сделать опыт использования приложения наиболее приятным и плавным для пользователя.

2.3.5 Docker

Установка различных инструментов, их подготовка к использованию программным обеспечением и запуск — отнюдь не легкая задача. Мы не можем гарантировать, что компоненты нашей системы будут работать одинаково на всех операционных системах и во всех средах.

Docker — программное обеспечение для контейнеризации приложений, выпущенное в 2013 году, оно поддерживается и обновляется до сих пор. Оно было создано, чтобы решить проблемы, возникающие при настройке рабочего окружения во время разработки и внедрения систем. Контейнеризация позволяет создать контейнер с легковесной операционной системой, который запускает в себе сервис и настраивает всё необходимое для его работы окружение, при этом гарантирующий его консистентность [9].

Все компоненты BRM-системы, в том числе и её клиентская часть, размещены в Docker контейнере. Таким образом, любой пользователь может с

легкостью запустить её для использования, при условии, что на целевом сервере установлена платформа Docker. В качестве сервера клиентской части используется nginx. Dockerfile, который используется для создания и настройки окружения образа с фронтендом, представлен на рисунке 3. Он затем запускается в контейнере вместе с другими компонентами системы с помощью утилиты docker-compose:

```
#STAGE 1
FROM node:latest AS build
WORKDIR /usr/src/app
COPY package.json package-lock.json ./
RUN npm install
COPY . .
RUN npm run build

#STAGE 2
FROM nginx:stable-alpine3.17-slim
WORKDIR /usr/share/nginx/html
RUN rm /etc/nginx/conf.d/default.conf
COPY --from=build /usr/src/app/dist/brm/browser .
#overriding default nginx configuration
COPY nginx.conf /etc/nginx/conf.d/
```

Рисунок 3 – Dockerfile клиентской части приложения

2.4 Авторизация и аутентификация. JWT-токены

Для безопасности данных компаний и их сотрудников, препятствия вредительским действиям злоумышленников приложение CRM-системы должно реализовывать механизмы авторизации и аутентификации.

Аутентификация — процесс проверки личности пользователя на обладание правом доступа. Авторизация определяет, имеет ли пользователь с определёнными правами доступ к каким-либо операциям над информацией. Аутентификация и авторизация работают вместе, существует несколько

способов их реализации. Самым распространенным на данный момент можно назвать токенизацию (процесс замены конфиденциального элемента данных на неконфиденциальный эквивалент, называемый токеном, который не имеет самостоятельного значения для внешнего или внутреннего использования), а именно JSON Web Tokens (JWT).

JWT — это открытый стандарт для создания токенов доступа, основанный на формате JSON. JWT состоит из трех частей, которые разделены точкой:

1) Header или заголовок — информация о токене, тип токена и алгоритм шифрования;

2) Payload или полезные данные — данные, которые мы хотим передать в токене. Например, имя пользователя, его роль, истекает ли токен. Эти данные представлены в виде JSON-объекта;

3) Signature или подпись — подпись токена, которая позволяет проверить, что токен не был изменен.

Существует два вида токенов access-токен и refresh-токен. При запросе клиента к серверу, клиент передает access-токен в специальном заголовке, а сервер проверяет подпись access-токена клиента, и если токен подписан неверно, запрос отклоняется. Важной особенностью access-токена является и то, что он имеет определенный период действия (обычно несколько часов), по истечении которого он становится невалидным и перестает приниматься сервером. Для обновления access-токена используется refresh-токен, который передаётся клиентом серверу, в ответ же клиент получает новый access-токен. Refresh-токен обычно имеет больший период действия (от недели до месяца). Если же истекают оба токена, пользователю будет необходимо заново пройти аутентификацию.

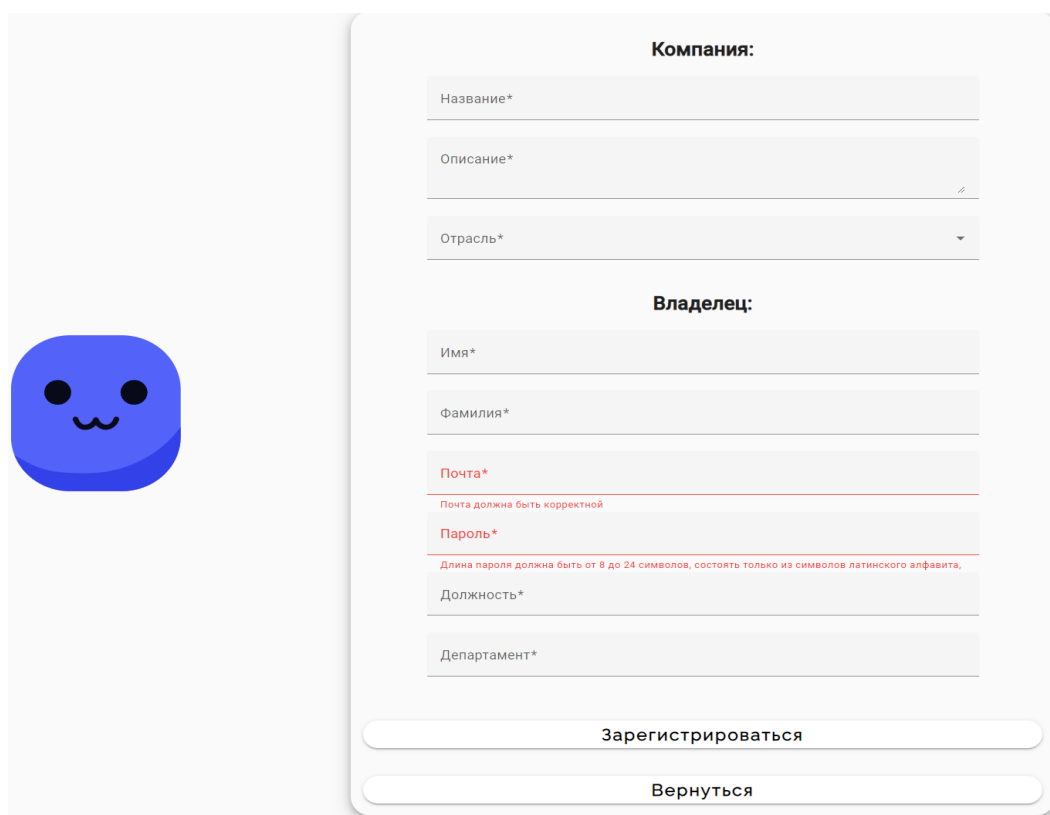
3 КЛИЕНТСКАЯ ЧАСТЬ CRM-СИСТЕМЫ

У клиентской части системы есть следующие страницы:

- 1) регистрация компании и владельца;
- 2) авторизация в системе;
- 3) главная страница компании;
- 4) информация о компании;
- 5) контакты;
- 6) объявления;
- 7) сделки;
- 8) уведомления.

