Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Институт компьютерных и инженерных наук
Кафедра информационных и управляющих систем
Направление подготовки 09.04.04 – Программная инженерия
Направленность (профиль) образовательной программы Управление разработкой программного обеспечения

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой
_____ А.В. Бушманов
«___» _____2025 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему:Проектирование и реализация информационной системы управления недвижимостью в микрорайоне

Исполнитель
студент группы 3105-ом2 _____ Ань Минцзе
(подпись, дата)

Руководитель
доцент, канд. техн. наук _____ А.В. Бушманов
(подпись, дата)

Руководитель научного содержания программы магистратуры
профессор, доктор техн. наук _____ И.Е. Ерёмин
(подпись, дата)

Нормоконтроль
инженер кафедры _____ В.Н. Адаменко
(подпись, дата)

Рецензент
доцент, канд. техн. наук _____ Т.В. Труфанова
(подпись, дата)

Благовещенск, 2025

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Институт компьютерных и инженерных наук
Кафедра информационных и управляющих систем

УТВЕРЖДАЮ
Зав. кафедрой
_____ А.В. Бушманов
«___» _____2024 г.

З А Д А Н И Е

К магистерской диссертации студента группы 3105-ом2 Ань Минцзе

1.Тема магистерской диссертации: Проектирование и реализация информационной системы управления недвижимостью в микрорайоне
(Утверждено приказом от 06.03.2025 № 609-уч)

2. Срок сдачи студентом законченной работы (проекта): 10.06.2025

3.Исходные данные к магистерской диссертации: документация разработчиков, интернет ресурсы, учебная литература

4.Содержание магистерской диссертации (перечень подлежащих разработке вопросов): Управление недвижимостью, проектирование алгоритма решения задачи, разработки системы управления недвижимостью

5. Перечень материалов приложения (наличие чертежей, таблиц, графиков, схем, программных продуктов, иллюстративного материала и т.п.):

6. Рецензент магистерской диссертации: Труфанова Т.В., доцент, канд. техн. наук

7. Дата выдачи задания 29.01.2024

Руководитель выпускной квалификационной работы: А.В. Бушманов, доцент, канд. техн. наук
(фамилия, имя, отчество, должность, уч.степень, уч.звание)

Заявление принял к исполнению _____

РЕФЕРАТ

Магистерская диссертация содержит 77 страница, 12 рисунок, 6 таблицы, 30 источников

ПРОФЕССИОНАЛЬНАЯ ДЕЯТЕЛЬНОСТЬ, МЕНЕДЖМЕНТ

Сегодня, когда общество продолжает развиваться, управлять сообществом становится все сложнее и сложнее. Хорошее управление сообществом имеет набор собственных инструментов управления, которые помогают менеджерам и владельцам управлять сообществом вместе с большими и малыми делами. Именно поэтому система управления имуществом Hope Community Property Services реализована с помощью современных технологий. Анализируя потребности владельцев, система в основном включает в себя две роли - администраторов и жителей. Администраторы отвечают за управление и поддержание основной информации района, информации о жилье, платежной информации, информации о персонале и т. д. Владельцы могут запрашивать через систему информацию о своем жилье, платежные реквизиты, жалобы и предложения и т. д. Она также может обеспечить эффективность получения информации владельцами, что позволяет сэкономить время сотрудников отдела недвижимости на хранение документов. Она экономит время застройщиков, сокращая дублирование работы. Эта система в основном сокращает время, затрачиваемое администратором на управление сообществом, и помогает владельцам иметь представление о состоянии сообщества.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Сегодня, когда общество продолжает развиваться, управление микрорайонами становится все более сложной задачей. Хорошее управление сообществом имеет набор собственных инструментов управления, которые помогают менеджерам и владельцам управлять сообществом вместе с большими и малыми делами. Поэтому мы надеемся реализовать систему управления имущественными услугами сообщества с помощью современных технологий. Анализируя потребности владельцев, система в основном включает в себя две роли - администраторов и жителей. Администраторы отвечают за управление и поддержание основной информации района, информации о жилье, платежной информации, информации о персонале и т. д. Владельцы могут запрашивать через систему информацию о своем жилье, платежные реквизиты, жалобы и предложения и т. д. Она также может обеспечить эффективность получения информации владельцами, что позволяет сэкономить время персонала по хранению документов. Это экономит время застройщиков, сокращая объем повторяющейся работы. Эта система в основном сокращает время, затрачиваемое администратором на управление сообществом, и помогает владельцам иметь представление о состоянии сообщества. Общинная собственность «Надежда» - это место для предоставления общественных услуг, основной функцией которого является предоставление услуг по оказанию помощи для удовлетворения различных потребностей владельцев. В связи с постоянным реформированием качества жизни, особенно управления недвижимостью, обслуживания жилья и развития и глубины управления недвижимостью работа становится все более тяжелой и сложной. Каждый регион создает свое собственное системное программное обеспечение, с другой стороны, он также постоянно развивает и совершенствует систему программного обеспечения, формируя определенный масштаб системы информационного строительства. Благодаря использованию Интернета и технологии Интернета вещей, интеллектуальному управлению и автоматизированной работе,

изменяется режим традиционного управления недвижимостью, повышается удовлетворенность владельцев и уровень обслуживания недвижимости. В то же время система не только помогает управляющим собирать, анализировать и обрабатывать данные для достижения всеобъемлющего управления информацией, но и обеспечивает жильцам более качественные услуги и условия проживания. Сотрудники могут просматривать содержание принадлежащей им работы и быстрее просматривать информационный контент.

1 ИССЛЕДОВАНИЯ В ОБЛАСТИ СИСТЕМ УПРАВЛЕНИЯ НЕДВИЖИМОСТЬЮ

1.1 Современное состояние исследований в стране и за рубежом

Чэнь Шухун предложил, что с развитием удобного строительства общины управление имуществом перешло от ручной обработки к управлению компьютерной системой, а информатизация управления имуществом стала основным средством управления и сбора ресурсов в общине. Лю Ючжэнь в работе «Разработка и внедрение системы управления имуществом общины на основе Java» предложил, что Java имеет много важных преимуществ и функциональную итеративную оптимизацию, чтобы обеспечить сильную техническую поддержку и гарантию для развития системы управления имущественными услугами в надежде на систему управления имуществом общины. Чжоу Nanyou в «лагерь для повышения» на индустрии управления имуществом и влияние стратегий преодоления для управления имуществом для персонала, чтобы уменьшить дублирование строительства проблемы, экономия времени и затрат. Лу Синай и др. при разработке и внедрении интеллектуальной системы управления недвижимостью предложили, что самое главное в современном управлении недвижимостью это обеспечить владельцам удобство и простоту использования.

Арати Пол предположил, что система управления недвижимостью в основном имеет дело с географическими областями, которые включают мониторинг и управление информацией, связанной с недвижимостью, которая основана на платформе ГИС на базе Oracle с использованием бесплатной настройки ArcGIS Explorer (AGX). Алиса Кристудасон заявила, что влияние сингапурского законодательства на управление общей недвижимостью говорит о том, что страна сама небрежно относится к статистике иностранного населения и имеет неполную информацию о своих жителях, что приводит к неполной функциональности системы. Прежде чем внедрять такую технологию, необходимо провести всестороннюю перепись населения и подсчитать

информационные ресурсы, чтобы в полной мере раскрыть ценность системы. Tian Yanxiao, Shi Qingsong в навыки компьютерного программирования и поддержания важности высокой параллельности Java веб-разработки модели на основе SpringBoot также имеет приложения в различных областях. Такие как SpringBoot основе промышленного контроля конфигурации программного обеспечения веб-публикации системы проектирования и реализации SpringBoot в других аспектах роли SpringBoot, который показывает, что в современном управлении информацией, SpringBoot в качестве структуры разработки имеет широкий спектр применения значение.

Подводя итог, можно сказать, что современная отечественная система управления недвижимостью имеет недостатки в функциональности, а также некоторые проблемы в безопасности системы, в то время как зарубежная система управления недвижимостью имеет комплексные функции, но построение инфраструктуры еще не совершенна. Поэтому нам необходимо проанализировать ситуацию с системой управления недвижимостью в стране и за рубежом, в сочетании с новейшими компьютерными технологиями, чтобы добиться улучшения функций системы управления недвижимостью и строительства инфраструктуры. Это может повысить эффективность управления недвижимостью и качество услуг, удовлетворить спрос жителей на услуги по управлению недвижимостью, способствовать развитию индустрии управления недвижимостью, а также предоставить новые идеи и направления для глобализации управления недвижимостью.

1.2 Содержание исследования

В соответствии со спросом пользователей на услуги по управлению недвижимостью анализируются различные аспекты, обсуждаются различные бизнес-процессы, а система управления недвижимостью сообщества в основном представлена в виде диаграмм. Достичь полностью функциональной, удобной для пользователя, масштабируемой системы управления имущественными услугами, в то время как система имеет связь между моделью базы данных для детального мышления и обучения. Реализация надежды на то, что система

управления имущественными услугами сообщества: наиболее популярная Java для разработки, а также описание кода, демонстрация администратора и т.д.

1.3 Инструменты разработки и технологический профиль

Инструменты: Tomcat, Mysql.

Технология Введение: Эта система в основном использует технологию Java, в настоящее время является одним из самых основных основных технических языков, технология в основном применима к взаимодействию между сервером и пользователем, как его определение принадлежит к объектно-ориентированной технологии, технология больше подходит для услуг, чтобы взаимодействовать друг с другом. весна безопасности рамки: рамки принадлежит к набору безопасности пользователя аутентификации рамки, использование рамки могут быть настроены на пользователя Весна безопасности Framework: эта структура относится к набору безопасности пользователя аутентификации рамки, использование рамки могут настроить пользователя с различными идентификационными данными, Ge Meng, Ван Ин в компьютерном программировании навыки и обслуживание на основе SpringBoot + SSM рамки для строительства и реализации веб-приложений, предлагаемых различных идентификационных данных могут быть предоставлены пользователю с различными разрешениями для контроля пользователей для достижения пользователя к взаимному взаимодействию. Кай Ванг, занимаясь разработкой и внедрением системы управления учебной базой на основе SpringBoot для высших профессиональных колледжей и университетов, объясняет, что, помимо всего прочего, система Spring Security является безопасной.

1.4 Анализ осуществимости функциональности системы

Анализ осуществимости позволяет определить, насколько осуществима реализация проекта и соответствует ли он реальным потребностям. Анализ осуществимости необходим для определения как достоинств, так и недостатков проекта.

1.4.1 Техническая осуществимость

В качестве бэкэнда в данном проекте используется язык Java и фреймворк

SpringBoot. Java - один из самых популярных языков программирования, обладающий мощными инструментами разработки и вспомогательными библиотеками, которые помогают разработчикам быстро создавать эффективные программные системы. Таким образом, использование системы управления имущественными услугами на базе Java для сообщества «Надежда» технически возможно. В свете глобальных изменений в технологическом ландшафте, таких как внедрение Интернета вещей(IoT),блокчейн-технологий и искусственного интеллекта, система управления недвижимостью также претерпеваетзначительныеизменения.

Все большее внимание уделяется автоматизации процессов, включая управление ресурсами, анализ потребностей пользователей и предсказания на основе больших данных. В современных условиях задачи управления недвижимостью становятся все более сложными, что требует от специалистов гибкости и способности интегрировать различные технологические решения в единую систему. Это включает в себя не только мониторинг состояния объектов недвижимости, но и более глубокий анализ данных, что в свою очередь помогает принять болееобоснованные и своевременные решения. Например, с помощью анализа данных можно прогнозировать потребности в обслуживании

1.4.2 Экономическая целесообразность

Основной код реализации этой системы загружается с официального сайта GitHub, а сопутствующее программное обеспечение этой разработки можно найти в Интернете бесплатно, что в целом удобнее. Таким образом, разработка системы управления имущественными услугами на базе Java для сообщества «Надежда» экономически целесообразна.

1.4.3 Возможность эксплуатации

Данная система предоставляет множество функций. Пользователи могут работать с центром пользователя, просмотром уведомлений, управлением ремонтом, обработкой карт доступа, услугами по считыванию показаний счетчиков, управлением оплатой счетов, оценкой услуг и т.д. на первой странице в соответствии с представлением системы. Администраторам и пользователям

удобно быстро работать. Таким образом, система управления услугами ЖКХ «Надежда» на базе Java является практически осуществимой.

1.4.4 Юридическая осуществимость

Юридическая осуществимость означает, соответствует ли формулировка или реализация закона нормативным требованиям правовой системы и может ли он быть признан правовыми институтами и различными слоями общества. При формулировании закона учитываются такие факторы, как законность самого закона, возможность его реализации, адаптируемость к конкретной ситуации, ожидаемая реакция общества и соответствие закону. Анализ юридической целесообразности, необходимый для данного проекта, в основном касается законов и нормативных актов, связанных с защитой личной конфиденциальной информации пользователей и информации о собственности компании. При сборе и хранении данных необходимо соблюдать законы и правила, касающиеся защиты личной информации. Во-вторых, при разработке и реализации система строго соблюдает соответствующие законодательные положения об авторских правах на программное обеспечение и правах интеллектуальной собственности и не нарушает ничьих прав и интересов. В то же время методы хранения и передачи данных, принятые в системе, соответствуют соответствующим стандартам сетевой безопасности и информационной безопасности, которые защищают конфиденциальность и безопасность данных пользователей. Дизайн и функции системы в основном направлены на управление услугами по управлению имуществом общины, что соответствует соответствующим нормам и стандартам управления услугами по управлению имуществом. В то же время для обеспечения конфиденциальности и безопасности пользовательской информации были приняты такие строгие меры, как аутентификация личности и шифрование данных. Наконец, при разработке и внедрении системы использовался язык программирования Java, который является открытым языком программирования, опирающимся на Интернет и открытые сообщества для технической поддержки и обновления, и обладает высокой степенью стабильности и безопасности, что соответствует национальной политике и

законам и правилам развития программных технологий. Таким образом, система управления имущественными услугами на базе Java для общины «Надежда» является юридически обоснованной.

1.5 Анализ системных требований

1.5.1 Ролевой модуль администратора системы

Регистрация учетной записи: Администратор системы обязуется зарегистрировать свою учетную запись в системе управления. После успешной регистрации, которая включает в себя проверку личных данных и прав доступа, администратор получает возможность войти в систему и начать работу с личным кабинетом, где отображаются все ключевые данные и операции, доступные в рамках его роли. В личном центре можно отслеживать состояние учетной записи, получать уведомления и обновления о текущих задачах, а также управлять личными настройками безопасности.

Управление объявлениями: Администратор обладает правом публиковать объявления в рамках системы управления. Эти объявления могут касаться различных событий в микрорайоне, таких как планы ремонта, новости о состоянии инфраструктуры или изменения в обслуживании. Владелец недвижимости, в свою очередь, может просматривать эти объявления на главной странице системы. Это позволяет эффективно информировать владельцев о текущих и предстоящих событиях, повышая уровень прозрачности и взаимодействия в сообществе.

Аудит владельцев: Для того чтобы владельцы могли получить доступ к системе, им необходимо пройти процесс регистрации, после чего администратор проводит аудит данных, представленных владельцами. Аудит включает проверку персональной информации, а также правомочности владения недвижимостью. Только после успешного завершения аудита владельцы получают доступ к полному функционалу системы. Владелец может изменить свои личные данные, обновить информацию о жилье и запросить различные услуги, доступные через систему.

Управление пользователями: Система управления поддерживает функции

добавления, редактирования, удаления и проверки данных пользователей. Администратор может добавлять новых пользователей, изменять их данные в случае необходимости и удалять учетные записи, которые больше не актуальны или нарушают правила использования системы. Эта функция дает возможность гибко управлять правами доступа и обеспечивать безопасность системы.

Управление персоналом: Владельцы недвижимости могут добавлять информацию о персонале, который работает на территории. Эта информация включает данные о сотрудниках, их должностях и контактных данных. Администратор имеет доступ к этому разделу и может редактировать или удалять данные о персонале, что позволяет контролировать и поддерживать актуальность информации о тех, кто обслуживает жилье и инфраструктуру.

