**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

**Выпускная квалификационная работа**

на тему «Клиент-серверное приложение для взаимодействия жильцов многоэтажного дома»

по направлению подготовки 09.03.04 «Программная инженерия»

|  |  |
| --- | --- |
| Научный руководитель  Доцент департамента программной инженерии,  кандидат технических наук  Х. М. Салех  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Подпись, Дата | Выполнил  студент группы БПИ161  4 курса бакалавриата  образовательной программы «Программная инженерия»  Н. Р. Никитин  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Подпись, Дата |

**Москва 2020**

**Реферат**

В настоящий момент огромное количество жителей многоэтажных домов испытывают трудности во взаимодействии с другими жильцами и с управляющими организациями. На рынке уже существуют решения, предоставляющие возможность коммуникации между жителями и управляющими организациями, однако их функциональность зачастую очень ограничена.

Проблема отсутствия коммуникации между жителями стоит не менее остро, и решения на рынке практически отсутствуют. Зачастую люди самостоятельно находят друг друга и создают в мессенджерах чаты для обсуждения текущих проблем, которые также не обладают достаточными функциональными возможностями, способными решить организационные вопросы. Ритм жизни современного человека делает практически невозможным организацию встреч жильцов в удобное для всех время для решения возникающих вопросов.

Данное приложение ставит перед собой цель решить проблему взаимодействия жильцов многоэтажного дома для решения бытовых и организационных проблем.

Работа содержит 49 страниц, 4 главы, 32 рисунка, 18 источников, 4 приложения.

**Ключевые слова**: жители, многоэтажный дом, управляющая организация, взаимодействие, коммуникация.

**Abstract**

Nowadays, a vast number of residents of multi-storey buildings experience difficulties in interaction with other residents, as well as with managing organizations. There are lots of different solutions that provide the opportunity for communication between residents and management organizations, but their functionality is often minimal.

However, the problem of lack of communication between residents is no less important; there are practically no solutions on the market. Often, residents independently find each other and create chats in instant messengers to discuss current problems. Instant messengers don’t have sufficient functionality that may help to solve organizational issues. The rhythm of life of a modern person makes it almost impossible to organize meetings of residents at a convenient time for everyone to solve emerging issues.

This application aims to solve the problem of interaction of tenants of a multi-storey building to solve domestic and organizational issues.

The paper contains 49 pages, 4 chapters, 32 figures, 18 sources, 4 annexes.

**Keywords**: residents, multi-storey building, managing organization, interaction, communication.

**Основные определения, термины и сокращения**

**Управляющая организация** – это доверенное лицо собственников помещений в многоквартирном доме, их полномочный представитель в отношениях с третьими лицами, осуществляющими оказание работ и услуг.

**HTTP** – это протокол передачи данных от сервера к клиенту.

**API** – интерфейс для взаимодействия клиента и сервера.

**Жилищно-коммунальное хозяйство** (*ЖКХ*) — комплекс отраслей экономики, обеспечивающий функционирование жилых зданий, создающих безопасное, удобное и комфортабельное проживание и нахождение в них жителей (потребитель). Включает в себя также объекты социальной инфраструктуры для обслуживания жителей.

**Жилец** - лицо, живущее в доме, квартире, снимающее жилое помещение (по отношению к тому, у кого снимается это помещение).

**Управляющая организация** - юридическое лицо, созданное для управления и/или эксплуатации, технического и санитарного содержания многоквартирных домов на основе возмездного договора с собственниками.

**Товариществом собственников жилья** *(ТСЖ)* - некоммерческая организация, объединение собственников помещений в многоквартирном доме для совместного управления комплексом недвижимого имущества в многоквартирном доме, обеспечения эксплуатации этого комплекса, владения, пользования и, в установленных законодательством пределах, распоряжения общим имуществом в многоквартирном доме.

**Совет многоквартирного дома** – выборный орган собственников помещений в многоквартирном доме. Его создание и полномочия регулируются статьей 161.1 Жилищного кодекса Российской Федерации

**Председатель** – председатель совета дома или председатель ТСЖ.

**Содержание**

[Реферат 2](#_Toc40977754)

[Abstract 3](#_Toc40977755)

[Основные определения, термины и сокращения 4](#_Toc40977756)

[Введение 7](#_Toc40977757)

[Глава 1. Анализ предметной области и определение требований 9](#_Toc40977758)

[1.1. Обзор рынка приложений 9](#_Toc40977759)

[1.1.1 Telegram и WhatsApp 10](#_Toc40977760)

[1.1.2 PropertyVISTA 10](#_Toc40977761)

[1.1.3 Hudson 11](#_Toc40977762)

[1.2 Постановка задач для разработки. 11](#_Toc40977763)

[2 Информационная модель 13](#_Toc40977764)

[2.1 Описание требований 13](#_Toc40977765)

[2.2 Способ хранения данных 16](#_Toc40977766)

[2.3 Взаимодействие серверной и клиентской частей приложения. 20](#_Toc40977767)

[3 Серверная часть. 22](#_Toc40977768)

[3.1 Стек технологий 22](#_Toc40977769)

[3.2 Обеспечение безопасности и разграничение прав. 22](#_Toc40977770)

[3.3 Реализация функциональных требований 24](#_Toc40977771)

[3.3.1 Регистрация пользователя и ведение истории регистраций. 24](#_Toc40977772)

[3.3.2 Авторизация пользователя. 27](#_Toc40977773)

[3.3.3 Поддержка различных ролей пользователей. 27](#_Toc40977774)

[3.3.4 Участие в чатах. 28](#_Toc40977775)

[3.3.5 Поднятие тем на обсуждение в чатах. 30](#_Toc40977776)

[3.3.6 Голосование в чатах. 30](#_Toc40977777)

[3.3.7 Отображение профиля УО. 30](#_Toc40977778)

[3.3.8 Внесение предложение и обозначение проблем. 30](#_Toc40977779)

[3.3.9 Рассылка объявлений жильцам. 31](#_Toc40977780)

[3.3.10 Уведомление о новых событиях. 32](#_Toc40977781)

[3.3.11 Отображение профиля пользователя. 32](#_Toc40977782)

[3.3.12 Вызов мастера на дом. 32](#_Toc40977783)

[3.3.13 Оценка работы приложения посредством формы обратной связи. 33](#_Toc40977784)

[3.4 Документирование API приложения. 33](#_Toc40977785)

[4 Описание работы клиентской части приложения 38](#_Toc40977786)

[4.1 Стек технологий. 38](#_Toc40977787)

[4.2 Работа с данными. 38](#_Toc40977788)

[4.3 Реализация функциональных требований 39](#_Toc40977789)

[4.3.1 Регистрация и авторизация. 39](#_Toc40977790)

[4.3.2 Поддержка различных ролей пользователей приложения. 40](#_Toc40977791)

[4.3.3 Участие в чатах. 41](#_Toc40977792)

[4.3.4 Создание голосований и участие в них. 42](#_Toc40977793)

[4.3.5 Отображение профиля УО и создание проблем и предложений. 43](#_Toc40977794)

[4.3.6 Рассылка объявлений жильцам. 43](#_Toc40977795)

[4.3.7 Ведение истории регистраций. 44](#_Toc40977796)

[4.3.8 Уведомление о новых событиях. 44](#_Toc40977797)

[4.3.9 Отображение профиля пользователя с информацией о нем. 44](#_Toc40977798)

[4.3.10 Вызов мастера на дом. 45](#_Toc40977799)

[4.3.11 Оценка работы приложения посредством формы обратной связи. 45](#_Toc40977800)

[Заключение 47](#_Toc40977801)

[Список использованных источников 48](#_Toc40977802)

**Введение**

На сегодняшний день в многоэтажных домах проживает огромное количество людей и их численность из года в год только возрастает. Например, по данным исследования GOMATRIX [1] в 2018 году среднее число этажей в новых домах на территории Москвы - 17. У жителей зачастую возникает проблема отсутствия каналов для коммуникации с другими жителями и управляющей организацией.

Для обсуждения и решения возникающих проблем жителям чаще всего приходится идти в офис управляющей компании, несмотря на то что на рынке уже существуют готовые решения этой проблемы. Как бы то ни было, существующие приложения имеют весьма скудную функциональность.

Не менее важная и острая проблема – отсутствие инструментов для взаимодействия между самими жильцами многоэтажного дома. В быту каждый человек нуждается в обсуждении проблем и обменом той или информацией, в любой момент может потребоваться помощь в нештатной ситуации или просто совет. Наиболее распространенным решением является использование мессенджеров и социальных сетей. Этот подход имеет массу недостатков: людей приходится находить вручную, нет никаких гарантий того, что в чат не сможет попасть человек, не проживающий в данном многоэтажном доме, а также все взаимодействие ограничивается лишь общением, так как реальных инструментов, которые помогут в решении возникающих проблем, данные приложения не предоставляют.

Главной целью приложения является решение двух вышеописанных проблем, а также предоставление жителям и управляющим организациям дополнительных инструментов, облегчающих управление многоэтажным домом и проживание в нем.

Для достижения цели необходимо решение следующих задач:

1. Изучить рынок аналогов. Отметить преимущества и недостатки существующих решений;
2. Определить требования к серверной и клиентской частям приложения;
3. Исследование инструментов для реализации поставленных требований в виде клиент-серверного приложения;
4. Спроектировать архитектуру будущего приложения;
5. Разработать клиентскую часть;
6. Разработать серверную часть;
7. Настроить взаимодействия обеих частей приложения с помощью API по HTTP протоколу;
8. Протестировать полученное приложение;
9. Написать техническую документацию;
10. **Анализ предметной области и определение требований**

В данной главе представлен сравнительный анализ существующих на рынке аналогов, а также список функциональных требований для разрабатываемого приложения.

* 1. **Обзор рынка приложений**

Для решения организационных и бытовых вопросов в большинстве многоэтажных домов существует совет многоквартирного дома. Это выборный орган, который является связующим звеном между собственниками жилья и управляющими организациями, в обязанности которого входит подготовка вопросов на рассмотрение общему собранию и обеспечение реализации принятых решений и контроля качества над ними. Руководящим лицом совета является председатель.

Согласно законодательству Российской Федерации, решение всех вопросов, связанных с управлением многоквартирным домом, решается с помощью общего собрания. Здесь возникает проблема – собрать большинство собственников в многоэтажных домах зачастую не предоставляется возможным. При недостатке кворума очное голосование не признается состоявшимся. В таких случаях, стоит прибегнуть к заочной форме голосования. Очевидно, что данный процесс является очень сложным и может быть упрощен. Данное приложение предоставляет возможность внесения тем на голосование и онлайн-голосования, что значительно облегчит жизнь собственникам. На данный момент, на рынке практически отсутствуют приложения с подобной функциональностью.

Не менее важным для организационного процесса являются объявления. Пользователь должен иметь возможность получать экстренные уведомления немедленно, так как в многоэтажном доме произойти может буквально все. Лишь самые современные решения в этой области поддерживают подобную функциональность, хотя это возможность является крайне необходимой для жителей многоэтажных домов. Данное приложение позволяет управляющим организациям своевременно оповещать жильцов о происходящих событиях.

Помимо голосований и объявлений, жители нуждаются в возможности обсуждения различных бытовых вопросов и проблем. Сейчас для этого используются мессенджеры, однако данное решение не является безопасным и удобным. Различные CRM системы ЖКХ, в свою очередь, вовсе не предоставляют возможность общения пользователей. Данное приложение поддерживает механизм ведения чатов, разделенных по тематике, для удобного использования, а также механизм авторизации, во избежание доступа к информации третьих лиц.

Также стоит учесть, что жители многоэтажных домов нуждаются как во взаимодействии между собой, так и с управляющими организациями. Мессенджеры и социальные сети – неподходящий инструмент для этих проблем. Данное приложение предоставляет возможность коммуникации с управляющими организациями через чат, а также позволяет пользователю обозначать проблемы и оставлять предложения для управляющих организаций.

На данный момент, на рынке практически не существует приложений, способных помочь жильцам в экстренной ситуации. Данное решение позволяет пользователю вызвать мастера на дом с возможностью отслеживать статус заявки.

В качестве конкурентов рассматривались только те ресурсы, которые доступны с любых устройств, независимо от их операционной системы. Очень важно, чтобы данное приложение работало одинаково хорошо на всех платформах, чтобы не вынуждать жителей и управляющие организации подстраиваться под конкретное решение.

* + 1. **Telegram и WhatsApp**

Одними из наиболее популярных мессенджеров являются Telegram [2] и WhatsApp [3]. Данные сервисы позволяют удобно и быстро обмениваться информацией с другими людьми, но однозначно не подходят для взаимодействия между жильцами. Одна из основных проблем данных решений состоит в том, что пользователям необходимо самостоятельно организовывать и управлять чатами, что является очень непростой задачей для многоэтажных домов с большим количеством жильцов. Также не существуют гарантий, что доступ к переписке не получит человек, не проживающий в данном доме. Помимо этого, Telegram и WhatsApp не предоставляют никаких инструментов, упрощающих управление организационными процессами.

* + 1. **PropertyVISTA**

PropertyVISTA [4] – яркий пример отличного приложения для управления домом. Данный сервис обладает удобным интерфейсом и широким набором инструментов для комфортного управления домом. Однако данное решение разработано лишь для управления и не ставит цели наладить взаимодействие жильцов между собой. PropertyVISTA позволяет автоматизировать большое количество процессов, но не предоставляет инструментов для коммуникации пользователей. Также стоит отметить внушительную стоимость ежемесячного использования сервиса.

* + 1. **Hudson**

Отдельного упоминания заслуживает приложение Hudson [5]. Это наиболее полное решение с точки зрения предлагаемых возможностей и больше остальных схоже с разрабатываемым в рамках данной работы приложением, хотя, как и другие CRM системы для ЖКХ, данное приложение не обладает инструментами для общения между пользователями. Данный сервис пока не обладает большим набором функций, однако, по заверению создателей, будет дорабатываться в будущем. Так, например, в приложение будут добавлены такие возможности как снятие данных со счетчиков и возможность заказать пропуск для гостей.