3.1 Страницы регистрации и авторизации

Для регистрации компании в системе владельцу необходимо заполнить информацию про себя и компанию. Форма регистрации представлена на рисунке 4.



The registration form is presented on a light gray background. On the left side, there is a blue rounded square icon with a white smiley face. The form itself is a white rounded rectangle with a subtle shadow. It is titled 'Компания:' and 'Владелец:'. The 'Компания:' section has three input fields: 'Название*' (text), 'Описание*' (text with a small 'A' icon), and 'Отрасль*' (dropdown menu). The 'Владелец:' section has six input fields: 'Имя*' (text), 'Фамилия*' (text), 'Почта*' (text with a red error message 'Почта должна быть корректной'), 'Пароль*' (text with a red error message 'Длина пароля должна быть от 8 до 24 символов, состоять только из символов латинского алфавита.'), 'Должность*' (text), and 'Департамент*' (text). At the bottom of the form, there are two buttons: 'Зарегистрироваться' and 'Вернуться'.

Рисунок 4 – Страница регистрации компании и владельца

Отрасль можно выбрать из выпадающего списка, содержащего следующие значения:

- 1) информационные технологии;
- 2) юридические услуги;
- 3) транспорт;
- 4) промышленность;
- 5) еда и напитки;
- 6) одежда и обувь;
- 7) развлечения;
- 8) туризм;
- 9) медицина;
- 10) рестораны и бары;
- 11) гостиницы.

Список представлен на рисунке 5.

Компания:

Название*

Описание*

Отрасль*

- Еда и напитки
- Информационные технологии
- Медицина
- Одежда и обувь
- Промышленность

Почта*

Почта должна быть корректной

Пароль*

Должность*

Департамент*

Зарегистрироваться

Вернуться

Рисунок 5 – Выпадающий список отраслей компаний

Для построения формы использовался модуль реактивных форм из фреймворка Angular. Была построена FormGroup-а, указанная на рисунке 6, которая затем конвертируется в JSON и передается в запросе на сервер.

Для формы настроена валидация по обязательности заполнения полей, максимальной и минимальной длине текста. При нарушении валидности пользователю отображаются ошибки, которые нужно исправить для того, чтобы продолжить регистрацию.

```
form = this.fb.nonNullable.group({
  company: this.fb.nonNullable.group({
    description: this.fb.nonNullable.control('',
[Validators.maxLength(1000), Validators.required]),
    industry: this.fb.nonNullable.control('',
[Validators.required]),
    name: this.fb.nonNullable.control('',
[Validators.maxLength(100), Validators.required]),
  }),
  owner: this.fb.nonNullable.group({
    department: this.fb.nonNullable.control('',
[Validators.maxLength(100), Validators.required]),
    email: this.fb.nonNullable.control('', [Validators.email,
Validators.required]),
    first_name: this.fb.nonNullable.control('',
[Validators.maxLength(100), Validators.required]),
    job_title: this.fb.nonNullable.control('',
[Validators.maxLength(100), Validators.required]),
    password: this.fb.nonNullable.control('',
[Validators.minLength(8), Validators.maxLength(24),
Validators.required]),
    second_name: this.fb.nonNullable.control('',
[Validators.maxLength(100), Validators.required]),
  }),
});
```

Рисунок 6 – Листинг кода FormGroup-ы регистрации

Страница авторизации содержит поля для ввода данных, необходимых для авторизации в соответствии с рисунком 7.

```
form = this.fb.nonNullable.group({  
    email: ['', Validators.required],  
    password: ['', Validators.required],  
});
```

Рисунок 7 – Листинг кода FormGroup-ы авторизации

Кнопка “Войти” отвечает за отправку на сервер данных авторизации, заполненных в форме, представленной на рисунке 8.

В ответ на этот запрос сервер возвращает пару refresh и access токенов, которые затем сохраняются на фронте и используются для последующих запросов на получение информации. С помощью кнопки “Зарегистрироваться” пользователь переходит на страницу регистрации.