Управление картами доступа: В системе предусмотрена возможность управления картами контроля доступа, которые могут использоваться для контроля доступа в различные части жилого комплекса или здания. Администратор проверяет заявки владельцев на получение карт доступа и обеспечивает их обработку. В процессе работы можно добавлять или удалять карты, назначать их владельцам и контролировать доступ на основе авторизованных данных.

Снятие показаний счетчиков: Владельцы могут подавать заявки на снятие показаний счетчиков, таких как вода, электричество, газ и другие коммунальные ресурсы. Администратор имеет доступ к этой информации и может просматривать запросы владельцев, подтверждать или отклонять их. Эта функция позволяет обеспечить точность учета потребленных ресурсов и своевременно реагировать на запросы владельцев.

Управление платежами: Когда владельцы вносят платежи за коммунальные услуги или аренду, система управления платежами отслеживает статус этих платежей. Администратор может просматривать информацию о том, какие платежи были произведены успешно, а какие находятся в процессе. Эта функция упрощает процесс учета финансовых операций и способствует прозрачности в финансовых вопросах.

Декларации об обслуживании и отзывы: Владелец может разместить запрос на техническое обслуживание или подать жалобу через систему. Администратор, в свою очередь, отслеживает эти запросы и назначает необходимый персонал для выполнения работ. После завершения обслуживания владелец может оставить комментарий или оценку выполненной работы. Администратор имеет доступ к этим отзывам и использует их для оценки качества предоставляемых услуг и для улучшения работы обслуживающего персонала.

1.5.2 Модуль ролей владельца

Регистрация учетной записи: Владелец недвижимости должен зарегистрировать свою учетную запись в системе управления. После регистрации он предоставляет свою личную информацию, такую как контактные данные, информацию о недвижимости и другие данные, необходимые для системы. Однако для того чтобы начать использовать систему, владельцу необходимо пройти процесс аудита, проводимый администратором. Этот процесс включает в себя проверку данных и права собственности. Только после успешного завершения аудита владелец получает доступ к личному кабинету и полномочиям в системе.

Декларация о техническом обслуживании: Владельцы могут размещать декларации о необходимости технического обслуживания их объектов недвижимости. Эти декларации могут касаться различных типов работ, таких как ремонт, очистка, улучшение инфраструктуры и других нужд. Администратор проверяет каждую декларацию и назначает соответствующего персонала для выполнения требуемых работ. Владелец получает уведомление о назначении и ходе выполнения работ. Администратор может составить план технического обслуживания, который будет включать график и сроки выполнения работ, а также отслеживать статус выполнения работ и запрашивать отчеты о ходе работы, чтобы убедиться в их своевременном завершении.

Карты контроля доступа: Владельцы могут запросить карты контроля доступа для своих помещений или для определенных зон жилого комплекса,

таких как парковки, подъезды или общие зоны. Эти карты предоставляют владельцам возможность ограничивать доступ в определенные зоны. Администратор недвижимости рассматривает заявки владельцев на получение карт, проверяет их данные и одобряет их выдачу. В случае необходимости, администратор может отправиться в здание и завершить процесс выдачи карт, проверяя и подтверждая их точность и соответствие.

Снятие показаний счетчиков: Владелец недвижимости может объявить о предоставлении услуг по снятию показаний счетчиков для различных коммунальных услуг, таких как вода, электричество, газ и другие. Эта информация необходима для правильного расчета оплаты коммунальных услуг. Администратор проверяет заявки владельцев и собирает информацию о снятых показаниях. Это помогает обеспечить точность расчетов и своевременные платежи. Владелец также может запрашивать информацию о прошлых показаниях для контроля расходов.

Управление платежами: Владелец имеет доступ к управлению своими платежами в системе. Он может просматривать счета, производить оплату за коммунальные услуги, аренду или другие расходы. Когда владелец производит оплату, администратор недвижимости может подтвердить успешность операции и уведомить владельца о статусе его платежа. Система также поддерживает хранение записей о всех произведенных платежах и формирует статистические отчеты по этим данным. Это позволяет владельцам иметь четкое представление о своих финансовых обязательствах и отслеживать расходы.

Оценка обслуживания: После того как выполнены работы по техническому обслуживанию или другие услуги, владельцы могут оставить свою оценку и комментарии о качестве выполненных работ. Эта функция помогает повысить качество обслуживания и дает администрации информацию о том, как улучшить услуги. Владелец может указать, насколько ему понравилось выполнение работ, а также предложить улучшения. Администратор может просмотреть эти комментарии и принять меры для улучшения качества обслуживания в будущем.

Обсуждение информации: В системе предусмотрена возможность

обсуждения различных вопросов и проблем между владельцами и администраторами. Владельцы могут оставлять комментарии по различным вопросам, связанным с их недвижимостью, а администратор может отвечать на эти комментарии или добавлять новые обсуждения. Это создает открытую коммуникацию между владельцами и управляющим персоналом, что способствует повышению уровня доверия и улучшению обслуживания.

1.5.3 Анализ требований к безопасности системы

С развитием цифровизации и интеллектуализации безопасность систем управления имуществом становится особенно важной, особенно в сценариях, связанных с конфиденциальными данными пользователей, такими как личная информация владельцев, платежные записи, заявки на ремонт и т.д. Чтобы гарантировать защиту этих данных от несанкционированного доступа или утечки, система должна использовать продуманную многоуровневую архитектуру безопасности. Эта архитектура проектируется вокруг аутентификации пользователей, шифрования данных, управления правами доступа и аудита, а также сочетает современные технологии для обеспечения целостности и безопасности данных.

Аутентификация пользователей и динамическая авторизация — это первый шаг к безопасности системы. Используя модель ролевого управления доступом (RBAC), система разделяет пользователей на роли: администраторы, владельцы, ремонтные сотрудники и т.д., где каждая роль имеет свои права. Например, администраторы обладают высшими привилегиями, могут получать доступ и изменять все модули, владельцы — только просматривать информацию о своей собственности и счета, а ремонтные сотрудники — работать только с заявками. Для усиления безопасности система также внедряет многофакторную аутентификацию (MFA), используя пароли, SMS-коды и другие методы проверки, чтобы подтвердить легальность пользователя. Кроме того, технология JSON Web Token (JWT) управляет сессиями, обеспечивая строгий контроль над каждым входом и принудительную повторную аутентификацию через определенное время, чтобы предотвратить перехват сессий и другие уязвимости.

Шифрование данных и безопасность передачи — ключевые инструменты защиты конфиденциальности. Все чувствительные данные (например, номера паспортов, банковские карты) шифруются алгоритмом AES-256 при хранении, а ключи управляются через отдельную систему управления ключами (KMS). При передаче данных система обязательно использует протокол HTTPS, обеспечивая шифрование канала через TLS 1.3, чтобы избежать атак «человек посередине» и перехвата данных.

Детализированный контроль доступа и аудит — важные компоненты безопасности. Детализированные права доступа гарантируют, что каждая роль получает минимально необходимые привилегии. Например, владелец видит только ремонтные записи своей собственности, а администратор — журналы действий всех пользователей. Каждая операция записывается в аудит-логи и анализируется в реальном времени через ELK-стек (Elasticsearch, Logstash, Kibana) для обнаружения аномалий. Это не только предотвращает утечки, но и позволяет отправлять оповещения администраторам.

Соответствие требованиям защиты конфиденциальности — еще одно критическое требование. Чтобы соответствовать законам, таким как Закон о защите персональных данных и GDPR, система предоставляет функцию «права на забвение», позволяя пользователям запрашивать удаление своих данных. Удаление выполняется логически и физически в установленные сроки. На странице политики конфиденциальности четко указываются цели обработки данных, а пользователи могут просматривать, изменять или экспортировать свои данные, повышая доверие и удовлетворенность.

1.5.4 Анализ требований к производительности и масштабируемости системы

С расширением пользовательской базы и усложнением бизнес-сценариев сообщества управляющих компаний производительность системы в условиях высокой нагрузки и её способность к масштабированию становятся критически важными. Особенно в пиковые периоды, такие как массовая оплата счетов собственниками или подача заявок на ремонт в праздничные дни, система

должна обеспечивать быстрое реагирование и гибкое масштабирование, чтобы избежать снижения качества обслуживания или полного простоя. На основе стресс-тестирования и анализа реальных данных эксплуатации были выявлены множественные узкие места в исходной архитектуре, что позволило разработать комплексное решение — от оптимизации баз данных до перехода на микросервисную архитектуру, а также внедрить стратегии аварийного восстановления и постепенного развертывания (gray release) для обеспечения стабильности системы при высоких нагрузках и росте бизнеса.

На этапе первоначального стресс-тестирования с использованием JMeter для имитации 1000 одновременных пользователей система выявила три ключевые проблемы. Во-первых, низкая эффективность запросов к базе данных: среднее время отклика интерфейса запроса счетов собственников составило 1,2 секунды из-за отсутствия индексации часто используемого поля user_id, что приводило к полному сканированию таблицы. Во-вторых, интерфейс обработки платежных уведомлений (callback) из-за синхронной блокировки демонстрировал пропускную способность всего 50 TPS (транзакций в секунду), что в пиковые периоды вызывало накопление задач. В-третьих, после 8 часов непрерывной работы использование кучи JVM (Java Virtual Machine) превышало 80_%, с частыми вызовами полной сборки мусора (Full GC), что создавало риск резкого роста задержек или даже сбоев. Эти проблемы напрямую влияли на пользовательский опыт и надежность системы, требуя немедленной оптимизации.

Для решения проблем с производительностью баз данных была реализована многогранная стратегия. На уровне индексации для часто запрашиваемых полей (user_id, payment_status) были созданы составные индексы, что сократило время отклика запросов счетов до 300 миллисекунд. Одновременно был внедрен механизм кэширования Redis для «горячих» данных (например, объявления управляющей компании или тарифы), повысив уровень попадания в кэш до 92_% и снизив нагрузку на базу данных. Для таблицы платежных записей (payment_records), объем которой продолжает расти, было

применено горизонтальное шардирование по идентификатору жилого комплекса, ограничив размер отдельной таблицы 5 миллионами записей и использовав алгоритм консистентного хеширования для равномерного распределения запросов. В результате этих мер производительность базы данных при тестировании с 2000 одновременных пользователей показала QPS (запросов в секунду) 1800, что в 3 раза выше исходного уровня.

На уровне системной архитектуры основное внимание уделялось асинхронной обработке и оптимизации управления ресурсами. Интерфейс обработки платежных уведомлений был переведен с синхронного режима на асинхронную модель с использованием очереди сообщений RabbitMQ, что увеличило пропускную способность до 1200 TPS при сохранении согласованности транзакций. Пакетные задачи, такие как генерация ежемесячных счетов, были перенесены в модуль Quartz для планирования заданий, что исключило блокировку основного потока. Для JVM параметры кучи были увеличены до 4 ГБ (-Xmx4096m) с активацией сборщика мусора G1 и настройкой MaxGCPauseMillis=200ms, что сократило частоту Full GC с 12 раз в час до 1 раза, а использование памяти стабилизировалось на уровне 65_%–75_%.

Для поддержки бизнес-целей удвоения пользовательской базы в течение трех лет архитектура системы была полностью переработана в микросервисном стиле. Согласно принципам Domain-Driven Design (DDD), ядро системы разделено на четыре модуля: сервис пользователей (аутентификация и управление правами), сервис заявок (отслеживание жизненного цикла ремонтных работ), сервис платежей (интеграция с Alipay и WeChat Pay) и сервис уведомлений (SMS и push-сообщения). Каждый сервис развернут в Docker-контейнерах и управляется кластером Kubernetes, обеспечивающим автоматическое масштабирование. Например, в пиковые периоды оплаты количество экземпляров платежного сервиса увеличивается с 2 до 5, а в обычное время сокращается до 2, повышая использование ресурсов на 40_%. Взаимодействие между сервисами осуществляется через API-шлюз с использованием регистра Nacos для обнаружения сервисов и балансировки

нагрузки, что гарантирует отказоустойчивость.

Для обеспечения высокой доступности разработана многоуровневая система аварийного восстановления. Базы данных развернуты в мультизоновой мастер-репликационной конфигурации с полусинхронной репликацией, где задержка между мастером и репликой не превышает 200 мс, что позволяет выполнять переключение за секунды. Кэш-слой Redis работает в режиме Sentinel с тремя узлами для автоматического восстановления и сохранения данных. При обновлениях версий используется стратегия постепенного развертывания: 10_% производственного трафика перенаправляется на новую версию, а после 15 минут мониторинга (при уровне ошибок ниже 0,5_%) доля трафика постепенно увеличивается. Это сократило среднее время простоя при обновлениях с 20 минут до 47 секунд.

Благодаря этим оптимизациям система показала значительные улучшения при тестировании с десятками тысяч одновременных пользователей: среднее время отклика ключевых интерфейсов — менее 800 мс, доступность системы достигла 99,95_%, а максимальное количество обработанных заявок за день превысило 120 тысяч. Механизмы автоматического масштабирования снизили затраты на оборудование на 35_%, а архитектура оставляет возможности для интеграции новых функций, таких как управление умными устройствами или AI-планирование задач. Эти улучшения не только устранили текущие узкие места, но и заложили технологическую основу для будущих инноваций, укрепляя позиции системы как надежной платформы для умных сообществ.

2 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА СИСТЕМЫ УПРАВЛЕНИЯ НЕДВИЖИМОСТЬЮ

2.1 Дизайн функций системы

Для лучшего понимания работы системы на рисунке 1 показана структура основных функциональных модулей, которые обеспечивают эффективное взаимодействие между пользователями и администраторами. Система условно разделена на два основных компонента: модуль администратора и пользовательский модуль. Каждый из этих компонентов имеет уникальные и важные функции, которые способствуют бесперебойной работе системы и обеспечивают удобство для всех участников. Модуль администратора включает в себя инструменты, предназначенные для эффективного управления пользователями, контроля и мониторинга состояния всей системы, настройки параметров, а также генерации отчетов для анализа работы системы. Администратор имеет возможность отслеживать работу всех сервисов, проверять их текущий статус, управлять тарифами и организовывать техническое обслуживание, чтобы обеспечить бесперебойную работу системы в целом. Важно, что благодаря этому модулю администратор может оперативно вмешиваться в работу системы, настраивать ее под потребности пользователей и следить за качеством предоставляемых услуг. Пользовательский модуль, в свою очередь, предоставляет простые и удобные инструменты для конечных пользователей, которые позволяют им легко управлять своими данными и взаимодействовать с системой. Пользователи могут оценивать качество услуг, управлять оплатой счетов, вводить показания счетчиков и обрабатывать карты доступа. Это дает им возможность напрямую взаимодействовать с системой, получать актуальную информацию о своих счетах, а также следить за состоянием своих данных. Таким образом, каждый компонент системы играет свою ключевуюроль в обеспечении качественного обслуживания и взаимодействия между администраторами и пользователями.

Рисунок 1 – Схема функциональных модулей системы управления недвижимостью на базе Java

2.2 Тестирование системного каркаса

Система состоит из нескольких ключевых функциональных модулей, которые обеспечивают эффективное и бесперебойное взаимодействие между владельцами, администраторами и пользователями. Одним из важнейших аспектов работы системы является модуль считывания показаний счетчиков, который позволяет пользователям и владельцам точно регистрировать данные о потребленных ресурсах. Этот модуль гарантирует, что информация будет собрана с максимальной точностью, что способствует правильному расчету и учету потреблений. В дополнение к этому, система интегрирована с модулем оплаты счетов, что позволяет пользователям и владельцам быстро и удобно оплачивать свои финансовые обязательства. При этом система предоставляет актуальную информацию о статусе платежей, помогая избежать ошибок и задержек в расчетах. Также важно, что система предусматривает различные уровни доступа, что позволяет владельцам и пользователям иметь разные права. Владельцы могут создавать заказы на обслуживание, которые администраторы проверяют и принимают в работу, обеспечивая качественное выполнение всех заявок. В свою очередь, пользователи могут оставлять комментарии о предоставляемых услугах, что способствует повышению качестваобслуживания, так как эти комментарии внимательно анализируются администраторами. Для того чтобы пользователи и владельцы всегда были в курсе текущих событий, администраторы регулярно публикуют объявления, касающиеся технического обслуживания, обновлений тарифов и других изменений, которые могут повлиять на их взаимодействие с системой. Владельцы и пользователи могут просматривать эти объявления, а

также оставлять свои отзывы, что помогает создавать прозрачную и эффективную коммуникацию. Все эти модули тесно взаимосвязаны между собой, что позволяет владельцам публиковать задания, которые затем могут быть принятыадминистраторами для выполнения, а администраторам — публиковать объявления, которые владельцы и пользователи могут анализировать и комментировать. Этот процесс способствует активной обратной связи и поддерживает гармоничное взаимодействие всех участников системы.

