Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

|  |
| --- |
| Институт информационных технологий и анализа данных |
| наименование института |

|  |  |
| --- | --- |
| Допускаю к защите |  |
| Руководитель | подпись |
|  | Л.С. Вахрушева |
|  | И.О. Фамилия |

|  |
| --- |
| Разработка программного обеспечения |
| для учёта и анализа частотно-временной информации |

наименование темы

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту по дисциплине

|  |  |  |
| --- | --- | --- |
| Технологии разработки программных комплексов | | |
|  | 1.13.00.00 - ПЗ |  |

обозначение документа

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил студент |  |  |  |  |  | А.И, Ханхаев |
|  |  | шифр группы |  | подпись |  | И.О. Фамилия |
| Нормоконтроль |  |  |  |  |  | Л.С. Вахрушева |
|  |  |  |  | подпись |  | И.О. Фамилия |

Курсовой проект защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Иркутск 2024 г.

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

ЗАДАНИЕ

НА КУРСОВОЕ ПРОЕКТИРОВАНИЕ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| По курсу | Технологии разработки программных комплесов | | | |
| Студенту |  | | | |
|  | (фамилия, инициалы) | | | |
| Тема проекта | | Разработка CRM системы для аренды жилья | | |
|  | | | | |
| Исходные данные: | | |  | |
| ???? | | | | |
|  | | | | |
| |  |  |  | | --- | --- | --- | | Рекомендуемая литература: |  | | | 1. Гутгарц Р.Д Проектирование автоматизированных систем обработки информации и управления: учебное пособие для академического бакалавриата. – М.: Издательство Юрайт, 2019. | | | 1. Проектирование АСОИУ [Электронный ресурс] : методические указания по выполнению курсового проекта: 09.03.01 «Информатика и вычислительная техника» для бакалавров по специальности «Автоматизированные системы обработки информации и управления» / Иркут. нац. исслед. техн. ун-т ; сост. Р. Д. Гутгарц. - Электрон. дан. - Иркутск : ИРНИТУ, 2018. | | | 1. Проектирование информационных систем : учебник и практикум для академического бакалавриата / Д. В. Чистов, П. П. Мельников, А. В. Золотарюк, Н. Б. Ничепорук ; под общ. ред. Д. В. Чистова. – М. : Издательство Юрайт, 2016. | | | 1. Рудинский И.Д. Технология проектирования автоматизированных систем обработки информации и управления : учебное пособие для вузов / И. Д. Рудинский. - Москва : Горячая линия - Телеком, 2011. | | | | | |  |

Графическая часть на \_\_\_\_\_\_\_\_\_\_\_\_\_\_ листах.

Дата выдачи задания « 20 » сентября 2024 г.

|  |  |  |  |
| --- | --- | --- | --- |
| Задание получил |  |  |  |
|  | подпись |  | И.О. Фамилия |

Дата представления проекта руководителю 2024 г.

|  |  |  |  |
| --- | --- | --- | --- |
| Руководитель курсового проектирования |  |  | Л.С.Вахрушева |

**Введение**

Рынок жилой недвижимости является одной из наиболее динамично развивающихся отраслей экономики, в которой участвуют множество игроков: застройщики, риэлторы и конечные покупатели. Каждый из них сталкивается с уникальными задачами и вызовами. Застройщики стремятся продвигать свои объекты, риэлторы — искать и предлагать актуальные варианты своим клиентам, а покупатели — находить подходящее жильё в огромном объёме данных.

На текущий момент значительная часть информации о продаже квартир размещается застройщиками в виде Google-таблиц. В этих таблицах содержатся данные о свободных квартирах, их характеристиках (площадь, этаж, планировка) и статусах продаж. Однако использование таких таблиц в отрыве друг от друга приводит к проблемам в работе всех участников рынка.

Существующая разрозненность и неструктурированность данных создаёт сложности, которые можно устранить с помощью автоматизированной системы. Она не только упрощает процессы управления информацией, но и повышает эффективность взаимодействия между застройщиком, риэлтором и клиентом.

**Цель проекта:**  
Создать веб-платформу (CRM-систему), которая объединяет данные о продаже квартир, упрощает их обработку и обновление, а также делает взаимодействие всех участников рынка недвижимости более удобным.