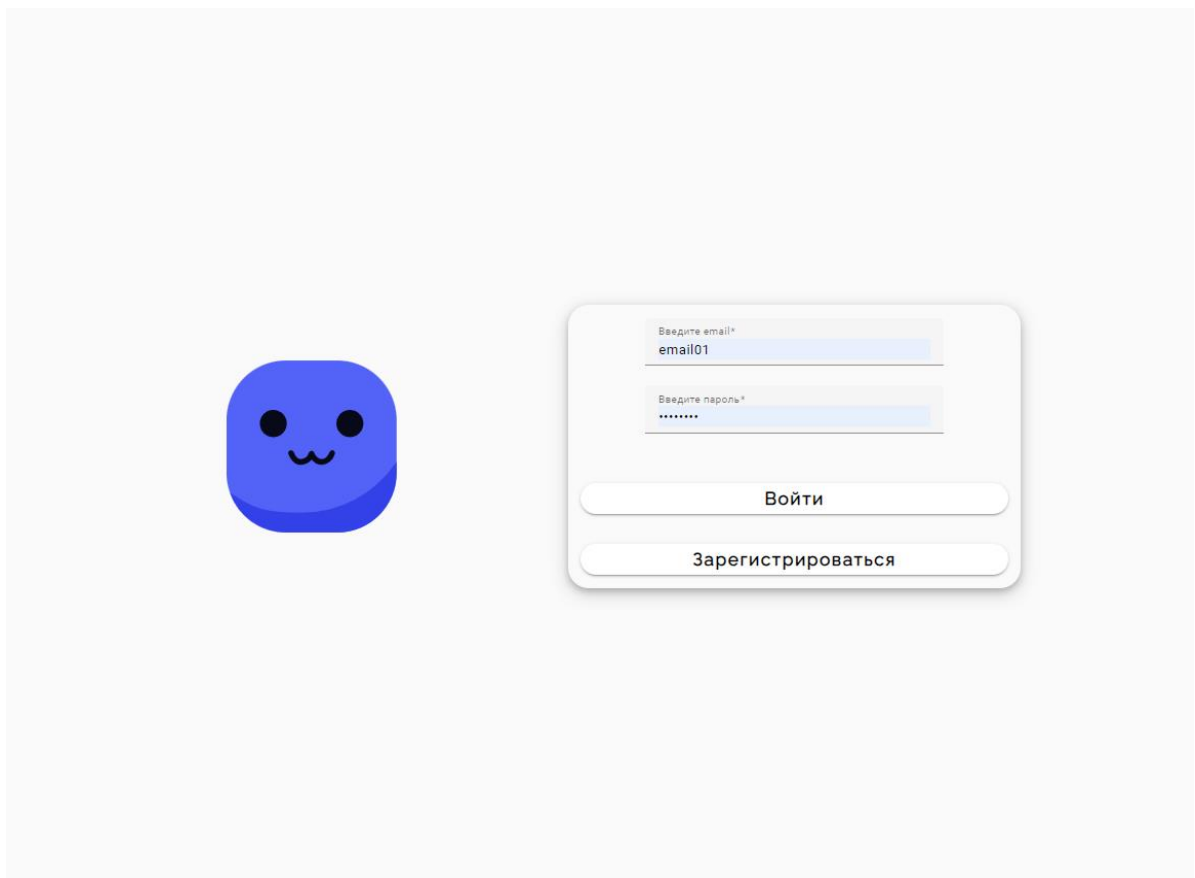


Рисунок 8 – Страница авторизации

3.2 Главная страница компании

На главной странице, представленной на рисунке 9, отображается статистика компании по основным параметрам: количество активных сделок, закрытых сделок, сумма сделок в соответствии с рисунком 10. Компании могут посмотреть количество активных объявлений, которые были оставлены ими на торговой площадке и свой рейтинг.

Рейтинг компании — важный показатель в нашей CRM-системе, зависящий от сделок, которые проводит компания. По рейтингу компании, доступному для просмотра в объявлениях, клиент может судить о её надёжности и принимать решение о возможности совершения сделки.