2.3 Проектирование базы данных

2.3.1 Концептуальный дизайн базы данных

Атрибуты сущности «Владелец» включают уникальный идентификатор владельца, что позволяет системе точно отличать каждого владельца. Также важным атрибутом является имя владельца, которое используется для идентификации пользователя в системе, а также пароль для входа, обеспечивающий безопасность учетной записи. Контактный номер телефона и адрес электронной почты позволяют системе поддерживать связь с владельцем, отправлять уведомления и обеспечивать обратную связь. Статус учетной записи является важным атрибутом, поскольку он может быть изменен на включенный или выключенный, что регулирует доступ владельца к системе. Эти данные позволяют системе надежно управлять доступом и следить за актуальностью учетных записей владельцев.

Атрибуты сущности «Администратор» включают уникальный идентификатор администратора, который точно определяет его в системе, и контактный номер телефона, необходимый для связи с администратором, что упрощает решение оперативных вопросов и поддержание связи в случае необходимости. Атрибуты сущности «Сотрудник» включают ID сотрудника, имя и должность, что позволяет четко идентифицировать каждого сотрудника в системе. Также важным атрибутом является рабочий статус сотрудника, который может быть активным или неактивным, что позволяет отслеживать состояние персонала, отвечающего за выполнение различных задач.

Атрибуты сущности «Объявление» включают уникальный идентификатор, который позволяет системе точно отслеживать каждое объявление. Заголовок и содержание объявления являются основными атрибутами, благодаря которым пользователи могут быстро понять суть публикации. Идентификатор администратора, который опубликовал объявление, и временная метка создания объявления дают полную картину о том, кто и когда разместил информацию. Эти атрибуты помогают системе точно отслеживать публикации и обеспечивать их актуальность.

Атрибуты сущности «Считывание показаний счетчиков» включают уникальный идентификатор задачи, который позволяет точно идентифицировать каждое считывание, и идентификатор сотрудника, который выполняет задачу. Важным атрибутом является информация о владельце, который связан с этим счетчиком, а также идентификатор администратора, который управляет задачей считывания. Также важным атрибутом является изображение снятых показаний, которое может быть хранится в виде ссылки или пути, обеспечивая точность данных.

Атрибуты сущности «Платеж» включают уникальный идентификатор платежа, который позволяет отслеживать каждый платеж в системе, и идентификатор администратора, который управляет записью о платеже. Сумма платежа и временная метка, когда был произведен платеж, играют важную роль в управлении финансовыми операциями, так как они позволяют точно зафиксировать дату и сумму каждого платежа.

Атрибуты сущности «Обслуживание» включают уникальный идентификатор задачи обслуживания, который необходим для отслеживания заявок на обслуживание, и описание ремонта, которое дает полную информацию о выполняемых работах. Идентификатор администратора, который управляет задачей, а также идентификаторы сотрудника, выполняющего задание, и владельца, который инициирует задачу, позволяют отслеживать ответственных за выполнение работ. Также учитывается статус ремонта, который может быть

незавершенным или завершенным, что дает возможность следить за состоянием задачи.

Атрибуты сущности «Комментарий» включают уникальный идентификатор комментария, который позволяет точно идентифицировать каждый отзыв, и временную метку его оставления, что помогает определить, когда был опубликован комментарий. Идентификаторы администратора и владельца дают возможность узнать, кто оставил комментарий, а содержание комментария позволяет анализировать взаимодействие пользователей с системой. Все эти атрибуты помогают системе эффективно хранить и обрабатывать информацию, улучшая качество обслуживания и взаимодействие между пользователями и администраторами.

2.3.2 Логическая структура базы данных

Отношения между сущностями в системе имеют четко организованную структуру, что способствует эффективному управлению данными и обеспечивает бесперебойную работу всей системы. Например, отношение между «Владельцем» и «Комментарием» является отношением «один-комногим», где один владелец может публиковать несколько комментариев. Это позволяет владельцам оставлять множественные отзывы или комментарии, касающиеся различных объявлений или услуг, что дает возможность лучшего понимания мнений владельцев и помогает улучшить качество обслуживания. Отношение «один-ко-многим» также существует между «Администратором» и «Объявлением», где один администратор может публиковать несколько объявлений, которые информируют пользователей о различных обновлениях и событиях в системе. Это способствует прозрачности и оперативности в коммуникации между администрацией и пользователями. Кроме того, отношение «один-ко-многим» между «Сотрудником» и «Считыванием показаний» позволяет одному сотруднику работать с несколькими задачами по снятию показаний счетчиков, что увеличивает производительность и эффективность работы. Точно так же отношение «один-ко-многим» существует между «Администратором» и «Оплатой», где один администратор может

обрабатывать множество платежей, следя за их статусами и обеспечивая точность расчетов.

Также отношение «один-ко-многим» существует между «Владельцем» и «Обслуживанием», где один владелец может инициировать несколько задач на обслуживание, создавая заявки на ремонт или другие услуги. Это отношение позволяет легко управлять заявками и поддерживать систему в рабочем состоянии. Аналогичное отношение имеется и между «Администратором» и «Обслуживанием», где один администратор может быть ответственным за выполнение нескольких задач, связанных с техническим обслуживанием и ремонтом.

Эти связи обеспечивают удобство работы с системой и повышают общую эффективность управления. Важным компонентом системы является таблица User, которая описывает данные о пользователе, включая такие атрибуты, как имя владельца, криптовалюта, местонахождение кейса, номер телефона и адрес электронной почты. Эти данные помогают точно идентифицировать владельца и обеспечивают возможность связи с ним для дальнейшего обслуживания и учета его предпочтений в системе.

Таблице 1 – Пользовательский интерфейс

серийный номер	Английское название	Русское имя	тип данных	длина	ключевые слова
1	id	id	int	10	первичный ключ
2	name	Имя владельца	varchar	20	
3	password	криптовалюта	int	10	

Администратор в основном включает в себя имя администратора, номер телефона администратора и т.д. Как показано в таблице 2.

Таблице 2 –Схема управления

серийный номер	Английское название	Русское имя	тип данных	длина	ключевые слова
1	id	id	int	20	первичный ключ
2	name	имя и фамилия	varchar	20	
3	job	офис	varchar	60	
4	valid	полож	int	10	

Оплата в основном включает в себя стоимость похудения, время создания платежа и администратора и т.д. Как показано в таблице 3.

Таблице 3 – Тарифная сетка

серийный номер	Английское название	Русское имя	тип данных	длина	ключевые слова
1	content	элемент	varchar	255	первичный ключ
2	price	цены	decimal(10,0)	0	
3	startTime	раз	varchar	50	
4	admin	смотрители	int	20	

Показания счетчиков в основном состоят из фотографий, сотрудников и идентификаторов показаний счетчиковКак показано в таблице 4.

Таблице 4 – Показания счётчика

серийный номер	Английское название	Русское имя	тип данных	длина	ключевые слова
1	id	id	int	11	первичный ключ
2	employee	рабочие	int	20	

Ремонт включает в себя содержимое ремонта, статус Идентификатор сотрудника , Идентификатор ремонта , Идентификатор администратора , Идентификатор владельцаКак показано в таблице 5.

Таблице 5 – Ремонтный лист

серийный номер	Английское название	Русское имя	тип данных	длина	ключевые слова
1	id	id	int	11	
2	content	элемент	varchar	255	

Отзывы в основном включают в себя среднее время, содержание отзывови т.д. Как показано в таблице 6.

Таблице 6 – Форма комментариев

серийный номер	Английское название	Русское имя	тип данных	длина	ключевые слова
1	id	id	int	12	первичный ключ
2	time_	раз	int	20	
3	user	домовладельцы	int	30	

В проектировании базы данных оптимизация логической структуры не только способствует эффективной работе системы, но и напрямую влияет на скорость доступа к данным, их согласованность и масштабируемость системы. В системе управления жилой недвижимостью данные охватывают множество аспектов, и эффективная организация этих взаимосвязей, обеспечивающая стабильную и производительную работу системы при высокой нагрузке и больших объемах данных, является ключевым моментом проектирования.

Во-первых, основная задача логического проектирования заключается в обеспечении четких и эффективных связей между таблицами. Например, в системе управления жилой недвижимостью основные сущности включают

владельцев, администраторов, оборудование, платежи, сервисные заявки и другие, причем между ними существуют различные связи. Например, отношение между владельцем и сервисной заявкой может быть «один ко многим» — один владелец может подавать несколько заявок. Отношение между администратором и платежами также может быть «один ко многим», поскольку один администратор может обрабатывать платежи нескольких владельцев. В то же время связь между оборудованием и сервисной заявкой может быть «один к одному», если каждый ремонт оборудования выполняется конкретным сотрудником. Четкое определение этих связей позволяет быстро находить нужную информацию при запросах, повышая эффективность работы системы.

Кроме того, важным этапом оптимизации логической структуры базы данных является продуманное проектирование полей. Каждое поле в таблице должно быть лишено избыточности, а данные должны соответствовать принципам нормализации. Например, следует избегать дублирования одной и той же информации в разных таблицах, а вместо этого использовать внешние ключи для связывания данных. Это не только уменьшает избыточность, но и обеспечивает целостность и согласованность данных. Также для разных типов данных можно задавать подходящие типы полей и ограничения, такие как ограничение длины, запрет на пустые значения и другие, что повышает надежность данных.

Для обеспечения стабильной работы системы в периоды пиковой нагрузки необходимо применять методы оптимизации, такие как партиционирование таблиц и индексация. Партиционирование позволяет разбивать большие таблицы на меньшие, ускоряя доступ к данным, а индексы ускоряют поиск конкретных записей. Грамотное использование индексов значительно сокращает время отклика при запросах, особенно при сложных выборках с объединением таблиц. Для системы управления жилой недвижимостью, где объем данных велик, оптимизация производительности базы данных критически важна, поскольку напрямую влияет на пользовательский опыт и стабильность системы.

Помимо этих базовых принципов проектирования, логическая структура должна поддерживать обновление данных в реальном времени и обработку высокой нагрузки. Для удовлетворения потребностей в оперативной обработке данных можно использовать кэширование, сохраняя часто запрашиваемые данные в памяти, что снижает количество обращений к базе и ускоряет работу системы. Кроме того, во избежание конфликтов данных при высокой нагрузке механизм транзакций должен быть тщательно проработан, чтобы гарантировать согласованность и целостность данных.

2.3.3 Физическая структура базы данных

В процессе проектирования базы данных, помимо концептуальной и логической структуры, физическая структура является ключевым фактором, обеспечивающим эффективную работу базы данных в реальных приложениях. Физическая структура затрагивает хранение данных, эффективность доступа, обеспечение согласованности данных и другие аспекты, которые напрямую влияют на производительность системы при высокой нагрузке и больших объемах данных.

Основная задача физического проектирования заключается в выборе подходящих носителей данных и стратегии распределения информации. В современных системах управления недвижимостью, где объем данных огромен, а операции сложны, традиционные однопользовательские базы данных могут не справляться с требованиями к быстрому доступу и эффективному хранению. Поэтому распределенные базы данных становятся важным технологическим решением. Они позволяют распределять данные между несколькими узлами, повышая отказоустойчивость и надежность системы за счет балансировки нагрузки и избыточности данных.

Для оптимизации хранения данных ключевым инструментом является разделение данных (партиционирование). Разделение больших таблиц на более мелкие, легко управляемые части ускоряет доступ к информации. Методы разделения могут включать диапазонное, хеш- или списковое партиционирование, каждый из которых подходит для разных типов данных и

запросов. В системах управления недвижимостью данные можно разделять по географическому расположению, типу собственников или времени, что делает запросы более эффективными.

Индексы также играют важную роль в повышении скорости запросов. Создание индексов для часто используемых полей значительно ускоряет поиск, особенно при сложных запросах с объединением таблиц. Индексы сокращают количество полных сканирований базы данных, снижая нагрузку на систему. Типы индексов (одноколоночные, составные, полнотекстовые) имеют свои преимущества в разных сценариях, поэтому выбор стратегии должен основываться на конкретных требованиях.

Избыточность данных обеспечивает безопасность и надежность системы. В управлении недвижимостью это может быть реализовано через репликацию master-slave и резервное копирование. Репликация гарантирует высокую доступность: при сбое основного сервера резервный продолжает работу. Регулярные резервные копии минимизируют риск потери данных в случае аварии.

С распространением облачных технологий облачные базы данных стали важным компонентом современных систем. Они предоставляют гибкое хранилище и эффективную обработку данных через распределенную архитектуру. Облачные платформы позволяют динамически масштабировать ресурсы, адаптируясь к нагрузке, что повышает гибкость системы.

Управление транзакциями и согласованность данных также критичны. Операции, такие как оплата услуг или учет ремонтов, требуют строгой согласованности. Механизмы транзакций, основанные на принципах ACID (атомарность, согласованность, изоляция, долговечность), гарантируют надежность даже при высокой нагрузке.

Кэширование — ещё один важный элемент. Хранение часто запрашиваемых данных (информация о собственниках, платежах) в памяти снижает нагрузку на базу и ускоряет отклик системы.

Эти оптимизации позволяют создать физическую структуру базы данных, которая обеспечивает высокую производительность системы управления недвижимостью даже при масштабных нагрузках, сохраняя стабильность и эффективность.

2.4 Функциональный дизайн и реализация

2.4.1 Разработка и реализация интерфейса входа и регистрации

При переходе на страницу login.html, пользователь отправляет запрос на сервер через браузер, и этот запрос обрабатывается бэкэндом. Сервер получает данные, переданные пользователем, например, имя пользователя и пароль, и начинает их проверку. Бэкэнд взаимодействует с базой данных, чтобы убедиться, что введенные данные соответствуют существующей учетной записи. После этого сервер принимает решение о том, авторизован ли пользователь, и в зависимости от результата отправляет соответствующий ответ. Если данные правильные, сервер генерирует сессию или токен доступа, которые используются для идентификации пользователя в дальнейшем, и возвращает успешный ответ с перенаправлением на главную страницу. В случае неверных данных, сервер отправляет сообщение об ошибке, информируя пользователя о необходимости повторить попытку. Этот процесс гарантирует безопасность и корректность авторизации, а также обеспечивает взаимодействие между фронтендом и бэкэндом.

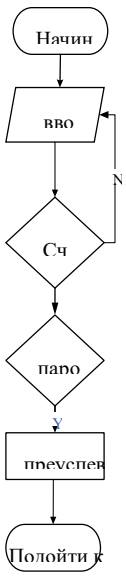

Рисунок – 2 Блок-схема процедуры входа в систему

На схеме, представленной ниже, показаны все ключевые этапы процесса аутентификации, начиная с того, как пользователь отправляет или обновляет свою идентификационную информацию. Этот шаг важен, так как включает введение логина и пароля или других данных, необходимых для подтверждения личности. После отправки данных начинается этап проверки, на котором система осуществляет проверку предоставленных данных. Этот процесс включает в себя взаимодействие с базой данных для сверки информации и проверки ее корректности. Проверку данных выполняет администратор системы, который в случае успешной проверки подтверждает идентификацию пользователя, что означает, что его учетные данные прошли проверку и он может получить доступ к сервису. Если же проверка не прошла, система уведомляет пользователя о необходимости повторной подачи или исправления введенной информации. Весь этот процесс организован таким образом, чтобы обеспечивать точность, безопасность и полноту предоставленных данных. На рисунке 5 представлена детальная схема, которая наглядно демонстрирует все этапы запроса и проверки, что позволяет точно отслеживать и контролировать процесс аутентификации и идентификации пользователя.

