МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Кафедра «Прикладная математика и информатика»
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки, специальности)

Цифровая трансформация бизнеса
(направленность (профиль)/специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему: «Разработка подсистемы визуализации заявок для системы аварийно–диспетчерской службы»

Обучающийся И.В. Балаев
(И.О. Фамилия) (личная подпись)

Руководитель к.т.н. Н.В. Хрипунов
(ученая степень, звание, И.О. Фамилия)

Консультант к.п.н. М.В. Дайнеко
(ученая степень, звание, И.О. Фамилия)

Тольятти 2025

Аннотация

Название темы бакалаврской работы: «Разработка подсистемы визуализации заявок для системы аварийно–диспетчерской службы».

Целью данной выпускной квалификационной работы является проектирование и разработка подсистемы визуализации заявок для системы аварийно–диспетчерской службы, направленной на повышение эффективности обработки заявок, улучшение прозрачности работы службы и упрощение взаимодействия пользователей с платформой «Квартплата 24».

Актуальность работы обусловлена необходимостью автоматизации обработки заявок, предоставлением более наглядной и оперативной информации об инцидентах, а также отсутствием эффективных средств визуализации для служб подобного рода. Работа включает введение, три главы, заключение и приложения.

Выпускная квалификационная работа состоит из нескольких глав, каждая из которых посвящена различным этапам анализа, проектирования и реализации подсистемы визуализации заявок.

В первой главе проводится анализ предметной области, включая описание организации и задачи, анализ текущей модели бизнес–процессов (AS–IS), обзор существующих решений, разработку функциональных требований и проектирование целевой модели (TO–BE).Вторая глава посвящена проектированию подсистемы визуализации заявок. В ней описывается формализация пользователей, логическая модель работы системы и разработка ERD–модели базы данных.Третья глава описывает реализацию подсистемы, включая выбор технологий (HTML, CSS, JavaScript, Leaflet.js, OpenStreetMap), программную реализацию интерфейса и функционала, демонстрационный пример работы системы и расчет экономического эффекта от внедрения.

Бакалаврская работа состоит из 60 страниц и включает 27 рисунков, 4 таблицы, 26 источников.

Abstract

The title of the graduation work is *Developing a subsystem to visualize applications for the emergency dispatch service system.*

The graduation work consists of an introduction, 3 parts, 27 figures, 4 tables, a conclusion, and a list of 26 references including foreign sources.

The aim of this study is to improve the application processing effectiveness through the development of the subsystem for visualizing applications on an interactive city map.

The object of the graduation work is the emergency dispatch service system of Kvartplata 24 platform.

The subject of the research is the application visualization subsystem, which includes application clustering, geolocation features, and a template–based application interface.

The key issue of the graduation work is to reduce the processing time and increase the transparency of the emergency dispatch service workflows by means of visualization and automation.

The graduation work may be divided into several logically connected parts which are: analytical review, system design, and implementation with economic efficiency evaluation.

The first part describes the current state of the emergency dispatch process (AS–IS), analyzes the existing systems, defines the system requirements, and presents the improved model (TO–BE).

The second part is devoted to designing the logical system architecture, the user roles, the UML diagrams, and developing the database scheme.

The third part covers the implementation using HTML, CSS, JavaScript, Leaflet.js, and OpenStreetMap APIs, the application visualization and clustering, as well as the economic effect assesment.

In conclusion, it should be highlighted that that the developed subsystem improves the usability and the effectiveness of the emergency dispatch service.

Содержание

Введение

В ходе преддипломной практики была выполнена работа по теме выпускной квалификационной работы: «Разработка подсистемы визуализации заявок для системы аварийно–диспетчерской службы». Основное внимание уделено созданию функционала, способствующего повышению эффективности работы диспетчерской службы и улучшению взаимодействия пользователей с системой компании «Квартплата 24».

Объектом исследования выступает процесс формирования и обработки заявок, поступающих от пользователей через личный кабинет. Целью практики стало проектирование и разработка подсистемы визуализации заявок, обеспечивающей автоматизацию обработки заявок, отображение их на карте города и внедрение дополнительных инструментов для упрощения взаимодействия [1].

Для достижения поставленной цели решены следующие задачи:

- изучение предметной области и существующих бизнес– процессов системы аварийно–диспетчерской службы [20];
- анализ текущей модели процессов подачи и обработки заявок (AS–IS) и разработка целевой модели (TO–BE);
- создание прототипа подсистемы визуализации заявок с ключевой функциональностью;
- проведение анализа аналогичных решений для учета лучших практик;
- оценка затрат на разработку и определение сроков окупаемости проекта.

Разработанная подсистема позволяет автоматизировать процесс обработки заявок, повысить прозрачность работы диспетчерской службы и удобство использования сервиса для пользователей

Глава 1 Анализ предметной области

1.1 Описание организации

Компания «Квартплата 24» – это цифровая платформа для управления оплатой жилищно–коммунальных услуг, работающая в России. Основные функции платформы включают оплату ЖКУ, передачу показаний приборов учета, отслеживание истории начислений и оплат, а также подачу заявок на обслуживание [14]. Платформа интегрирована с государственными системами, такими как ГИС ЖКХ, и обслуживает более миллиона лицевых счетов в 70 регионах РФ [8].Структурной схемы организации компании представлена на рисунке 1.

Рисунок 1–Иерархическая структура компании «Квартплата 24»

Основной вид деятельности компании по классификатору ОКВЭД относится к сфере предоставления услуг по обработке платежей и расчётов, а также информационных услуг в сфере ЖКХ [23]. Коды ОКВЭД, используемые компанией, включают:

63.11 – «Деятельность по обработке данных, предоставление услуг по

размещению информации и связанная с этим деятельность».

–Включает обработку данных, услуги облачных решений, предоставление платформ для взаимодействия между пользователями.

66.19 – «Другая вспомогательная деятельность в сфере финансовых услуг, кроме страхования и пенсионного обеспечения».

–Подходит для компаний, занимающихся обработкой коммунальных платежей.

68.32 – «Управление недвижимым имуществом за вознаграждение или на договорной основе».

–Может применяться, если компания занимается управлением платежами за коммунальные услуги в ЖКХ.

62.09 – «Деятельность, связанная с использованием вычислительной техники и информационных технологий, прочая».

– Используется для компаний, разрабатывающих IT–решения, включая платформы для расчётов ЖКХ.

Задача разработки подсистемы визуализации заявок для аварийно–диспетчерской службы является частью общей стратегии компании по автоматизации и улучшению взаимодействия с пользователями [3]. Подсистема позволит повысить эффективность обработки заявок, улучшить прозрачность работы диспетчерской службы и упростить взаимодействие пользователей с системой. Основная цель – автоматизация процесса обработки заявок, отображение их на карте города и внедрение дополнительных инструментов для упрощения взаимодействия.

1.2 Модель AS IS

В данной части работы будет рассмотрена текущая модель работы системы аварийно–диспетчерской службы (АДС), которая будет описана как модель «AS–IS». Эта модель отражает существующие процессы и

взаимодействия между пользователями, системой и сотрудниками организации до внедрения новых функциональных возможностей. На основе этой модели будет разработана новая, улучшенная модель «TO–BE», которая будет учитывать предложенные изменения и улучшения. Контекстная диаграмма бизнес–процесса по созданию заявки на оказание услуги AS–IS представлена на рисунке 2.

Рисунок 2– «Контекстная диаграмма бизнес–процесса по созданию заявки на оказание услуги AS–IS»

На данный момент система работы с заявками в рамках аварийно–диспетчерской службы выглядит следующим образом:

Заявка в виде текстового документа – пользователи могут оставить заявку, описав проблему в виде текстового комментария. Этот способ подачи заявки является стандартным, но в текущей системе нет возможности более точно и быстро формулировать запрос, что может приводить к

недоразумениям или задержкам в обработке.

Информация о регистрации заявки – после подачи заявки система регистрирует данные о заявке, такие как адрес, дата и время подачи, а также сам текстовый комментарий. Эти данные важны для того, чтобы сотрудники диспетчерской службы могли отслеживать статус каждой заявки и правильно организовать их обработку.

Информационный запрос – когда заявка зарегистрирована, система может отправить запрос на дальнейшую обработку, например, на проверку состояния оборудования или на подтверждение наличия проблемы. Эти запросы используются для дальнейшей работы с заявками.

Информация от пользователя (текстовые комментарии) – заявки в основном основаны на текстовых комментариях от пользователей. Это создаёт некоторые трудности в процессе обработки, так как текст может быть не всегда структурированным и содержать недостаточно конкретной информации. В связи с этим важно, чтобы система могла дополнительно собирать уточняющие данные от пользователей для более точной диагностики.

Сотрудники организации – сотрудники аварийно–диспетчерской службы участвуют в процессе обработки заявок. Они проверяют информацию, реагируют на запросы и координируют действия по устранению проблемы. Однако в текущей системе нет достаточно эффективного способа отслеживания количества заявок по одному адресу или типу проблемы, что усложняет процесс реагирования на массовые жалобы.

Система обработки заявок – для управления заявками используется система, которая фиксирует их статус, проводит анализ и назначает задачи сотрудникам. Однако она не позволяет эффективно визуализировать заявки по географическому положению, что ограничивает возможности диспетчерской службы в быстром реагировании на массовые проблемы в определённых районах.