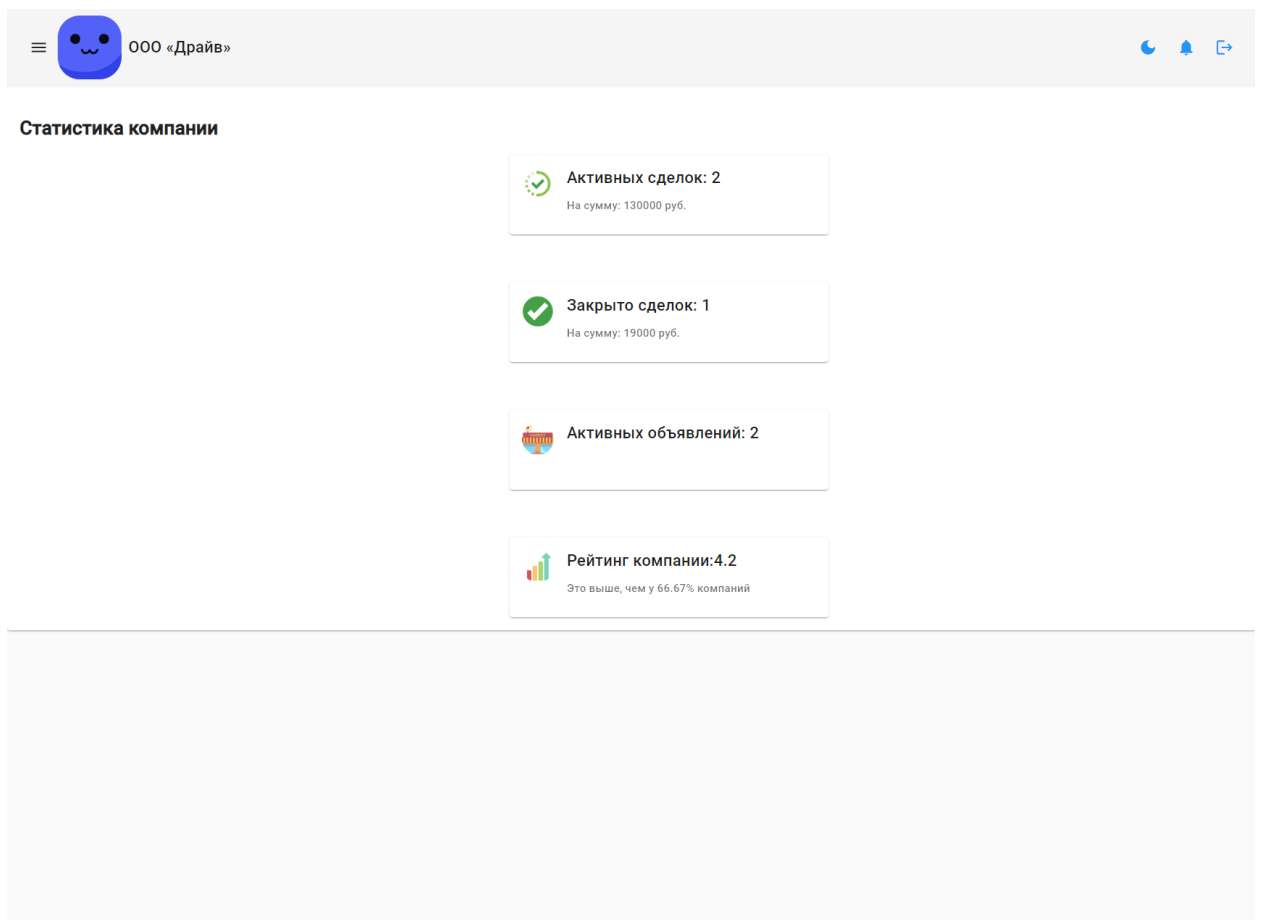


Рисунок 9 – Главная страница компании

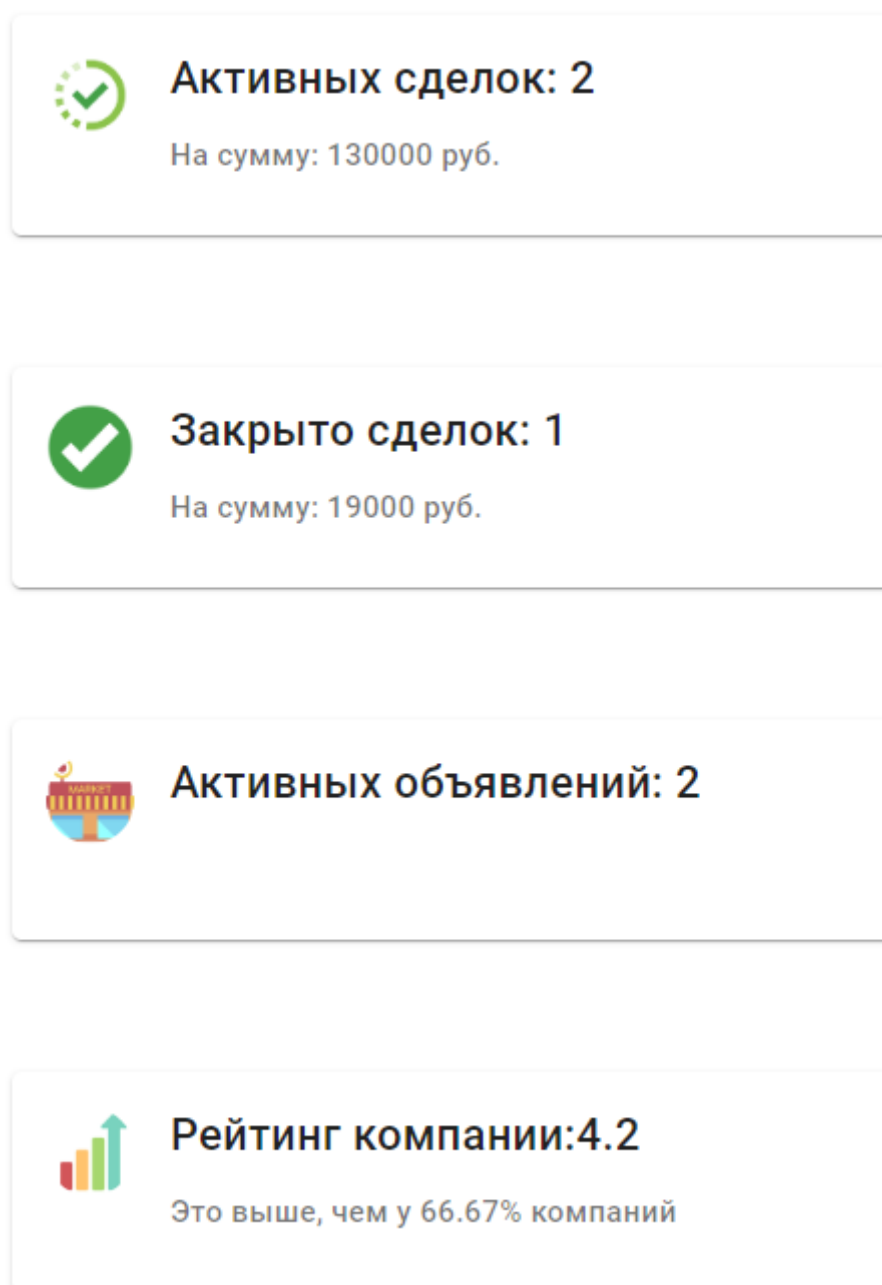







Рисунок 10 – Статистика компании

3.3 Информация о компании

На странице информации о компании можно изменить название, описание и отрасль компании. Страница представлена на рисунке 11.


 ИП "Пекарь"



Компания


Название*

ИП "Пекарь"





Описание*

Компания по производству мучных изделий




Отрасль*

Еда и напитки



Рейтинг

4,2








Вы можете изменить информацию о своей компании

Рисунок 11 – Информация о компании

При изменении какого-либо поля появляется кнопка “Сохранить”, которая отправляет запрос на сервер для изменения информации о компании (рисунок 12).

Компания

Название*	ООО «Банкет»	
Описание*	Компания по организации банкетов и деловых обедов 	
Отрасль*	Информационные технологии	 
Рейтинг	4,9	

Вы можете изменить информацию о своей компании





Обновить информацию о компании

Рисунок 12 – Обновление информации о компании

3.4 Контакты

На странице контактов сотрудник может просматривать контактные данные избранных работников как своей, так и другой компании в соответствии с рисунком 13.

Контакты

Компания: ООО «Драйв» Департамент: Продажи Должность: Менеджер по продажам Макар Тимофеев 
 
 makar_timofeev@yandex.ru





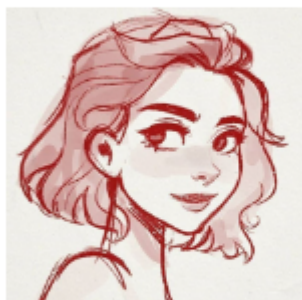
Компания: ООО «Драйв» Департамент: Продажи Должность: Менеджер по продажам Артём Кузнецов 
 
 artyom_kuznetsov@yandex.ru



Рисунок 13 – Страница контактов

Контактная запись состоит из фотографии сотрудника, компании, департамента, заметок и почты в соответствии с рисунком 14.

Компания: ООО «Драйв»
Департамент: Продажи
Должность: Менеджер по продажам

Елизавета Романова



 Очень ответственно подходит к делу 


 liza_romanova@yandex.ru

Рисунок 14 – Карточка контакта

При нажатии на кнопку карандаша открывается диалоговое окно, представленное на рисунке 15, позволяющее добавить или редактировать заметки о контакте.