**Задачи для достижения цели:**

1. Автоматизировать процесс сбора и обновления данных из Google-таблиц застройщиков.
2. Обеспечить риэлторам удобный инструмент для поиска и публикации актуальной информации.
3. Разработать функционал для покупателей, позволяющий легко находить подходящие квартиры.
4. Повысить точность и оперативность обновления данных о статусе продаж.
5. Обеспечить застройщикам эффективное продвижение их объектов через единую платформу.

**1. Анализ**

**1.1. Описание предметной области**

**Рынок жилой недвижимости и его участники**  
Рынок недвижимости включает в себя три ключевые группы участников:

* **Застройщики** создают объекты недвижимости и размещают информацию о них для продажи. Их основная задача — привлечение внимания риэлторов и покупателей.
* **Риэлторы** выступают посредниками между застройщиками и покупателями, публикуя актуальную информацию о квартирах и помогая клиентам сделать выбор.
* **Покупатели** ищут подходящее жильё, ориентируясь на доступные данные о квартирах, их характеристиках и ценах.

Основная проблема заключается в разрозненности информации: застройщики ведут свои данные в Google-таблицах, которые не связаны между собой и требуют ручного обновления риэлторами.

**Проблемы текущей ситуации**

1. **Для застройщиков:**
   * Ограниченные возможности распространения информации.
   * Риск потери потенциальных клиентов из-за устаревших данных.
2. **Для риэлторов:**
   * Высокие временные затраты на проверку и обновление данных.
   * Сложности при поиске и публикации актуальных объектов.
3. **Для покупателей:**
   * Трудности в поиске жилья из-за отсутствия единой базы данных.
   * Отсутствие уверенности в актуальности представленной информации.

**1.2. Анализ существующих решений**

Существующие решения для управления данными о продаже квартир зачастую не охватывают весь спектр потребностей участников рынка. Наиболее популярные методы:

* **Google-таблицы застройщиков.** Удобны для внутреннего использования, но плохо масштабируются.
* **Крупные онлайн-платформы.** Такие сайты, как Avito и ЦИАН, требуют сложной интеграции и предоставляют риэлторам ограниченные инструменты для управления информацией.
* **CRM-системы для риэлторов.** Многие из них требуют ручного ввода данных и не синхронизируются с источниками информации от застройщиков.

**1.3. Проблема и актуальность разработки**

Разрозненность информации и неэффективность её обработки приводят к потере времени, увеличению стоимости продаж и неудовлетворённости клиентов. Проект предлагаемой CRM-системы станет решением этой проблемы. Он автоматизирует процесс сбора и обработки данных, создавая удобный инструмент для застройщиков, риэлторов и покупателей.

**Актуальность разработки** обусловлена следующими факторами:

* Рынок недвижимости продолжает расти, что увеличивает объём данных для обработки.
* Застройщики и риэлторы стремятся к автоматизации своих процессов для повышения конкурентоспособности.
* Покупатели ожидают прозрачности и удобства при поиске жилья.

**Заключение:**  
Разработка данной CRM-системы позволит устранить существующие проблемы, повысить эффективность работы участников рынка недвижимости и создать удобный инструмент для взаимодействия всех сторон.

**2. Обзор существующих программных средств**

**2.1. Общий обзор текущих решений**

На рынке существуют различные программные средства, направленные на поддержку продаж недвижимости. Эти решения можно разделить на три основные категории:

1. **Крупные онлайн-платформы для размещения объявлений.**  
   Примеры: Avito, ЦИАН. Они позволяют застройщикам, риэлторам и частным лицам публиковать объявления о продаже недвижимости, предоставляя доступ к широкой аудитории покупателей.
2. **CRM-системы для риэлторов.**  
   Примеры: AmoCRM, Pipedrive, Bitrix24. Такие системы ориентированы на управление продажами, но требуют ручного ввода данных, что снижает их эффективность в работе с большим количеством объектов.
3. **Собственные решения застройщиков.**  
   Застройщики часто создают Google-таблицы или используют внутренние платформы для хранения информации о своих объектах. Эти решения упрощают управление данными внутри компании, но плохо подходят для совместной работы с риэлторами и клиентами.

**2.2. Анализ функциональности Avito**

Avito — крупнейшая платформа для размещения объявлений о продаже недвижимости. Она охватывает широкий круг пользователей: от частных продавцов до крупных застройщиков.