Модель IDEF0 была использована для подробного анализа текущего

процесса формирования заявки на оказание услуги в личном кабинете Квартплата24 [22]. На основе контекстной диаграммы процесс был декомпозирован до уровня отдельных действий (A1–A3), что позволило более детально изучить каждую операцию и выявить проблемные области.Результат декомпозиции представлен на рисунке 3 [16].

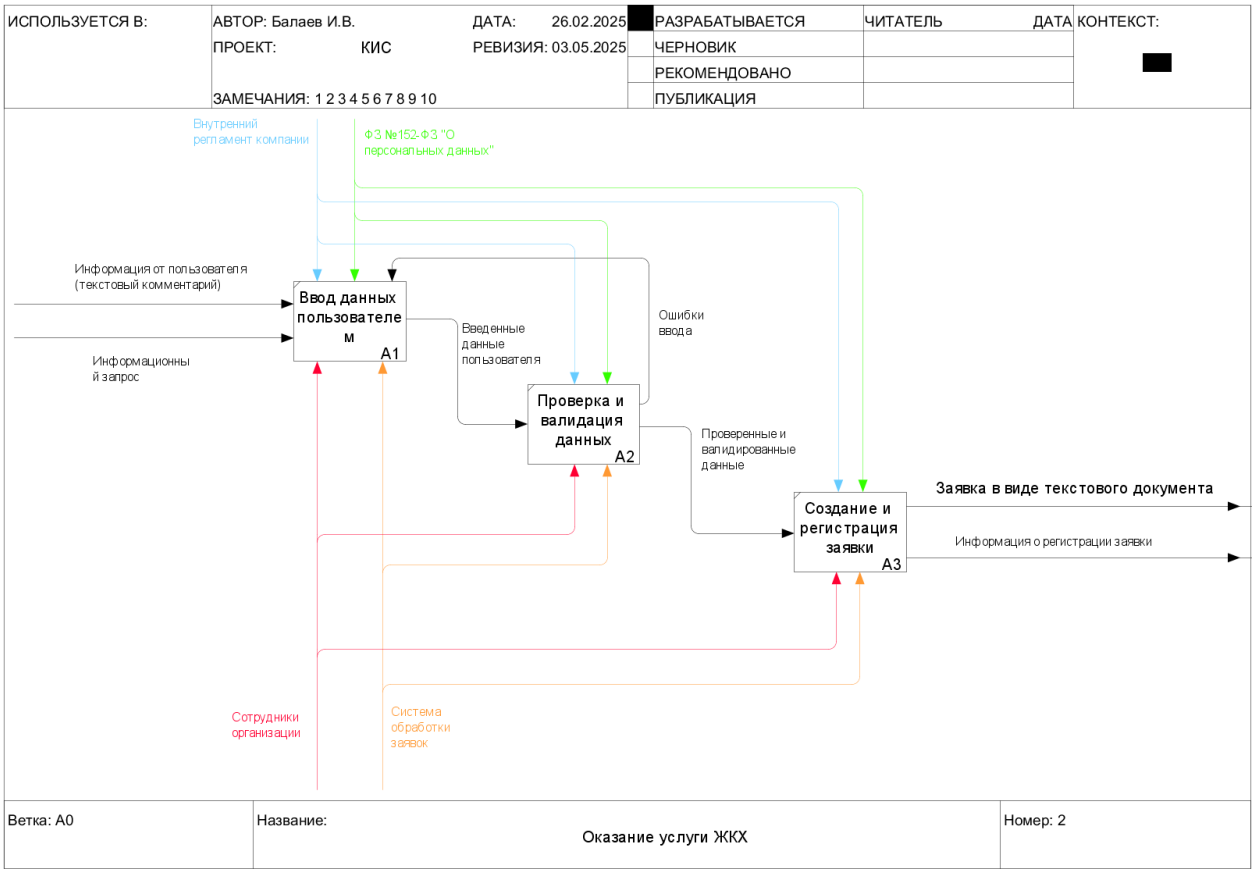


Рисунок 3 – Процесс «Декомпозиция бизнес–процесса по оказанию услуги ЖКХ AS–IS»

Основной процесс в системе аварийно–диспетчерской службы связан с подачей и обработкой заявок от пользователей. Этот процесс состоит из последовательных этапов, начиная с ввода данных пользователем и заканчивая регистрацией заявки в системе. Для анализа текущего состояния важно рассмотреть каждый этап отдельно, выделив входные и выходные данные, а также задействованные механизмы и средства контроля. Это необходимо для понимания структуры процесса и последующей

декомпозиции.

Первым этапом является ввод данных пользователем. На этом этапе в систему поступает текстовая информация – описание проблемы, комментарии, а также возможные уточняющие запросы. Эти данные являются исходным материалом для формирования заявки. На уровне механизмов реализованы регламенты, которые обеспечивают соответствие вводимой информации требованиям безопасности и обработки персональных данных.

Далее следует этап проверки и валидации данных. Система анализирует полученную информацию на предмет полноты и корректности. В случае обнаружения ошибок – например, отсутствия обязательных полей или некорректных значений – пользователю возвращается уведомление о необходимости доработки. При успешной проверке данные передаются на следующий этап.

Следующим этапом является создание и регистрация заявки. Проверенные данные преобразуются в текстовый документ, который сохраняется в системе. Заявке присваивается уникальный идентификатор, фиксируется дата и время регистрации, а также начальный статус. После этого заявка поступает в систему для дальнейшей обработки.

Для обеспечения прозрачности процесса система отправляет пользователю уведомление о регистрации заявки с указанием её идентификатора и статуса. Это позволяет пользователю отслеживать этапы обработки заявки в реальном времени.

На завершающем этапе зарегистрированные заявки передаются сотрудникам для выполнения. Система позволяет отслеживать их статус, обеспечивая базовое взаимодействие между автоматизированной частью и исполнителями. Контроль за обработкой заявок осуществляется как автоматически, так и вручную – со стороны ответственных сотрудников.

Данное описание отражает текущее (AS–IS) состояние модели подачи и обработки заявок, что позволяет определить направления для построения улучшенной модели (TO–BE).

1.3 Обзор и анализ существующих решений

Для обоснования необходимости разработки новой подсистемы визуализации заявок важно рассмотреть существующие решения, представленные на рынке в сфере ЖКХ. Сравнительный анализ поможет выявить их сильные и слабые стороны, а также определить, какие функции уже реализованы, а какие остаются неудовлетворенными или отсутствуют вовсе. Это позволит точнее сформулировать требования к разрабатываемой системе, ориентируясь не только на текущие потребности пользователей, но и на потенциальные возможности улучшения качества обслуживания. Ниже приведён обзор нескольких популярных платформ, используемых для взаимодействия граждан с коммунальными службами.

«ГИС ЖКХ» – представляет собой федеральную систему, предназначенную для обеспечения прозрачности в сфере ЖКХ, включая взаимодействие с гражданами, управление платежами, расчетами и подачей заявок на услуги. Платформа предоставляет функционал для подачи показаний счетчиков, подачи жалоб и заявок, а также контроля за начислениями и платежами. Веб–страница «ГИС ЖКХ» представлена на рисунке 4.

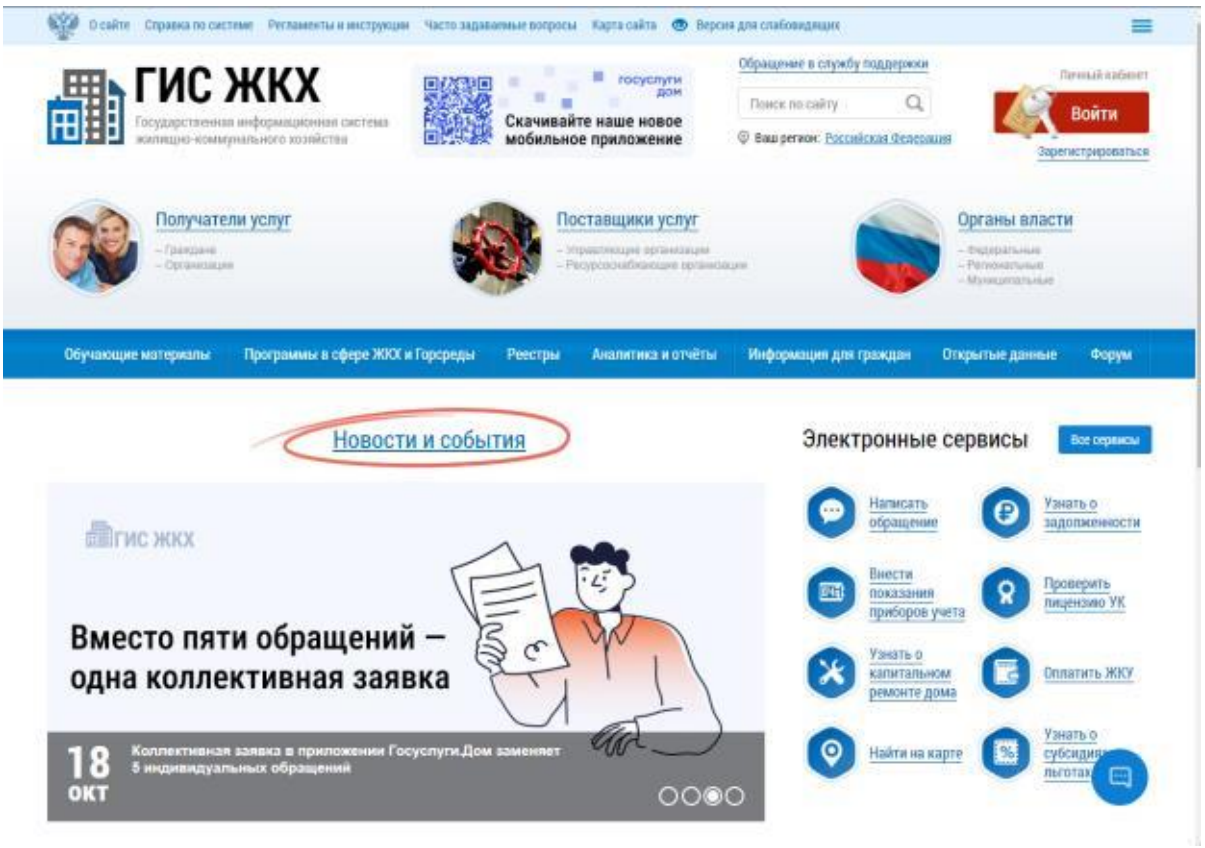


Рисунок 4 – ГИС ЖКХ

Критерии оценки:

- простота управления: 4 – система интуитивно понятна, но для полного комфорта требуется некоторое время на привыкание;
- производительность: 5 – отлично справляется с большим количеством пользователей и запросов, обеспечивая стабильную работу;
- удобство навигации: 3 – интерфейс может быть не всегда удобным, особенно для пожилых пользователей, что снижает общую доступность;
- юзабилити: 3 – функционал системы разнообразен, но не всегда удобен в использовании, что может требовать дополнительного времени на освоение;
- функциональность: 4 – обеспечивает все необходимые функции, но возможности для персонализации и дальнейших улучшений пока ограничены.

Общая оценка: 19

«ЖКХ Контроль» – предоставляет платформу для жителей, чтобы они могли контролировать качество коммунальных услуг и взаимодействовать с управляющими компаниями. Платформа дает возможность подавать жалобы и следить за выполнением работ, а также имеет функции для оценки качества предоставляемых услуг. Веб–страница «ЖКХ Контроль» представлена на рисунке 5.

Рисунок 5 – ЖКХ Контроль

Критерии оценки:

- простота управления: 4 – интерфейс интуитивно понятен и удобен для пользователей, что упрощает взаимодействие с системой;
- производительность: 4 – система работает стабильно, однако иногда могут возникать небольшие задержки в обработке запросов;
- удобство навигации: 4 – навигация логична и понятна, все основные разделы легко найти и использовать;

– юзабилити: 4 – пользователи быстро осваиваются с системой, но периодическое обновление интерфейса могло бы улучшить общий опыт;
– функциональность: 3 – система ограничена в возможностях для полноценного управления заявками, что требует доработок.

Общая оценка: 19

«Диспетчер 24» – платформа ориентирована на управление работой управляющих компаний, ТСЖ и других организаций, связанных с ЖКХ. Включает функционал для автоматизации приема заявок от жильцов, создания отчетности, управления контактами с исполнителями, а также расчета тарифов и выставления счетов. Платформа предлагает контактный центр для обратной связи с жильцами и позволяет оптимизировать процессы взаимодействия. Веб–страница «Диспетчер 24» представлена на рисунке 6.

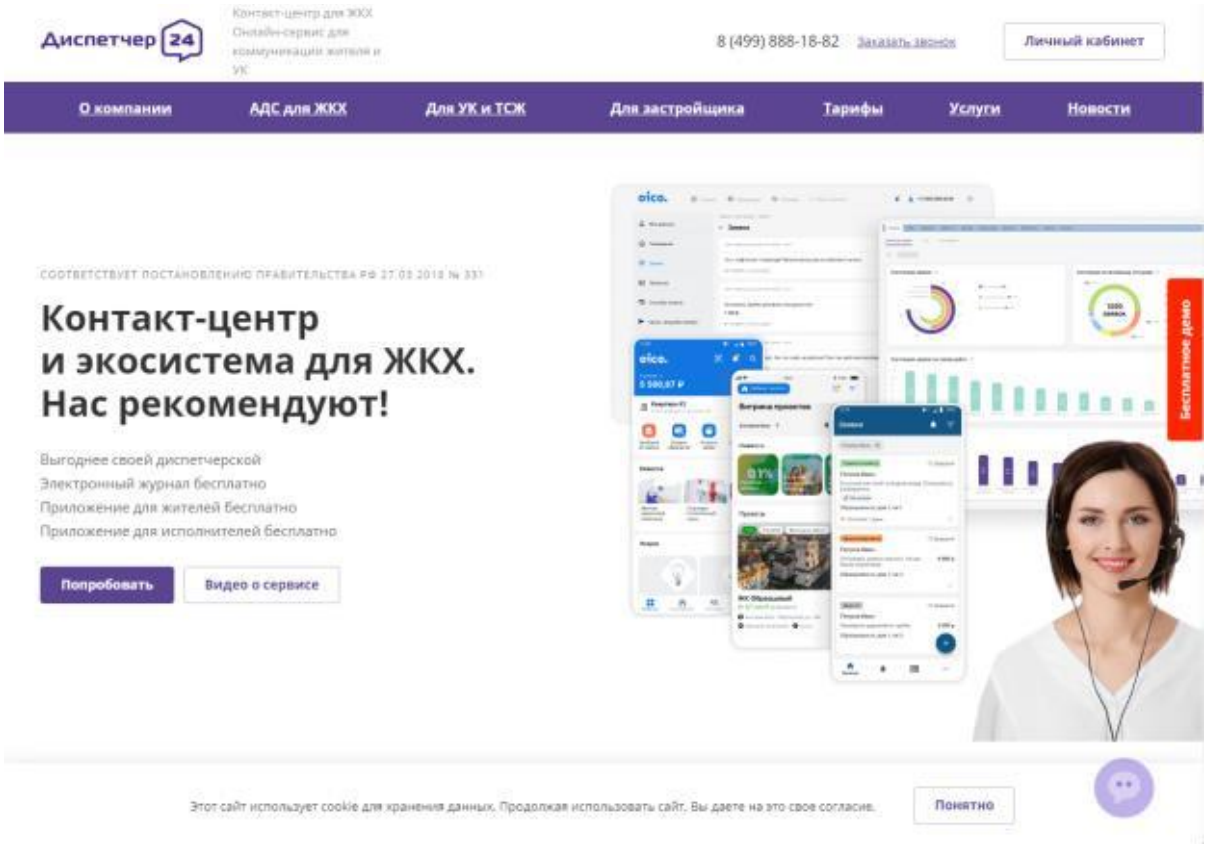


Рисунок 6 – Диспетчер 24

Критерии оценки:

- простота управления: 4 – интерфейс интуитивно понятен и удобен для пользователей разных уровней подготовки;
- производительность: 5 – система отлично справляется с высокой нагрузкой и быстро обрабатывает запросы, не давая сбоев;
- удобство навигации: 3 – навигация доступна, но освоить ее быстро может быть сложно без предварительного обучения;
- юзабилити: 4 – платформа позволяет эффективно работать, однако интерфейс мог бы быть более удобным и интуитивно понятным;
- функциональность: 4 – платформе не хватает некоторых дополнительных функций для более эффективной обработки заявок, но базовые возможности представлены хорошо.

Общая оценка: 20

Результаты проведённого анализа позволяют наглядно сравнить ключевые характеристики рассматриваемых систем [9]. Для удобства восприятия полученные данные сведены в таблицу, где оценены основные критерии, влияющие на эффективность и удобство использования платформ. Результаты сравнительного анализа представлены в таблице 1.

Таблица 1 – Анализ систем, представленных на рынке

Критерий	ГИС ЖКХ	ЖКХ Контроль	Диспетчер 24
Простота	4	4	4
Производительность	5	4	5
Удобство навигации	3	4	3
Юзабилити	3	4	4
Функциональность	4	3	4
Итого	19	19	20

На основе проведенного анализа существующих решений можно сделать вывод, что хотя все платформы обладают определенными преимуществами, ни одно из них не полностью удовлетворяет потребности в улучшении функционала аварийно–диспетчерской службы. Это подчеркивает

необходимость разработки новой системы, которая обеспечит дополнительные функции, такие как создание шаблонов для быстрого формирования заявок, визуализация заявок на карте города и интеграция с системой для отображения однотипных заявок.

Таким образом, реализация предложенного проекта для Квартплата 24 позволит значительно улучшить сервис и ускорить процесс подачи и обработки заявок.

1.4 Разработка требований

После анализа существующих решений и выявления их сильных и слабых сторон, необходимо определить основные требования к разрабатываемой системе. Для этого используется модель FURPS+, которая позволяет структурировать требования по функциональным и нефункциональным характеристикам [17], обеспечивая более полное представление о будущей системе. Основные требования к подсистеме визуализации заявок представлены в таблице 2.