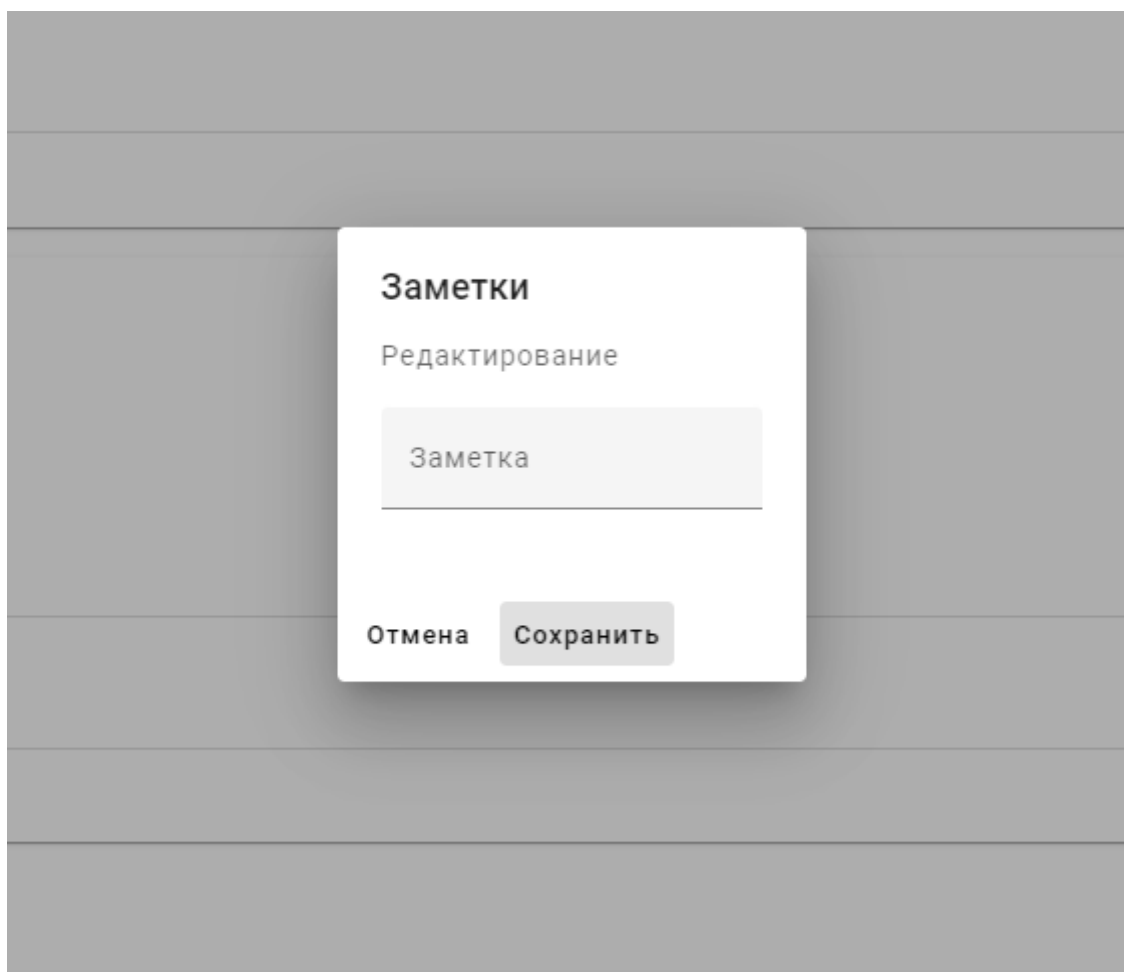


Рисунок 15 – Диалоговое окно редактирования заметок о контакте

3.5 Сотрудники

На странице сотрудников отображаются пользователи системы, которые работают в одной компании с авторизованным пользователем. Страница сотрудников изображена на рисунке 16. На этой странице владелец компании может добавлять новых сотрудников с помощью кнопки “Добавить сотрудника”, заполнив данные в открывшемся диалоговом окне, представленном на рисунке 17.

Сотрудники

[Добавить сотрудника](#)

Департамент: Продажи
Должность: Менеджер по продажам

Макар Тимофеев



✉ makar_timofeev@yandex.ru

[Добавить в контакты](#) [Редактировать](#)

Департамент: Продажи
Должность: Менеджер по продажам

Алиса Шарова



✉ alicesharova@yandex.ru

[Добавить в контакты](#) [Редактировать](#)

Рисунок 16 – Страница сотрудников

Новый сотрудник

Загрузи картинку

Имя*

Фамилия*

Почта

Департамент*

Должность*

email01

Пароль*

•••••

[Выйти](#)

[Сохранить](#)

Рисунок 17 – Диалоговое окно добавления сотрудника

3.6 Объявления

Страница объявлений в BRM-системе для удобства пользователя делится на две группы: “объявления компании пользователя” и “все объявления” в соответствии с рисунком 18.

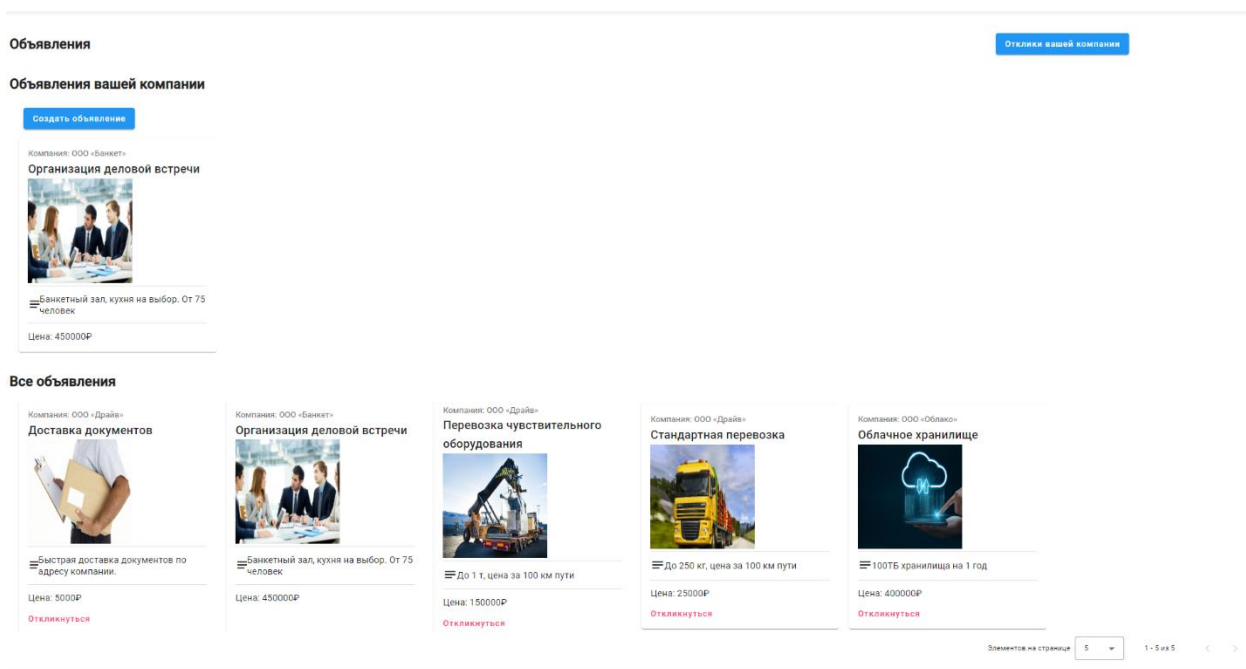


Рисунок 18 – Страница объявлений

Карточка объявления, изображенная на рисунке 19, состоит из наименования компании-продавца, фотографии товара или услуги, описания и цены. Для отклика на объявление необходимо нажать на кнопку откликнуться.