Таблица 1. Сравнительный анализ.



* 1. **Постановка задач для разработки.**

Житель многоэтажного дома должен иметь возможность зарегистрироваться и авторизоваться в приложении, иметь доступ к чатам, обсуждениям, голосованиям и объявлениям. Кроме того, пользователь должен иметь возможность общения с управляющей компанией, оставлять пожелания или указывать на проблемы в ее профиле. Также житель должен иметь возможность оставить отзыв о работе приложении в целом. При необходимости собственник жилья должен иметь возможность вызвать мастера на дом или посмотреть имеющуюся о нем информацию в профиле.

Председатель дома должен обладать теми же возможностями, что и житель. Кроме того, председатель должен иметь уникальный доступ к созданию и выдвижению тем на обсуждение и созданию голосований.

Администратор системы должен иметь доступ к списку заявлений на регистрацию квартир и домов в системе и к истории регистраций. Также он должен иметь доступ к авторизации и странице профиля.

Менеджер УО должен иметь доступ к системе авторизации и регистрации, профилю пользователя. Помимо этого, менеджер должен иметь возможность общения с жильцами и просмотру профиля УО. Также менеджер УО должен иметь возможность рассылки объявлений жильцам.

После изучения необходимых функций для каждой роли пользователя приложения, были поставлены следующие задачи:

1. Регистрация в приложении;
2. Авторизация в приложении;
3. Поддержка различных ролей пользователей приложения;
4. Участие в чатах, разделенных по тематике;
5. Поднятие тем на обсуждение в чатах;
6. Голосование в обсуждениях;
7. Отображение профиля УО;
8. Внесение предложений и обозначение проблем управляющей организации;
9. Рассылка объявлений жильцам;
10. Ведение истории регистраций;
11. Уведомление о новых событиях;
12. Отображение профиля пользователя с информацией о нем;
13. Вызов мастера на дом;
14. Оценка работы приложения посредством формы обратной связи;

**Выводы по главе:**

В первой главе был произведен сравнительный анализ существующих аналогов, на основании которого были сформулированы основные задачи приложения.

В следующей главе будет описана информационная модель данного приложения.

1. **Информационная модель**

В данной главе приведено описание требований, способ хранения данных, а также способ взаимодействия серверной и клиентской частей приложения.

* 1. **Описание требований**

Прежде всего, важно убедиться в корректности предъявляемых к программному продукту требований и структурировать их. Для этого решено было построить диаграмму прецедентов. Так как данное приложение содержит большое количество функций, то одну диаграмму прецедентов было решено разбить на несколько для лучшего понимания процессов.

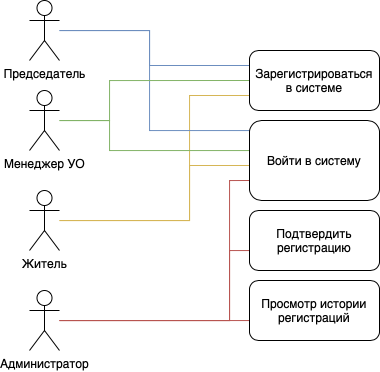


Рисунок 1. Схема прецедентов, рассматривающая процесс регистрации пользователя.

В схеме из рис. 1 стоит обратить внимание на то, что каждую регистрацию пользователя в системе должен одобрить администратор. До тех пор, пока администратор системы не подтвердит регистрацию, аккаунт считается недействительным, и у пользователя отсутствует возможность входа в систему.

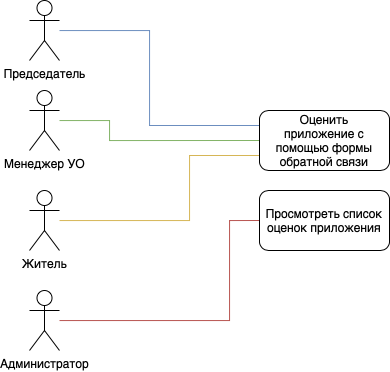


Рисунок 2. Схема прецедентов, рассматривающая работу механизма оценивания.

Как видно из рис. 2, пользователь с ролью администратора не имеет возможности оставлять отзывы о системе, в отличии от других пользователей приложения. Администратор, в свою очередь, может просматривать список с существующими отзывами о приложении.

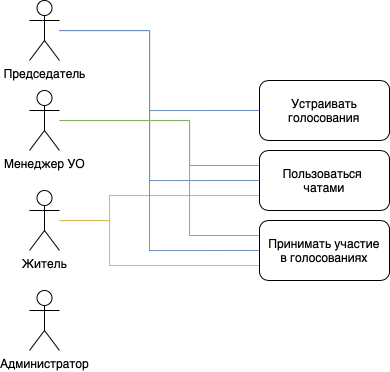


Рисунок 3. Схема прецедентов, рассматривающая работу чатов.

На рис. 3 видно, что администратор не имеет доступа к системе чатов. Это связано с тем, что каждый чат в системе связан с домом, а аккаунт администратора не предполагает наличие у него квартиры в системе.

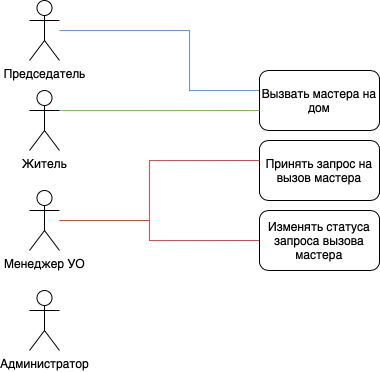


Рисунок 4. Схема прецедентов, рассматривающая механизм вызова специалиста на дом.

Вызвать мастера может председатель или житель многоэтажного дома. Менеджер УО имеет возможность принять запрос, в таком случае статус заявки изменится. Дальнейшее изменение статуса запроса производится менеджером вручную.

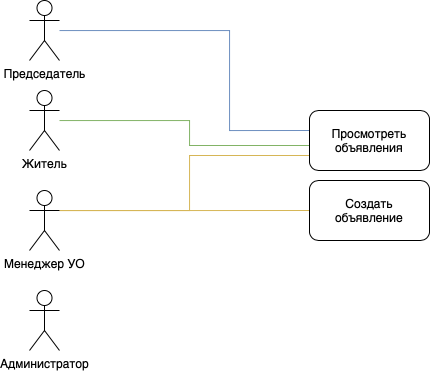


Рисунок 5. Схема прецедентов, рассматривающая механизм работы объявлений.

Возможность просмотра объявлений есть у жителей и председателей данного дома, а также у менеджера УО. Создавать новое объявление может только менеджер.

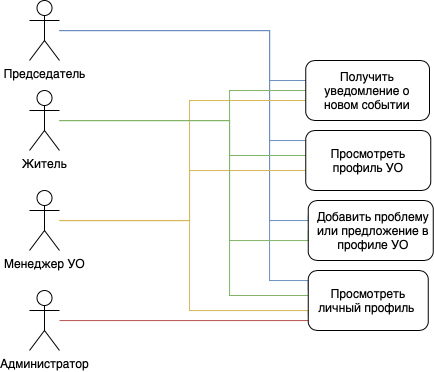


Рисунок 6. Схема прецедентов, рассматривающая оставшуюся функциональность приложения.

Все пользователи приложения могут просматривать личный профиль в системе. Просмотр профиля УО может осуществляться председателем, жителем и менеджером УО. Добавлять проблемы и предложения в профиле УО могут председатели и жители.

Также существует функция получения уведомления о новом событии. В рамках данного приложения за событие принимается только обновляемые в реальном режиме данные, то есть сообщения в чатах и объявления. Именно поэтому администратор не может воспользоваться данной функцией, так как не имеет доступа к объявлениям и чатам.

* 1. **Способ хранения данных**

Приложение использует реляционную базу данных PostgreSQL [6], спроектированную для работы высоконагруженных систем с большим количеством данных.

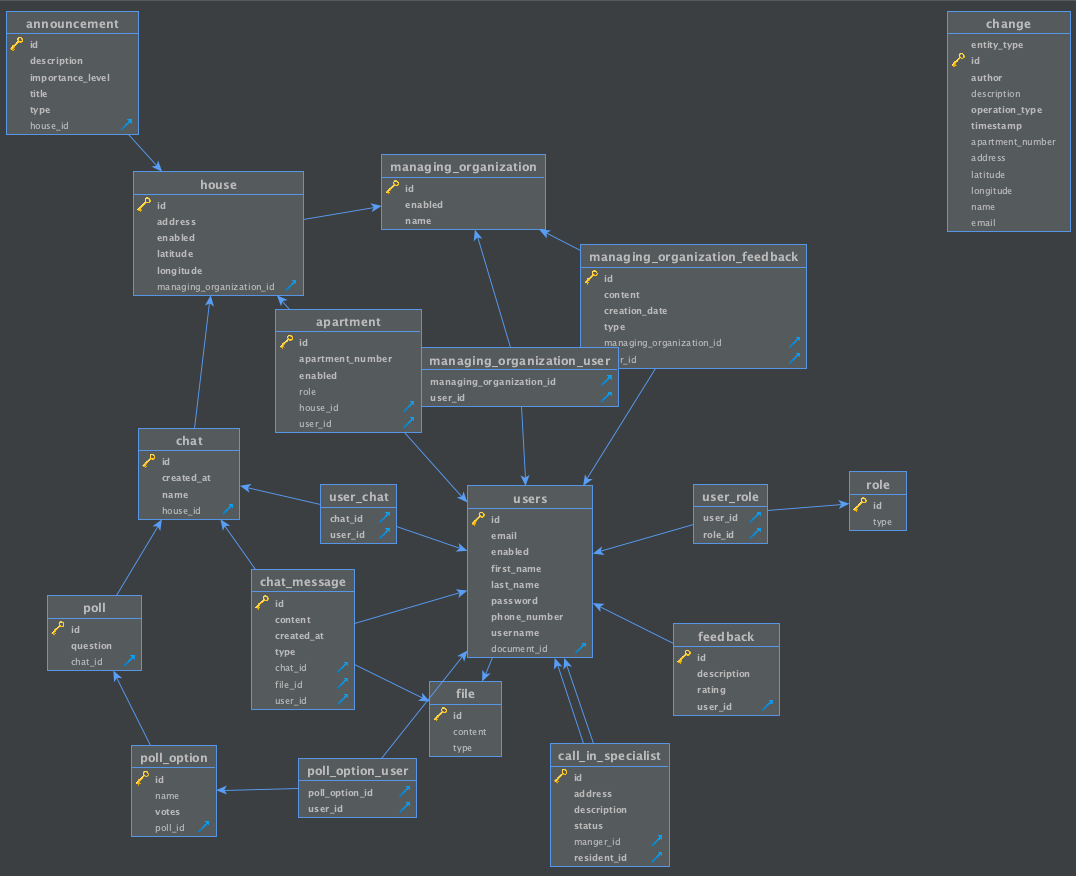


Рисунок 7. Диаграмма сущностей

На диаграмме сущностей (рис. 7) представлены все существующие в базе данных таблицы. Для поддержки связи многие ко многим (many-to-many) в базе данных были созданы таблицы-связки.

Таблица 2. Таблицы базы данных с их описанием.

|  |  |
| --- | --- |
| **Таблица** | **Описание** |
| announcement | Таблица объявлений. Каждая запись содержит данные об уровне важности объявления (перечисление), подробном описании, заголовке, типе объявления (перечисление). |
| house | Таблица домов. Каждая запись содержит данные об адресе, широте и долготе, внешний ключ для связи с таблицей управляющих организаций и имеет флаг подтверждения в системе. |
| chat | Таблица чатов. Каждая запись содержит данные о дате создания чата, его наименовании, а также внешний ключ для связи с таблицей домов. |
| poll | Таблица опросов. Каждая запись содержит тему опроса. |
| poll\_option | Таблица вариантов ответов на вопрос. Каждая запись содержит наименование ответа, количество голосов, а также внешний ключ для связи с таблицей опросов. |
| managing\_organization | Таблица управляющих организаций. Каждая запись содержит имя управляющей организации, а также флаг подтверждения в системе. |
| apartment | Таблица квартир. Каждая запись содержит данные о номере квартиры, роли владельца квартиры (председатель/житель), внешний ключ для связи с таблицей домов, внешний ключ для связи с таблицей пользователей. |
| user\_chat | Таблица-связка для сущностей user и chat. |
| chat\_message | Таблица сообщений в чате. Каждая запись содержит текстовое содержимое сообщения, дату создания, тип (перечисление), внешний ключ для связи с таблицей чатов, внешний ключ для связи с таблицей файлов, внешний ключ для связи с таблицей пользователей. |
| poll\_option\_user | Таблица-связка для сущностей poll\_option и user. |
| managing\_organization\_user | Таблица-связка для сущностей managing\_organization и user. |
| users | Таблица пользователей. Каждая запись содержит email, имя, фамилию, пароль, номер телефона и внешний ключ для связи с таблицей файлов. |
| file | Таблица файлов. Каждая запись содержит содержимое файла и его тип. |
| call\_in\_specialist | Таблица вызовов специалистов. Каждая запись содержит адрес, описание проблемы, статус заявки (перечисление), и два внешних ключа для связи с таблицей пользователей. |
| managing\_organization\_feedback | Таблица проблем и предложений УО. Каждая запись содержит текстовое содержание, тип записи (проблема/предложение), внешний ключ для связи с таблицей управляющих организаций, внешний ключ для связи с таблицей пользователей. |
| user\_role | Таблица-связка сущностей user и role. |
| feedback | Таблица отзывов о приложении. Каждая запись содержи описание, рейтинг и внешний ключ для связи с таблицей пользователей. |
| role | Таблица ролей пользователей. Каждая запись содержит тип роли (перечисление). |
| change | Таблица изменений в системе (создана для сохранения истории действий администратора при регистрации). Каждая запись содержит автора, описание, тип операции (перечисление), дату создания, номер квартиры, адрес, долгота и широта, имя и email. Такое большое количество столбцов обусловлено тем, что изменения фиксируются при регистрации дома, квартиры, УО и пользователя в системе. |

Примечание: не все колонки таблиц обязательны к заполнению, данные могут отсутствовать в зависимости от логики работы сервера.