Таблица 2 – Требования к информационной системе для подсистемы визуализации заявок в методологии FURPS+

Вид требований	Содержание требований
Функциональные требования к информационной системе для подсистемы визуализации заявок	– шаблоны заявок – возможность выбора готовых шаблонов для быстрого формирования заявки; – визуализация заявок – отображение заявок на карте с указанием статуса и частоты возникновения проблем;
Удобство использования информационной системы для подсистемы визуализации заявок	– интуитивный интерфейс – упрощённый процесс подачи заявки с минимальным количеством действий; – визуальные элементы – цветовая маркировка статусов заявок на карте для наглядности.

Продолжение таблицы 2

Вид требований	Содержание требований
Удобство использования информационной системы для подсистемы визуализации заявок	– интуитивный интерфейс – упрощённый процесс подачи заявки с минимальным количеством действий; – визуальные элементы – цветовая маркировка статусов заявок на карте для наглядности;
Надежность информационной системы для подсистемы визуализации заявок	– стабильность работы – обеспечение доступности системы 24/7; – безопасность данных – шифрование данных пользователей и защита от несанкционированного доступа;
Производительность информационной системы для подсистемы визуализации заявок	– высокая скорость работы – загрузка данных и карты не более 2 секунд; – масштабируемость – поддержка большого количества пользователей без ухудшения производительности;
Поддерживаемость информационной системы для подсистемы визуализации заявок	– совместимость – работа на всех популярных браузерах и мобильных устройствах; – обновляемость – возможность развёртывания обновлений без прерывания работы пользователей;
(+) – Для подсистемы визуализации заявок	– пользовательский интерфейс – карта города с интерактивными метками и шаблоны заявок.

На основе анализа существующих решений, модели AS–IS и особенностей предметной области были выявлены ключевые проблемы текущего процесса оказания услуг ЖКХ [3]. Ручная обработка заявок, отсутствие автоматизации и недостаточная прозрачность приводят к задержкам, ошибкам и снижению удовлетворенности пользователей [15]. В текущей системе предусмотрен лишь текстовый формат обращения, что затрудняет точную формулировку проблемы и замедляет её обработку. Эти недостатки подтверждают необходимость модернизации системы за счёт внедрения современных решений [11]: шаблонов заявок, визуализации обращений на карте и отображения количества однотипных заявок.

Формирование требований к новой модели TO–BE осуществляется с

применением подхода FURPS+, который охватывает как функциональные, так и нефункциональные характеристики системы – такие как надёжность, удобство использования, производительность, поддержка и расширяемость. Целью разработки TO–BE–модели является повышение качества сервиса аварийно–диспетчерской службы за счёт внедрения ключевых улучшений: списка типовых заявок для быстрого формирования обращений, автоматического выявления массовых проблем (с присвоением обращению статуса «общедоступной» при превышении порога в 5 заявок по одному адресу) и визуализации заявок на карте города. Эти изменения позволят ускорить процесс обработки, повысить прозрачность и улучшить взаимодействие между жителями и службами ЖКХ.

1.5 Модель TO–BE

В новой модели TO–BE внесены значительные улучшения, направленные на модернизацию процесса создания заявки. Эти изменения отражены на схеме синими стрелками, показывающими усовершенствованные связи между этапами. Контекстная диаграмма бизнес–процесса по созданию заявки на оказание услуги представлена на рисунке 7.

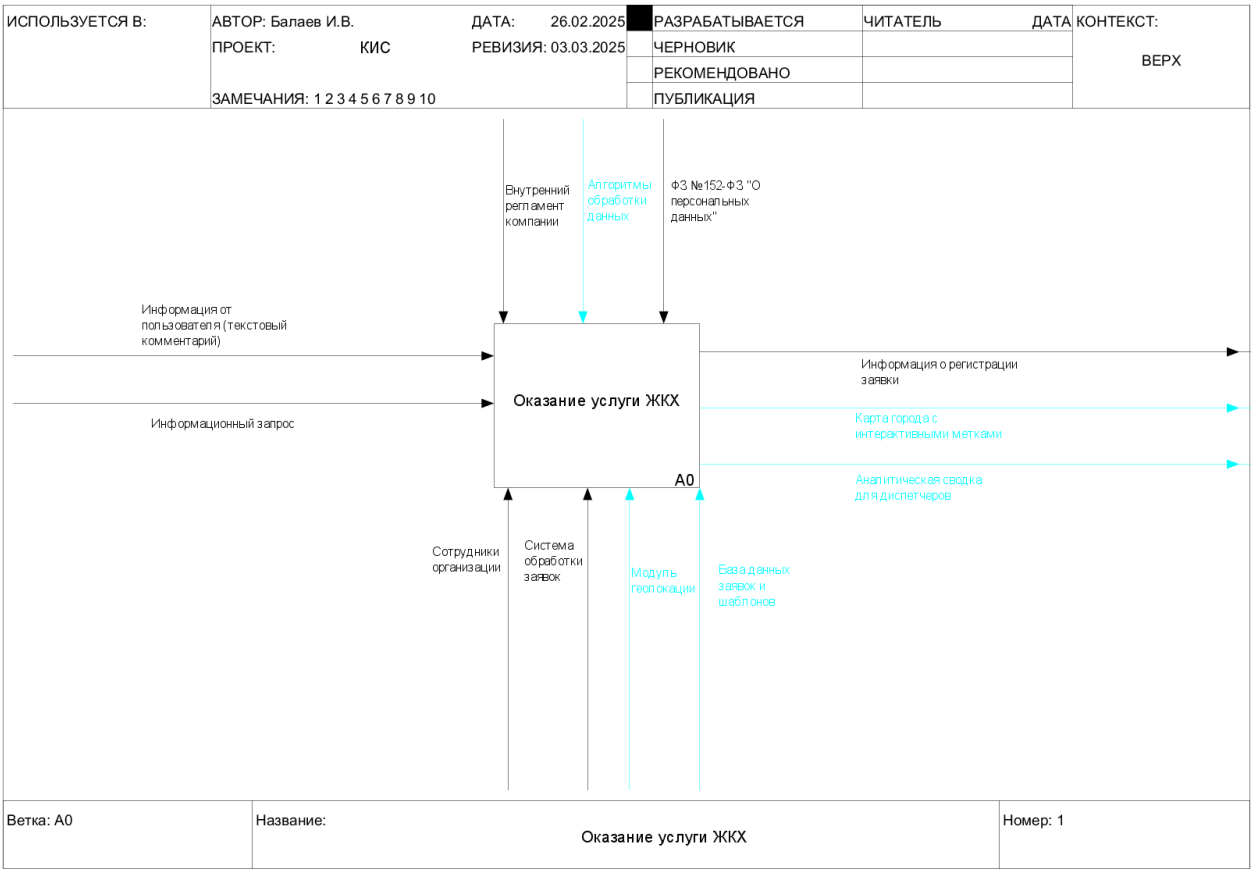


Рисунок 7 – Контекстная диаграмма бизнес–процесса по созданию заявки на оказание услуги TO–BE

Модель «TO–BE» включает в себя несколько значительных улучшений, которые обеспечат более эффективную работу как для пользователей, так и для сотрудников диспетчерской службы.

Модуль геолокации – одним из ключевых улучшений является добавление модуля геолокации. Этот элемент позволяет пользователям и системе более точно определять местоположение проблемы, что значительно ускоряет обработку заявок. Теперь информация о заявке будет автоматически ассоциироваться с географическими данными, что позволит диспетчерам быстрее реагировать на проблемы, происходящие в разных частях города. Модуль геолокации будет интегрирован с картой города, которая будет отображать заявки в реальном времени.

Карта города с интерактивными метками – новый функционал, включающий карту города с интерактивными метками, позволит

визуализировать заявки на карте в реальном времени. Это позволит диспетчерам и операторам аварийно–диспетчерской службы легко отслеживать количество заявок по конкретным районам, а также оперативно реагировать на накопившиеся проблемы. Каждая заявка будет отображаться с соответствующей меткой на карте, что позволит быстро увидеть, где возникает наибольшее количество инцидентов.

Аналитическая сводка для диспетчеров – для улучшения процесса принятия решений, в модели «ТО–ВЕ» появляется аналитическая сводка для диспетчеров. Эта функция предоставляет сводную информацию о всех поступивших заявках, включая их статус, частоту возникновения проблем и другие важные метрики. Это даст диспетчерам возможность более эффективно распределять ресурсы, а также оперативно реагировать на часто возникающие проблемы в определенных районах или по конкретным типам заявок.

База данных заявок и шаблонов – важным улучшением является создание базы данных заявок и шаблонов, которая будет хранить информацию о предыдущих заявках и часто используемых шаблонах. Это позволит пользователям быстрее создавать заявки, выбирая из уже готовых шаблонов для типовых ситуаций, таких как поломка лифта или неисправность отопления. Такая база данных повысит скорость подачи заявок и снизит вероятность ошибок в описаниях проблем.

Интеграция с системой обработки заявок – в модели «ТО–ВЕ» также предусмотрено улучшение интеграции с системой обработки заявок. Эта система будет более эффективно обрабатывать поступающие запросы, автоматически назначать статусы заявкам, а также направлять их к соответствующим специалистам для дальнейшего реагирования.