2.4.2 Разработка и внедрение системы управления информацией о владельце

Разработка и внедрение системы управления информацией о владельцах (OIMS) включали в себя создание мощного и удобного интерфейса для администраторов, который обеспечивал эффективное управление данными пользователей. Этот интерфейс позволяет администраторам системы легко добавлять новых пользователей, вводя необходимые данные, такие как имена пользователей, пароли, номера телефонов и другие важные сведения. Важно отметить, что перед добавлением нового пользователя система автоматически выполняет проверку на наличие уже существующих учетных записей с аналогичными данными. Если введенные данные совпадают с уже зарегистрированными пользователями или если система обнаруживает дублирование, она информирует администратора о проблеме и блокирует

дальнейшее добавление записи. Этот механизм проверки помогает предотвратить создание дублированных учетных записей, что способствует поддержанию целостности и точности данных. Помимо этого, система включает автоматическую проверку вводимых данных на правильность, например, проверку формата телефонных номеров или минимальной длины пароля. Это важно для снижения количества ошибок ввода и повышения безопасности системы. Администраторы также имеют возможность редактировать информацию о существующих пользователях, обновлять ее или удалять, чтобы поддерживать актуальность данных в базе. Вся эта система управления обеспечивает надежность, точность и эффективность в работе с данными пользователей, улучшая качество обслуживания и общую стабильность работы системы.

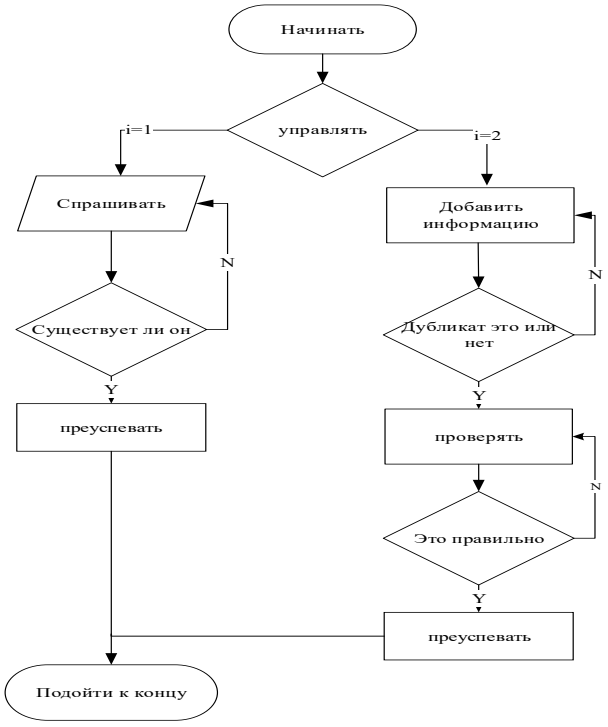

Рисунок 3 – Блок-схема процесса управления пользователями

2.4.3 Разработка и внедрение системы управления взносами

Управление платежами является неотъемлемой частью системы управления имуществом, играя ключевую роль в обеспечении стабильного функционирования системы и высококачественного обслуживания арендаторов. Совершенная система управления платежами не только эффективно

обрабатывает все виды платежей, поступающих от арендаторов, но и систематически управляет различными имущественными сборами, гарантируя прозрачность и отслеживаемость каждой финансовой операции. Важной особенностью такого модуля является поддержка множества способов оплаты, включая онлайн-платежи, оплату наличными, банковскими картами и другие методы, что позволяет удовлетворить различные потребности жильцов. Это помогает сделать процесс оплаты более гибким и удобным для всех пользователей. Модуль управления платежами также включает такие функции, как автоматическое формирование счетов, напоминания о предстоящих платежах и уведомления о просрочках, что помогает арендаторам своевременно узнать статус своих платежей и избежать задержек с оплатой. Вдобавок, в систему могут быть интегрированы возможности для создания отчетности и статистических данных, которые предоставляют управляющим недвижимостью подробный анализ доходов в режиме реального времени. Эти данные помогают лучше понять текущую финансовую ситуацию, а также дают возможность принимать оперативные решения по улучшению управления сообществом. Эффективная система управления платежами должна быть ориентирована на удобство пользователей, предлагая интуитивно понятный интерфейс, позволяющий арендаторам легко и быстро совершать платежи, просматривать историю оплат и получать информацию о состоянии своих счетов. Система должна быть достаточно гибкой, чтобы минимизировать сложность процессов и сделать их максимально доступными для жильцов. В то же время, особое внимание следует уделить защите данных: необходимо обеспечить надежный механизм защиты, который бы исключал утечку личной информации арендаторов или неправильное использование их платежных данных. В целом, такой модуль должен быть простым в использовании, безопасным и надежным, предоставляя арендаторам всю необходимую информацию о платежах. В завершение стоит подчеркнуть, что модуль управления платежами играет важную роль в повышении эффективности работы системы управления недвижимостью, удовлетворенности арендаторов и улучшении качества предоставляемых услуг,

являясь неотъемлемой частью современного управления недвижимостью. Блок-схема программы «Список платежей» показана на рисунке 4, что позволяет лучше понять архитектуру системы и ее работу в реальных условиях.

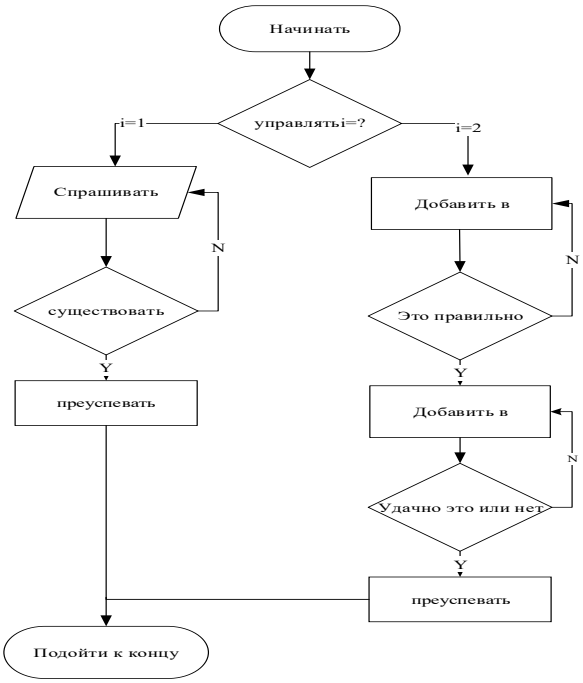

Рисунок 4 – Схема платежей

2.4.4 Блок-схема процесса заказа на техническое обслуживание

Пользователи могут создавать заявки на техническое обслуживание, указывая в них подробности о необходимых работах или о том, что должно быть выполнено. Система управления заявками на обслуживание занимает важное место в структуре системы управления недвижимостью, так как она служит связующим звеном между пользователями и обслуживающим персоналом, обеспечивая своевременное и эффективное выполнение всех заявленных задач. После подачи заявки администратор заходит в систему, чтобы проверить ее содержание и оценить, что требуется для выполнения работ. Каждая заявка изначально имеет статус «0», что означает, что она еще находится в процессе обработки. Администратор назначает соответствующего технического специалиста для выполнения работы, после чего статус заявки обновляется до «1», что свидетельствует о том, что обслуживание было успешно завершено. Кроме того, система предоставляет дополнительные полезные функции, такие

как ведение истории всех заявок, возможность автоматического уведомления пользователей о текущем статусе их заявок, а также возможность формирования аналитических отчетов для администраторов. Это не только улучшает прозрачность и информированность пользователей о процессе обслуживания, но и помогает оптимизировать распределение ресурсов, снижая вероятность задержек и повышая качество обслуживания в целом. В итоге система управления заявками на обслуживание стала незаменимым инструментом для повышения эффективности управления недвижимостью, улучшения качества обслуживания и повышения уровня удовлетворенности пользователей, так как она способствует упрощению процесса обработки запросов и делает его более оперативным и организованным.

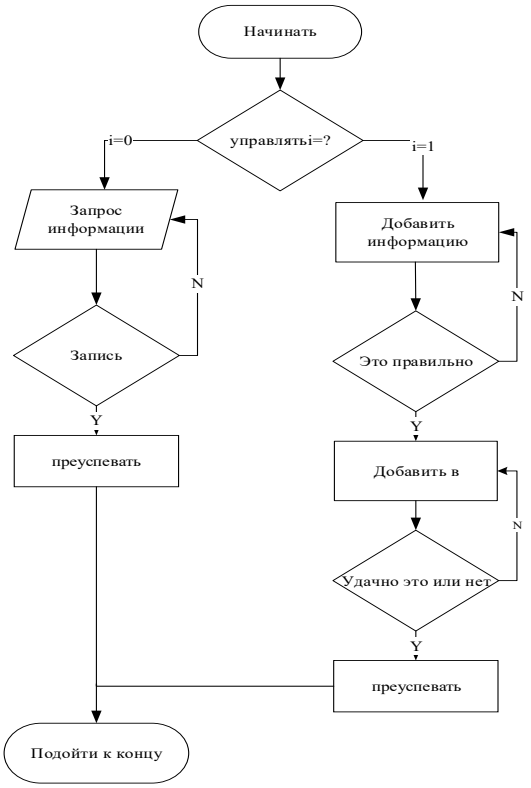

Рисунок 5 – Блок-схема процесса заказа на ремонт

2.4.5 Блок-схема процесса оценки услуг

Когда администратор завершает обслуживание, статус задачи изменяется на завершенный, и одновременно с этим в бэкэнде автоматически создается запись обзора услуг. Эта запись позволяет владельцам оценивать качество выполненной работы через модуль обзора услуг, где они могут оставить свои

отзывы и поставить оценку в зависимости от качества обслуживания. Главная цель этого процесса — предоставить администраторам возможность получать обратную связь от владельцев в реальном времени, что помогает оперативно реагировать на замечания и улучшать качество предоставляемых услуг. Это также способствует более открытому и дружественному взаимодействию между администраторами и жителями, укрепляя отношения в сообществе. После того как владельцы оставляют свою оценку, администратор имеет возможность ответить на их комментарии, что помогает улучшить коммуникацию и показывает, что мнение владельцев учитывается. Такой подход способствует созданию гармоничных и взаимовыгодных отношений между администрацией и владельцами, создавая атмосферу доверия и уважения.

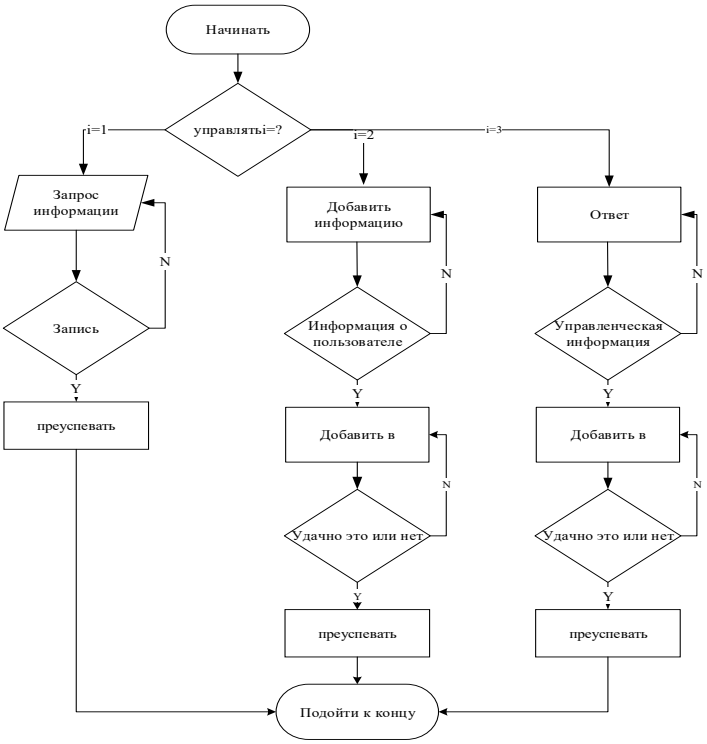

Рисунок 6 – Блок-схема процесса оценки услуг

2.5 Тестирование системы

Для обеспечения стабильной и бесперебойной работы Java-системы Hope Community Property Service System, а также для гарантии быстрого и эффективного реагирования на действия пользователей, проводится всестороннее тестирование производительности, отзывчивости бэкэнда и поэтапное тестирование всех бизнес-процессов. Это необходимое условие для

подтверждения эффективности системы, ее способности справляться с нагрузкой в условиях реальной эксплуатации и соответствия ожиданиям пользователей. Особенно важно это для веб-платформ, используемых в образовательных или практических целях, где тестирование становится неотъемлемой частью процесса разработки, обеспечивая высокое качество и надежность системы. Обычно системное тестирование проводится после завершения интеграционного тестирования, чтобы проверить, как все компоненты системы взаимодействуют друг с другом. Это включает проверку взаимодействия аппаратного и программного обеспечения, внешних интерфейсов, баз данных и других взаимосвязанных элементов. Такой подход позволяет не только оценить функциональность системы, но и проанализировать ее стабильность, надежность, удобство использования и безопасность. Эти аспекты имеют критическое значение для обеспечения корректной работы системы в реальных условиях эксплуатации, особенно при высоких нагрузках и интенсивном использовании. Согласно книге «Инженерия информационных систем, проектирование и применение сетевой учебной платформы на основе технологии разработки компьютерного программного обеспечения» Цяожэня Вана, система должна соответствовать заранее установленным параметрам и критериям, чтобы она могла эффективно взаимодействовать с базой данных и всеми компонентами в реальных условиях. Это включает проверку производительности, масштабируемости и отказоустойчивости системы. Важным этапом тестирования является бизнестестирование, которое помогает убедиться, что система соответствует всем бизнес-требованиям и выполняет свои функции в соответствии с ожиданиями заинтересованных сторон. Бизнес-тестирование также позволяет выявить потенциальные дефекты или недочеты, которые могут повлиять на качество работы системы.

После успешного бизнес-тестирования проводится тестирование на производительность, которое позволяет оценить, насколько система способна выдерживать нагрузки и эффективно работать при максимальных объемах данных. Это особенно важно для систем, которые должны обрабатывать большое

количество запросов одновременно, таких как Hope Community Property Service System. Тестирование на производительность включает проверку времени отклика системы, пропускной способности, устойчивости к нагрузкам и способности восстанавливаться после сбоев. После этого проводится еще несколько раундов тестирования, которые направлены на проверку корректности проектирования данных, выявление дефектов в функциональности, а также на оценку того, не возникнут ли проблемы с производительностью, которые могут повлиять на стабильную работу системы в долгосрочной перспективе. В работе Сюй Сяохуэя, Лю Цзяна и Гао Ханя, которые занимались компьютерным моделированием системы на основе SpringBoot + Vue для добычи газа, подчеркивается важность тщательной отладки системы на всех этапах разработки. Они также акцентируют внимание на необходимости применения незаменимых технических методов для корректной настройки и оптимизации работы системы. Тестирование и отладка — это ключевые этапы в процессе разработки, которые не только позволяют выявить и устранить потенциальные проблемы, но и обеспечивают стабильность и надежность системы в долгосрочной перспективе. Это способствует успешной эксплуатации системы в реальных условиях, повышает удовлетворенность пользователей и укрепляет доверие к платформе. Кроме того, регулярное тестирование и мониторинг системы после ее внедрения позволяют оперативно выявлять и устранять возникающие проблемы, что способствует поддержанию высокого уровня качества обслуживания и обеспечивает долгосрочную эффективность системы. Таким образом, всестороннее тестирование и отладка являются неотъемлемой частью жизненного цикла разработки системы Hope Community Property Service System. Они обеспечивают не только высокую производительность и надежность, но и соответствие системы всем бизнестребованиям и ожиданиям пользователей. Это делает систему не только функциональной, но и устойчивой к нагрузкам, что является ключевым фактором для ее успешной эксплуатации в реальных условияхАппаратная средаБренд Pride представляет собой высококачественную и мощную конфигурацию, оснащенную процессором Intel Core i7, что

обеспечивает отличную производительность и скорость обработки данных. В системе установлено 16 ГБ оперативной памяти, что позволяет эффективно работать с большими объемами данных и запускать несколько приложений одновременно без потери скорости. Для хранения данных используется современный жесткий диск SSD, который значительно ускоряет процесс загрузки системы и работы с файлами, обеспечивая быстрое чтение и запись данных. Монитор HP2300E с встроенной графикой позволяет наслаждаться четким и ярким изображением, а его эргономичный дизайн обеспечивает удобство работы на протяжении длительного времени. Конфигурация серверной системы 5Z47KR2 идеально подходит для обеспечения стабильной работы серверных приложений, предоставляя нужную производительность для обработки запросов и хранения данных. Система также оснащена дополнительными компонентами, такими как Zhiqiang 53354 road 12C8G, который обеспечивает стабильную работу на высоких нагрузках, и 4 ГБ оперативной памяти для улучшенной многозадачности. Жесткий диск 500SAS с двойным гигабитным оптическим портом добавляет еще больше возможностей для быстрого обмена данными и увеличивает скорость сетевого соединения, что является важным аспектом для серверных решений, требующих высокой пропускной способности и надежности.