**Преимущества Avito:**

1. **Широкая аудитория.**
   * Ежемесячно платформу посещают миллионы пользователей.
   * Подходит для продвижения объектов на массовом рынке.
2. **Удобство для покупателей.**
   * Простая навигация, фильтры для поиска по цене, площади, району и другим параметрам.
   * Возможность прямого контакта с продавцом.
3. **Рекламные инструменты.**
   * Продвижение объявлений через платные услуги, такие как "VIP-размещение".
   * Статистика просмотров объявлений.

**Ограничения Avito:**

1. **Нет автоматизации работы с данными.**
   * Застройщики и риэлторы должны вручную загружать данные о квартирах и обновлять их статус. Это затрудняет управление объектами с высоким уровнем динамики продаж.
2. **Отсутствие специализированного функционала для застройщиков.**
   * Платформа ориентирована на размещение отдельных объявлений, а не на управление большими портфелями объектов.
3. **Платность услуг.**
   * Использование большинства инструментов требует оплаты, что может быть затратным для застройщиков с большим количеством объектов.
4. **Сложности с актуализацией информации.**
   * Если статус объекта изменился (например, квартира продана), обновление данных занимает время, что может вводить клиентов в заблуждение.

**3. Процесс TO BE**

**3.1.1. Импорт данных от застройщиков**

**Описание:**

Застройщики загружают данные о квартирах через:

* + API интеграцию с Google Sheets.
  + Загрузку Excel-файлов с предопределённой структурой.

**Этапы:**

1. Застройщик вводит ссылку на Google-таблицу или загружает файл.
2. Система проверяет данные на ошибки и уведомляет застройщика об успехе или необходимости исправлений.
3. Информация синхронизируется с базой данных и становится доступной для поиска клиентами.

**3.1.2. Поиск квартир**

**Описание:**

Клиенты вводят параметры поиска (цена, район, метраж, этаж и т.д.). Система обрабатывает запросы, выводя списки объектов и отображая их на карте.

**Этапы:**

1. Клиент задаёт параметры фильтра на главной странице (например, цена от 5 до 10 млн, наличие стеклопакетов).
2. Система отображает список квартир, соответствующих критериям.
3. Результаты визуализируются на карте для удобства выбора.

**3.1.3. Взаимодействие с клиентами**

**Описание:**

После выбора квартиры клиенты могут оставить заявку на просмотр или задать вопрос через встроенный мессенджер. Менеджеры получают заявки и фиксируют их в CRM.

**Этапы:**

1. Клиент нажимает на кнопку "Оставить заявку".
2. Менеджер получает уведомление в личном кабинете.
3. Через мессенджер уточняются детали встречи, и менеджер фиксирует статус клиента (лид, договор, продажа).

**3.1.4. Администрирование данных застройщика**

**Описание:**

Застройщики управляют своими проектами и объектами недвижимости, обновляют данные о квартирах, ведут работу с клиентами и настраивают рекламные кампании.

**Этапы:**

1. Застройщик входит в свою административную панель.
2. Добавляет или редактирует проекты (описание, характеристики квартир, статус продаж).
   1. Загружает данные об объектах недвижимости через Google-таблицы, Excel или вручную.
3. Настраивает параметры фильтров (например, площадь, этаж, наличие стеклопакетов).
4. Управляет рекламными кампаниями:
   1. Настраивает баннеры или продвигает проекты на платформе.
   2. Отслеживает эффективность рекламы через аналитику.
5. Работает с клиентами:
   1. Обрабатывает заявки.
   2. Ведёт переговоры через встроенный мессенджер.
   3. Переводит клиентов между статусами ("лид", "заключена сделка").
6. Анализирует статистику по проектам и продажам.

**3.1.5. Аналитика и отчётность**

**Описание:**

Система анализирует данные о продажах, активности пользователей и эффективности застройщиков, предоставляя отчёты в удобной форме.

**Этапы:**

1. Система собирает данные о запросах, просмотрах и продажах.
2. Генерирует отчёты для застройщиков, показывая эффективность проектов.
3. На основе аналитики застройщики могут скорректировать свои стратегии (например, изменить цены или добавить больше описаний).

**3.2.** **Схема TO BE**

Графическое представление функциональной структуры системы для клиента представлено на рисунке 1.