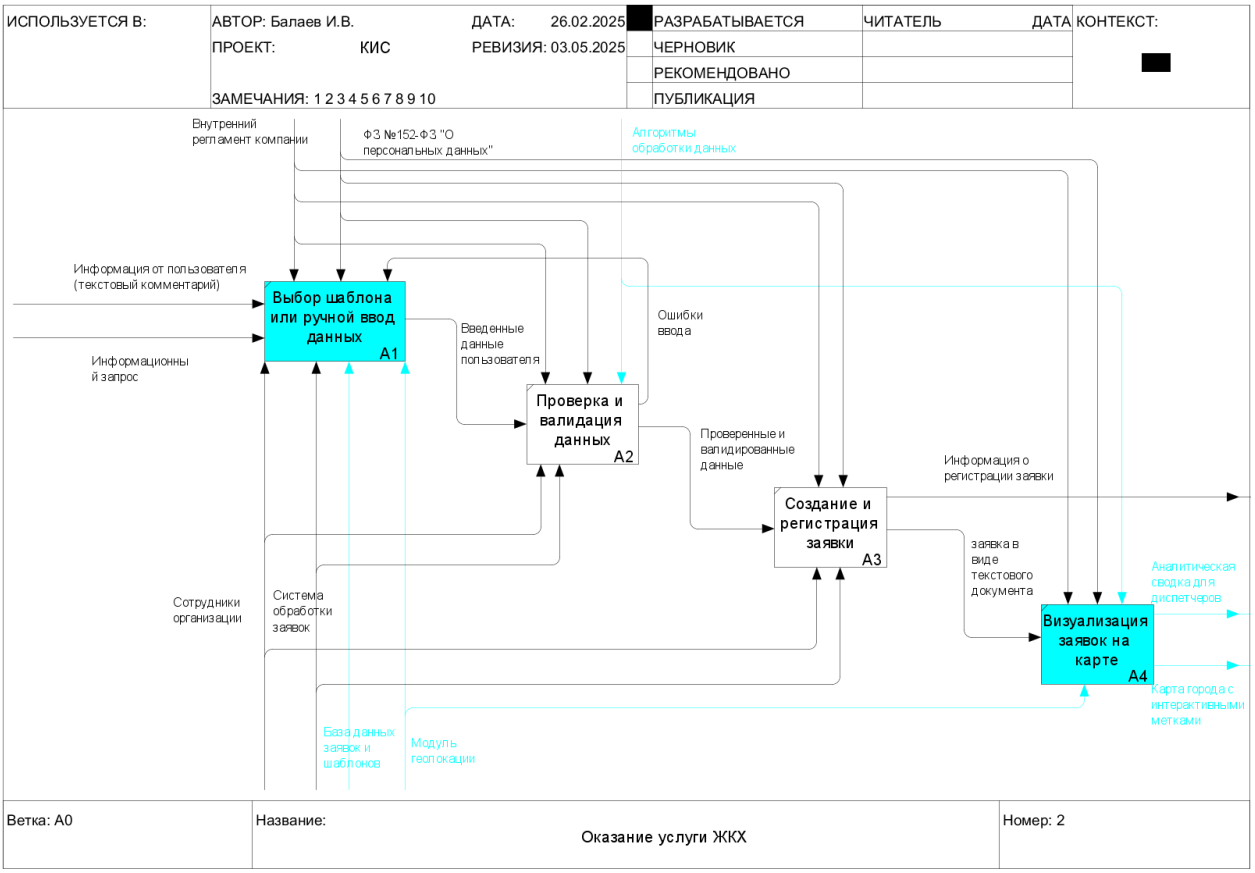


Рисунок 8 – Декомпозиция бизнес–процесса по созданию заявки на оказание услуги ТО– ВЕ

Для более детального понимания, как предложенные улучшения будут интегрированы в систему [4], необходимо разобрать каждый компонент модели «ТО–ВЕ» на более низком уровне. Декомпозиция позволит детализировать ключевые этапы и процессы, включая новые функции, такие как геолокация, интерактивная карта и аналитическая сводка. Рассмотрим, как эти элементы будут взаимодействовать между собой и с текущими компонентами системы, чтобы обеспечить более эффективное функционирование аварийно–диспетчерской службы.

Выбор шаблона или ручной ввод данных – для упрощения подачи заявок пользователям предоставлены два варианта: использование готовых шаблонов или ручной ввод. Шаблоны охватывают типовые ситуации (например, «поломка лифта», «утечка воды») и автоматически заполняют ключевые поля, сокращая время оформления. При ручном вводе система предлагает подсказки

на основе предыдущих заявок, что снижает риск ошибок. Данные передаются в модуль геолокации для автоматической привязки к координатам.

Модуль геолокации – этот блок устраняет неточности в определении адреса. Пользователи могут выбрать местоположение через GPS или указать его на интерактивной карте. Координаты автоматически связываются с заявкой, что ускоряет обработку и позволяет диспетчерам мгновенно идентифицировать район возникновения проблемы.

Визуализация заявок на карте – все заявки отображаются на карте города в режиме реального времени с цветовыми метками:

- красный – заявки высокой срочности, требующие немедленного реагирования;
- жёлтый – заявки средней важности, выполнение которых допускает краткую отсрочку;
- зелёный – заявки низкой срочности, не требующие оперативного вмешательства.

Диспетчеры могут фильтровать заявки по типу проблемы, району или дате, что упрощает мониторинг и распределение ресурсов.

Аналитическая сводка – система автоматически формирует отчеты, включающие данные о статусе заявок, частоте проблем в районах и загруженности бригад. Это позволяет оптимизировать маршруты сотрудников и оперативно реагировать на повторяющиеся инциденты.

База данных заявок и шаблонов – архив заявок и шаблонов типовых ситуаций ускоряет подачу запросов. Пользователи выбирают подходящий шаблон, что сокращает время заполнения и минимизирует ошибки. Данные из базы используются для анализа трендов и прогнозирования нагрузок.

Интеграция с системой обработки – улучшенная интеграция обеспечивает автоматическое назначение статусов заявкам, распределение задач между бригадами на основе геолокации и уведомление пользователей о ходе работ.

В рамках анализа бизнес–процессов формирования заявок аварийно–

диспетчерской службы компании «Квартплата 24» были построены функциональные модели AS–IS и TO–BE, которые позволили выявить ключевые проблемы текущей системы и предложить пути их решения. На основе этих моделей был сделан вывод о необходимости внедрения новых функциональных возможностей, таких как автоматизация обработки данных, визуализация заявок на карте и интеграция с внешними сервисами. Для реализации этих улучшений требуется переход к проектированию подсистемы визуализации заявок с использованием современных инструментов моделирования.

После детального анализа текущего состояния (AS–IS) и проектирования будущей модели (TO–BE) был сделан вывод о необходимости перехода к более детальному проектированию системы. Для этого были выбраны диаграммы UML (Unified Modeling Language), которые позволяют наглядно описать архитектуру, взаимодействие компонентов и поведение системы [18]. UML–диаграммы помогают структурировать процесс разработки, определить ключевые элементы системы и их взаимосвязи, а также обеспечить четкое понимание требований к функциональности подсистемы.

Вывод по 1 главе:

В данной главе проведено детальное исследование предметной области системы аварийно–диспетчерской службы компании «Квартплата 24», которая является ключевым объектом разработки. Основное внимание было уделено анализу текущих бизнес-процессов, их особенностей и ограничений, что позволило сформировать четкое представление о текущем состоянии системы и выявить направления для её совершенствования.

Проведён тщательный анализ текущих бизнес–процессов (модель AS–IS), в результате которого были выявлены основные недостатки, препятствующие эффективной работе службы. К ним относятся отсутствие автоматизированных инструментов для обработки заявок, что увеличивает

время их обработки, недостаточная прозрачность процессов, затрудняющая контроль за статусами заявок, а также сложность визуализации данных, что ограничивает оперативность реагирования на обращения пользователей.

Для поиска решений был выполнен обзор платформ ГИС ЖКХ, ЖКХ Контроль и Диспетчер 24, показавший, что ни одна из них не полностью удовлетворяет потребности компании в эффективной визуализации и автоматизации обработки заявок, подчеркивая актуальность новой подсистемы.

На основе полученных данных с использованием модели FURPS+ были сформулированы функциональные и нефункциональные требования к разрабатываемой подсистеме визуализации заявок. Функциональные требования включают создание шаблонов для упрощения подачи заявок, визуализацию на интерактивной карте с цветовой индикацией статуса и автоматическое определение массовых обращений. Нефункциональные требования охватывают высокую производительность (загрузка данных до 2 секунд) и удобство интерфейса для пользователей с разным уровнем подготовки.

Разработана целевая модель бизнес–процессов (TO–BE), включающая модуль геолокации для точной привязки заявок, интерактивную карту для наглядного отображения и аналитическую сводку для диспетчеров. Эти нововведения направлены на повышение оперативности и прозрачности работы аварийно–диспетчерской службы, что способствует улучшению взаимодействия с пользователями.

Анализ предметной области и четко сформулированные требования создали прочную основу для дальнейшего проектирования подсистемы. Это обеспечивает её соответствие реальным потребностям пользователей и стратегическим целям компании, а также закладывает фундамент для успешной реализации проекта.