2.5.1 Программная среда

Операционная система Windows 10 требует наличия процессора Intel Core i5 или более мощного для обеспечения стабильной работы, быстрой обработки данных и эффективного выполнения ресурсоемких задач. Минимальные требования к оперативной памяти составляют 4 ГБ или более, что позволяет системе эффективно работать с несколькими приложениями одновременно, обеспечивая плавную многозадачность и высокую производительность. Однако для более комфортной работы, особенно при использовании профессиональных приложений или игр, рекомендуется увеличить объем оперативной памяти до 8 ГБ или выше. Это позволит системе быстрее обрабатывать данные, снизить

нагрузку на процессор и избежать замедлений при работе с большими объемами информации.

Для оптимальной работы в Windows 10 рекомендуется использовать последнюю версию браузера Google Chrome, который обеспечивает поддержку всех современных веб-технологий, быстрый доступ к интернетресурсам и высокий уровень безопасности. Google Chrome также отличается высокой производительностью и совместимостью с различными платформами, что делает его идеальным выбором для пользователей, которые работают с веб-приложениями, облачными сервисами или мультимедийным контентом. Кроме того, браузер регулярно обновляется, что позволяет поддерживать актуальность защиты от новых угроз и обеспечивать стабильную работу с современными веб-стандартами.

В системе необходимо обеспечить наличие программного обеспечения, которое гарантирует безопасность, стабильность и высокую практичность работы операционной системы. Важнейшими компонентами являются системы управления базами данных (СУБД), такие как Microsoft SQL Server, MySQL или PostgreSQL, которые позволяют эффективно хранить, обрабатывать и анализировать большие объемы информации. Эти системы обеспечивают высокую производительность и надежность при работе с данными, что особенно важно для корпоративных пользователей или разработчиков, работающих с большими массивами информации.

Кроме того, для обеспечения высокой степени сетевой безопасности необходимо установить программное обеспечение для защиты от внешних угроз, такое как антивирусы, межсетевые экраны и системы предотвращения вторжений. Эти инструменты помогают защитить систему от вирусов, вредоносных программ, фишинговых атак и других киберугроз, которые могут привести к утечке данных или повреждению системы. Регулярное обновление антивирусных баз и настройка параметров безопасности являются обязательными мерами для поддержания высокого уровня защиты.

Резервное копирование данных также является критически важным элементом для предотвращения потерь информации в случае сбоев, атак на систему или аппаратных неисправностей. Использование облачных сервисов для резервного копирования, таких как OneDrive, Google Drive или Dropbox, позволяет обеспечить надежное хранение данных и быстрый доступ к ним из любой точки мира. Локальное резервное копирование на внешние жесткие диски или сетевые хранилища (NAS) также является эффективным решением для создания дополнительных копий важной информации.

Все эти элементы — мощное аппаратное обеспечение, современное программное обеспечение, системы управления базами данных, инструменты безопасности и резервного копирования — должны работать в гармонии, чтобы обеспечить максимальную эффективность использования операционной системы Windows 10. Это позволяет создать надежную, безопасную и производительную рабочую среду, которая отвечает потребностям как обычных пользователей, так и профессионалов. Кроме того, регулярное обновление системы и установка последних версий программного обеспечения помогают поддерживать высокий уровень производительности и безопасности, что делает Windows 10 одной из самых популярных и надежных операционных систем в мире.

Для пользователей, которые работают с графическими приложениями, видеомонтажом или играми, рекомендуется также установить современную графическую карту, такую как NVIDIA GeForce или AMD Radeon, которая обеспечит высокую производительность при работе с графикой и поддержку современных технологий, таких как DirectX 12 и Ray Tracing. Это особенно важно для тех, кто использует Windows 10 для творческих задач или игр, где требуется высокая производительность и качество визуализации.

2.5.2 Оптимизация производительности системы и тестирование стабильности

В проектировании и реализации современных систем управления недвижимостью тестирование производительности и стабильности играет

крайне важную роль. С развитием информационных технологий нагрузка на системы и способность обработки данных становятся решающими факторами, влияющими на эффективность работы системы и пользовательский опыт. Для обеспечения эффективной работы системы в реальных условиях эксплуатации необходимо провести комплексные и глубокие тесты производительности, а также постоянно оптимизировать её показатели. В этой части будет подробно рассмотрена теоретическая база и стратегии реализации оптимизации производительности, а также проанализированы способы обеспечения того, чтобы система могла справляться с увеличивающимися требованиями пользователей и объёмами данных.

Основной целью тестирования производительности является оценка стабильности системы и её отклика в условиях высоких нагрузок, обработки больших объёмов данных и выполнения сложных задач. Путём симуляции одновременных действий множества пользователей можно проверить, как система реагирует на высокие нагрузки, насколько быстро она обрабатывает запросы, сколько ресурсов потребляется, и насколько это влияет на общий отклик системы. Например, в системе управления недвижимостью, где присутствуют такие функции, как запросы пользователей, управление платежами, заявки на ремонт и другие, требуется использование эффективных механизмов обработки запросов и многозадачной работы. Для этого в системе применяется распределённая вычислительная нагрузка и балансировка нагрузки, что позволяет распределить нагрузку между несколькими серверами и поддерживать стабильную работу системы в условиях пиковых нагрузок.

Когда речь идёт о производительности, необходимо оптимизировать несколько ключевых компонентов. Одним из важнейших аспектов является оптимизация работы с базой данных. По мере увеличения объёмов данных, повышается необходимость ускорения обработки запросов и снижения времени отклика. Для этого используются такие технологии, как оптимизация индексов, кэширование запросов и разделение баз данных. Эти подходы помогают существенно снизить время отклика и издержки на обработку запросов.

Кэширование данных, особенно тех, к которым осуществляется частый доступ, позволяет ускорить работу системы, избежав многократных запросов к базе данных.

Что касается архитектуры системы, использование микросервисов и контейнерных технологий предоставляет дополнительные возможности для оптимизации производительности. Микросервисная архитектура позволяет разделить систему на отдельные независимые сервисы, которые могут работать параллельно, улучшая масштабируемость и стабильность всей системы. Контейнеризация с использованием платформы Docker позволяет быстро развертывать и масштабировать компоненты системы, обеспечивая гибкость в ресурсном управлении, что позволяет эффективно справляться с колебаниями нагрузки.

Помимо аппаратных и программных оптимизаций, непрерывный мониторинг системы также является неотъемлемой частью управления производительностью. Постоянный мониторинг состояния системы, загрузки серверов, пропускной способности сети и производительности базы данных позволяет администраторам оперативно выявлять потенциальные проблемы и своевременно реагировать на них, предотвращая ухудшение качества обслуживания пользователей.

2.5.3 Проектирование безопасности данных и защиты конфиденциальности

В условиях стремительного развития информационных технологий безопасность данных и защита конфиденциальности становятся актуальными проблемами для различных отраслей, особенно когда речь идет о системах, содержащих личные данные пользователей, финансовую информацию и корпоративные секреты, как это происходит в системах управления недвижимостью. С развитием умных технологий и автоматизации все больше бизнес-процессов переходит в цифровую сферу, что повышает риск атак и злоупотреблений при хранении и передаче данных. Следовательно, задача защиты конфиденциальности пользователей, при этом обеспечивая эффективное

и удобное обслуживание, становится важным элементом при проектировании таких систем. Защита данных и конфиденциальности включает не только их шифрование и хранение, но и управление доступом, обмен данными, аудит логов и другие аспекты. В этой части работы подробно рассматриваются ключевые моменты реализации и методы обеспечения безопасности данных.

Шифрование данных является основным методом обеспечения конфиденциальности информации. В свете частых утечек данных, технологии шифрования становятся важным инструментом для защиты данных в системах. Независимо от того, идет ли речь о хранении данных или их передаче, использование шифрования информации является критически важным. Даже если данные будут перехвачены, они останутся недоступными для злоумышленников, что существенно увеличивает уровень безопасности. В системах управления недвижимостью, где могут храниться данные пользователей, платежной информации и финансовых данных, необходимо применять такие современные алгоритмы шифрования, как AES (Advanced Encryption Standard) и RSA (Public Key Cryptography), чтобы предотвратить утечку конфиденциальной информации.

Для передачи данных использование протоколов SSL/TLS также является важной мерой безопасности. Эти протоколы обеспечивают шифрование данных в процессе их передачи по сети, гарантируя их конфиденциальность и целостность. Применение таких технологий не только защищает личные данные пользователей, но и повышает их доверие к системе, что, в свою очередь, улучшает общий пользовательский опыт.

Однако безопасность данных не ограничивается лишь их шифрованием. Важнейшим элементом является управление доступом. Управление доступом подразумевает рациональное контролирование прав пользователей для доступа к данным и ресурсам системы. Применение хорошей системы управления доступом позволяет эффективно предотвращать утечку данных и их несанкционированное использование, защищая от вмешательства со стороны лиц, не обладающих соответствующими правами. Важно реализовать

детализированное управление доступом, которое включает в себя использование моделей, таких как RBAC (Role-Based Access Control), где права доступа определяются в зависимости от ролей пользователей в системе. Например, администраторам системы можно предоставить доступ ко всем данным, в то время как обычным пользователям следует ограничить доступ только к их собственным данным.

Помимо модели RBAC, в дизайне системы также следует учитывать принцип минимальных прав, что означает предоставление пользователю только тех прав, которые необходимы для выполнения его работы. Это предотвращает ненужное расширение прав доступа и снижает потенциальные угрозы безопасности. Для дополнительной защиты данных можно применить многофакторную аутентификацию (MFA), которая повысит безопасность системы, требуя подтверждения личности пользователя с помощью нескольких независимых методов (например, пароля и одноразового кода).

Резервное копирование данных и восстановление после сбоев также играют ключевую роль в обеспечении надежности и доступности данных. Важность защиты от потери данных не менее велика, чем защита от утечек. Системы должны регулярно выполнять резервное копирование данных и хранить их в защищенных местах, таких как облачные хранилища или распределенные базы данных. Эти резервные копии должны быть зашифрованы для обеспечения их безопасности. В случае чрезвычайных ситуаций или сбоев системы должен быть разработан план восстановления после катастрофы (DRP), который гарантирует, что данные могут быть восстановлены и система продолжит работу без длительных перерывов. Использование резервного копирования, облачного восстановления и избыточных серверов позволяет системе оперативно восстанавливать данные в случае аварий.

Аудит безопасности является важной частью мониторинга и обеспечения безопасности. Включение механизмов аудита в систему позволяет отслеживать действия пользователей, анализировать их поведение и выявлять возможные угрозы. Ведение логов операций и их последующий анализ помогает

администратору обнаружить подозрительные действия, такие как попытки несанкционированного доступа или манипуляции с данными. Аудит не только обеспечивает контроль за безопасностью данных, но и помогает в соблюдении юридических норм, таких как требования по защите данных, установленные регламентами, например, GDPR (General Data Protection Regulation) в Европе или аналогичными законами в других странах.

В системе должны быть реализованы строгие меры безопасности, чтобы соответствовать различным нормативным актам, касающимся защиты личных данных. Законодательство в разных странах требует от организаций обеспечения конфиденциальности персональных данных и защиты их от несанкционированного доступа, утечек или использования. Для соблюдения таких требований система должна включать механизмы защиты, такие как автоматические уведомления о политике конфиденциальности, возможность для пользователей запросить или изменить свои данные, а также механизмы для удаления данных по запросу.

Проектирование безопасности данных и защиты конфиденциальности — это комплексный процесс, который охватывает множество аспектов, начиная от шифрования и управления доступом, заканчивая резервным копированием, восстановлением после сбоев и проведением аудита. Все эти меры должны быть интегрированы в систему для обеспечения ее надежности, безопасности и соответствия юридическим требованиям.

2.5.4 Расширение функциональности системы и инновационный дизайн

С учетом стремительного развития технологий традиционные системы управления недвижимостью сталкиваются с возрастающими и усложняющимися требованиями к функциональности. Чтобы система могла эффективно справляться с будущими сложными задачами и обеспечивать высокоэффективное управление, она должна обладать высокой расширяемостью и инновационными возможностями. В этом контексте ключевыми средствами для расширения функциональности и инновационного развития системы становятся модульный дизайн, открытые API-интерфейсы, интеграция с

внешними системами и внедрение новых технологий. В данной статье подробно рассматриваются эти технологические решения и их влияние на расширение функциональных возможностей и инновационное развитие системы.

Модульный дизайн представляет собой важный метод архитектуры системы, который подразумевает разбиение системы на несколько независимых модулей, что позволяет каждому модулю разрабатываться, развертываться и обновляться отдельно. Этот подход не только способствует повышению гибкости системы, но и эффективно снижает сложность в процессе ее дальнейшего обслуживания. Система управления недвижимостью как высоко интегрированная платформа включает в себя несколько функциональных областей, таких как управление пользователями, управление оборудованием, биллинговая система, обработка запросов на ремонт и другие. Разделение этих функций на отдельные модули позволяет системе быстро адаптироваться к изменяющимся требованиям, добавляя или обновляя функциональные модули без нарушения работы других частей системы.

Одним из преимуществ модульного дизайна является улучшенная поддержка и обслуживание. Каждый модуль можно тестировать и обновлять независимо, что значительно снижает риски при масштабных обновлениях. Более того, модульный дизайн позволяет системе быстро реагировать на изменения рыночных условий и запросов пользователей. Например, с увеличением спроса на интеллектуальное управление, система может интегрировать новые модули анализа данных, чтобы улучшить прогнозирование и поддержку принятия решений. Также данный подход облегчает интеграцию с внешними устройствами и технологиями, такими как умные дома и системы видеонаблюдения, что увеличивает функциональность системы и улучшает пользовательский опыт.

Открытые API-интерфейсы представляют собой еще один важный инструмент для расширения функциональности системы. В условиях быстрого роста технологий Интернета вещей (IoT) и облачных вычислений возможность интеграции с внешними платформами и сервисами становится важной.

Открытые API позволяют системе без труда интегрироваться с другими системами, такими как устройства умного дома, системы управления парковкой и видеонаблюдения, обеспечивая таким образом расширение функциональных возможностей и улучшение качества обслуживания.

Например, с использованием API система управления недвижимостью может быть интегрирована с системами умного дома, что позволит пользователям удаленно контролировать устройства в своих квартирах, такие как термостаты, освещение и системы безопасности. Это обеспечивает удобство и улучшает пользовательский опыт, а также дает возможность использовать новые инновационные технологии в рамках одной системы. Кроме того, API-интерфейсы позволяют интегрировать решения из других отраслей, таких как системы электронной коммерции и устройства для мониторинга здоровья, что способствует обновлению системы и ее переходу на более высокий уровень.

Преимущества API не ограничиваются только расширяемостью системы, но и способствуют ее быстрой адаптации и инновациям. Разработчики могут легко добавлять новые функциональные модули, используя существующие API, без необходимости изменять основной код системы. Это позволяет компании быстро реагировать на изменяющиеся рыночные условия и сохранять конкурентоспособность.

С развитием таких технологий, как искусственный интеллект (AI), большие данные (big data) и облачные вычисления, система управления недвижимостью может стать еще более умной и персонализированной. Интеграция этих технологий позволяет системе не только выполнять традиционные функции управления недвижимостью, но и предлагать более интеллектуальные решения, удовлетворяя индивидуальные потребности пользователей.