* 1. **Взаимодействие серверной и клиентской частей приложения.**

Серверная и клиентская часть данного приложения не связаны между собой, все взаимодействие осуществляется посредствам REST API. Со стороны клиента выполняется запрос по HTTP протоколу к заранее определенному контроллером сервера URL. Сервер получает и отправляет данные в формате JSON [7].

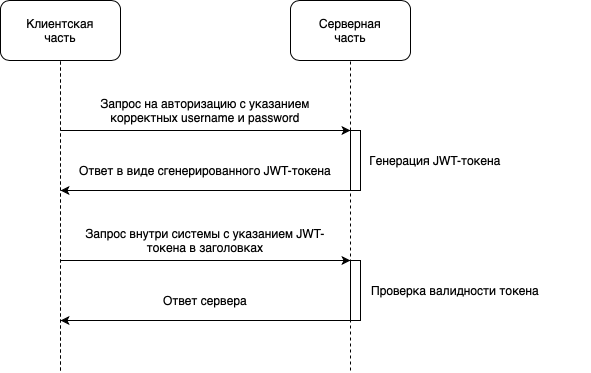


Рисунок 8. Обобщенная модель взаимодействия клиента и сервера.

В ответ на запрос авторизации, содержащий ошибку в пользовательских данных входа, на клиент придет JSON с сообщением об ошибке и ее описанием.

В ответ на запрос внутри системы с указанием недействительного токена на клиент придет сообщение об ошибке с кодом 403 в формате JSON.

На рис. 9 изображена структура приложения.

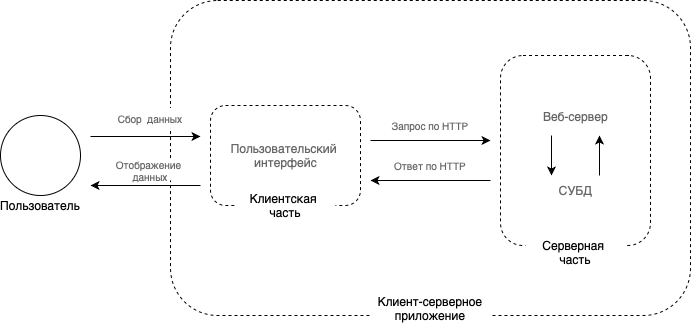


Рисунок 9. Структура приложения.

**Выводы по главе:**

Во второй главе было приведено описание требований, способ хранения данных, а также способ взаимодействия серверной и клиентской частей приложения.

В следующей главе будет описана серверная часть приложения.

1. **Серверная часть.**

В серверной части приложения содержится основная логика его работы. В данной главе приведено описания стека технологий для реализации сервера, механизма обеспечения безопасности, способов реализации функциональных требований, а также список доступных для взаимодействия методов контроллеров.

* 1. **Стек технологий**

Серверная часть приложения реализована с помощью Spring Framework [8], одним из самых используемых и универсальных фреймворков с открытым исходным кодом. Spring – не является единым фреймворком и состоит из целого ряда фреймворков, ответственных за разную часть работы. Данный инструмент значительно облегчает написание серверной части приложения, освобождая программиста от написания низкоуровнего кода и позволяет ему сосредоточиться на реализации бизнес-логики приложения.

В качестве языка программирования серверной части был выбран Kotlin [9]. Код, написанный на языке Kotlin, компилируется в байткод JVM, и программы, написанные на этом языке, могут использовать все существующие Java-фреймворки и библиотеки. В свою очередь, данный язык имеет несколько преимуществ над Java [10], такие как null-безопасность и большое количество синтаксического сахара.

Взаимодействия приложения с базой данных происходит с помощью Spring Data JPA. С помощью данного инструмента используя аннотации и Spring Repository можно удобно применять базовые операции к объектам в базе данных без написания прямых запросов к ней. В случае необходимости, данный инструмент позволяет создавать прямые запросы к базе данных с помощью аннотации @Query.

* 1. **Обеспечение безопасности и разграничение прав.**

Данное приложение поддерживает механизм разграничения прав. Серверная реализация использует инструменты Spring Security для настройки доступа.

В переопределенном методе configure из стандартной конфигурации WebSecurity существует настройка, позволяющая получать доступ к авторизации и регистрации всем пользователям приложения - .antMatchers("/api/auth/\*\*").permitAll(), а также настройка, запрещающая доступ к не попавшим в исключения адресам неавторизированным пользователям - .anyRequest().authenticated().

Также в приложении существуют отдельные контроллеры для администраторов. Доступ к ним имеют только пользователи с ролью ADMIN. За это отвечает следующая настройка: .antMatchers("/api/administration/\*\*").hasAuthority("ADMIN").

После всех настроек происходит добавление фильтрации полученных запросов с помощью метода doFilterInternal класса, наследующего абстрактный класс OncePerRequestFilter.

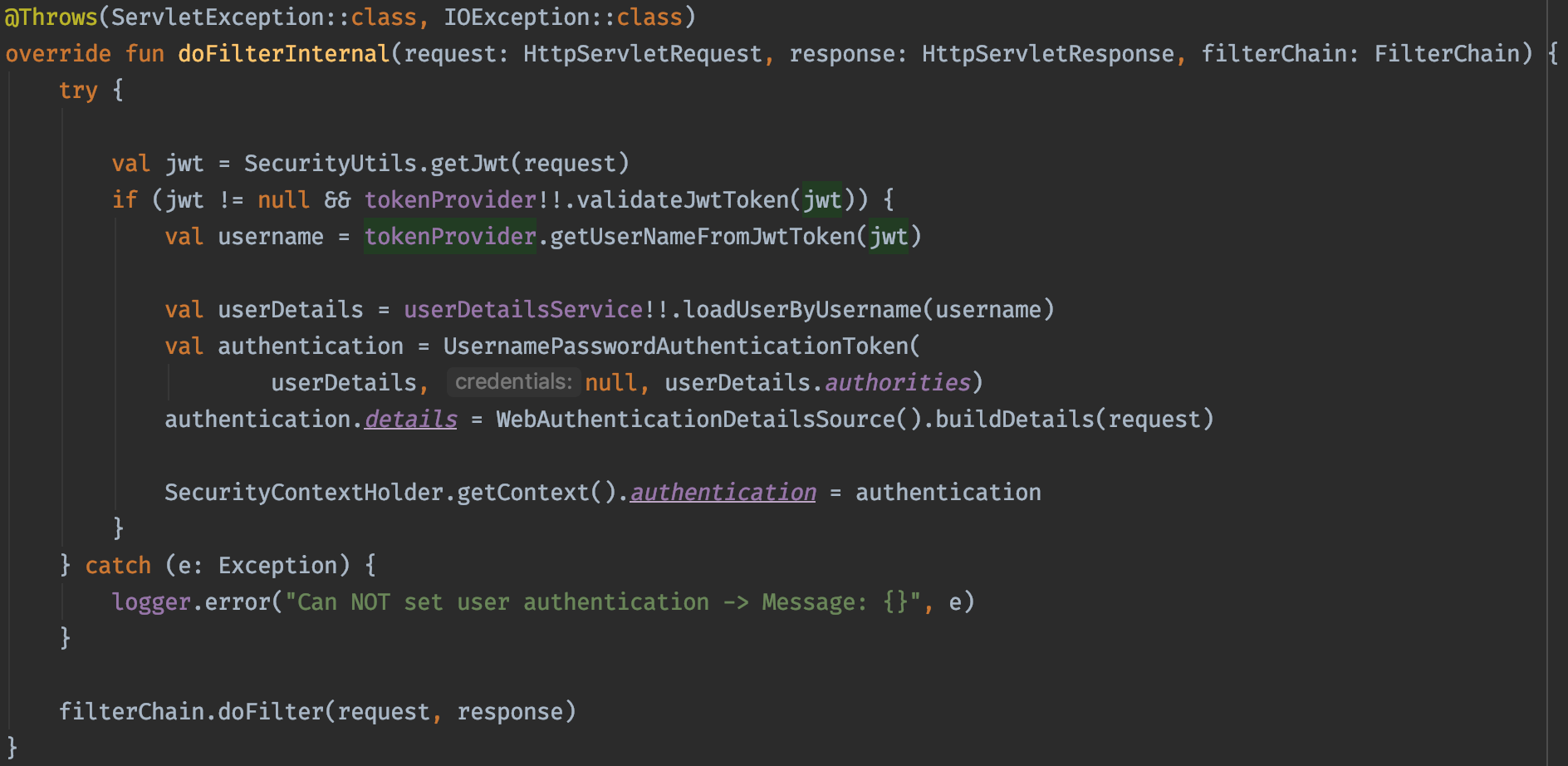


Рисунок 10. Метод фильтрации входящих запросов.

Данный метод вызывает внутри себя метод проверки токена на корректность, и, в случае положительного результата, пытается получить информацию о пользователе из этого токена, предотвращая несанкционированный доступ к ресурсам неавторизированным пользователям.

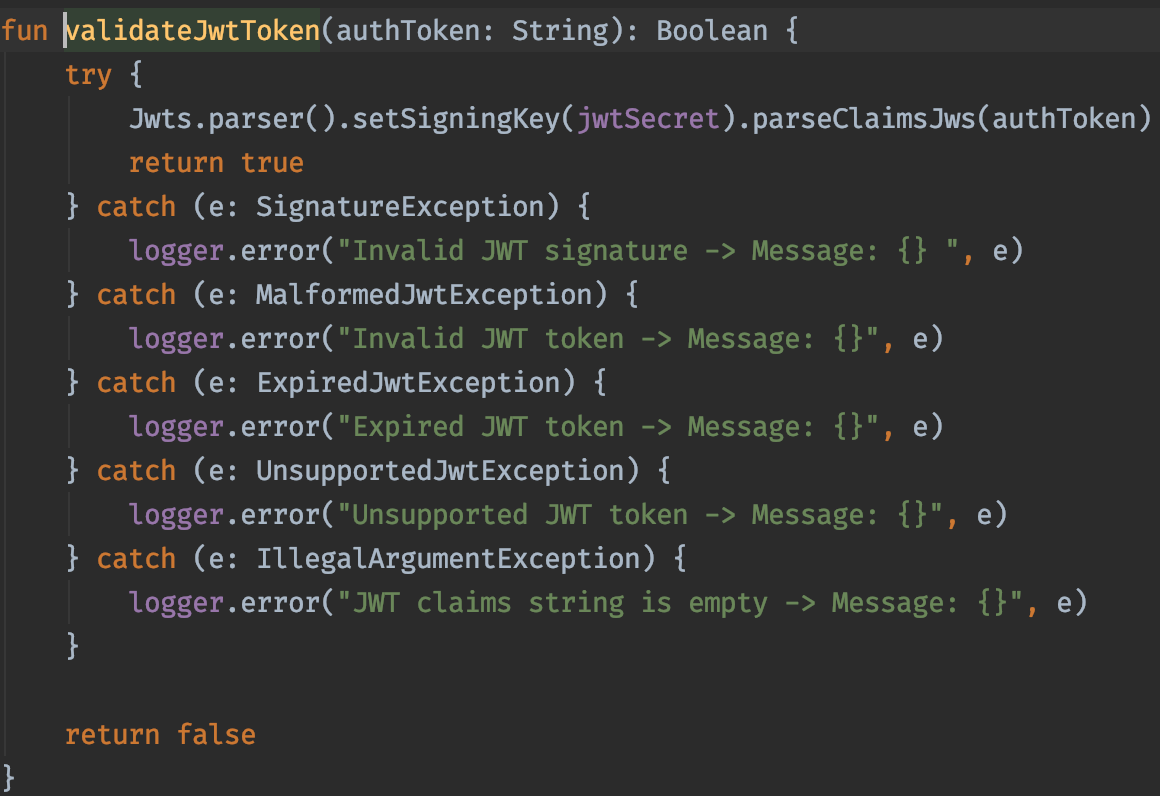


Рисунок 11. Проверка JWT-токена на корректность.

* 1. **Реализация функциональных требований**
     1. **Регистрация пользователя и ведение истории регистраций.**

В приложении возможна регистрация председателя совета дома, жителя дома и менеджера УО. Регистрация авторизация может быть произведена только ручным редактированием базы данных.

Первоначально происходит проверка на наличие пользователя с данным username и email в системе. Поле username принимает значение email, чтобы избавить пользователя от ввода лишней информации.

Изначально в системе сохраняется документ, подтверждающий владение квартирой или подтверждающий принадлежность УО, затем вся личная информация о пользователе, а именно username, firstName, lastName, email, phoneNumber, password, roles.

После этого происходит проверка на наличие дома в системе. Проверка производится с помощью поиска по latitude и longitude. В том случае, если дом в системе не присутствует, то создается новая сущность дома со значением enabled равным false. В ином случае, система получает дом из базы данных и продолжает процесс регистрации, используя уже существующий в системе дом.

С точки зрения базы данных в приложении существует только 3 роли пользователя: “ADMIN”, “USER” и “MANAGER”. Поэтому с технической точки зрения процесс регистрации жителя и председателя дома отличается только выбором роли на форме. В зависимости от этого выбора будет проставлено значение role сущности apartment.