Компания: ООО «Облако»

Облачное хранилище



≡ 100ТБ хранилища на 1 год

Цена: 400000₽

Откликнуться

Рисунок 19 – Карточка объявления

При успешном отклике на объявление отображается уведомление, представленное на рисунке 20.

Цена: 25000₽

Откликнуться

Цена: 400000₽

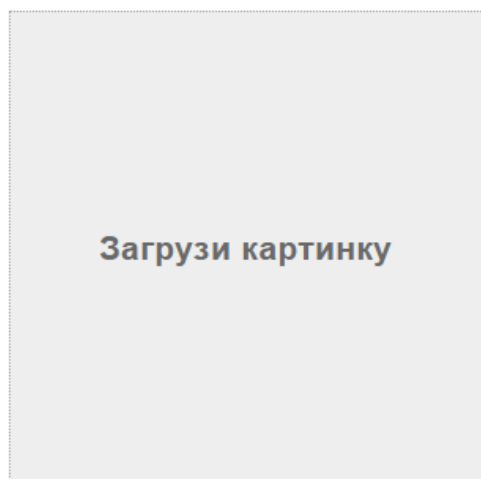
Откликнуться

Вы успешно откликнулись на объявление

Рисунок 20 – Уведомление об успешном отклике на объявление

Также на странице можно разместить новое объявление от имени своей компании, нажав на кнопку “Создать объявление” и заполнив информацию в диалоговом окне, представленном на рисунке 21.

Объявление



Название товара*

Отрасль*

Цена*

Описание

Выйти

Разместить на торговой площадке

Рисунок 21 – Создание объявления

При нажатии на кнопку “Отклики вашей компании” пользователя перенаправляет на страницу с объявлениями, на которые его компания уже откликнулась. Страница представлена на рисунке 22.

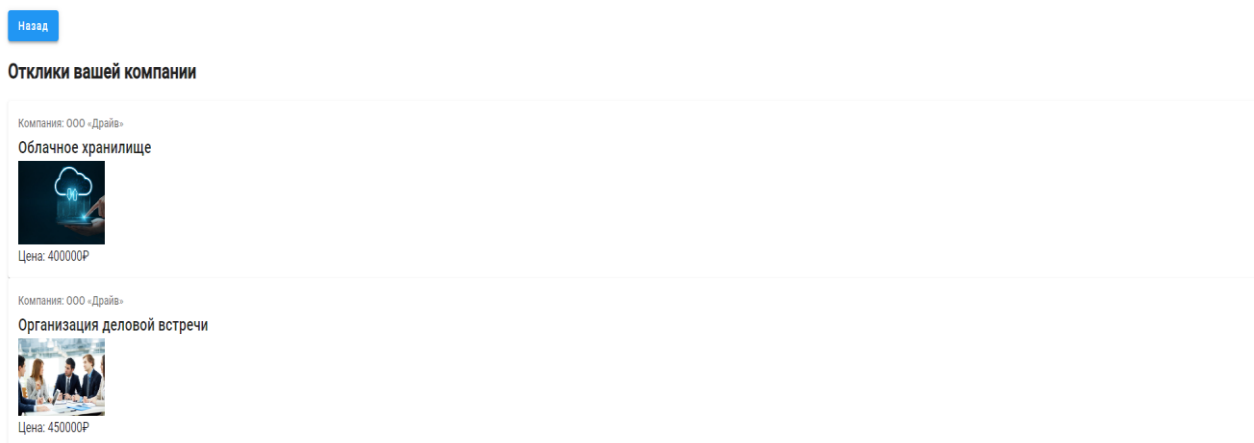


Рисунок 22 – Страница откликов компании

3.7 Сделки

Сделки в клиентской части CRM-системы представлены в виде доски по статусам, которыми они могут обладать. Страница изображена на рисунке 23. Сделка может находиться в следующих статусах:

- 1) новая сделка — начальный статус сделки, говорит о том, что сотрудники не рассматривали созданную сделку;
- 2) установка контакта — ответственный за сделку связывается с клиентом;
- 3) обсуждение деталей — контакт установлен, обе стороны договариваются о различных ситуациях, возникновение которых возможно в процессе выполнения сделки;
- 4) заключительные детали — большая часть сделки со стороны поставщика выполнена, обсуждаются нюансы, например сроки оплаты и поставки;
- 5) завершено — заключительный этап, означает, что обе стороны выполнили свои обязательства и сделка закрыта.

НОВАЯ СДЕЛКА	УСТАНОВКА КОНТАКТА	ОБСУЖДЕНИЕ ДЕТАЛЕЙ	ЗАКЛЮЧИТЕЛЬНЫЕ ДЕТАЛИ	ЗАВЕРШЕНО	ОТКЛОНЕНО
	Перевозка сервера для компании «Банкет» Цена: 100000Р Жёсткие диски, серверное оборудование Редактировать	Перевозка оборудования для компании «Банкет» Цена: 300000Р Мебель, интерьер, около 200 кг Редактировать		Перевозка кабелей и оборудования для коммутации Цена: 190000Р Около 150 кг Редактировать	

Рисунок 23 – Доска сделок

Важно отметить, что созданную сделку нельзя удалить, но существует возможность перевести её в один из заключительных этапов: «Завершено» или «Отклонено». При подтверждении закрытия сделки со стороны клиента рейтинг компании повышается.

Для изменения статуса сделки и информации о ней пользователь может воспользоваться формой на рисунке 24.