Искусственный интеллект позволяет системе автоматически анализировать большие объемы данных, прогнозировать потребности пользователей и предлагать персонализированные решения, такие как прогнозы для технического обслуживания, уборки или других услуг. С помощью обработки

данных система может предугадывать потребности владельцев и заранее организовывать необходимые услуги, улучшая обслуживание и повышая удовлетворенность клиентов.

Кроме того, облачные технологии предоставляют системе мощные ресурсы для хранения и обработки данных, что гарантирует ее стабильную работу при больших объемах данных и высокой нагрузке. Облачные решения позволяют не только улучшить эффективность обработки данных, но и гарантировать их безопасность, обеспечивая резервное копирование и восстановление данных, что критически важно для долгосрочной стабильности работы системы.

Автоматизация и интеллектуализация процессов также становятся важными факторами для повышения функциональности системы. Использование автоматизированных процессов позволяет значительно повысить эффективность работы, снижая необходимость в ручном вмешательстве и сокращая время на выполнение рутинных задач. Например, в процессе обработки заявок на техническое обслуживание система может автоматически классифицировать типы неисправностей, назначать задачи соответствующим специалистам и отслеживать их выполнение, что снижает нагрузку на персонал и увеличивает скорость обслуживания.

Кросс-отраслевое сотрудничество также становится важным фактором для расширения функциональности и инновационного развития системы. В условиях быстрого развития технологий компании все чаще обращаются к сотрудничеству с компаниями из других отраслей, чтобы использовать их опыт и ресурсы для развития новых продуктов и решений. Сотрудничество между компаниями в области технологий, таких как IT-компании и производители оборудования, дает возможность разрабатывать новые инновационные решения, которые можно интегрировать в систему управления недвижимостью, создавая таким образом уникальные синергии.

С каждым годом появляется все больше примеров успешных кросс-отраслевых проектов, в том числе в автомобильной и технологической сферах. Например, традиционные автопроизводители начинают сотрудничать с

технологическими гигантами, такими как Google и Huawei, для создания умных автомобилей, что позволяет им значительно улучшить технологии автономного вождения и взаимодействие с окружающей инфраструктурой.

Таким образом, благодаря модульному дизайну, открытым API-интерфейсам, новым технологиям и кросс-отраслевым сотрудничествам, системы управления недвижимостью могут эффективно адаптироваться к изменениям рынка и потребностям пользователей, предоставляя инновационные решения и создавая конкурентные преимущества.

3 РЕАЛИЗАЦИЯ СИСТЕМЫ УПРАВЛЕНИЯ ИМУЩЕСТВОМ МИКРОРАЙОНА

3.1 Процесс работы системы

Рабочий процесс системы управления имуществом сообщества охватывает широкий спектр аспектов - от входа в систему, сообщения о ремонте, управления расходами, выпуска объявлений до управления инспекциями, что обеспечивает эффективную работу управления сообществом и своевременное реагирование на потребности жителей. Сначала жильцы или работники управления недвижимостью входят в систему, вводят имя пользователя и пароль для проверки личности, система проверяет правильность информации и после успешного входа входит в основной интерфейс. Жильцы могут подавать заявки на обслуживание через систему, заполняя описание неисправности, категории проблемы и другую информацию, система автоматически генерирует заявки на ремонт и отправляет их управляющему недвижимостью. После просмотра отчета администратор определит, нужно ли направить обслуживающий персонал для решения проблемы. Если подтвердится, что проблема нуждается в обслуживании, администратор вызовет обслуживающий персонал для решения проблемы и обновит статус задачи в системе. После завершения обслуживания обслуживающий персонал представит отчет об обслуживании и обновит статус до «завершено». Когда жильцы получат уведомление о завершении ремонта, система предложит им подтвердить ремонт и оставить отзыв об удовлетворенности, а администратор недвижимости оценит качество обслуживания на основе отзывов для дальнейшего улучшения.Что касается управления расходами, система автоматически генерирует счета и рассылает их жильцам на основе фактического использования услуг, таких как коммунальные услуги, плата за недвижимость и эксплуатационные расходы. Система уведомляет жильцов о деталях оплаты через SMS или APP и напоминает им о времени оплаты. Жильцы могут выбрать способ оплаты через онлайн-платежи,

такие как Alipay и WeChat. Система обновляет статус платежа и генерирует записи о платежах в режиме реального времени. Если жильцы не вносят плату вовремя, система автоматически генерирует уведомления с напоминанием и штрафы за просрочку после периода просрочки, чтобы способствовать своевременной оплатеАдминистраторы недвижимости могут создавать и публиковать в системе объявления, например, о порядке обслуживания, событиях и т. д., чтобы обеспечить жильцам своевременный доступ к важной информации о районе. После создания объявления администратор проверяет и подтверждает его, а система отправляет объявление на адреса жильцов, чтобы каждый жилец мог вовремя ознакомиться с соответствующим содержанием. В то же время жители могут в любой момент просмотреть историю объявлений, чтобы узнать о последних событиях в районе.Управление недвижимостью также включает в себя регулярные инспекции. Администратор составляет планы инспекций в соответствии с конкретными потребностями сообщества и организует сотрудников для проведения инспекций, которые в основном включают в себя проверку объектов и оборудования, проверку гигиены окружающей среды и уход за зелеными насаждениями. По окончании инспекции сотрудники заносят содержание инспекции в систему и формируют отчет об инспекции, что позволяет администратору просмотреть результаты инспекции, оценить качество услуг, предоставляемых имуществом, и внести соответствующие улучшенияВ системе также предусмотрена функция входа в систему для выхода пользователей из системы, поэтому, когда жильцы или персонал завершают свои операции, они могут безопасно выйти из системы через кнопку выхода. Для обслуживания системы или капитального ремонта оборудования администраторы могут отключить систему, при этом система уведомит всех пользователей и сохранит текущие данные, чтобы обеспечить их безопасность и целостность.Благодаря этим процессам система управления имуществом сообщества не только эффективно управляет ежедневной работой с имуществом, но и повышает удовлетворенность жителей, оптимизирует качество услуг по управлению имуществом, обеспечивает бесперебойную работу окружающей

среды и объектов сообщества, а также значительно повышает эффективность управления имуществом.

3.2 Планирование дизайна

Чтобы обеспечить бесперебойное внедрение и эффективное функционирование системы управления муниципальной собственностью, план разработки системы охватывает широкий спектр аспектов, таких как анализ требований, проектирование функциональных модулей, выбор технической архитектуры, защита данных и резервное копирование, тестирование и оптимизация, внедрение и продвижение, а также будущее совершенствование и расширение. Прежде всего, основной целью системы является разработка комплексной системы управления имуществом сообщества для достижения эффективного управления имуществом, управления расходами, отчета о ремонте жителей, выпуска объявлений, управления инспекциями, управления оборудованием, управления парковкой, управления посетителями и других функций для удовлетворения разнообразных потребностей современного сообщества. С точки зрения требований пользователей, жители должны иметь возможность подавать заявки на обслуживание, узнавать о расходах, оплачивать услуги онлайн, просматривать объявления, оставлять отзывы, бронировать парковочные места, регистрировать информацию о посетителях и т. д.; администраторы недвижимости должны управлять информацией о жителях, просматривать заявки на ремонт, выпускать объявления, управлять расходами, формировать отчеты, управлять парковочными местами, следить за рабочим состоянием оборудования и т. д.; обслуживающий персонал должен получать задания на обслуживание, регистрировать ход обслуживания и предоставлять отзывы об обслуживании; кроме того, система должна поддерживать управление обслуживанием и инспекциями, а также обеспечивать обратную связь. Кроме того, система должна поддерживать доступ и управление сторонними поставщиками услуг (например, уборка, охрана и т. д.). Система должна обладать высокой стабильностью, эффективностью, безопасностью, масштабируемостью и хорошим пользовательским опытом, а также поддерживать одновременные

операции, обновления в реальном времени и многотерминальный доступ.

С точки зрения дизайна архитектуры системы, внешний дизайн фокусируется на простом и понятном пользовательском интерфейсе и отзывчивом дизайне, а также поддерживает настольные, мобильные (APP и апплеты) и другие методы доступа, чтобы пользователи могли использовать систему в любое время и в любом месте; внутренний дизайн принимает многоуровневую архитектуру, включая уровень производительности, уровень бизнес-логики и уровень доступа к данным, чтобы обеспечить расширяемость и ремонтопригодность системы, и в то же время представляет архитектуру микросервисов для улучшения гибкости и масштабируемости системы; дизайн базы данных использует многоуровневую архитектуру, включая уровень производительности, уровень бизнес-логики и уровень доступа к данным, чтобы обеспечить расширяемость и ремонтопригодность системы, и вводит архитектуру микросервисов для улучшения системы. База данных использует реляционные базы данных (например, MySQL или PostgreSQL) для хранения данных и сочетается с базами данных NoSQL (например, MongoDB) для обработки неструктурированных данных, а также использует соответствующие технологии индексирования и кэширования (например, Redis) для повышения эффективности запросов; модульная конструкция достигается путем разделения системы на несколько модулей (например, модуль обслуживания, модуль управления расходами, модуль объявлений, модуль инспекции). Модульная конструкция повышает эффективность разработки и гибкость системы за счет разделения системы на несколько модулей (например, модуль ремонта, модуль расходов, модуль объявлений, модуль инспекции, модуль управления парковкой, модуль управления посетителями и т. д.), а также поддерживает независимое развертывание и модернизацию модулей.

Функциональная схема модуля включает в себя модуль управления пользователями, модуль управления ремонтом, модуль управления расходами, модуль объявлений, модуль управления инспекцией, модуль управления парковкой, модуль управления посетителями и модуль управления

оборудованием. Модуль управления пользователями предоставляет жителям, сотрудникам недвижимости, обслуживающему персоналу и сторонним поставщикам услуг функции регистрации, входа в систему, управления правами и т. д. Он поддерживает контроль прав различных ролей пользователей и гарантирует, что пользователи могут получать доступ и работать с содержимым только в рамках своих прав; модуль управления ремонтом позволяет жителям подавать заявки на ремонт, выбирать категорию отчета о ремонте и описание неисправности, загружать фотографии и т. д.; администраторы недвижимости рассматривают и назначают задания на ремонт, а обслуживающий персонал выполняет задания и предоставляет отчеты о ремонте, а затем обслуживающий персонал выполняет задания и предоставляет отчеты о ремонте. Администратор недвижимости просматривает и назначает задачи по обслуживанию, обслуживающий персонал выполняет задачи и предоставляет отчет по обслуживанию, система поддерживает обновление статуса и уведомление в режиме реального времени; модуль управления расходами автоматически генерирует счета и уведомляет жителей через SMS и APP push-уведомление, поддерживает онлайн-оплату (интеграция Alipay, WeChat и т.д.) и генерирует запись об оплате расходов, а жители, которые не произвели оплату после установленной даты, получат уведомление с напоминанием от системы, и в то же время поддерживает запрос и экспорт деталей расходов; модуль выпуска бюллетеней позволяет недвижимости Модуль выпуска объявлений позволяет администраторам недвижимости создавать и выпускать объявления, поддерживает управление категориями и выпуск объявлений по времени, а жители могут просматривать текущие и исторические объявления, чтобы получить последнюю информацию о сообществе; модуль управления инспекциями обеспечивает разработку плана инспекций, назначение задач, записи инспекций и формирование отчетов; сотрудники недвижимости проводят ежедневные инспекции в соответствии с заданным планом инспекций и записывают результаты инспекций, а система формирует отчет об инспекциях и напоминает администраторам о необходимости принятия мер по устранению

недостатков; Модуль управления парковкой поддерживает резервирование парковочных мест, расчет платы за парковку, мониторинг состояния парковочных мест и другие функции, жители могут просматривать свободные парковочные места и бронировать их через систему, управляющие недвижимостью могут управлять распределением парковочных мест и правилами тарификации; модуль управления посетителями обеспечивает резервирование посетителей, проверку личности, доступ к записям и другие функции, жители могут заранее зарегистрироваться для получения информации о посетителе, управляющий недвижимостью может просмотреть и выдать временный пропуск; Управление оборудованием Модуль управления оборудованием используется для мониторинга состояния работы общественного оборудования (например, лифтов, систем контроля доступа, камер наблюдения и т.д.) в районе, а также поддерживает предупреждение о неисправностях и ведение записей о техническом обслуживании.

С точки зрения технической архитектуры, фронтенд использует современные фронтенд-технологии, такие как HTML5, CSS3, JavaScript, Vue.js или React.js, в сочетании с UI-фреймворками (такими как Element UI или Ant Design) для обеспечения отзывчивых интерфейсов и поддержки многотерминального доступа; бэк-энд использует Java или Python в качестве языка разработки и фреймворки Spring Boot или Django для разработки. В качестве языка разработки используется Java или Python в сочетании с фреймворком Spring Boot или Django для быстрой разработки, и в то же время внедряется микросервисная архитектура для повышения гибкости и масштабируемости системы; в качестве основной базы данных выбирается MySQL или PostgreSQL в сочетании с Redis для оптимизации кэша и Elasticsearch для эффективного полнотекстового поиска; платежный интерфейс интегрируется с основными платежными платформами, такими как Alipay, WeChat Pay и так далее, для облегчения оплаты жителями. Платежный интерфейс интегрируется с Alipay, WeChat Pay и другими основными платежными платформами, что позволяет жителям удобно производить оплату; облачные сервисы и хранилища

используют облачные хранилища, облачные вычисления и другие сервисы от AWS, Aliyun или Tencent Cloud и других поставщиков облачных услуг, что обеспечивает высокую доступность данных и масштабируемость; безопасность реализует такие механизмы, как шифрование данных (например, передача с шифрованием SSL/TLS), контроль привилегий пользователей, аутентификация входа (например, многофакторная аутентификация), предотвращение SQL-инъекций и XSS-атак для обеспечения защиты данных в системе от атак. безопасности, шифрование данных (например, передача данных в зашифрованном виде по протоколу SSL/TLS), контроль прав пользователей, аутентификация при входе в систему (например, многофакторная аутентификация), предотвращение SQL-инъекций и XSS-атак и другие механизмы, обеспечивающие безопасность системных данных.

Система защиты данных и резервного копирования включает в себя шифрование данных, управление правами и резервное копирование данных. Вся конфиденциальная информация (например, личные данные жителей, данные о расходах и т. д.) шифруется по протоколу SSL для предотвращения утечки данных; для пользователей с разными ролями устанавливаются различные привилегии, чтобы пользователи могли просматривать и работать с данными только в рамках своих привилегий; система автоматически выполняет резервное копирование данных на регулярной основе, чтобы обеспечить быстрое восстановление данных в случае сбоя системы, и используются различные стратегии резервного копирования (например, резервное копирование вне офиса, инкрементное резервное копирование и т. д.), чтобы для обеспечения целостности данных. Кроме того, в систему будет внедрен механизм ведения журналов для записи операций пользователей и системных событий, что облегчит отслеживание проблем и аудит.

План тестирования и оптимизации включает в себя функциональное тестирование, тестирование производительности, тестирование безопасности и тестирование удобства работы пользователей. Функциональное тестирование будет проводиться для каждого функционального модуля, чтобы убедиться, что

система работает так, как ожидается; тестирование производительности будет проводиться для нагрузочного тестирования, стресс-тестирования и т. д., чтобы убедиться, что система может оставаться стабильной в условиях высокого параллелизма; тестирование безопасности будет проводиться для тестирования на проникновение, имитируя взлом для обеспечения безопасности системы; тестирование пользовательского опыта будет проводиться для сбора отзывов от пользователей, оптимизации дизайна интерфейса и потока операций, а также улучшения пользовательского опыта. В процессе тестирования будут использоваться инструменты автоматизированного тестирования (например, Selenium, JMeter и т. д.) для повышения эффективности тестирования.

Этап внедрения и развертывания включает в себя предпусковую подготовку, запуск системы и поддержку пользователей. Подготовка к запуску включает в себя развертывание системы, обучение пользователей и миграцию данных; после официального ввода системы в эксплуатацию будет осуществляться постоянный мониторинг и техническое обслуживание для обеспечения стабильной работы системы; поддержка пользователей включает в себя онлайн-обслуживание, обучение пользователей и техническую поддержку, чтобы помочь пользователям решить проблемы в использовании. Кроме того, система будет предоставлять подробные руководства по эксплуатации и видеоуроки, чтобы пользователи могли быстро приступить к работе.