Процесс регистрации жильца и менеджера УО все же отличается – менеджеру необходимо заполнить данные о наименовании УО, а жильцу ввести свой номер квартиры.

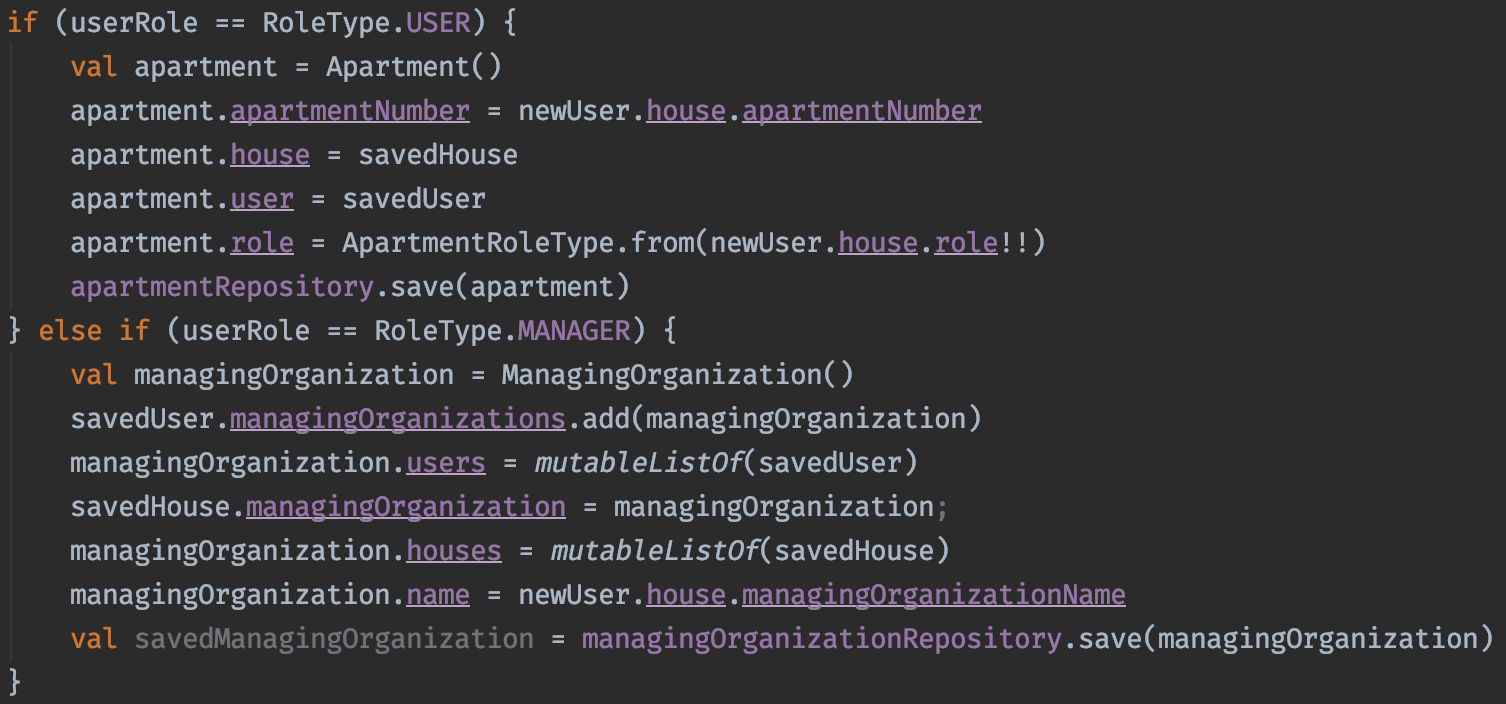


Рисунок 12. Отличия в регистрации жильца (и председателя) и менеджера УО

В случае успешной регистрации, сервер вернет клиенту ответ с кодом 200, иначе – с кодом 400.

На этом процесс не заканчивается, администратор системы должен подтвердить регистрацию.

Администратор получает 3 списка на подтверждение – список запросов на регистрацию дома, список запросов на регистрацию квартир и список запросов на регистрацию УО. Первый этап – подтверждение дома, без него невозможен дальнейший процесс регистрации. Если дом не подтвержден в системе, то в списке заявок на регистрацию квартиры не будут отображаться квартиры, располагающиеся в этом доме. Аналогичная логика работы реализована и с запросами на подтверждение УО – в случае отсутствия дома в системе в списке запросов на регистрацию УО не отображаются УО, управляющие этим домом.

Если регистрация дома был подтверждена в системе, то происходит выставление enabled = true у сущности дом и переход на второй этап регистрации – подтверждения квартиры или УО. В противном случае дом удаляется из системы, а затем и запрос на регистрацию квартиры или УО и аккаунт пользователя.

На втором этапе происходит подтверждение квартиры или УО в системе. После успешного подтверждения данного этапа происходит выставление enabled = true у сущности квартиры или УО и у пользователя. С этого момента он может быть авторизован. В ином случае квартира или УО не подтверждается и удаляется из системы вместе с аккаунтом пользователя. При этом не происходит удаление ранее подтвержденного дома из системы.

Все действия в данном процессе сохраняются в журнал истории и затем могут быть доступны для просмотра администратору.

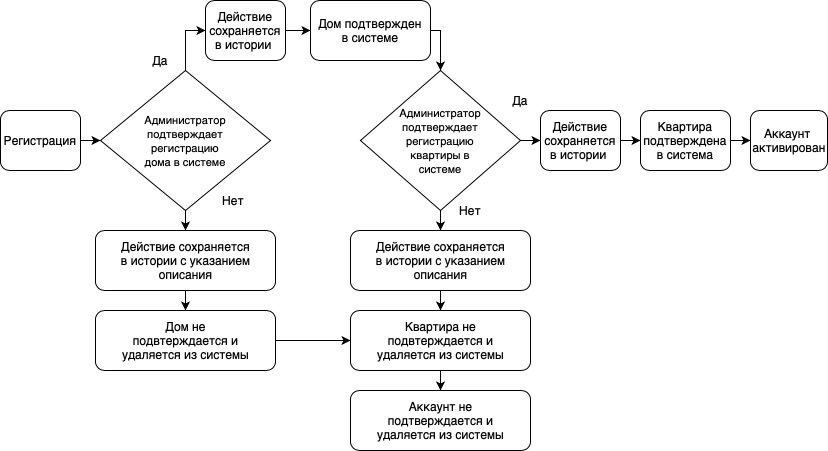


Рисунок 13. Процесс регистрации жильца или председателя в системе

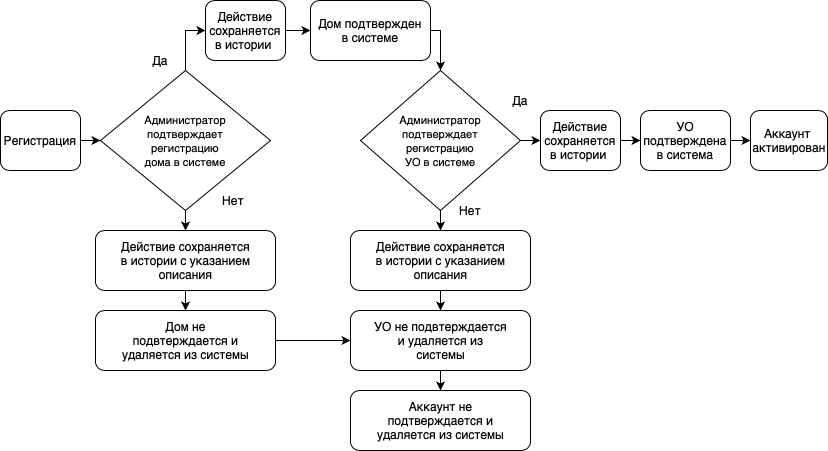


Рисунок 14. Процесс регистрации менеджера в системе

Помимо сохранения истории действий администратора, при подтверждении дома в системе происходит не связанный напрямую с регистрацией процесс – создание чатов для общения. А при подтверждении управляющей организации или квартиры – добавление пользователей в чаты. Данная логика будет описана в документе далее.

* + 1. **Авторизация пользователя.**

В качестве данных для авторизации сервер ожидает имя пользователя и пароль. При наличии пользователя с данным username в системе происходит получение объекта Authentication средствами Spring Security на основе введенных пользователем username и password, после с использованием этого объекта генерируется JWT-токен.



Рисунок 15. Метод генерации токена, вызываемый из контроллера авторизации.

Как показано на рис. 15, в токен записывается информация об алгоритме шифрования, типе токена, имени пользователя, дате генерации токена, дате окончания действия токена и роли пользователя.

* + 1. **Поддержка различных ролей пользователей.**

Данное приложение должно поддерживать следующие роли: администратор, житель, председатель и менеджер УО. У сущности user существует отдельная таблица связка с сущностью role – user\_role. Отношение many-to-many обусловлено тем, что приложение спроектировано таким образом, что пользователь может быть одновременно менеджером и жителем или же жителем и председателем. Это, в свою очередь, связано с тем, что у пользователя может быть не только одна квартира в приложении, а управляющая организация может управлять сразу несколькими домами. На данном этапе существования приложения такая функциональность уже поддерживается архитектурой и логикой серверной части.

Как уже было описано ранее, доступные роли для пользователя – это менеджер, администратор и жилец. Роль председателя реализована иначе – она может быть выбрана в поле роль сущности apartment. Это связано с тем, что с точки зрения логики сервера председатель также является жителем, и в случае существования у пользователя нескольких квартир в разных домах и разных ролей в системе невозможно будет определить в каком доме он является председателем совета дома, а в каком просто жильцом.

* + 1. **Участие в чатах.**

При регистрации нового дома в системе автоматически создаются 4 новых чата, принадлежащих данному дому, со следующими названиями: “General”, “Repair”, “ Flea Market” и “Managing Organization”. Разделение на несколько чатов с различными названиями сделано во избежание излишнего захламления. Последний из них создан для общения между жильцами и менеджерами УО, в трех других могут общаться только жильцы дома.

При подтверждении запроса на регистрацию УО, менеджер добавляется в список пользователей в чат с названием “Managing Organization” того дома, управление которым осуществляется управляющей организацией.



Рисунок 16. Подтверждение или отклонение регистрации УО

При подтверждении квартиры, жилец или председатель добавляется в список пользователей всех чатов данного дома.

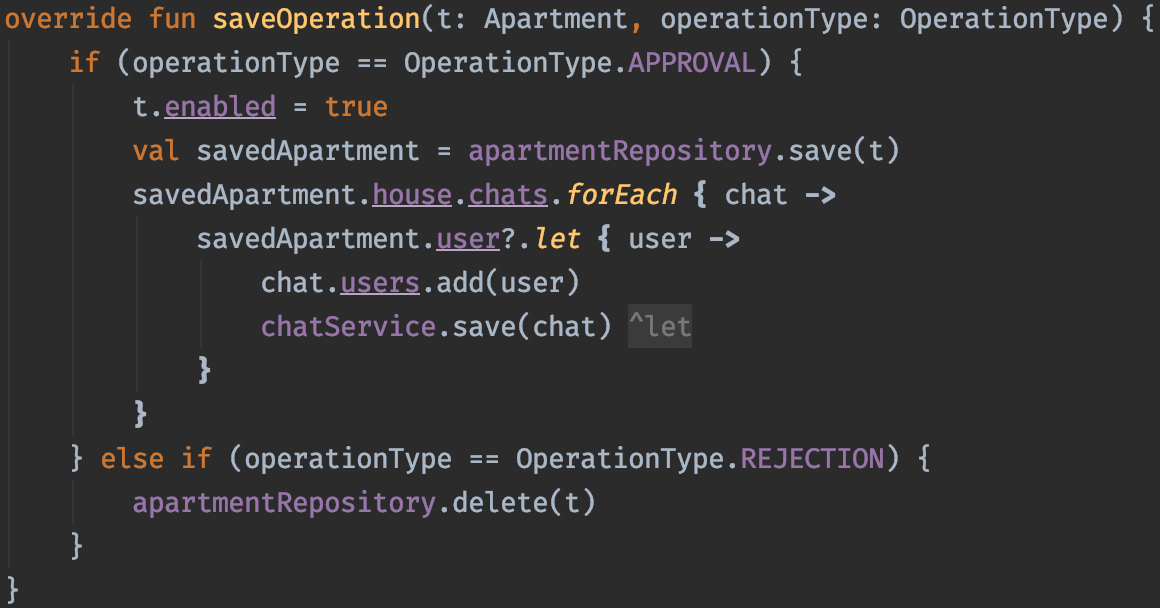


Рисунок 17. Подтверждение или отклонение регистрации квартиры

Для удобного общения в чатах используется WebSocket. Данная технология позволяет мгновенно обновлять данные на клиент, не требуя дополнительных запросов к серверу.

После успешного подключения к каналу чата пользователь будет получать новые сообщения от других пользователей, а также иметь возможность отправлять собственные.