Сделка

Название товара*	Деловая встреча компании «Драйв»
Статус	Завершено
Ответственный	Захарова Мария
Цена	500000
Описание	Банкет на 90 человек

Компания-клиент: ООО «Банкет»

Ответственный: Панов Григорий

[Добавить в контакты](#)[Выйти](#) [Сохранить](#)

Рисунок 24 – Окно редактирования сделки

3.8 Уведомления

На странице уведомлений пользователь может ознакомиться с произошедшими в системе событиями. Страница уведомлений представлена на рисунке 25.

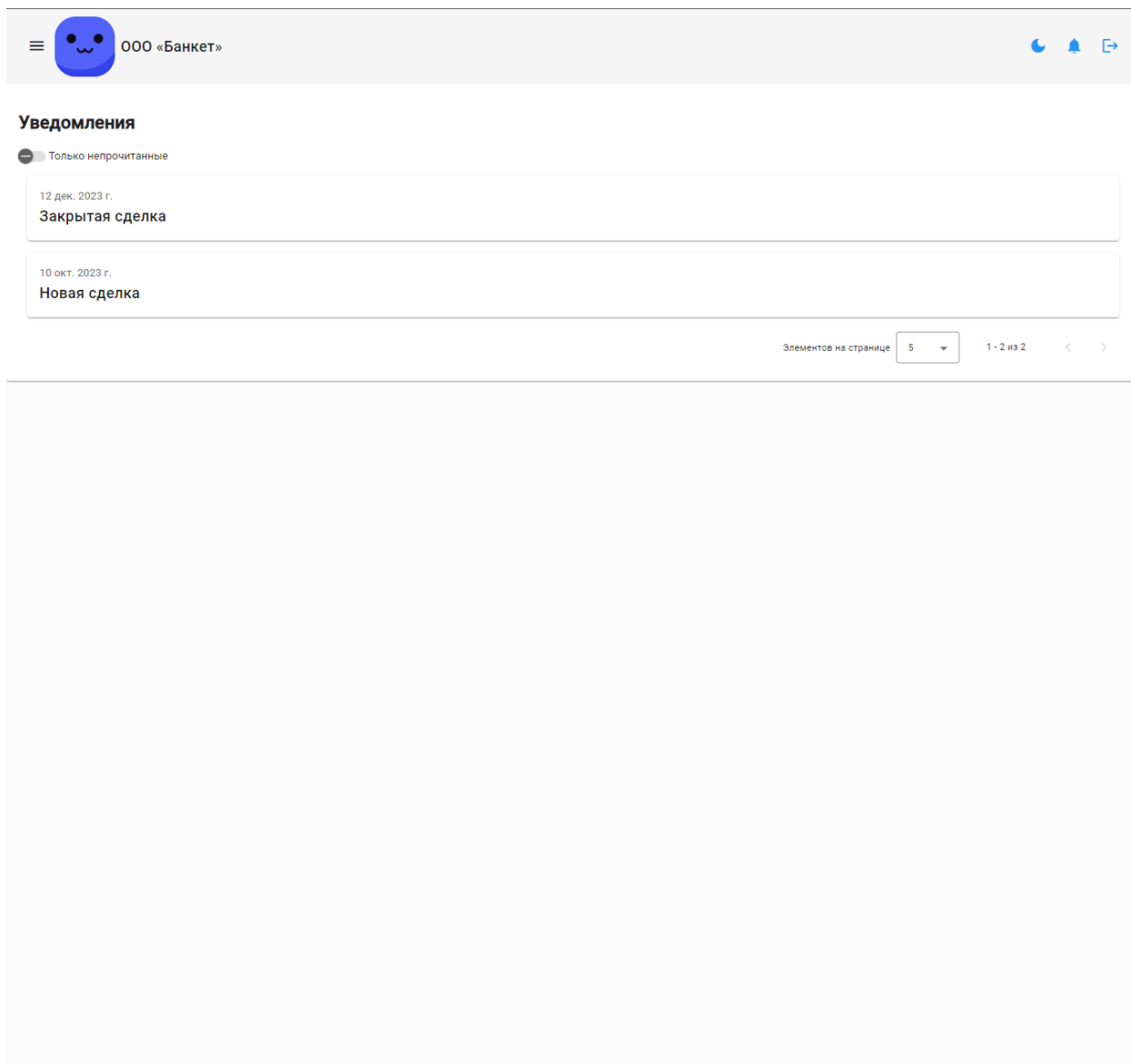


Рисунок 25 – Уведомления

Для удобства пользователя реализована возможность фильтрации уведомлений по непрочитанным.

В диалоговом окне уведомления компания-клиент может либо подтвердить закрытие сделки, то есть поставщик честно отработал и заслуживает повышения рейтинга, либо не подтвердить, и тогда поставщика

ждёт понижение рейтинга за мошеннические действия. Интерфейс данного окна представлен на рисунке 26.

Уведомление

12 дек. 2023 г.

Закрытая сделка

Компания «ООО «Банкет»» закрыла сделку по объявлению «Организация деловой встречи».

Вы подтверждаете закрытие сделки? Отменить это действие нельзя.

Подтвердить

Закрыть

Рисунок 26 – Уведомление о закрытой сделке

Для уведомления об отклике на объявление вашей компании открывается другое диалоговое окно, представленное на рисунке 27.

Уведомление

10 окт. 2023 г.

Новая сделка

Компания «ООО «Драйв»» откликнулась на ваше объявление «Организация деловой встречи». На её отклик была создана сделка.

Закрыть

Рисунок 27 – Уведомление об отклике на объявление

ЗАКЛЮЧЕНИЕ

В результате выполнения дипломной работы разработана CRM-система с функционалом торговой площадки. С помощью этой системы компании могут взаимодействовать друг с другом, продавая услуги и товары, что подтверждает B2B направленность разработанной системы. Приложение позволяет проводить сделки, обеспечивать коммуникацию между компаниями, обладает возможностью сохранения и последующего анализа статистики взаимодействия компаний с клиентами, что и является основным и определяющим функционалом для CRM-системы.

Интерфейс приложения разработан с учётом всего необходимого функционала и удобства для целевых пользователей. Он обладает рядом ключевых характеристик, таких как интуитивная навигация, четкая структура, понятный дизайн и эргономичное расположение элементов управления. Плавность взаимодействия с приложением и его быстроедействие достигается благодаря одностраничному подходу к реализации.