**Изображение выглядит как снимок экрана, диаграмма, линия, текст

Автоматически созданное описание**

Рисунок 1. Схема TO BE для клиента.

Графическое представление функциональной структуры системы для клиента представлено на рисунке 2.

Изображение выглядит как снимок экрана, текст, диаграмма, линия

Автоматически созданное описание

Рисунок 2. Схема TO BE для застройщика.

**4.Описание вариантов использования**

Диаграмма прецедентов, отражающая функциональные требования к системе представлена на рисунках 3 и 4

Изображение выглядит как текст, диаграмма, рисунок, шаблон

Автоматически созданное описание

Рисунок 3. Диаграмма прецедентов

Изображение выглядит как текст, диаграмма, рисунок

Автоматически созданное описание

Рисунок 4. Диаграмма прецедентов

**5.Основные функциональные требования**

**5.1.1 Регистрация и авторизация пользователей:**

* Пользователи (застройщики, менеджеры, клиенты, администраторы) должны иметь возможность регистрироваться на платформе, заполняя личные данные, подтверждая регистрацию через email или телефон.
* Платформа должна поддерживать восстановление пароля через email или SMS, а также распределение ролей (застройщик, менеджер, клиент, администратор) с соответствующими правами доступа.

**5.1.2 Управление объектами недвижимости:**

* Застройщики должны иметь возможность создавать, редактировать и удалять проекты и объекты недвижимости, указывая такие параметры, как площадь, этаж, цена, статус (доступно, продано) и дополнительные характеристики (например, наличие стеклопакетов).
* Система должна поддерживать автоматическую загрузку данных из Google-таблиц или Excel-файлов.
* Данные о недвижимости должны отображаться в виде удобного фильтра для клиентов и визуализироваться на карте.

**5.1.3 Работа с клиентами:**

* Менеджеры должны иметь возможность управлять списками клиентов (лиды, договоры, продажи) и назначать встречи или переговоры через встроенный календарь.
* Платформа должна поддерживать встроенный мессенджер для общения с клиентами, включая возможность прикрепления файлов и ведения истории переписки.
* Система должна позволять менеджерам передавать клиентов между собой (с согласованием или без).

**5.1.4 Рекламные кампании и продвижение:**

* Застройщики должны иметь возможность настраивать рекламные кампании, включая загрузку баннеров и отслеживание их эффективности через встроенную аналитику.
* Администратор должен управлять глобальными рекламными блоками и регулировать их показ.

**5.1.5 Сбор и анализ статистики:**

* Система должна собирать статистику по активности застройщиков, эффективности их продаж и взаимодействия с клиентами.
* Для застройщиков и администраторов должна быть доступна аналитика по проектам, в том числе рейтинги популярности квартир, данные о продажах и статистика просмотров.
* Система должна предоставлять отчёты застройщикам и администраторам в удобной форме (графики, диаграммы, таблицы).

**5.2 Нефункциональные требования**

**5.2.1 Интуитивный интерфейс:**

* Система должна быть простой в использовании для всех типов пользователей, с минималистичным интерфейсом и доступной навигацией.

**5.2.2 Безопасность данных:**

* Платформа должна обеспечивать защиту персональных данных пользователей (клиентов, застройщиков, менеджеров), включая шифрование паролей и конфиденциальных данных, защиту от SQL-инъекций и DDoS-атак.

**5.2.3 Производительность:**

* Система должна обеспечивать обработку большого объёма данных (множество проектов, пользователей и клиентов) без значительных задержек.

**5.2.4 Масштабируемость:**

* Платформа должна быть готова к расширению функционала, поддержке большего количества пользователей и данных без необходимости полной переработки системы.

**5.3 Постановка задачи**

На основе сформированных требований задача проекта заключается в создании единой информационной платформы, которая позволит:

* Объединить информацию о продаже недвижимости в одном месте для удобного поиска и анализа клиентами;
* Автоматизировать процессы загрузки данных о недвижимости (Google-таблицы, Excel) для застройщиков, а также упрощённое управление объектами;
* Внедрить систему фильтров для быстрого поиска подходящей недвижимости клиентами;
* Предоставить аналитические данные и статистику по продажам и рекламным кампаниям для застройщиков и администраторов;
* Обеспечить безопасное хранение данных, включая защиту персональной информации клиентов, менеджеров и застройщиков.