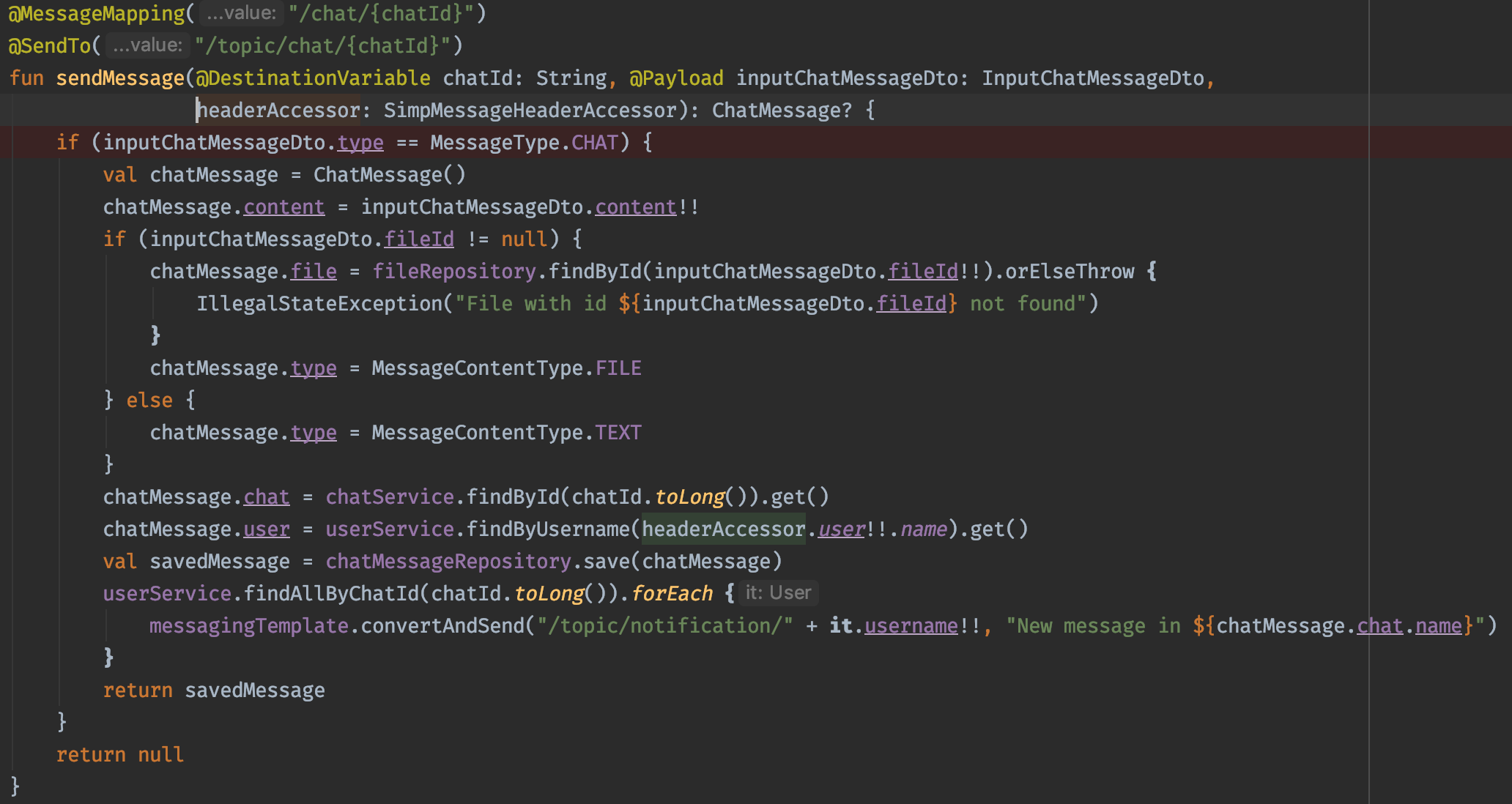


Рисунок 18. Метод контроллера для отправки сообщений по WebSocket

При получении сообщения в контроллере, оно сначала сохраняется в базу данных, а затем отправляется в необходимый топик.

Сервер поддерживает отправку текстовых сообщений, а также сообщений с файловыми вложениями. В таком случае сначала происходит сохранение файла через отдельный контроллер, а затем передача его id в сообщении серверу.

* + 1. **Поднятие тем на обсуждение в чатах.**

Возможностью создания чатов обладает только председатель совета дома. Для создания чата необходимо передать серверу JSON, содержащий вопрос, идентификатор чата в системе, а также строковый массив возможных ответов на вопрос.

* + 1. **Голосование в чатах.**

Принять участие в чатах могут все пользователи, имеющие доступ к чату. Инструмент голосования был добавлен в систему для решения проблемы голосования на общем собрании собственников квартир, однако не существует ограничений на использование данной функции в чатах, поэтому при создании голосования в чате с УО принять участие в голосовании сможет также и менеджер УО.

При голосовании необходимо отправить запрос к серверу, содержащий id выбираемого варианта. После этого счетчик голосов за выбранный ответ будет увеличен на 1.

* + 1. **Отображение профиля УО.**

Для отображения профиля УО клиент отправляет запрос содержащий id УО в системе. Ответом на данный запрос являются данные об управляющей организации.

Для того, чтобы отправить id, пользователь изначально выбирает интересующую его организацию из списка. Логика получения списков для менеджеров и для жителей дома отличается. Так как сервер предусматривает наличие нескольких ролей у пользователя приложения, то существует два различных метода контроллера для получения списка УО – для менеджера и для жителя. Подобные вещи делают использование приложения более комфортным для конечного пользователя.

* + 1. **Внесение предложение и обозначение проблем.**

В профиле управляющей организации жильцы дома могут оставлять свои предложения, а также обозначать существующие проблемы. Обратная связь разделена на две категории для удобства просмотра.

Для сохранения проблемы или предложения клиенту необходимо передать на сервер запрос, содержащий id управляющей организации, введенный пользователем текст, а также тип сообщения – проблема или предложение.

Менеджеры УО могут только просматривать данные списки.

* + 1. **Рассылка объявлений жильцам.**

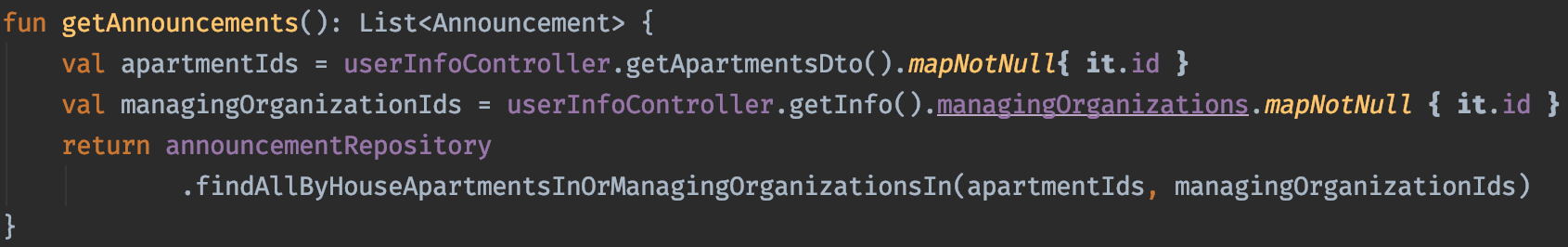


Рисунок 19. Получение списка объявлений.

При получении списка объявлений сначала происходит получение списка домов, в которых у пользователя имеются квартиры, затем выполняется поиск списка домов, управляемых УО, в которой состоит пользователь. После этого выполняется еще один запрос к базе данных для получения списка объявлений из идентификаторов полученных раннее списков.

Создавать новые объявления может только менеджер УО.

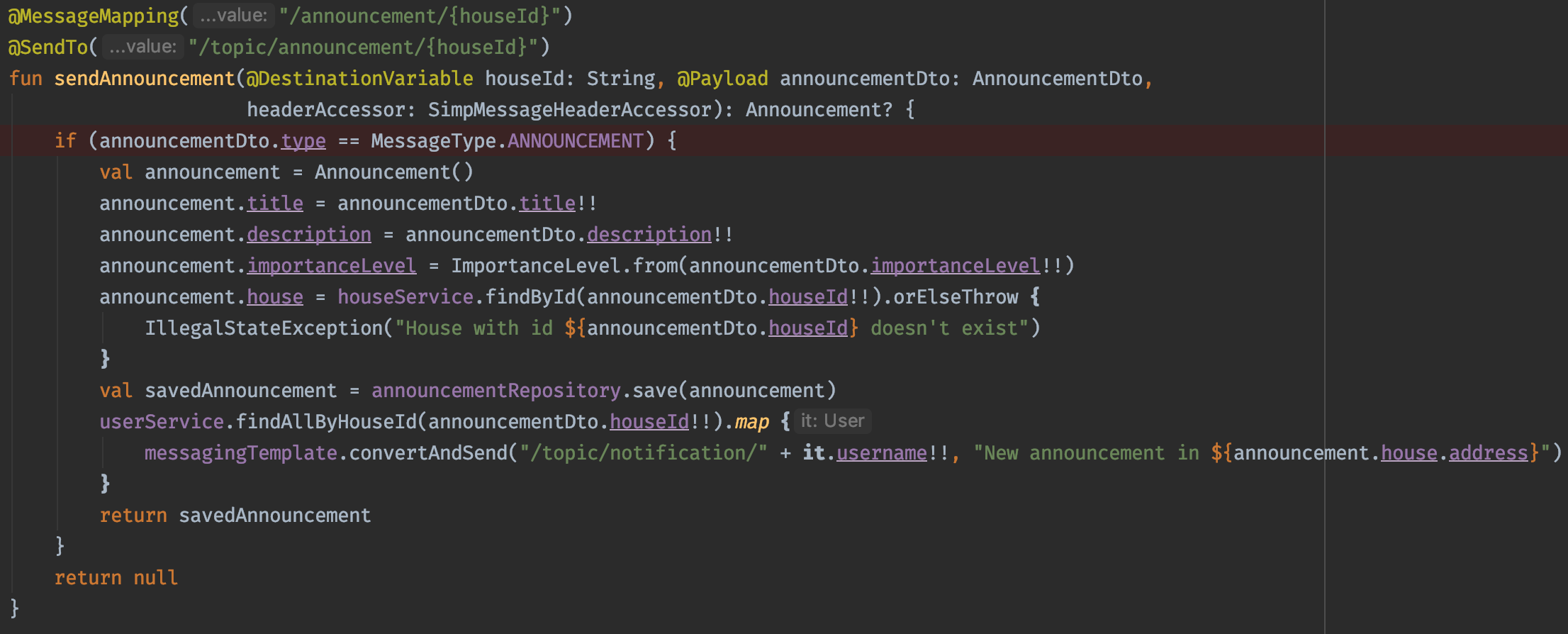


Рисунок 20. Метод контроллера для отправки объявлений по WebSocket.

Отправка объявлений – одна из важнейших функций приложения. Поэтому для данной сущности также поддерживается отправка с помощью WebSocket. Аналогично механизму ведения чатов объявление сперва сохраняется в базу данных, а затем отправляется в необходимый топик. При отправке уведомления необходимо указать идентификатор дома в системе, описание, заголовок, степень важности уведомления, а также его тип. В данный момент единственный возможный тип уведомления – текстовый.

* + 1. **Уведомление о новых событиях.**

Как было описано ранее, за событие принимается только обновляемые в реальном режиме данные, то есть сообщения в чатах и объявления. Для каждого пользователя существует свой уникальный топик для отправки уведомлений.



Рисунок 21. Метод контроллера для отправки уведомлений по WebSocket

На рис. 18 показан процесс отправки сообщения в метод контроллера для работы с уведомлениями. При получении нового сообщения в чате происходит поиск всех пользователей данного чата, после чего для каждого из них вызывается метод контроллера, который отправляет полученное сообщение в персонифицированный топик.

На рис. 20 показан аналогичный процесс, однако метод контроллера вызывается уже при получении объявления. Стоит отметить, что при получении объявления уведомление отправляется не всем пользователям, которым доступно данное объявление, а только жителям и председателям. Данная логика обусловлена тем, что создать объявление может только менеджер УО, а значит, что его можно считать отправителем по умолчанию. Следовательно, данный пользователь не нуждается в отправке уведомления о данном событии.

* + 1. **Отображение профиля пользователя.**

Для получения информации о пользователе клиент делает GET-запрос без параметров к соответствующему методу контроллера. Идентификатор текущего пользователя получается средствами Spring Security, затем по его id выполняется запрос к базе данных на получение сущности user, после чего полученный результат возвращается клиенту.

* + 1. **Вызов мастера на дом.**

Житель многоэтажного дома и председатель совета имеют возможность вызова мастера по адресу дома. Для создания заявки необходимо передать на сервер адрес дома и описание проблемы. Жителю и председателю доступен список поданных им обращений с указанием их статуса.

Менеджер УО не может создавать новые заявки на вызов специалиста в системе, однако пользователь с данной ролью может просматривать список уже существующих заявок, а также изменять их статусы.

Первоначально заявка создается в статусе «на рассмотрении» (“UNDER\_REVIEW”). Для изменения статуса заявки клиент может обращаться к двум различным методам контроллера: accept и updateStatus. При обращении к первому произойдет перевод заявки в статус «подтверждена» («ACCEPTED»). Данный метод выделен отдельно для быстрого подтверждения заявок.

При запросе клиента ко второму методу с указанием статуса заявки сервер переводит заявку в указанный статус.

Сам процесс взаимодействия менеджера управляющей организации и специалиста не контролируется системой.

* + 1. **Оценка работы приложения посредством формы обратной связи.**

Жители, председатели и менеджеры УО имеют возможность оставить отзыв о приложении. Для этого со стороны клиента ожидается запрос к методу контроллера, содержащий в себе оценку по шкале от 0 до 5 с шагом 0.5, а также текстовое описание.

Администратор не может создавать новые отзывы, однако имеет доступ к списку имеющихся в системе отзывов.

* 1. **Документирование API приложения.**

Ниже отображены все доступные методы всех контроллеров приложения с указанием URL, по которым они доступны. Также приведен пример подробной спецификации одного из методов. Контроллеры сгенерированы с помощью Swagger [11], фреймворка для описания и документации REST API.