Благодаря многоуровневой архитектуре фронтенда, веб-приложение открыто для расширения и совершенствования. Изменения в структуре данных и объектах на сервере не опасны для клиентской части, так как изменения затронут лишь слой абстракции данных.

Весь запланированный функционал приложения был реализован.

Благодаря контейнеризации наша система может быть легко и быстро развернута для применения в любой компании.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Полное руководство по CRM-системам. — URL: <https://www.oracle.com/cis/cx/what-is-crm/> (дата обращения 11.01.2024).
2. CRM-система — что это и как работает. — URL: <https://megaplan.ru/news/articles/что-такое-crm-sistema/> (дата обращения 09.01.2024).
3. Шьям Шешадри. Angular: Up and Running: Learning Angular, Step by Step. // O'Reilly Media, Inc., 2018. – 300 с
4. Стефан Баумгартнер. TypeScript Cookbook. // O'Reilly Media, Inc., 2023. – 768 с
5. Angular и его преимущества. — URL: <https://habr.com/ru/companies/otus/articles/746076/> (дата обращения 23.04.2024).
6. Single Page Application: что это такое и как работает. — URL: <https://thecode.media/spa/> (дата обращения 24.04.2024).
7. What Is a SPA? Architecture, Benefits, and Challenges. — URL: <https://www.spiceworks.com/tech/devops/articles/what-is-single-page-application/> (дата обращения 26.04.2024).
8. Что такое RxJS и почему о нём полезно знать. — URL: <https://habr.com/ru/companies/ruvds/articles/341880/> (дата обращения 26.04.2024).
9. Что такое Docker? — URL: <https://www.oracle.com/cis/cloud/cloud-native/container-registry/what-is-docker/> (дата обращения 03.05.2024).
10. What Is TypeScript? — URL: <https://thenewstack.io/what-is-typescript/> (дата обращения 07.05.2024).
11. Why does TypeScript exist? — URL: <https://www.typescriptlang.org/why-create-typescript/> (дата обращения 12.05.2024).
12. What Is Angular Material. — URL: <https://upstackhq.com/blog/software-development/what-is-angular-material> (дата обращения 17.05.2024)

ПРИЛОЖЕНИЕ А
QR-код репозитория с исходным кодом



ПРИЛОЖЕНИЕ Б

Методы HTTP-запросов

Таблица 1 – Методы HTTP-запросов

Метод	Краткое описание
OPTIONS	<p>Используется для определения возможностей веб-сервера или параметров соединения для конкретного ресурса. Предполагается, что запрос клиента может содержать тело сообщения для указания интересующих его сведений. Формат тела и порядок работы с ним в настоящий момент не определён. Сервер пока должен его игнорировать. Аналогичная ситуация и с телом в ответе сервера.</p> <p>Для того чтобы узнать возможности всего сервера, клиент должен указать в URI звёздочку — «*».</p>
GET	<p>Используется для запроса содержимого указанного ресурса. С помощью метода GET можно также начать какой-либо процесс. В этом случае в тело ответного сообщения следует включить информацию о ходе выполнения процесса. Клиент может передавать параметры выполнения запроса в URI целевого ресурса после символа «?»: GET /path/resource?param1=value1&m2=value2 HTTP/1.1</p> <p>Согласно стандарту HTTP, запросы типа GET считаются идемпотентными — многократное повторение одного и того же запроса GET должно приводить к одинаковым результатам (при условии, что сам ресурс не изменился за время между запросами). Это позволяет кэшировать ответы на запросы GET.</p>

HEAD	<p>Аналогичен методу GET, за исключением того, что в ответе сервера отсутствует тело. Запрос HEAD обычно применяется для извлечения метаданных, проверки наличия ресурса (валидация URL) и чтобы узнать, не изменился ли он с момента последнего обращения.</p> <p>Заголовки ответа могут кэшироваться. При несовпадении метаданных ресурса с соответствующей информацией в кэше копия ресурса помечается как устаревшая.</p>
POST	<p>Применяется для передачи пользовательских данных заданному ресурсу. Например, в блогах посетители обычно могут вводить свои комментарии к записям в HTML-форму, после чего они передаются серверу методом POST и он помещает их на страницу. При этом передаваемые данные (в примере с блогами — текст комментария) включаются в тело запроса. Аналогично с помощью метода POST обычно загружаются файлы.</p> <p>В отличие от метода GET, метод POST не считается идемпотентным, то есть многократное повторение одних и тех же запросов POST может возвращать разные результаты (например, после каждой отправки комментария будет появляться одна копия этого комментария).</p> <p>При результатах выполнения 200 (Ok) и 204 (No Content) в тело ответа следует включить сообщение об итоге выполнения запроса. Если был создан ресурс, то серверу следует вернуть ответ 201 (Created) с указанием URI нового ресурса в заголовке Location.</p>

	<p>Сообщение ответа сервера на выполнение метода POST не кэшируется.</p>
PUT	<p>Применяется для загрузки содержимого запроса на указанный в запросе URI. Если по заданному URI не существовало ресурса, то сервер создаёт его и возвращает статус 201 (Created). Если же был изменён ресурс, то сервер возвращает 200 (Ok) или 204 (No Content). Сервер не должен игнорировать некорректные заголовки Content-* передаваемые клиентом вместе с сообщением. Если какой-то из этих заголовков не может быть распознан или не допустим при текущих условиях, то необходимо вернуть код ошибки 501 (Not Implemented).</p> <p>Фундаментальное различие методов POST и PUT заключается в понимании предназначений URI ресурсов. Метод POST предполагает, что по указанному URI будет производиться обработка передаваемого клиентом содержимого. Используя PUT, клиент предполагает, что загружаемое содержимое соответствует находящемуся по данному URI ресурсу.</p>

	Сообщения ответов сервера на метод PUT не кэшируются.
PATCH	Аналогично PUT, но применяется только к фрагменту ресурса.
DELETE	Удаляет указанный ресурс.
TRACE	Возвращает полученный запрос так, что клиент может увидеть, что промежуточные сервера добавляют или изменяют в запросе.
LINK	Устанавливает связь указанного ресурса с другими.
UNLINK	Убирает связь указанного ресурса с другими.