**6.Выбор и обоснование средств проектирования и реализации**

Для реализации проекта был выбран стек технологий, который соответствует современным стандартам разработки, а также обеспечивает гибкость, безопасность и производительность.

Средства реализации:

**Back-end**

PHP Laravel

Laravel - популярный PHP-фреймворк, который предоставляет удобные инструменты для создания сложных приложений, включая встроенные механизмы работы с базами данных, роутинг и поддержку REST API.

Обоснование выбора:

1. Легкость в разработке и поддержке благодаря читаемому коду и интуитивным интерфейсам.
2. Наличие встроенных решений, таких как ORM (Eloquent), система миграций и middleware, что ускоряет процесс разработки.
3. Большая документация и активное сообщество, что важно для решения возникающих задач.
4. Большое количество сайтов на данном языке, что дает вам большую ценность на рынке специалистов в будущем.

PostgreSQL

PostgreSQL - мощная проверенная объектно-реляционная система управления базами данных.

Обоснование выбора:

1. Поддержка сложных запросов и транзакций.
2. Надёжность, безопасность и высокая производительность для работы с большими объёмами данных.
3. Расширяемость и поддержка JSON для хранения структурированных данных.

Nginx + Ubuntu

Nginx – проверенный популярный веб-сервер, который также может быть использован как обратный прокси-сервер, так же и Ubuntu — одна из самых популярных и относительно простых серверных операционных систем.

Обоснование выбора:

1. Nginx обеспечивает высокую производительность и надёжность для обработки большого числа одновременных запросов.
2. Ubuntu предоставляет стабильную, безопасную и удобную среду для запуска приложений в купе с простотой использования и наличием графической оболочкой, идеальный вариант для начинающего веб-разработчика.

**Front-end**

React

React - библиотека JavaScript для построения пользовательских интерфейсов, особенно эффективная для одностраничных приложений (SPA).

Обоснование выбора:

1. Высокая производительность за счёт виртуального DOM.
2. Удобная интеграция с REST API, что соответствует потребностям проекта.
3. Широкий набор библиотек и компонентов, ускоряющий разработку.
4. Проверенная библиотека с большим количеством обучающих ресурсов, что упрощает обучение.

**Авторизация**

JWT (JSON Web Token)

JWT — это стандарт токенов для передачи данных между клиентом и сервером.

Обоснование выбора:

1. Лёгкость интеграции с REST API.
2. Безопасность и возможность работы в безсессионном режиме, что улучшает производительность.
3. Простота масштабирования, особенно для SPA.
4. Распространенность, благодаря чему уже готовые инструменты работы со страндартом.

**7.Проектирование архитектуры приложения**

На рисунке 1 представлена схема архитектуры проекта

**Изображение выглядит как текст, снимок экрана, диаграмма, дизайн

Автоматически созданное описание**

Рис. 1 – схема архитектуры системы

Схема включает в себя несколько компонентов:

* Front-end. Клиентская часть веб-приложения, разработанная с помощью библиотеки React на языке JS и сверстанная на HTML и CSS. Взаимодействие с сервером происходит через протокол http с проверкой авторизации по стандарту JWT.
* Back-end. Серверная часть веб-приложения, разработанная с помощью фреймворка Laravel на языке PHP, использующая для взаимодействия с Front-end стиль REST API. В качестве СУБД используется PostgreSQL, все это размещено на веб-сервера NGINX, который в свою очередь запущен в окружении Ubuntu.

**8.Проектирование хранилища данных**

На рисунке 2 представлена инфологическая модель базы данных

Изображение выглядит как текст, число, снимок экрана, диаграмма

Автоматически созданное описание

Рис. 2 – схема БД.

|  |  |
| --- | --- |
| **Сущность** | **Назначение** |
| **Users** | Хранение основных данных о пользователе сервиса. |
| **Roles** | Хранение данных о существующих ролях в сервисе, для ограничения прав пользователя. |
| **UserInfo** | Хранение второстепенных данных о пользователе сервиса. |
| **UserContacts** | Хранение контактных данных пользователя |
| **PropertyBuilders** | Хранение основных данных о застройщиках |
| **Projects** | Хранение данных и проектах застройщиков |
| **Apartments** | Хранение данных об квартирах построенных в рамках проекта |
| **ApartmentsImageURLs** | Хранение ссылок на изображение квартир благодаря которым можно найти изображение в бд |
| **AdvertisingDashboard** | Хранение данных о рекламном положении застройщика в сервисе |
| **FeedBack** | Хранение отзывов пользователей сервиса об застройщиках |