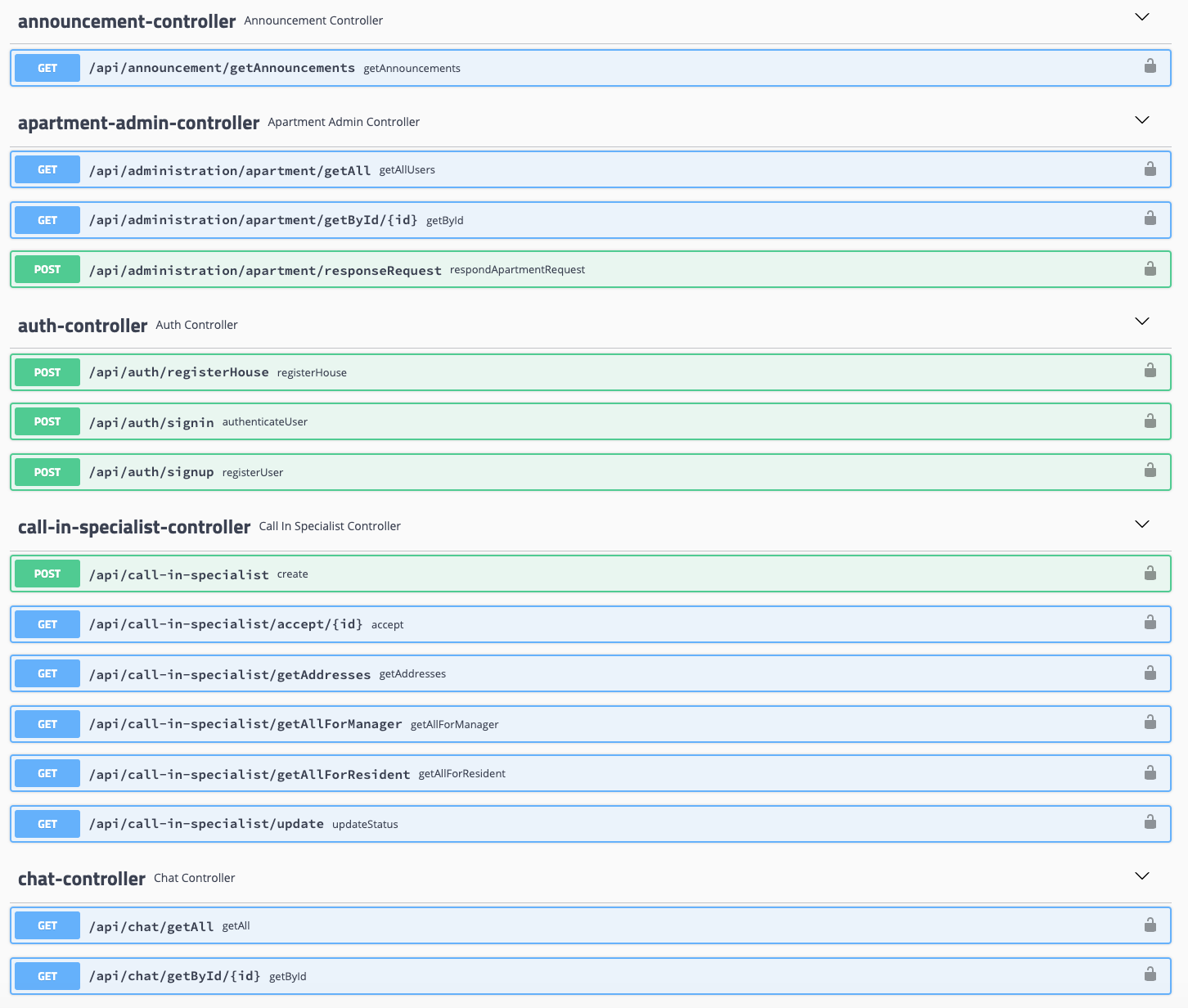


Рисунок 22. Контроллеры для работы с объявлениями, механизмом авторизации и регистрации, вызовами специалистов, чатами, а также контроллер администратора для работы с квартирами.

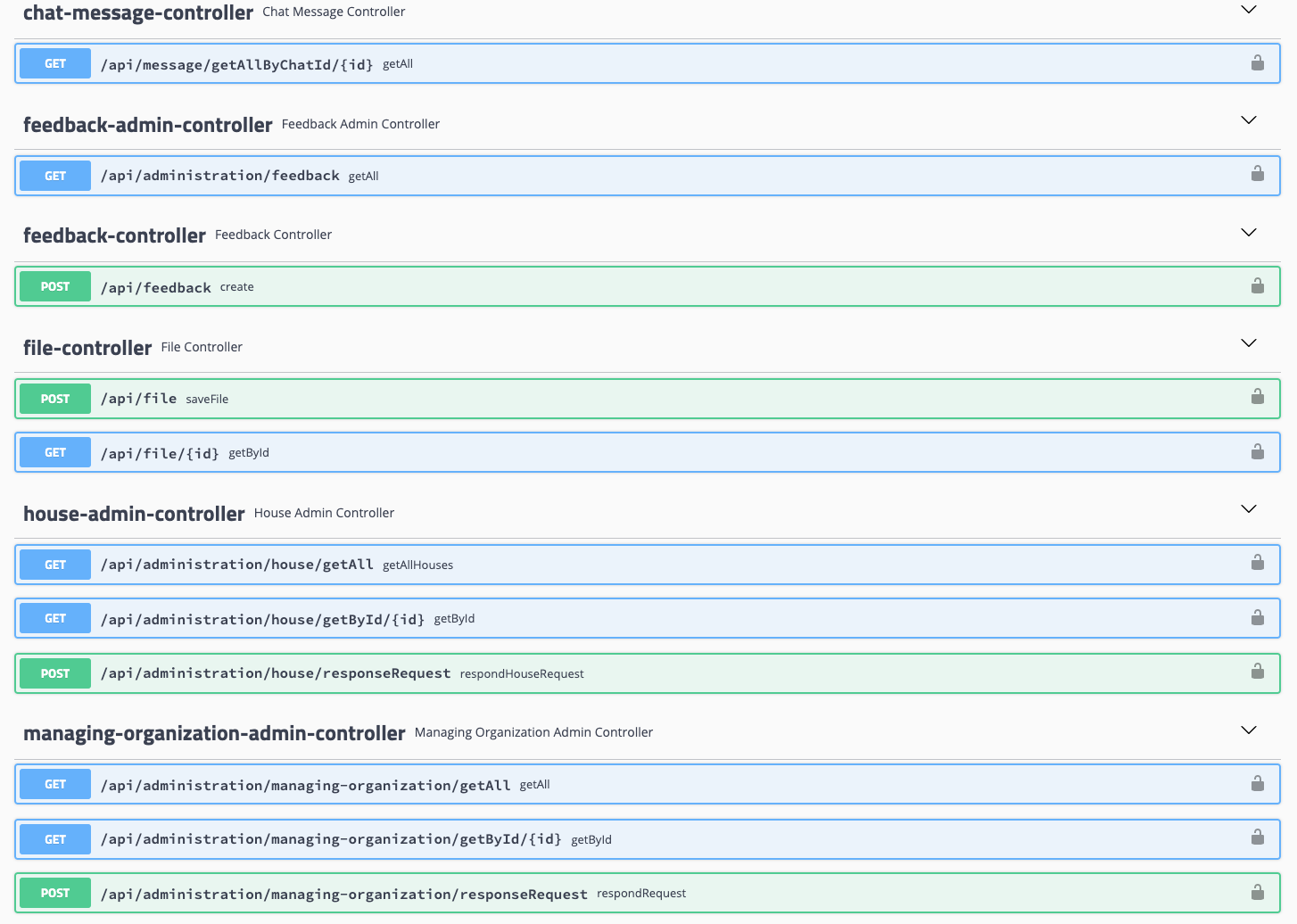


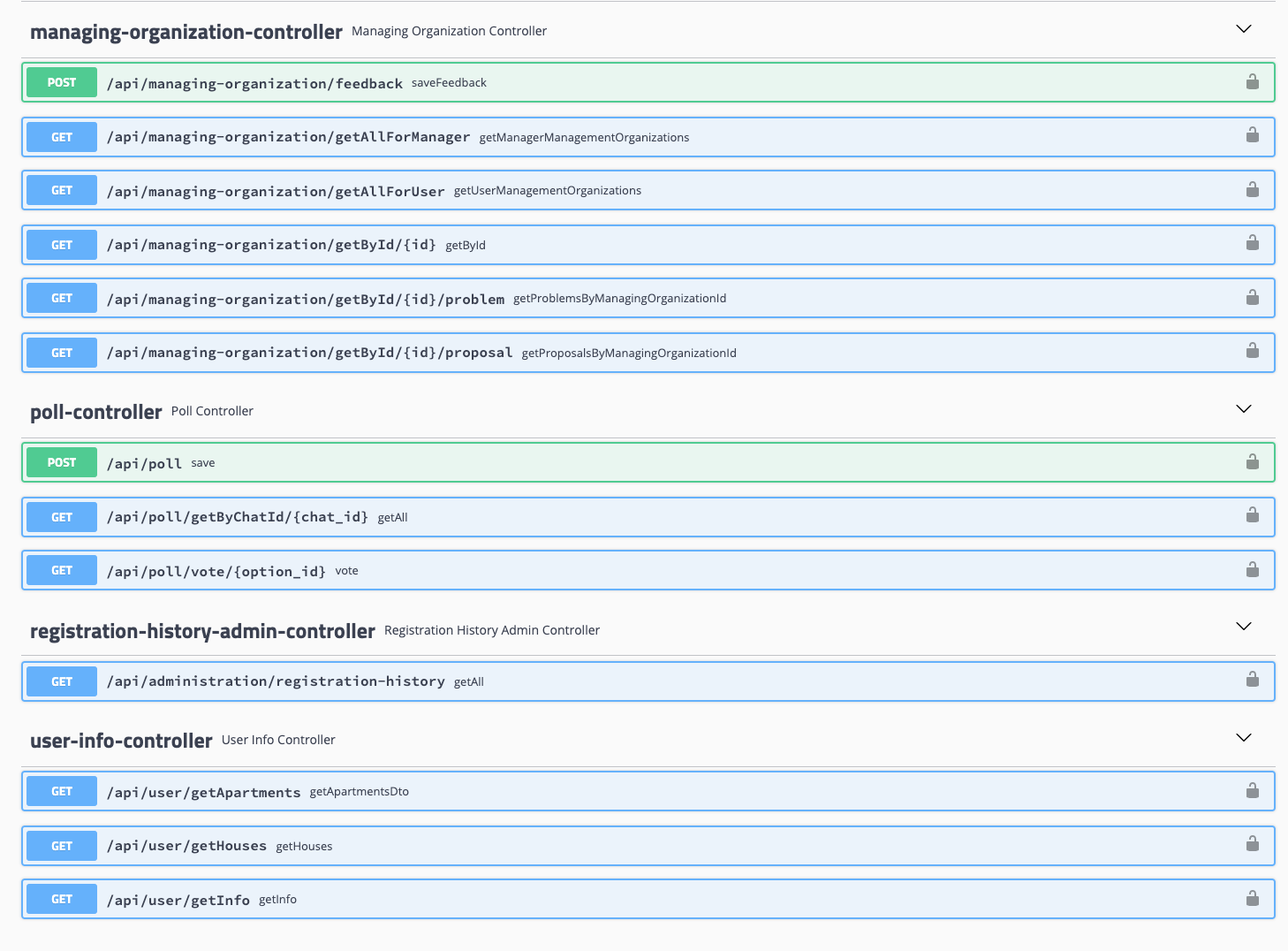
Рисунок 23. Контроллеры для работы с сообщениями, обратной связью, файлами и контроллеры администратора для взаимодействия с отзывами, домами и УО.

Рисунок 24. Контроллеры для работы с УО, опросами, информацией о пользователях, а также контроллер для получения истории регистрации.

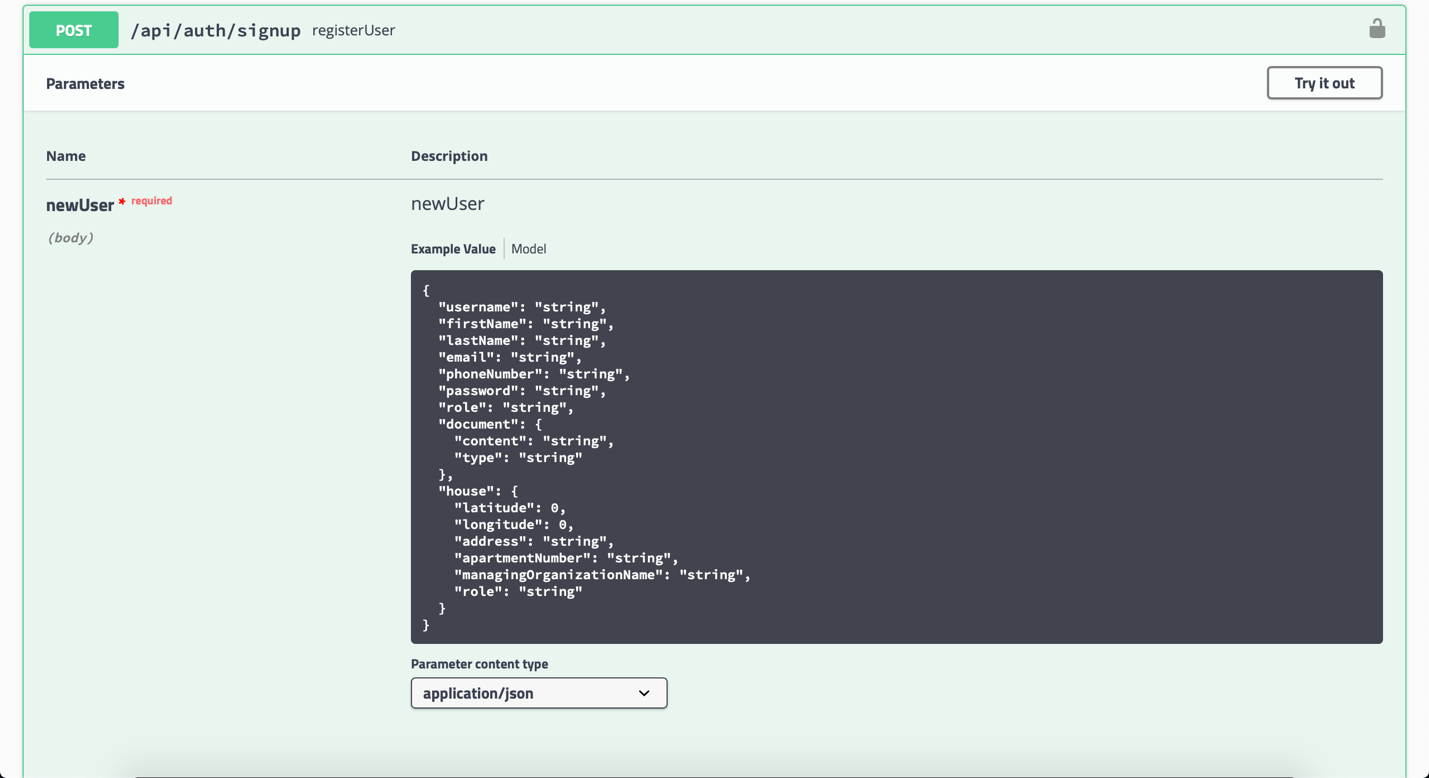


Рисунок 25. Пример спецификации API на основе метода регистрации.