Глава 2 Проектирование подсистемы визуализации заявок

2.1 Формализация пользователей

Формализация пользователей – это процесс определения ролей и задач, выполняемых различными группами пользователей в системе. В контексте разработки подсистемы визуализации заявок для компании «Квартплата 24» были выделены три ключевые категории пользователей: жители, диспетчеры и администраторы. Каждая из этих групп обладает своими уникальными функциями и требованиями, которые необходимо учитывать при проектировании.

Жители, являясь основными пользователями, взаимодействуют с системой через личный кабинет: они подают заявки, выбирая тип обращения из шаблонов и указывая уровень срочности, могут отслеживать статус заявок и получать уведомления об изменениях, просматривать свои обращения на карте города и в истории личного кабинета. Для этой категории особенно важны простота интерфейса, быстрая обратная связь и удобство визуализации – особенно для пожилых пользователей.

Диспетчеры отвечают за обработку заявок, их распределение между исполнителями и контроль выполнения. Они анализируют обращения, назначают исполнителей с учётом их специализации и загруженности, обновляют статусы заявок и используют карту города для более эффективного планирования. Также диспетчеры анализируют статистику по обращениям для выявления повторяющихся проблем. Им необходимы удобные инструменты для фильтрации, визуализации и интеграции с другими сервисами.

Администраторы управляют системой в целом: настраивают параметры, ведут учет пользователей, обеспечивают стабильную работу и интеграцию с внешними платформами, такими как ГИС ЖКХ, и проводят аналитику на основе накопленных данных. Они играют ключевую роль в обеспечении надежности, безопасности и масштабируемости системы.

Все три группы работают в рамках единой системы, но с разными уровнями доступа и функционалом [5]. В типичном сценарии житель создает заявку, которая автоматически поступает к диспетчеру, тот назначает исполнителя, обновляет статус по мере выполнения, а житель получает уведомления и может проследить весь процесс на карте. Администратор в свою очередь отслеживает общую эффективность и вносит необходимые настройки.

Формализация ролей позволила чётко распределить задачи между пользователями и обеспечить более логичное, удобное и эффективное взаимодействие с системой [2].

2.2 Календарный план проекта

Для того чтобы была разработана эффективная и функциональная подсистема визуализации заявок для системы аварийно–диспетчерской службы, был составлен календарный план проекта, включающий основные этапы, сроки их реализации, ключевые задачи и ожидаемые результаты. Этот план обеспечивает поэтапную организацию работ, контроль за их выполнением и достижение поставленных целей в установленные сроки. Календарный план проекта представлен в таблице 3.

Таблица 3 – Календарный план проекта

Этап проекта	Период(недели)	Основные задачи	Результаты этапа
Подготовительный этап	1 неделя	– анализ предметной области и бизнес–процессов; – изучение требований к подсистеме; – согласование индивидуального графика (плана).	– понимание текущих процессов (AS–IS); – сформулированные требования к подсистеме; – утвержденный план работы.

Продолжение таблицы 3

Этап проекта	Период(недели)	Основные задачи	Результаты этапа
Проектирование подсистемы	2–4 недели	– построение моделей AS–IS и TO–BE; – разработка UML–диаграмм (диаграмма развертывания, вариантов использования, последовательности, классов); – формализация пользователей (жители, диспетчеры, администраторы);	– модели текущих и целевых процессов; – UML–диаграммы, описывающие архитектуру и взаимодействие компонентов системы; – определение ролей и задач пользователей;
Разработка прототипа	5–8 недели	– выбор инструментов и технологий (HTML, CSS, JavaScript, Leaflet.js, OpenStreetMap); – разработка структуры веб–приложения (вкладки, интерфейс); – реализация функционала создания заявок (шаблоны, срочность, отправка);	– выбранные технологии и инструменты для реализации; – прототип интерфейса с основными вкладками; – рабочая форма создания и отправки заявок;
Визуализация заявок на карте	9–10 недели	– интеграция карты с использованием Leaflet.js и OpenStreetMap API; – реализация кластеризации маркеров с использованием Leaflet.markercluster; – внедрение системы подсчета заявок по адресу;	– карта с отображением заявок; – кластеризация маркеров для улучшения производительности; – функция подсчета заявок и обновления статуса при превышении порога;
Тестирование и доработка	11–12 недели	– тестирование функционала подсистемы (юзабилити, производительность, безопасность); – устранение выявленных ошибок и доработка функционала.	– отчет о тестировании с выявленными ошибками и предложениями по улучшению; – исправленный и оптимизированный прототип подсистемы.

Продолжение таблицы 3

Этап проекта	Период(недели)	Основные задачи	Результаты этапа
Расчет экономического эффекта	13 неделя	– расчет затрат на разработку (оплата труда, амортизация, электроэнергия); – оценка экономического эффекта и срока окупаемости;	– отчет о затратах на разработку; – расчет экономического эффекта и срока окупаемости;
Подготовка отчета	14–16 недели	– написание отчета по практике; – подготовка презентации и защита проекта.	– готовый отчет по преддипломной практике; – презентация проекта и успешная защита.

Разработанный календарный план обеспечивает структурированный подход к реализации проекта, позволяя эффективно распределить ресурсы и контролировать выполнение задач на каждом этапе [10].

2.3 Логическая модель работы подсистемы

В ходе проектирования информационной системы для реализации этой подсистемы основное внимание уделено улучшению визуализации заявок и упрощению взаимодействия пользователей с системой. Важными задачами являются создание списка часто используемых обращений для быстрого оформления заявок, отображение количества однотипных заявок по одному адресу, а также возможность отображения заявок на карте города. Все эти элементы существенно ускорят обработку заявок и повысят прозрачность работы службы.

Для проектирования системы будет использована технология моделирования UML (Unified Modeling Language), которая является

стандартом индустрии для описания архитектуры и поведения системы [19] через диаграммы. UML позволяет наглядно представить как взаимодействуют компоненты системы, какие шаги и процессы необходимы для реализации требуемого функционала.

В рамках использования UML будут созданы различные диаграммы, такие как диаграмма развертывания, которая отображает архитектуру выполнения системы, включая взаимодействие аппаратных и программных компонентов. Диаграмма развертывания для разрабатываемой подсистемы визуализации заявок в системе аварийно–диспетчерской службы представлена на рисунке 9.

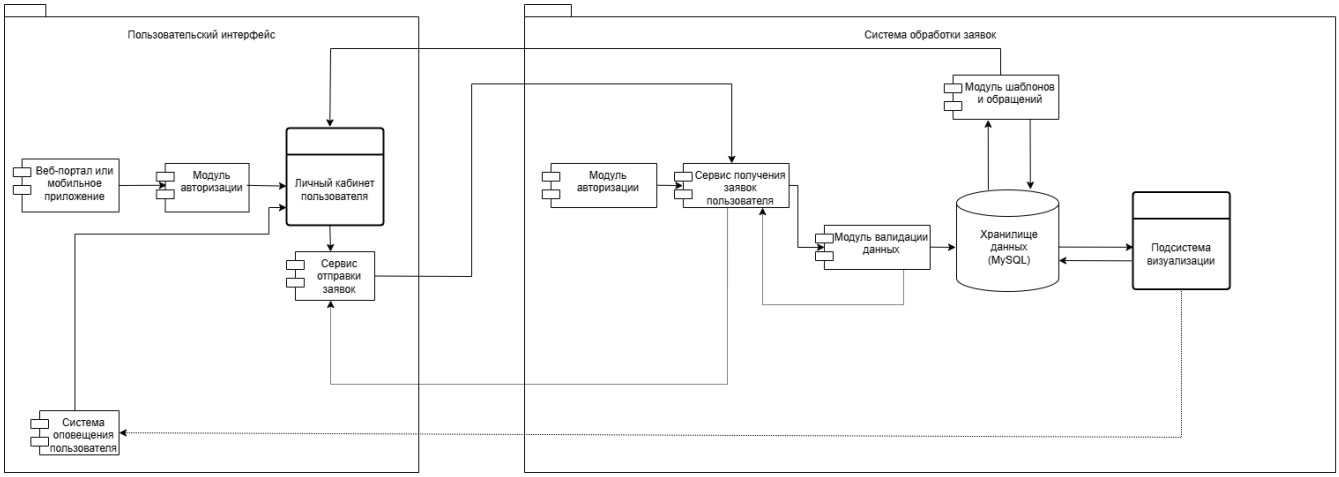


Рисунок 9 – UML диаграмма развертывания подсистемы визуализации заявок

На данной диаграмме развертывания представлены ключевые компоненты системы и их взаимодействие с различными подсистемами. Система состоит из двух основных частей: пользовательского интерфейса и системы обработки заявок.

Пользовательский интерфейс включает в себя веб–портал или мобильное приложение, где пользователи взаимодействуют с системой, а также модуль авторизации для входа в систему. После успешной авторизации пользователь попадает в личный кабинет. В этом кабинете доступен сервис

отправки заявок, который взаимодействует с системой и служит основным интерфейсом для подачи запросов. Система уведомлений информирует пользователя о статусе заявки и событиях, связанных с обработкой.

Система обработки заявок включает в себя сервис получения заявок пользователя, который взаимодействует с модулем шаблонов и обращений, а также с модулем валидации данных для проверки корректности введённых данных. Вся информация сохраняется в хранилище данных (MySQL), что является центральным элементом для хранения и обработки заявок. Для визуализации заявок используется подсистема визуализации, которая получает данные из хранилища и отображает их в соответствующем интерфейсе.