**9.Проектирование пользовательского интерфейса**

На рисунке 3 представлена схема страниц веб-приложения

**Изображение выглядит как текст, диаграмма, Параллельный, снимок экрана

Автоматически созданное описание**

Рис. 3 – иерархическая схема страниц

**Назначение экранов**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № экрана | |  | | --- | | Краткое название |  |  | | --- | |  | | |  | | --- | | Поля ввода для валидации |  |  | | --- | |  | | Описание экрана и его поведения |
| 1 | Вход | Email, password | Экран с двумя полями ввода пользовательских данных для авторизации. В случае корректного ввода перенаправляет на главную страницу, в случае некорректного выдает предупреждение о неправильном вводе. |
| 2.1 | Профиль | - | Страница с основными данными пользователя, отображает его аватар, почту и личные данные. |
| 2.2 | Настройки профиля | Email, личные данные, вставка изображения для аватара | Страница с полями для ввода новых данных профиля для его обновления, так же запрашивает пароль для обновления данных. Так же можно удалить аккаунт |
| 2.3 | Отзывы о застройщиках | Текст отзыва, название застройщика, оценка. | Страница с возможностью просмотра всех своих отзывов, так же можно написать отзыв, оставить 5-ти бальную оценку кампании и написать причину такой оценки. |
| 2.4 | Список желаемого | - | Страница просмотра всех квартир которые клиент рассматривает для аренды. На странице отображается их список с основными данными о них, так же можно удалять их нажатием одной кнопки. Можно переходить на полные страницы квартиры по клику по квартире из списка. |
| 2.5 | Контакты | Сообщение, поиск контакта | Страница со списком пользователей с которым вы ведете переписку внутри сайта, так же если контактов слишком много можно искать их в списке контактов. При выборе контакта открывается ваша переписка с ним. |
| 2.6 | Уведомления | - | Страница со списком уведомлений с краткой информацией о них. При клике по уведомлению вы увидите расширенную версию уведомления. Уведомления можно удалять и закреплять. |
| 3.1 | Профиль компании | - | Страница-визитка компании которая предоставляет свои аренд-услуги на сайте. На странице отображаются данные о компании и их собственное заполнение информации о себе. Их можно закрепить. Так же можно посмотреть список их проектов. |
| 3.2 | Статистика | - | Страница с статистикой продаж квартир, которой пользуется застройщик, для аналитики своей деятельности, на экране отображены разные графики и численные показатели а так же статистика сделок. |
| 3.3 | Сделки | Дата сделки, описание сделки, состояние сделки | Страница для создания сделок а так же навигации по другим сделкам. Отображается список всех сделок с краткой информацией по ним. При клике по определенной сделке открывается ее расширенная мини страничка с полным ее представлением. |
| 3.4 | Контакты | Сообщение, поиск контакта | Страница со списком пользователей с которым вы ведете переписку внутри сайта, так же если контактов слишком много можно искать их в списке контактов. При выборе контакта открывается ваша переписка с ним. |
| 3.5 | Проекты | Название проекта | Страница с списком всех проектов с основной небольшой информацией по ним, при клике по определенному проекту открывается полная информация по нему |
| 3.6 | Рекламный кабинет | Выбор квартиры для рекламы | Страница с возможностью купить рекламу на главной странице сайта, так же просмотр статистики по рекламе, пополнение баланса, выбор квартиры для рекламы, или выбор проекта |
| 3.7 | Квартира | - | Страница с полной информацией о квартире. Ее можно добавить в желаемое, так же через нее можно перейти к кампании застройщику. |
| 3.8 | Кабинет застройщика | - | Личная страница компании застройщика, где можно редактировать данные о компании и просматривать все недоступные другим пользователям данные. Через нее проводится доступ к другим страницам таким как Контакты, Сделки, Статистика, Проекты, Рекламный кабинет. |
| 4.1 | Квартиры | Стоимость, адрес, кол-во комнат, тип ремонта, тип аренды, этаж, название проекта. | Страница со списком всех актуальных квартир. Так же есть возможность использовать фильтр для поиска квартир по своим предпочтениям. При клике по квартире вы перейдете на ее страницу. |
| 4.2 | Застройщики | Название, репутация, область работы, мин-макс. квартир | Страница со списком всех застройщиков. Можно использовать фильтр для поиска предпочтительных застройщиков. При клике по пункту из списка вы перейдете на страницу-визитку застройщика. |