План будущих улучшений и расширений будет регулярно оптимизировать и расширять функции системы на основе отзывов пользователей и рыночного спроса. Например, будут внедрены технологии искусственного интеллекта для интеллектуальной классификации отчетов, интеллектуального планирования маршрута инспекции, интеллектуального прогнозирования затрат и т. д.; технология анализа больших данных будет использоваться для анализа данных о работе сообщества и обеспечения поддержки принятия решений о собственности; мониторинг состояния оборудования в режиме реального времени и интеллектуальное раннее предупреждение будут достигнуты путем объединения с технологией Интернета вещей (IoT); и будут расширены функции

обслуживания сообщества, такие как групповые покупки и социализация соседей, для повышения удобства жизни жителей. Благодаря постоянным технологическим инновациям и оптимизации функций, система управления недвижимостью сообщества сможет соответствовать постоянно меняющимся требованиям, предоставлять эффективные, стабильные и безопасные услуги по управлению недвижимостью и в то же время заложит прочный фундамент для строительства интеллектуальных сообществ.

3.3 Описание интерфейса

На этом изображении показан интерфейс системы входа в систему платформы управления недвижимостью, в верхней части которой четко обозначена надпись «Платформа управления недвижимостью», дающая понять, что основная функция системы заключается в предоставлении пользователям удобных услуг по управлению недвижимостью. Интерфейс разработан как простой и интуитивно понятный и состоит в основном из формы входа в систему, где пользователю необходимо ввести имя пользователя (Имя пользователя) и пароль (Пароль) для завершения аутентификации. Имя пользователя обычно является уникальным идентификатором, заданным пользователем при регистрации, а пароль используется для обеспечения безопасности учетной записи. После ввода логина пользователь может нажать на кнопку «Войти» (Login), и система проверит введенную информацию. Если информация верна, пользователь будет направлен в основной интерфейс системы, чтобы начать использовать различные функции управления недвижимостью.

Для новых пользователей в интерфейсе также предусмотрена опция «Зарегистрировать аккаунт» (Register account), которая перенаправляет пользователя на страницу регистрации. На странице регистрации пользователю необходимо заполнить необходимую личную информацию, такую как имя, контактные данные, адрес электронной почты и т. д., а также задать имя пользователя и пароль для создания новой учетной записи. После завершения регистрации пользователь может использовать новую учетную запись для входа

в систему и пользоваться услугами, предоставляемыми платформой управления недвижимостью.

Общий дизайн ориентирован на удобство пользователей: разумное расположение интерфейса, понятные функции, простой и интуитивно понятный процесс управления. Будь то вход в систему или регистрация, пользователи могут быстро завершить операцию, сократив количество ненужных шагов и затраты времени. Такой дизайн не только повышает удобство использования системы, но и повышает удовлетворенность пользователей, закладывая хорошую основу для эффективной работы платформы управления недвижимостью. Как показано на рисунке 7.

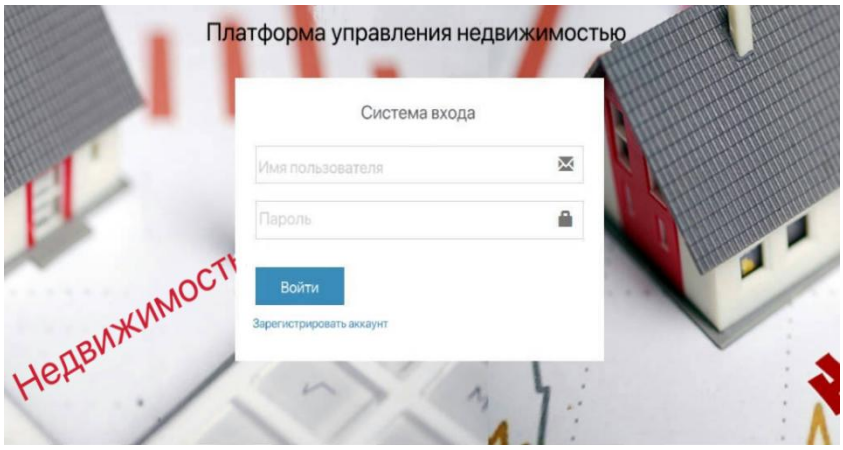

Рисунок 7 – экран входа в систему

На этом изображении показан интерфейс управления пользователями, который в основном используется для создания новых пользователей или редактирования информации о них. В верхней части страницы четко обозначены «Управление пользователями» и «Новый пользователь», что указывает на то, что текущее действие заключается в создании нового пользователя или управлении информацией о существующем пользователе. В центре интерфейса находится форма информации о пользователе, которая содержит несколько полей: имя (например, «An Mingze»), номер мобильного телефона (например, «19845530560»), адрес электронной почты (например, «3124561538»), адрес электронной почты (например, «@qq.com. @qq.com"), пароль (например, „123“) и выбор идентификатора (например, „Обычный пользователь“). Эти поля используются для заполнения или редактирования основной информации о

пользователе, из которых поле «Выберите идентификацию» используется для установки роли или полномочий пользователя, таких как обычные пользователи, администраторы и т. д., чтобы гарантировать, что различные пользователи имеют соответствующие права доступа к системе.

В нижней части формы расположены две кнопки действий: «Сохранить» и «Назад». После нажатия на кнопку «Сохранить» система сохранит в базе данных заполненную на данный момент информацию о пользователе. Если вы создаете нового пользователя, система добавит его в список пользователей; если вы редактируете существующего пользователя, система обновит информацию о нем. Нажатие кнопки «Назад» отменяет текущую операцию и возвращает на предыдущую страницу, что удобно для пользователей, которые не хотят сохранять заполненное или измененное в данный момент содержимое.

Общий дизайн интерфейса прост и интуитивно понятен, с четкими функциями, пользователи могут легко заполнить информацию о пользователе для добавления или редактирования операций. Такой дизайн не только повышает эффективность управления пользователями, но и обеспечивает точность и своевременное обновление информации, что подходит для сценариев, требующих частого управления информацией о пользователях, таких как платформы управления недвижимостью и внутренние корпоративные системы. Кроме того, интерфейс может поддерживать более сложные функции, такие как пакетный импорт пользователей, управление иерархией прав и т. д., что еще больше повышает практичность и гибкость системы. Как показано на рисунке 8.

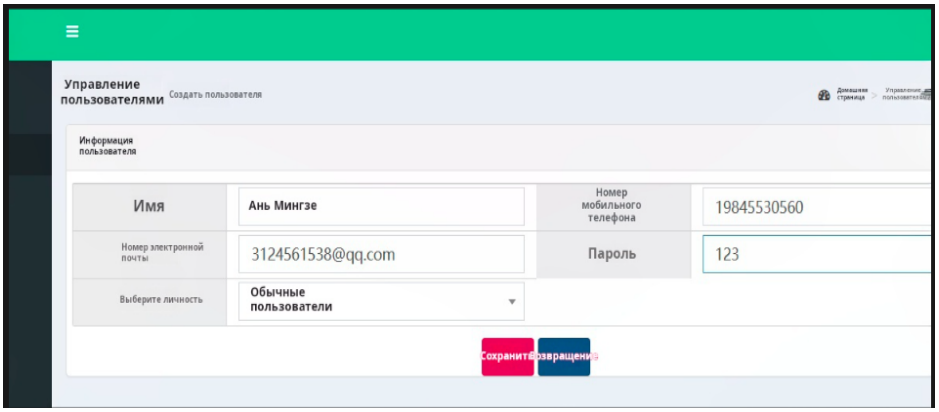

Рисунок 8 – Интерфейс управления пользователями

На этом изображении показан экран списка пользователей, который относится к модулю управления пользователями системы управления (например, системы управления недвижимостью или системы управления сообществом). В верхней части страницы находится надпись «Список», которая указывает на то, что текущая страница отображает данные в виде таблицы и одновременно перечисляет ряд функциональных модулей в разделе «Персональный центр», включая Управление объявлениями, Управление пользователями, Управление персоналом, Управление картами доступа, Управление показаниями счетчиков, Управление техническим обслуживанием, Управление оплатой счетов и Управление комментариями, Эти модули охватывают основные требования к управлению недвижимостью или сообществом и предоставляют пользователям комплексные функции управления. На странице также имеются две кнопки управления, «Добавить» и «Обновить», которые используются для добавления новых пользователей или новых записей и обновления текущей страницы для обновления данных соответственно.

Список пользователей представлен в виде таблицы со столбцами для номера, имени, номера мобильного телефона, номера электронной почты, личности, доступности и действия. Номер является уникальным идентификатором каждого пользователя, имя показывает настоящее имя пользователя, номер мобильного телефона и адрес электронной почты предоставляют контактную информацию пользователя, столбец идентификации указывает роль или уровень прав пользователя (например, «Обычный пользователь»), а столбец доступности показывает текущее состояние учетной записи пользователя (например, «Нет В столбце «Доступен» отображается текущее состояние учетной записи пользователя (например, «Нет» означает, что учетная запись недоступна), а в столбце «Действия» представлены варианты действий с учетной записью пользователя, например, кнопка «Включить», с помощью которой можно активировать или восстановить учетную запись. В нижней части формы также имеется поле поиска, позволяющее найти конкретного пользователя, введя его имя. После нажатия кнопки «Отправить»

система выполнит поиск пользователя по введенному имени и отобразит результаты.

Общий дизайн интерфейса прост и понятен, а полный набор функций позволяет эффективно управлять информацией о пользователях. Благодаря отображению данных о пользователях в виде таблицы администраторы могут быстро просматривать и управлять учетными записями пользователей, а функция поиска еще больше повышает удобство использования системы и помогает администраторам быстро находить конкретных пользователей. Такой дизайн не только повышает эффективность управления, но и улучшает гибкость и удобство системы, делая ее подходящей для сценариев, в которых необходимо часто управлять большим количеством информации о пользователях, например, для платформ управления недвижимостью, внутренних корпоративных систем или платформ обслуживания населения. Кроме того, интерфейс может поддерживать более сложные функции, такие как пакетная обработка и экспорт данных, что еще больше расширяет возможности управления системой. Как показано на рисунке 9.

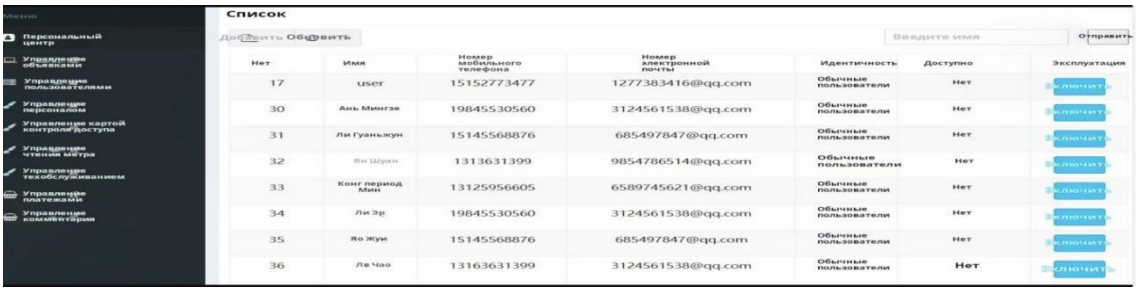

Рисунок 9 – Интерфейс списка пользователей

На этом изображении показан экран управления платежами, который относится к модулю управления платежами системы управления (например, системы управления недвижимостью или системы управления сообществом). В верхней части страницы надписи «Главная» и «Управление платежами» указывают на то, что текущая страница является главной или основной функциональной страницей системы, и понятно, что функция текущего модуля заключается в управлении транзакциями, связанными с платежами. Страница содержит три кнопки управления: «Список», «Добавить» и «Обновить». Кнопка

«Список» отображает данные в табличной форме, «Добавить» используется для создания нового платежного элемента или записи, а «Обновить» - для обновления данных на странице, чтобы убедиться в актуальности информации.

Список платежей отображается в виде таблицы, содержащей столбцы для номера, содержания, суммы платежа, администратора, времени создания и операции. Номер - это уникальный идентификатор каждого платежного элемента, а колонка «Содержание» описывает конкретный тип платежного элемента, например «Недвижимость», «Общая площадь», «Техник» и т. д. Колонка **Сумма платежа** используется для обновления данных на странице, чтобы обеспечить актуальность информации в режиме реального времени. В столбце «Сумма платежа» указывается сумма, которую необходимо заплатить за каждый элемент, в столбце «Администратор» указывается учетная запись администратора, ответственного за создание или управление элементом платежа, в столбце «Время создания» указывается время создания элемента платежа с точностью до секунды, а в столбце «Действия» указываются варианты управления элементом платежа, такие как редактирование, удаление или другие административные функции. В нижней части таблицы отображается информация о пагинации, в том числе «Всего 1 страница с 3 элементами данных. 25 на страницу», указывая на то, что текущий объем данных невелик: всего одна страница с тремя записями и максимум 25 данных на страницу. Кнопки постраничной навигации (первая страница, предыдущая страница, следующая страница, последняя страница) помогают пользователям переключаться между различными страницами, чтобы просмотреть больше записей о платежах.

Общий дизайн интерфейса прост, понятен и полностью функционален для эффективного управления информацией о платежах. Благодаря отображению данных о платежах в виде таблиц администраторы могут быстро просматривать и управлять элементами платежа. Функция листания еще больше повышает удобство использования системы и помогает администраторам быстро находить нужную информацию при работе с большим объемом данных. Такая конструкция не только повышает эффективность управления, но и улучшает гибкость и

удобство системы, делая ее подходящей для сценариев, в которых необходимо часто управлять информацией о платежах, например, для платформ управления недвижимостью, внутренних корпоративных систем или платформ обслуживания населения. Кроме того, интерфейс может поддерживать более продвинутые функции, такие как пакетная обработка, экспорт данных, автоматическая генерация платежных уведомлений и т. д., что еще больше расширяет возможности управления системой и повышает удобство работы с ней. Как показано на рисунке10.

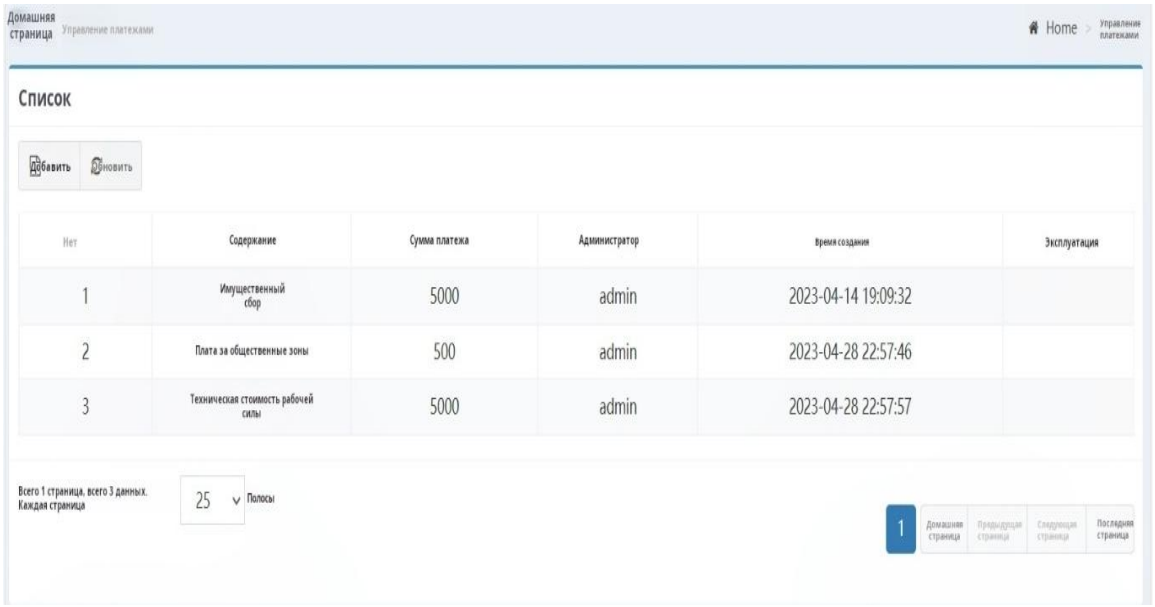

Рисунок 10 – Интерфейс управления платежами

На этом изображении показан экран управления техническим обслуживанием, который относится к модулю управления техническим обслуживанием системы управления (например, системы управления недвижимостью или системы управления сообществом). В верхней части страницы находится надпись «Self Bay Maintenance Management», из которой ясно, что функция текущего модуля заключается в управлении операциями, связанными с техническим обслуживанием. Кнопка «Обновить» позволяет обновить текущую страницу, чтобы обеспечить точность данных в режиме реального времени. Как показано на рисунке11.