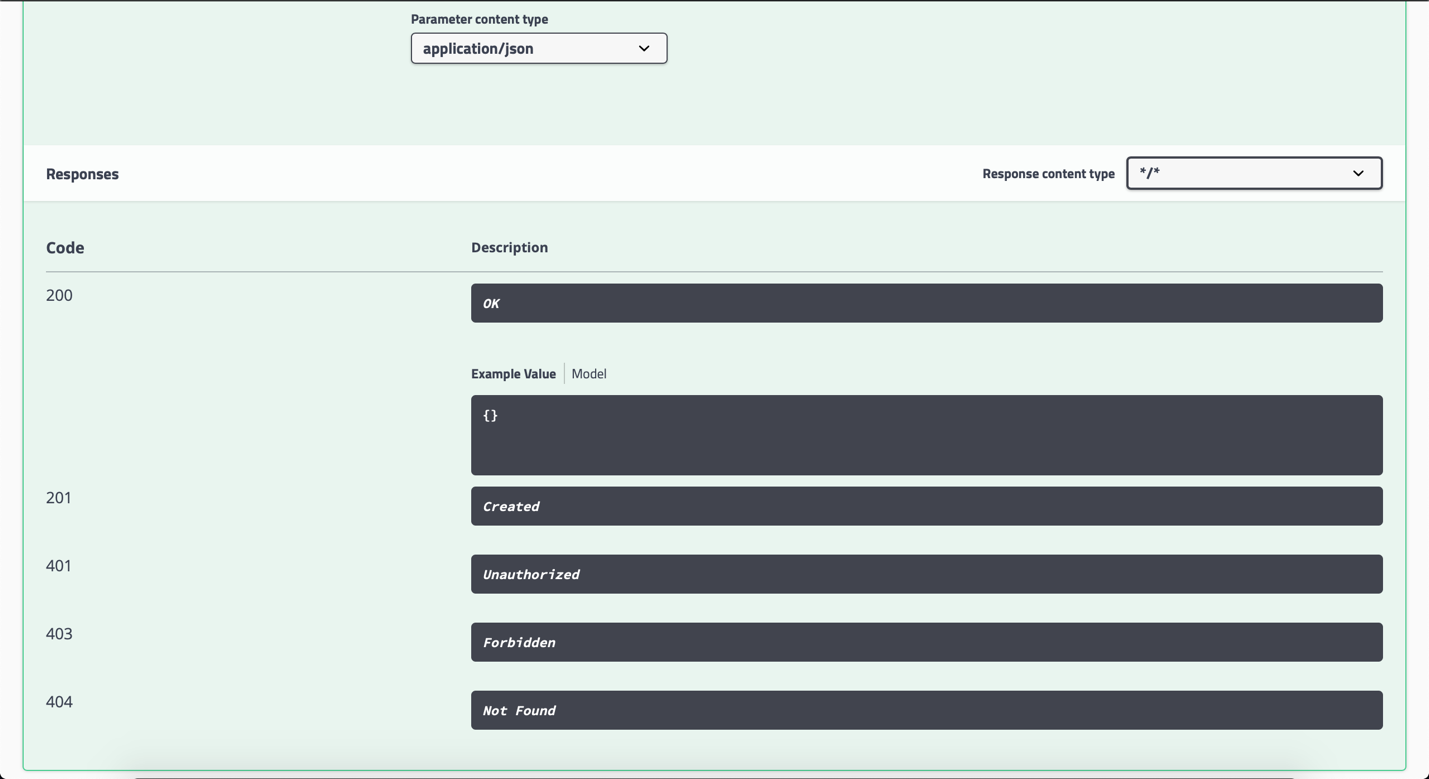


Рисунок 26. Пример спецификации API на основе метода регистрации.

**Выводы по главе:**

В третьей главе приведено описания стека технологий для реализации сервера, механизма обеспечения безопасности, способов реализации функциональных требований, а также список доступных для взаимодействия методов контроллеров.

В следующей главе будет описан подход к реализации клиентской части.

1. **Описание работы клиентской части приложения**

Клиентская часть является не менее важным составляющим приложения. В данной главе приведено описания стека технологий для реализации клиентской части, механизма работы с данными и способов реализации функциональных требований.

* 1. **Стек технологий.**

Клиентская часть приложения написана с использованием React.JS [12] – Javascript-фреймворком, разработанным для удобного создания пользовательских интерфейсов. В проекте также используется фреймворк Material-UI [13], предоставляющий разработчику огромный выбор готовых React-компонентов, выполненных в стиле material design, для разработки. В качестве исходной темы была выбрана тема Material Dashboard [14], реализованная с использованием компонентов из Material-UI. Для управления состоянием приложения используется React Redux [15].

Клиентская часть приложения получает данные с сервера исключительно с помощью HTTP-запросов. Вся логика по удалению, изменению, созданию или чтению объектов обязательно происходит посредством таких запросов, клиентская сторона не вносит изменения и не читает информацию из базы данных напрямую. Для создания HTTP-запросов в программе используется Javascript-библиотека Axios [16]. Она обладает легким и понятным синтаксисом, а также легко интегрируется в проект, значительно упрощая составление запросов к серверу и обработку ответов.

* 1. **Работа с данными.**

Обработка и хранение данных производится не только на серверной части приложения, но и на клиентской.

Клиент получает данные по мере необходимости при переходе на ту или иную страницу и не сохраняет получаемую информацию. Однако для произведения запросов необходимо добавлять в заголовок запроса токен, полученный при авторизации от сервера. Данный токен сохраняется в локальное хранилище браузера при получении от сервера и удаляется из хранилища при выходе из приложения.

Важнейшим инструментом для работы с данными является React Redux. Клиентская часть приложения состоит из множества компонентов, а это означает, что данные между ними необходимо каким-то образом передавать. Для этого создается store, хранящий в себе дерево состояний приложения. Наличие данного store позволяет не передавать данные из одного компонента в другой и позволяет уменьшить зависимость между компонентами.

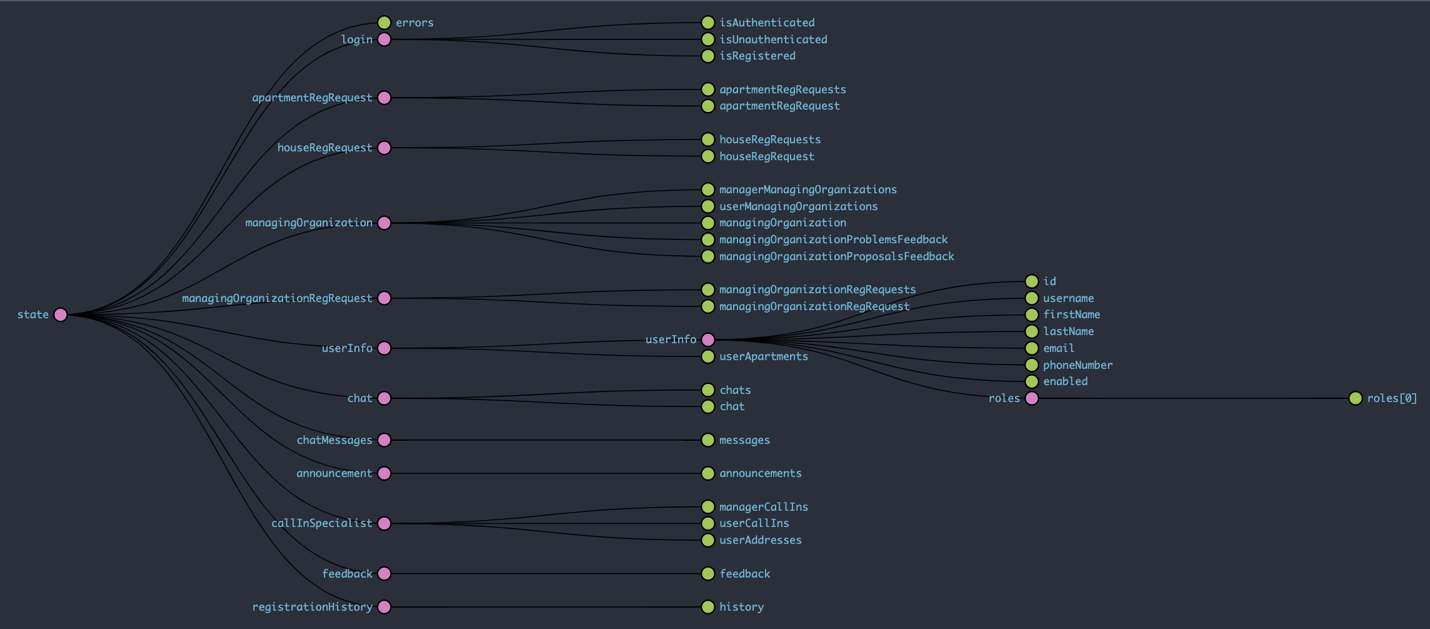


Рисунок 27. Дерево состояний приложения.

* 1. **Реализация функциональных требований**
     1. **Регистрация и авторизация.**

При регистрации пользователь должен заполнить форму с информацией о пользователе – first name, second name, phone number, email и password. Затем пользователь должен выбрать свой дом с помощью поиска по карте.

Для работы с картой используется библиотека Leaflet [17]. Данная библиотека использует данные OpenStreetMap [18], представляющая собой открытую базу данных картографических объектов. С помощью Leaflet пользователь вводит адрес дома, получая в ответ список существующих домов на выбор.

После выбора дома пользователь должен выбрать роль в системе. Доступные для выбора роли: житель, председатель, менеджер.

При выборе первых двух ролей пользователю будет доступно для ввода поле номера квартиры. Если была выбрана роль менеджера, то пользователь должен будет ввести наименование УО в соответствующее поле.

Затем пользователь должен приложить документ, подтверждающий владение квартирой или принадлежность УО.

Завершить регистрацию необходимо нажав чекбокс «Согласен с правилами» (“Agree with terms”) и кнопку регистрации. После нажатия на кнопку регистрации клиент отправляет запрос к серверу, передавая все введенные данные, и перенаправляет пользователя на форму входа в систему.

Для входа в систему пользователь должен ввести email и password. Если пользователь ввел данные корректно и его запрос на регистрацию был подтвержден администратором, то сервер отправляет пользователю JSON, содержащий токен. Данный токен сохраняется в локальное хранилище браузера, и пользователь перенаправляется на страницу просмотра профиля внутри системы.

* + 1. **Поддержка различных ролей пользователей приложения.**

Для фильтрации контента в зависимости от роли пользователя клиент получает данные из JWT-токена.



Рисунок 28. Получение ролей из токена.

При отображении контента используется условный рендеринг компонентов в зависимости от роли пользователя в системе.

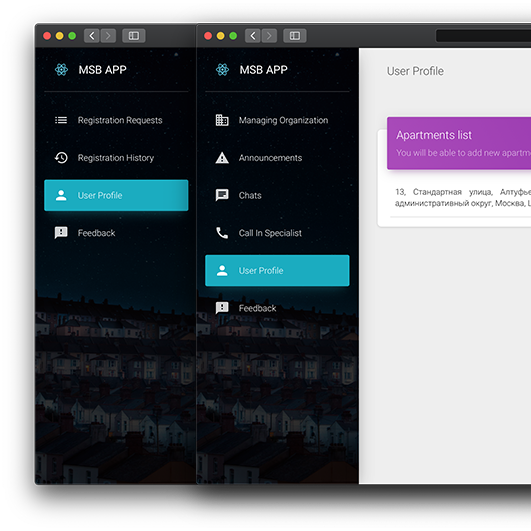


Рисунок 29. Разница в списках доступных страниц для пользователей с ролью «администратор» и остальными.

* + 1. **Участие в чатах.**

При переходе на страницу чатов клиент делает запрос на получение всех доступных пользователю чатов. При выборе одного из чатов сначала происходит загрузка истории сообщений с сервера, а затем – подключение к каналу чата по WebSocket. При поступлении новых сообщений в канале сообщение добавляется в store, где хранятся все сообщения выбранного чата, а затем происходит обновление списка сообщений.



Рисунок 30. Отображение списка сообщений.

Пользователь может не только получать, но и отправлять сообщения. Возможно отправление сообщений в текстовом и файловом форматах. В случае добавления файлового сообщения сначала происходит сохранение файла с помощью запроса к контроллеру файлов, а затем идентификатор этого файла добавляется к отправляемому сообщению. Отправке сообщения на сервер происходит по каналу WebSocket.