**10. Итоги работы**

На изображении 4 вы можете видеть реализацию REST API для взаимодействия сервера и клиентского приложения

Route::middleware(‘auth:sanctum’)->get(‘/user’, function (Request $request) {

  return $request->user();

});

/\*\*

 \* Регистрация, авторизация, выход из системы, обновление токенов

 \*/

Route::prefix(‘auth’)->group(function() {

  Route::post(‘register’, [AuthController::class, ‘register’]);

  Route::post(‘login’, [AuthController::class, ‘login’]);

  Route::get(‘logout’, [AuthController::class, ‘logout’])->middleware(‘auth:api’);

  Route::get(‘refresh’, [AuthController::class, ‘refresh’]);

  Route::get(‘verify\_email’, [AuthController::class, ‘verifyEmail’]);

});

Route::get(‘companies’, [CompanyController::class, ‘index’]);

Route::get(‘companies/{companyId}’, [CompanyController::class, ‘getOne’]);

Route::get(‘projects’, [ProjectController::class, ‘index’]);

Route::get(‘projects/{projectId}’, [ProjectController::class, ‘getOne’]);

Route::get(‘entities’, [EntityController::class, ‘index’]);

Route::get(‘entities/{entityId}’, [EntityController::class, ‘getOne’]);

Route::get(‘project/{projectId}/entities’, [EntityController::class, ‘getProjectEntities’]);

Route::get(‘appartments’, [AppartmentController::class, ‘index’]);

Route::get(‘appartments/{appartmentId}’, [AppartmentController::class, ‘getOne’]);

Route::get(‘entities/{entityId}/appartments’, [AppartmentController::class, ‘getEntityAppartments’]);

/\*\*

 \* Доступ авторизованным пользователям

 \*/

Route::middleware([‘auth:api’, ‘ban’])->group(function() {

  Route::get(‘users’, [UserController::class, ‘index’]);

  Route::get(‘users/me’, [UserController::class, ‘me’]);

  Route::put(‘users/{userId}’, [UserController::class, ‘update’]);

  /\*\*

   \* Не доделано. Проблемы с загрузкой файлов изображений на сервер

   \*/

  Route::get(‘project/{projectId}/images’, [ProjectImageController::class, ‘index’]);

  Route::get(‘payment\_types’, [PaymentTypeController::class, ‘index’]);

  Route::get(‘appartment\_types’, [AppartmentTypeController::class, ‘index’]);

});

/\*\*

 \* Доступ администраторам системы, менеджерам системы, SU

 \*/

Route::middleware([‘auth:api’, ‘role:ADMIN MANAGER SUPERUSER’])->group(function() {

  Route::get(‘users/{userId}’, [UserController::class, ‘getOne’]);

  Route::post(‘add\_company\_admin’, [UserController::class, ‘add’])->middleware(‘add\_role:COMPANY\_ADMIN’);

  Route::post(‘companies’, [CompanyController::class, ‘add’]);

  Route::put(‘companies/{companyId}’, [CompanyController::class, ‘update’]);

  Route::get(‘black\_list’, [BlackListController::class, ‘index’]);

  Route::post(‘black\_list’, [BlackListController::class, ‘ban’]);

  Route::delete(‘black\_list/{banId}’, [BlackListController::class, ‘unban’]);

});

/\*\*

 \* Доступ администраторам системы, SU

 \*/

Route::middleware([‘auth:api’, ‘role:ADMIN SUPERUSER’])->group(function() {

  Route::post(‘add\_manager’, [UserController::class, ‘add’])->middleware(‘add\_role:MANAGER’);

  Route::get(‘deals/last’, [AdminController::class, ‘getLastDeals’]);

  Route::get(‘admin/companies’, [AdminController::class, ‘getCompanies’]);