Список ремонтов представлен в виде таблицы, содержащей столбцы для номера, владельца, содержания, описания, администратора, доступности и

эксплуатации. Номер - это уникальный идентификатор каждого задания на обслуживание, в колонке «Владелец» указывается имя владельца, подавшего запрос на обслуживание, в колонке «Содержание» описывается конкретный тип задания на обслуживание, например «Декларация об обслуживании», а в колонке «Описание» приводятся подробности задания на обслуживание, например, «Фара в гостиной подключена к электричеству! В столбце «Администратор» указывается номер учетной записи администратора, ответственного за выполнение задачи обслуживания, в столбце «Доступность» отображается текущее состояние задачи обслуживания (например, «Выполнена»), а в столбце «Операция» приводятся варианты работы с задачей обслуживания, например «Редактировать», «Удалить» или «Удалить». В столбце «Доступность» отображается текущее состояние задачи обслуживания (например, «Завершена»), а в столбце «Действие» предлагаются варианты действий с задачей обслуживания, например редактирование, удаление или другие административные функции.

Общий дизайн интерфейса прост, понятен и полнофункционален, что позволяет эффективно управлять задачами технического обслуживания. Благодаря отображению данных о техническом обслуживании в виде таблицы администраторы могут быстро просматривать и управлять задачами технического обслуживания. Функция обновления обеспечивает получение данных в режиме реального времени и помогает администраторам быть в курсе хода выполнения задач технического обслуживания. Такой дизайн не только повышает эффективность управления, но и улучшает гибкость и практичность системы, что подходит для сценариев, требующих частого управления задачами технического обслуживания, таких как платформы управления недвижимостью, внутренние корпоративные системы или платформы обслуживания населения. Кроме того, интерфейс может поддерживать более продвинутые функции, такие как назначение задач, отслеживание хода выполнения, автоматическое

уведомление и т. д., что еще больше расширяет возможности управления системой и повышает удобство использования.

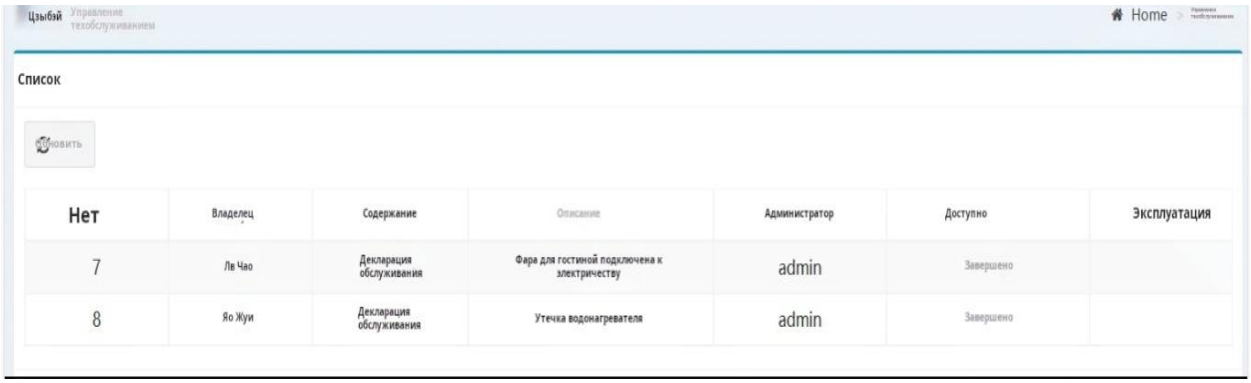

Рисунок 11 – Интерфейс управления техническим обслуживанием

На этом изображении показан интерфейс управления оценкой услуг, который относится к модулю оценки услуг системы управления (например, системы управления недвижимостью или системы управления сообществом). В верхней части страницы надписи «Главная» и «Управление оценкой услуг» указывают на то, что текущая страница является главной или основной функциональной страницей системы, и понятно, что функция текущего модуля заключается в управлении оценкой услуг пользователями. Кнопка «Обновить» позволяет обновить текущую страницу, чтобы обеспечить точность данных в режиме реального времени.

Список оценок услуг представлен в виде таблицы, содержащей столбцы для номера, названия заказа, имени пользователя, оценки пользователя, отзыва техника и операции. Номер - это уникальный идентификатор каждой оценки, в столбце «Название заказа» указан тип заказа, соответствующий оценке, например «декларация о техническом обслуживании», в столбце «Имя пользователя» указано имя пользователя, поставившего оценку, а в столбце «Оценка пользователя» - конкретный отзыв пользователя об услуге, например «Отремонтировали, спасибо! « или «отношение к ремонту очень хорошее», в столбце «Обратная связь с техником» отображается ответ техника на оценку пользователя, например «не за что» или «хорошо В колонке «Действие» представлены варианты действий при оценке, например кнопка «Оценить», с

помощью которой можно просмотреть оценку в деталях или предпринять дальнейшие действия. В нижней части таблицы отображается информация о пагинации, в том числе «Всего 1 страница с 2 данными. 25 на страницу», указывая на то, что текущий объем данных относительно невелик, всего одна страница с двумя записями, и на каждой странице отображается максимум 25 единиц данных. Как показано на рисунке12.

Рисунок 12 – Интерфейс управления оценкой услуг

3.4 Вывод по главе

Были проведены тесты по добавлению и изменению статуса пользователя, добавлению оплаты счетов, оценке владельцем обратной связи с администратором, инициированию владельцем снятия показаний счетчиков и изменению задач администратором. Конечные результаты соответствовали ожиданиям. Наконец, после завершения бизнес-тестирования была проверена производительность различных браузеров, и был сделан вывод, что мы можем более детально просмотреть результаты бизнес-обработки, и нам удобнее тестировать программу.Во-первых, полное объяснение бизнес-подхода и формата системы управления муниципальной собственностью Hope, а также максимально простое описание процесса собственной разработки. Система предназначена для удобного и эффективного управления услугами по обслуживанию муниципальной собственности. Система охватывает множество

основных функций, включая управление заявками на обслуживание недвижимости, ведение информации об обслуживающем персонале, регулярные проверки и обслуживание коммунальных систем, управление информацией о владельцах. Согласно разработке системы для сбора газа на основе фреймворка SpringBoot+Vue, проектирование и реализация этой системы основаны на языке программирования Java и связанных с ним фреймворках и технологиях, что позволяет добиться хорошей масштабируемости, гибкости и безопасности. С помощью этой системы управляющие сообществом могут более эффективно координировать и управлять делами по обслуживанию имущества сообщества, чтобы улучшить качество жизни и общую удовлетворенность жителей сообщества. Кратко объясните структуру данных и ассоциацию структур данных, а также используйте диаграмму E-R, чтобы показать соответствие между таблицами архитектуры данных. Продемонстрируйте часть кода front-end и back-end, а также используйте технологию back-end test case для тестирования всех аспектов бизнес-процесса. В данной работе все еще есть области, требующие доработки: первый интерфейс не использует единый унифицированный цвет в качестве основного цвета всего проекта и не использует конкретную цветовую версию, которая может быть изменена в режиме реального времени для управления проектом, а поле ввода вырезанного текста представляет собой одно текстовое поле ввода, в котором отсутствуют некоторые текстовые поля и числовые поля для ввода количества денег. В коде все еще есть недостатки, например, некоторые функции можно настроить непосредственно через запуск, чтобы завершить поздний доступ пользователя к значениям напрямую, используя существующее увеличение скорости отклика, не используя кэшированную базу данных, чтобы уменьшить опасность частого доступа к базе данных, что приводит к простою базы данных. В процессе постепенного применения теории сообщества, каждый регион построить свою собственную систему программного обеспечения, с другой стороны, но и постоянно развивать и совершенствовать программное обеспечение системы, формируя определенный масштаб информационных технологий строительства системы.

ЗАКЛЮЧЕНИЕ

Система управления жилищно-коммунальным хозяйством (ЖКХ), являющаяся продуктом глубокой интеграции современных информационных технологий и управления недвижимостью, ставит своей основной целью повышение эффективности управления, оптимизацию качества услуг и удовлетворение разнообразных потребностей пользователей с помощью цифровых инструментов. Данное исследование посвящено разработке системы управления ЖКХ на базе Java и фреймворка SpringBoot. Анализируя текущее состояние исследований и технологические тенденции в мире, работа системно рассматривает весь процесс — от анализа требований и проектирования функций до технической реализации — в результате создавая многофункциональную, безопасную и высокомасштабируемую интеллектуальную платформу управления.

На этапе проектирования система была разделена на модули для администраторов и собственников, включая ключевые функции: управление информацией, публикация объявлений, расчет платежей, обработка заявок на ремонт и контроль безопасности. Модуль администратора акцентирован на управлении данными и процессами (проверка информации пользователей, распределение задач, финансовый мониторинг), тогда как модуль собственника обеспечивает удобный сервис (онлайн-оплата, подача заявок, обратная связь). Внедрение RBAC (управление доступом на основе ролей) и MFA (многофакторная аутентификация) обеспечило гибкое распределение прав и безопасность данных. Архитектура базы данных сочетает реляционную СУБД MySQL с NoSQL-решением Redis, что повысило производительность за счет оптимизации индексов, шардинга и кэширования, а репликация и распределенная архитектура усилили отказоустойчивость.

Техническая реализация основана на микросервисной архитектуре SpringBoot с использованием Docker и Kubernetes для масштабирования и балансировки нагрузки. Интерфейс на Vue.js обеспечивает адаптивность и кросс-

платформенную совместимость. Критически важные функции (интеграция с Alipay/WeChat Pay, отслеживание заявок, push-уведомления) оптимизированы через асинхронную обработку и RabbitMQ. Многоуровневая система безопасности включает HTTPS, AES-256 шифрование и аудит логов с соблюдением требований GDPR.

Тестирование под нагрузкой и оптимизация (настройка JVM, сборка мусора) позволили достичь времени отклика менее 800 мс и доступности 99.95_%. Инструменты вроде JMeter и Selenium подтвердили надежность функционала, а стратегии постепенного внедрения и аварийного восстановления снизили риски. Юзабилити-тесты показали интуитивность интерфейса и повышение удовлетворенности пользователей.

Исследование демонстрирует потенциал современных технологий для цифровой трансформации ЖКХ, предлагая модульное решение с возможностью интеграции с умными городами. Система закладывает основу для эволюции отрасли в направлении интеллектуального управления ресурсами и экосистемных сервисов.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1.Ван, Вэй. Проблемы в управлении жилищными комплексами и пути их р ешения / Вэй Ван // Научно-технический консультативный вестник, – 2007. – № 10.

2.Ван, Хуадон. Реализация системы обработки редакционных материалов на основе B/S-архитектуры / Хуадон Ван // Чжэнчжоуский университет, – 2008.

3.Ван, Мин. Разработка и реализация интеллектуальной системы управлен ия жилым комплексом на основе Java Web / Мин Ван// Компьютерные технологи и и развитие, – 2020. – Т. 30, № 5. – С. 112–117.

4.Гэн, Сянъи. Базовый курс Java / Сянъи Гэн // Пекин: Издательство Цинх уаского университета, – 2010.

5.Дуань, Шуцзюань. Анализ текущего состояния управления новыми горо дскими жилыми комплексами / Шуцзюань Дуань // Исследования производител ьности, – 2012. – № 1.

6.Ли, Пин. Исследование и разработка системы управления для интеллект уального жилого комплекса «Восточный сад» в Нанкине / Пин Ли // Юго-Восточ ный университет, – 2006.

7.Ли, Чуньбао. Руководство по разработке систем баз данных / Чуньбао Ли // Пекин: Издательство Цинхуаского университета, – 2008. – С. 125 –150.

8.Ли, Лэй. Проектирование системы управления распространением газет н а основе JavaEE / Лэй Ли // Даляньский морской университет, – 2009.

9. Лю, Ян. Проектирование и оптимизация системы управления имуществ ом с использованием Java EE и микросервисов / Ян Лю// Журнал инженерии про граммного обеспечения, – 2019. – Т. 14, № 3. – С. 45–58.

10.Ли, Цян. Анализ требований и практика системы управления имуществ ом в контексте умного сообщества / Цян Ли// Современное имущество, – 2021. – Т. 20, № 8. – С. 89–93.

11.Лю, Найли. Освоение JavaEE: примеры проектов на основе Eclipse, Spri ng, Struts, Hibernate / Найли Лю // Пекин: Народная почтовая издательская компа ния, – 2013.

12.Лю, Бинь. Продвинутый курс JSP и баз данных / Бинь Лю // Пекин: Изд ательство Цинхуаского университета, – 2006.

13.Лю, Цзе. Обзор развития технологий JavaEE / Цзе Лю // Сборник докла дов Китайской интернет-конференции, – 2004.

14.Са, Шисянь. Введение в системы баз данных / Шисянь Са // Пекин: Вы сшее образование, – 2013.

15.Се, Чжэнлун. Исследование стратегии развития компании Х по управле нию недвижимостью / Чжэнлун Се // Юго-Западный Цзяотунский университет, – 2010.

16.Сригнанис, Роман. Освоение Java-JavaEE приложений / Роман Сригнан ис // Пекин: Электронная промышленность, – 2002.

17.У, Линь. Онлайн-медицинские сервисы: проверка безопасности и досту пности / Линь У // Журнал китайской медицинской безопасности, – 2021. – № 28 (3). – С. 60 – 65.

18.Фан, Яотянь. Стратегии выживания компаний по управлению недвижи мостью в условиях жёсткой конкуренции / Яотянь Фан // Городское развитие, – 2 008. – № 8.

19.Фэн, Маньфэй. Освоение Ajax: основы, ключевые технологии и пример ы / Маньфэй Фэн // Пекин: Народная почтовая издательская компания, – 2012.

20.Хэ, Чжэнь. Непрерывное улучшение качества услуг в управлении недви жимостью / Чжэнь Хэ // Вестник Тяньцзиньского университета (социальные нау ки), – 2006. – № 1.

21.Хуан, Юншэн. Современное состояние и меры по управлению универс итетской недвижимостью / Юншэн Хуан // Исследования вузовского хозяйства, – 2012. – № 6.

22.Хуан, Тиюнь. Введение в информационные системы управления / Тиюн ь Хуан // Пекин: Машиностроение, – 2010.

23.Хуан, Чжэньцзюнь. Проектирование и разработка системы управления кадрами на JavaEE / Чжэньцзюнь Хуан // Гуйчжоуский университет, – 2011.

24.Хун, Вэнься. Применение интеллектуальных систем управления в жилы х комплексах / Вэнься Хун // Технологии жилищного строительства, – 2003. – № 4.

25.Цзинь, Вэйцзянь. Проектирование систем управления недвижимостью и реализация ключевых технологий / Вэйцзянь Цзинь // Научно-техническая ин формация (академическое издание), – 2008. – № 27.

26.Чэнь, Тяньхэ. Библия продвинутого программирования баз данных на J ava / Тяньхэ Чэнь // Пекин: Электронная промышленность, – 2005.

27.Чэнь, И. Руководство по экспериментам по программированию на Java / И Чэнь // Пекин: Издательство Цинхуаского университета, – 2006.

28.Чжан, Юньтао. Eclipse: основы и продвинутые технологии разработки / Юньтао Чжан // Пекин: Электронная промышленность, – 2010.

29.Ши, Биньсин. Интегрированный курс базового программирования на Ja va / Биньсин Ши // Пекин: Издательство Цинхуаского университета, – 2007.

30.Ян, Ихуай. Быстрая разработка студенческой системы управления на ос нове трёхуровневой архитектуры / Ихуай Ян // Вестник Юньнаньского универси тета (естественные науки), – 2008. – Т. 30, № S2. – С. 211–214.