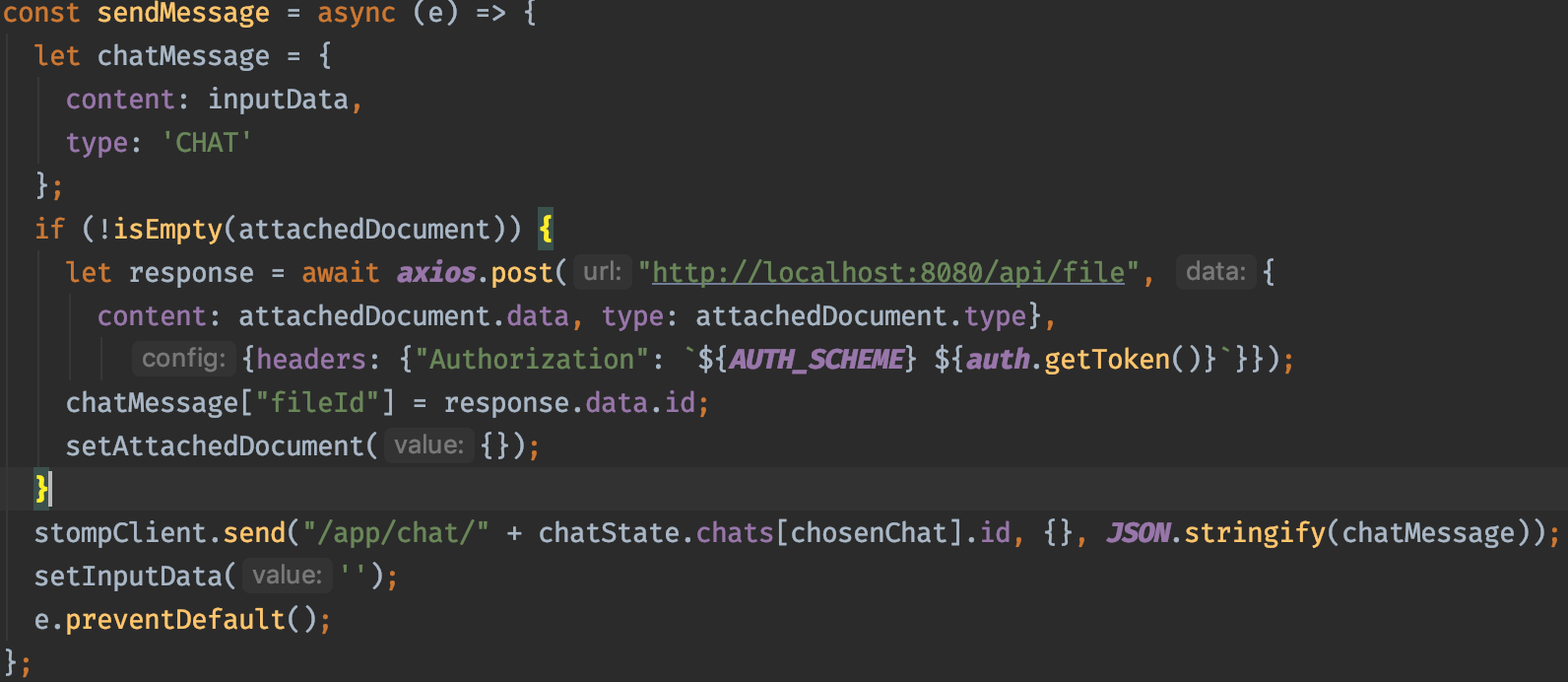


Рисунок 31. Отправление сообщений.

* + 1. **Создание голосований и участие в них.**

Создание голосований может быть произведено председателем совета в чате. Для этого необходимо прежде выбрать чат и перейти к списку сообщений в чате. Над данным списком необходимо нажать кнопку добавления нового опроса. После этого над списком сообщений появится компонент создания голосований. Председатель должен заполнить поле вопроса, а также добавить необходимые варианты ответа на него. Затем клиент отправляет запрос серверу, содержащий идентификатор чата, вопрос и список возможных ответов, а компонент создания голосований закрывается.

Для участия в голосовании необходимо открыть список доступных голосований. Для этого существует отдельная кнопка в интерфейсе. После нажатия данной кнопки над списком сообщений появляется список голосований. При выборе одного голосования из списка происходит открытие модального окна, в котором пользователь должен сделать выбор из доступных вариантов. Пользователь может проголосовать только в том случае, если до этого он не принимал участия в данном голосовании. После выбора необходимого варианта ответа на сервер отправляется запрос, содержащий id выбранного варианта ответа, а окно голосования и список голосований закрываются. При повторном просмотре данного голосования голос пользователя уже будет отмечен.

* + 1. **Отображение профиля УО и создание проблем и предложений.**

При переходе на страницу «Управляющие организации» клиент делает запрос на получение списка управляющих организаций по идентификатору пользователя. Для роли менеджера УО и остальных ролей (житель и председатель) адреса запросов и списки отличаются. Для менеджера УО загружается список УО, в которых он является менеджером, а для жителя и председателя – список УО, управляющих домами, в которых проживает данный пользователь.

После выбора определенной УО из списка происходит переход на форму просмотра профиля УО и клиент выполняет 3 запроса к серверу: для получения основной информации об управляющей организации, для получения списка проблем и для получения списка предложений.

В списках проблем и предложений у жителя и председателя присутствуют кнопки добавления новых записей. После нажатия на кнопку происходит открытие поля для введения описании проблемы или предложения. После нажатия кнопки «сохранить» происходит запрос добавления на сервер, а сама запись добавляется в store и отображается в соответствующем списке.

* + 1. **Рассылка объявлений жильцам.**

Механизм по рассылке объявлений похож на механизм работы чатов. При переходе на страницу объявлений происходит получение с сервера списка домов пользователя под управлением управляющей организации, в которой данной пользователь является менеджером, или домов, в которых проживает данный пользователь, и, используя идентификаторы этих домов, клиент подключается к WebSocket-каналам объявлений. Также при загрузке страницы объявлений происходит получение всех уже существующих объявлений с помощью запроса к серверу. Все полученные с помощью WebSocket и прямых запросов к серверу данные сохраняются в store приложения и отображаются в списке объявлений. Список объявлений в приложении представлен в виде отдельных друг от друга компонентов карт, чтобы пользователь сразу получал доступ к подробному описанию уведомлений.



Рисунок 32. Отображение списка объявлений.

Менеджер УО может создавать новые объявления. Для этого у пользователя с данной ролью доступна дополнительная карточка, в которой он должен указать заголовок, выбрать адрес дома из списка, заполнить подробное описание и выбрать уровень важности объявления. Объявления разного уровня важности отображаются на карточках разного цвета. Затем менеджеру необходимо нажать на кнопку «сохранить», после чего новое сообщение отправляется на сервер по каналу WebSocket.

* + 1. **Ведение истории регистраций.**

Логика работы истории регистрации целиком расположена на сервере. На клиентской части приложения существует отдельная страница со списком истории регистраций, доступная пользователю с ролью администратор. При переходе на данную страницу клиент делает запрос к серверу для получения данного списка.

* + 1. **Уведомление о новых событиях.**

Для вывода уведомлений существует отдельный список, доступный в панели навигации приложении, расположенной в верхней части экрана. Кнопка открытия списка уведомлений содержит в себе маркер количества непрочитанных. После авторизации в приложении клиент подписывается на канал уведомлений, используя имя пользователя. При появлении нового уведомления оно добавляется в store приложения, откуда уже попадает в список уведомлений. Маркер количества непрочитанных уведомлений привязан к размеру списка и обновляется автоматически.

После нажатия на кнопку открытия списка пользователю становится видны все непрочитанные уведомления, а после его закрытия данный список очищается в store, и количество непрочитанных обновлений становится равным 0.

* + 1. **Отображение профиля пользователя с информацией о нем.**

Страница профиля пользователя является стартовой при входе в приложение. При открытии данной страницы выполняется 2 запроса к серверу: получение основной информации о пользователе и списка квартир пользователя. Данная страница доступна для всех пользователей вне зависимости от их ролей.

* + 1. **Вызов мастера на дом.**

При переходе на страницу со списком вызовов мастера производится 3 запроса к серверу: получение списка запросов для жителей и председателя, получение списка запросов для менеджеров и получения списка адресов пользователя. Для менеджеров и остальных пользователей списки отличаются.

В списке запросов житель и председатель могут нажать на кнопку создания нового запроса. После этого пользователям с данными ролями необходимо ввести описание проблемы и выбрать адрес из списка. После нажатия кнопки «сохранить» происходит запрос к серверу на сохранение новой заявки, затем заявка добавляется в store и попадает в список заявок пользователя.

В списке запросов менеджер может управлять статусами запросов. Если статус запроса находится в статусе «на рассмотрении» (“UNDER\_REVIEW”), то у менеджера присутствует кнопка подтверждения запроса. При нажатии на нее происходит запрос к соответствующему контроллеру сервера. Также у менеджера присутствует возможность редактирования статуса заявки через окно просмотра подробной информации. После изменения статуса и нажатии кнопки сохранения происходит запрос на обновление статуса заявки с указанием нового статуса. После осуществления запросов на изменения статуса заявки происходит обновление данных в store и списке заявок.

* + 1. **Оценка работы приложения посредством формы обратной связи.**

Для обратной связи существует отдельная страница. Для роли администратор данная страница представляет собой список существующих в системе отзывов о приложении, полученных с помощью запроса к серверу.

Для остальных пользователей приложения данная страница представляет компонент оценки, после нажатия на который появляется поле для ввода описания. При нажатии кнопки сохранить данные отправляются на сервер, а вместо формы появляется надпись «Спасибо».

**Выводы по главе**

В четвертой главе приведено описания стека технологий для реализации клиентской части, механизма работы с данными и способов реализации функциональных требований.

**Заключение**

В рамках выполнения выпускной квалификационной работы было разработано клиент-серверное приложение для взаимодействия жильцов многоэтажного дома.

Для этого были решены следующие задачи:

1. Реализована регистрация в приложении;
2. Реализована авторизация в приложении;
3. Добавлена поддержка различных ролей в приложении;
4. Реализован механизм чатов, разделенных по тематике;
5. Добавлена возможность поднятия тем на обсуждение в чатах;
6. Добавлен механизм голосования;
7. Добавлена возможность просмотра профиля УО;
8. Реализован механизм внесения предложений и обсуждения проблем между жильцами и управляющей организацией;
9. Добавлена возможность рассылки объявлений жильцам;
10. Реализован механизм ведения истории регистраций;
11. Реализован механизм уведомлений;
12. Добавлена страница профиля пользователя с информацией о нем;
13. Реализована возможность вызова мастера на дом;
14. Добавлена возможность оценки работы приложения посредством формы обратной связи;

В дальнейшем планируется реализовать возможность управления несколькими квартирами в различных домах и наличие у пользователя нескольких ролей. Данное нововведение является крайне необходимым, однако внесет существенные изменения в обе части приложения. Архитектура и логика сервера поддерживает данное поведение, однако на стороне клиента необходимо внести существенные изменения. Также планируется добавить поддержку локализации.

**Список использованных источников**

1. GEOMATRIX. [Электронный ресурс]// URL: https://geomatrix-retail.com/statistics-and-reality-in-moscow-region/ (Дата обращения: 18.02.2020, режим доступа: свободный).
2. Telegram. [Электронный ресурс]// URL: https://telegram.org/ (Дата обращения: 18.02.2020, режим доступа: свободный).
3. WhatsApp. [Электронный ресурс]// URL: https://www.whatsapp.com/ (Дата обращения: 18.02.2020, режим доступа: свободный).
4. PropertyVISTA. [Электронный ресурс]// URL: https://www.propertyvista.com/ (Дата обращения: 18.02.2020, режим доступа: свободный).
5. Hudson. [Электронный ресурс]// URL: https://hudson.im/ (Дата обращения: 18.02.2020, режим доступа: свободный).
6. PostgreSQL. [Электронный ресурс]// URL: https://www.postgresql.org/ (Дата обращения: 18.02.2020, режим доступа: свободный)
7. JSON. [Электронный ресурс]// URL: https://www.json.org/ (Дата обращения: 18.02.2020, режим доступа: свободный)
8. Spring Framework. [Электронный ресурс]// URL https://spring.io (Дата обращения: 18.02.2020, режим доступа: свободный).
9. Kotlin. [Электронный ресурс]// URL: https://kotlinlang.org (Дата обращения: 18.02.2020, режим доступа: свободный)
10. Java. [Электронный ресурс]// URL: https://www.java.com/ru/ (Дата обращения: 18.02.2020, режим доступа: свободный)
11. Swagger [Электронный ресурс]// URL: https://swagger.io/ (Дата обращения: 19.05.2020, режим доступа: свободный)
12. React.JS. [Электронный ресурс]// URL: https://reactjs.org/ (Дата обращения: 18.02.2020, режим доступа: свободный)
13. Material-UI. [Электронный ресурс]// URL: https://material-ui.com (Дата обращения: 18.02.2020, режим доступа: свободный)
14. Material Dashboard. [Электронный ресурс]// URL: https://www.creative-tim.com/product/material-dashboard-react (Дата обращения: 18.02.2020, режим доступа: свободный)
15. React Redux. [Электронный ресурс]// URL: https://react-redux.js.org (Дата обращения: 18.02.2020, режим доступа: свободный)
16. Axios. [Электронный ресурс]// URL: https://github.com/axios/axios (Дата обращения: 18.02.2020, режим доступа: свободный)
17. Leaflet [Электронный ресурс]// URL: https://leafletjs.com/ (Дата обращения: 19.05.2020, режим доступа: свободный)
18. OpenStreetMap [Электронный ресурс]//URL: https://www.openstreetmap.org/ (Дата обращения: 19.05.2020, режим доступа: свободный)