Все эти компоненты и сервисы связаны между собой через физическое или виртуальное оборудование, где осуществляется обмен данными через сеть. Хранилище данных MySQL играет центральную роль в обеспечении доступа к данным, обеспечивая их сохранность и доступность для всех компонентов системы.

Диаграмма развертывания помогает четко представить, как различные компоненты системы будут развернуты и взаимодействовать на различных уровнях. Это дает понимание о том, как будут обеспечены взаимодействия между пользователями, системами и сервисами, а также как данные будут храниться и передаваться в рамках всей системы. Такой подход позволяет на стадии проектирования определить оптимальную структуру и конфигурацию системы для обеспечения её эффективной работы.

Диаграмма вариантов использования (Use Case Diagram) представлена для подсистемы визуализации заявок АДС и описывает взаимодействие различных пользователей (актеров) с системой, а также их действия в процессе работы с заявками. Диаграмма отражена в соответствующей схеме на рисунке 10

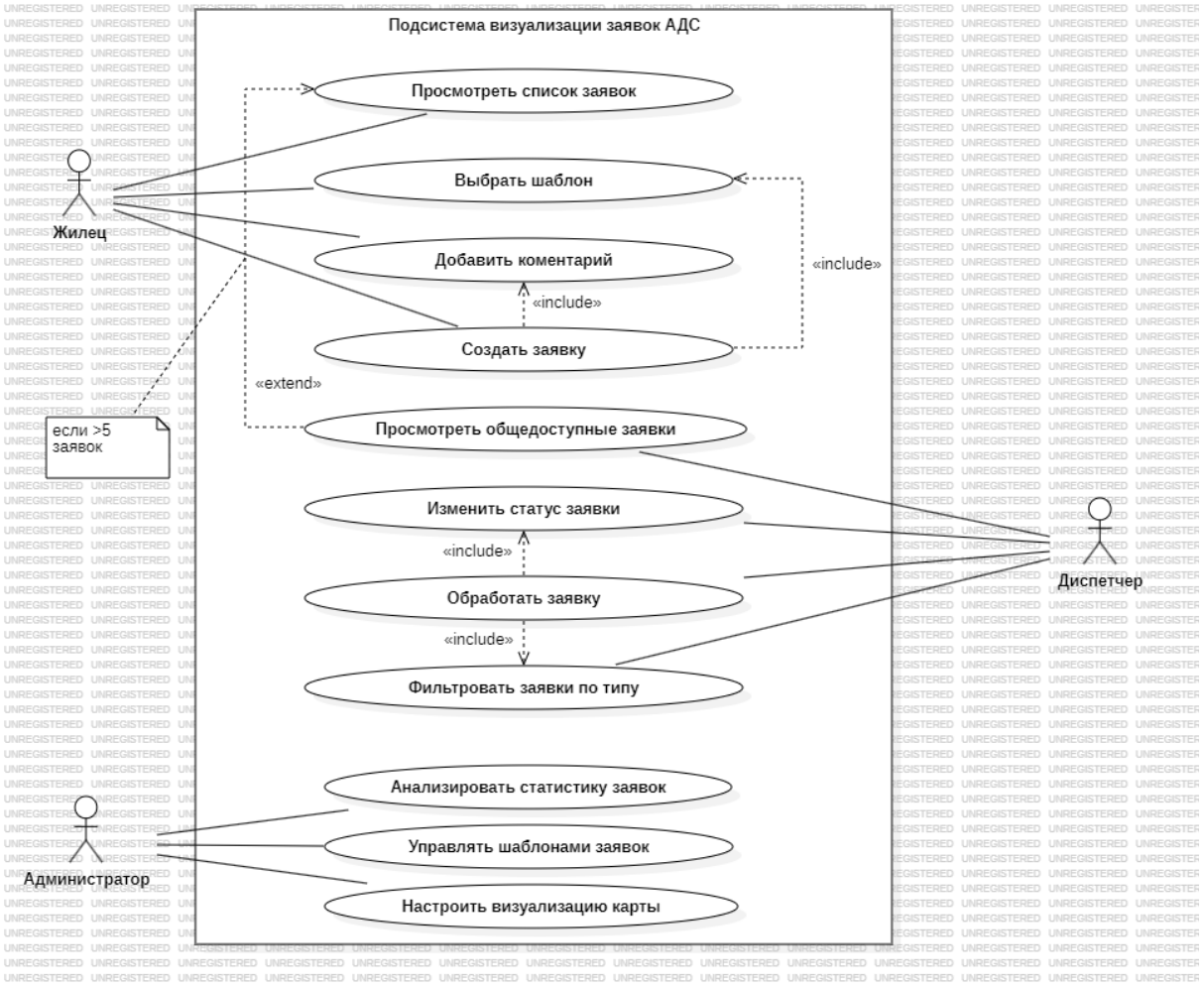


Рисунок 10 – UML диаграмма вариантов использования разрабатываемой подсистемы визуализации заявок в системе аварийно–диспетчерской службы.

Жилец является основным пользователем системы и имеет несколько вариантов взаимодействия:

- просмотреть список заявок – этот вариант использования позволяет жильцу просматривать все заявки в системе;
- выбрать шаблон – жилец может выбрать шаблон для создания новой заявки;
- добавить комментарий – возможность добавить комментарий к существующим заявкам;
- создать заявку – жилец может создать новую заявку, выбрав шаблон и добавив необходимые данные.

В случае, если количество заявок превышает 5, жилец может

фильтровать заявки по типу. Это условие представлено в виде extend, что указывает на дополнительную функциональность, активируемую при превышении определенного количества заявок.

Диспетчер имеет более широкий доступ к системе и может выполнять следующие действия:

- просмотреть общедоступные заявки – диспетчер может просматривать заявки, доступные для всех пользователей системы;
- изменить статус заявки – диспетчер может изменять статус заявки в процессе ее обработки;
- обработать заявку – диспетчер имеет возможность обработать заявку, которая поступила от жильца;
- фильтровать заявки по типу – диспетчер может фильтровать заявки по различным критериям для упрощения их обработки.

Администратор имеет доступ к управлению системой и может выполнять более высокоуровневые действия:

- анализировать статистику заявок – администратор может анализировать статистику по заявкам, чтобы оценить эффективность работы системы;
- управлять шаблонами заявок – администратор может добавлять, изменять или удалять шаблоны для заявок;
- настроить визуализацию карты – администратор может настроить отображение заявок на карте для более удобного визуального представления.

Отношение include обозначает обязательные действия, которые всегда выполняются в рамках основного процесса. Эти шаги являются неотъемлемой частью выполнения варианта использования.

- выбрать шаблон и Добавить комментарий в процессе Создать заявку – эти действия всегда включаются в процесс создания заявки, что означает, что для того, чтобы завершить создание заявки, пользователь должен выбрать шаблон или добавить комментарий;
- изменить статус заявки и обработать заявку – эти действия всегда

выполняются диспетчером в процессе работы с заявками, поскольку они являются необходимыми для изменения состояния заявки и её дальнейшей обработки.

Отношение extend используется для обозначения действий, которые выполняются только при определённых условиях. Они не являются обязательными и выполняются по мере необходимости. В этом случае:

Если заявок больше 5, то заявка получает статус общедоступной. Это условие отражает, что, если количество заявок превышает пять, она становится доступной для всех пользователей, и они могут её увидеть. В результате этого действия пользователи могут просматривать список заявок. Это дополнительная функциональность, которая активируется, когда количество заявок достигает определённого порога.

Таким образом, логика работы с заявками зависит не только от действий конкретного пользователя, но и от общей ситуации в системе – например, количества аналогичных обращений. Эта динамическая реакция системы на внешние условия позволяет гибко адаптировать интерфейс и доступные функции. Для более детального понимания порядка взаимодействия пользователей с системой и её внутренней логики используется диаграмма последовательности, которая описывает пошаговое выполнение операций.

Для пользователя диаграмма последовательности отражает ключевые шаги, такие как выбор вкладки для создания заявки, просмотр истории, работа с картой и получение новостей. Взаимодействие с базой данных позволяет обновлять информацию о заявках и адресах. Система обновляет состояние в зависимости от действий пользователя, что находит отражение в интерфейсе.