  Route::get(‘managers’, [AdminController::class, ‘getManagers’]);

});

/\*\*

 \* Доступ застройщикам (администраторам)

 \* role:COMPANY\_ADMIN – пока такое название роли, потом можно поменять

 \*/

Route::middleware([‘auth:api’, ‘role:COMPANY\_ADMIN’])->group(function() {

  Route::post(‘add\_company\_manager’, [CompanyAdminController::class, ‘createManager’]);

  Route::get(‘get\_company\_managers’, [CompanyAdminController::class, ‘getManagers’]);

  Route::delete(‘projects/{projectId}’, [ProjectController::class, ‘delete’]);

});

/\*\*

 \* Доступ застройщикам (администраторам, менеджерам)

 \* role:COMPANY\_MANAGER – пока такое название роли, потом можно поменять

 \*/

Route::middleware([‘auth:api’, ‘role:COMPANY\_MANAGER COMPANY\_ADMIN’])->group(function() {

  Route::post(‘projects’, [ProjectController::class, ‘add’]);

  Route::put(‘projects/{projectId}’, [ProjectController::class, ‘update’]);

  Route::post(‘deals’, [DealController::class, ‘add’]);

  // /\*\*

  //  \* Не доделано

  //  \* НУЖНО ДОБАВИТЬ ЗАГРУЗКУ ФАЙЛОВ ИЗОБРАЖЕНИЙ НА СЕРВЕР

  //  \*/

  // Route::prefix(‘projects/{projectId}’)->group(function() {

  //   Route::post(‘images’, [ProjectImageController::class, ‘add’]);

  //   Route::delete(‘images/{imageId}’, [ProjectImageController::class, ‘delete’]);

  // });

  Route::post(‘projects/{projectId}/entities’, [EntityController::class, ‘add’]);

  Route::put(‘projects/{projectId}/entities/{entityId}’, [EntityController::class, ‘update’]);

  Route::delete(‘entities/{entityId}’, [EntityController::class, ‘delete’]);

  /\*\*

   \* Надо написать парсер google sheets, для этого нужно создать сервисный аккаунт в google cloud platform

   \* Напишу, если успею

   \*/

  Route::post(‘entities/{entityId}/appartments’, [AppartmentController::class, ‘add’]);

  Route::post(‘entities/{entityId}/appartments/url’, [AppartmentController::class, ‘addAppartmentsWithUrl’]);

  Route::put(‘entities/{entityId}/appartments/{appartmentId}’, [AppartmentController::class, ‘update’]);

  Route::delete(‘appartments/{appartmentId}’, [AppartmentController::class, ‘delete’]);

});

/\*\*

 \* Доступ только SU, ADMIN

 \*/

Route::middleware([‘auth:api’, ‘role:SUPERUSER ADMIN’])->group(function () {

  Route::post(‘users’, [UserController::class, ‘add’]);

  Route::delete(‘users/{userId}’, [UserController::class, ‘delete’]);

  Route::put(‘users/{userId}/changeRole’, [UserController::class, ‘changeRole’]);

  Route::delete(‘info/{infoId}’, [RoleController::class, ‘delete’]);

  Route::get(‘roles’, [RoleController::class, ‘index’]);

  Route::post(‘roles’, [RoleController::class, ‘add’]);

  Route::put(‘roles’, [RoleController::class, ‘update’]);

  Route::delete(‘roles’, [RoleController::class, ‘delete’]);

  Route::delete(‘companies/{companyId}’, [CompanyController::class, ‘delete’]);

  Route::get(‘payment\_types/{paymentId}’, [PaymentTypeController::class, ‘getOne’]);

  Route::post(‘payment\_types’, [PaymentTypeController::class, ‘add’]);

  Route::put(‘payment\_types/{paymentId}’, [PaymentTypeController::class, ‘update’]);

  Route::delete(‘payment\_types/{paymentId}’, [PaymentTypeController::class, ‘delete’]);

  Route::post(‘appartment\_types’, [AppartmentTypeController::class, ‘add’]);

  Route::put(‘appartment\_types/{typeId}’, [AppartmentTypeController::class, ‘update’]);

  Route::delete(‘appartment\_types/{typeId}’, [AppartmentTypeController::class, ‘index’]);

});

Рис. 4 -реализация роутинга с помощью фреймворка Laravel