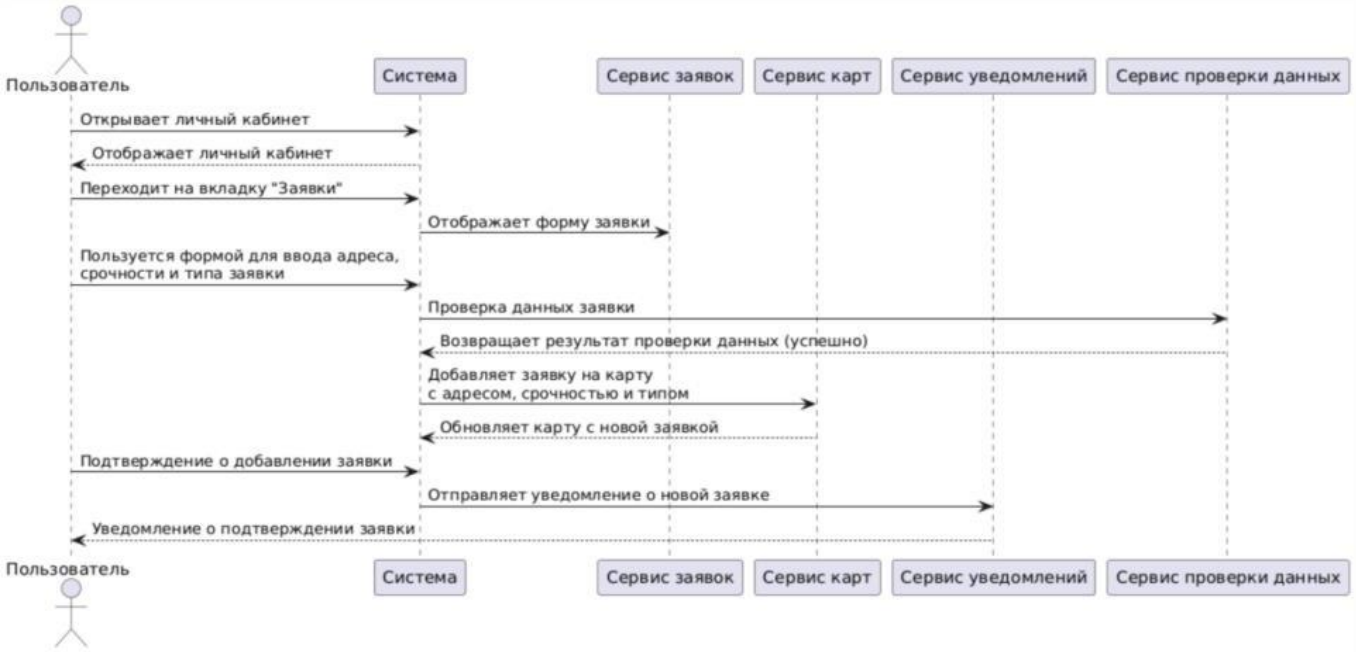


Рисунок 11 – UML диаграмма последовательности для серверной части приложения

Диаграмма последовательности, представленная на рисунке 11, наглядно демонстрирует, как программные компоненты работают вместе для выполнения задач пользователя.

2.4 Проектирование базы данных информационной системы.

Логическая модель базы данных представляет собой абстрактное описание структуры данных и их взаимосвязей [6], не включая технические детали реализации. Она помогает четко определить, какие сущности существуют в системе, как они взаимодействуют друг с другом и какие данные должны быть сохранены в каждой из сущностей. Важно отметить, что логическая модель формирует основу для разработки физической модели базы данных, где уже будет прописана конкретная структура таблиц, индексов и других технических аспектов. В данном случае, модель описывает систему, связанную с обработкой заявок, их статусами, пользователями, диспетчерами и локациями. Логическая модель базы данных представлена на рисунке 12.

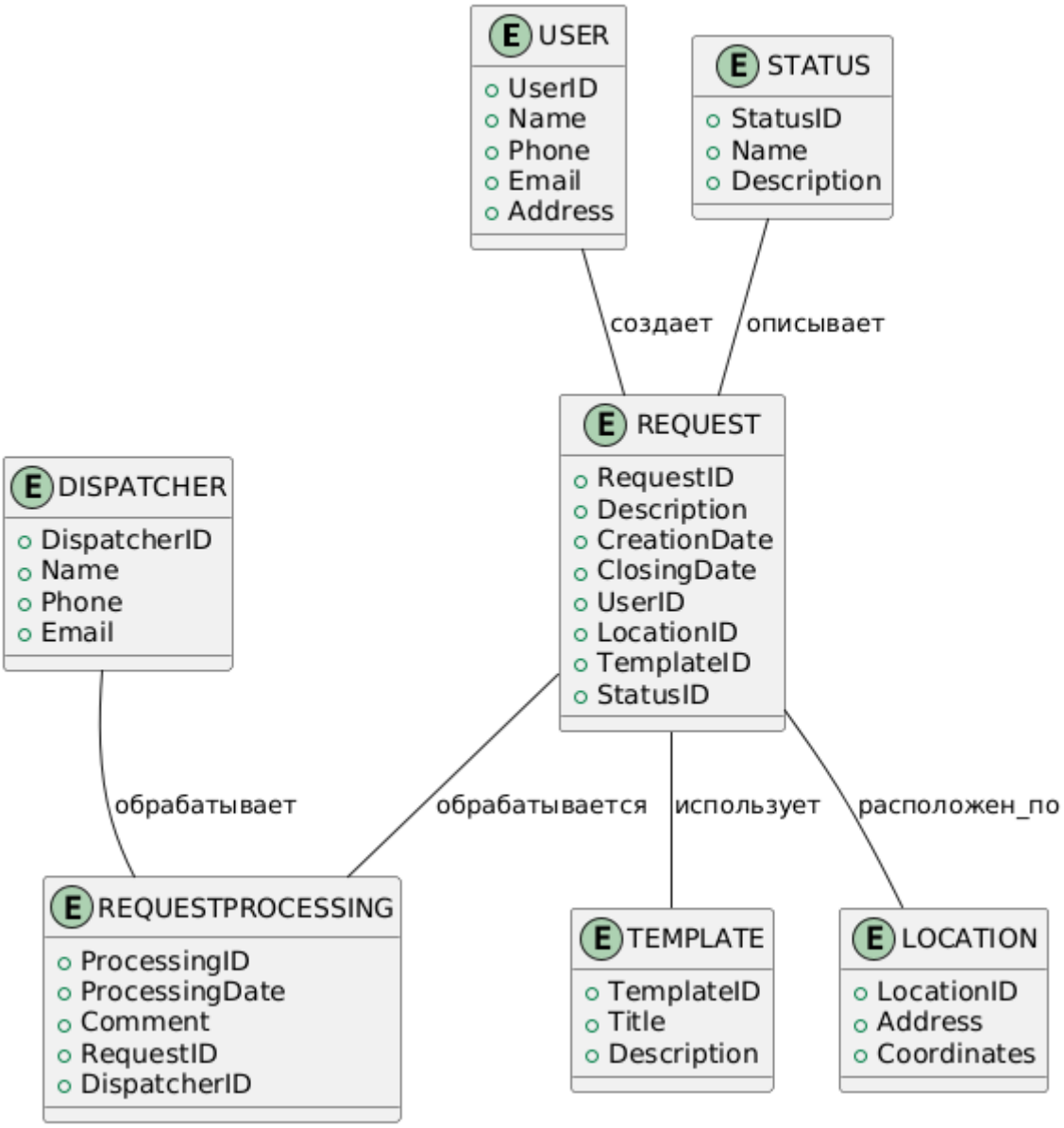


Рисунок 12 – Логическая модель базы данных информационной системы.

В модели присутствуют несколько ключевых сущностей, каждая из которых играет важную роль в процессе работы системы.

USER (Пользователь) – сущность представляет пользователей системы, которые могут создавать заявки. Атрибуты включают уникальный идентификатор пользователя (UserID), имя, телефон, email и адрес. Один пользователь может создать несколько заявок, что отражено связью «создает» с сущностью REQUEST.

STATUS (Статус) – сущность описывает различные статусы заявок (например, «в обработке», «завершено» и т.д.). Статус связан с заявкой через внешний ключ (StatusID), который указывает на текущий статус заявки. Один статус может быть связан с несколькими заявками, что выражается через связь «описывает» между сущностями STATUS и REQUEST.

REQUEST (Заявка) – это основная сущность модели, представляющая заявку, созданную пользователем. Она включает в себя идентификатор заявки (RequestID), описание, дату создания и закрытия, а также ссылки на пользователя, локацию, шаблон и статус заявки. Заявка может быть обработана диспетчером, что отражено через связь с сущностью REQUESTPROCESSING.

DISPATCHER (Диспетчер) – эта сущность представляет сотрудников, которые обрабатывают заявки. Атрибуты включают уникальный идентификатор диспетчера (DispatcherID), имя, телефон и email. Диспетчер обрабатывает заявку, и эта связь отображается через сущность REQUESTPROCESSING, которая соединяет диспетчера с заявками, которые он обрабатывает.

REQUESTPROCESSING (Обработка заявки) – это промежуточная сущность, которая связывает заявки с диспетчерами, указывая, какие заявки были обработаны и когда. Она также содержит комментарии, связанные с обработкой заявки. Эта сущность связывает заявку с диспетчером, выполняющим обработку.

TEMPLATE (Шаблон) – сущность шаблона используется для создания заявок, позволяя стандартизировать ввод данных. Каждый шаблон имеет уникальный идентификатор (TemplateID), название и описание. Заявка может быть основана на одном из шаблонов, что фиксируется через связь с сущностью REQUEST.

LOCATION (Локация) – сущность локации описывает географическое расположение, с которым связана заявка. Атрибуты включают идентификатор локации (LocationID), адрес и координаты. Каждая заявка может быть

привязана к одной локации, что отображается через связь «расположен по» с сущностью REQUEST.

Логическая модель служит основой для создания физической модели базы данных, где будут детализированы таблицы, ключи, индексы и связи на уровне базы данных [12]. В физической модели будут учтены конкретные технологии хранения данных, нормализация и оптимизация запросов, что позволит эффективно хранить и обрабатывать данные в реальной системе. Физическая модель базы данных представлена на рисунке 13.

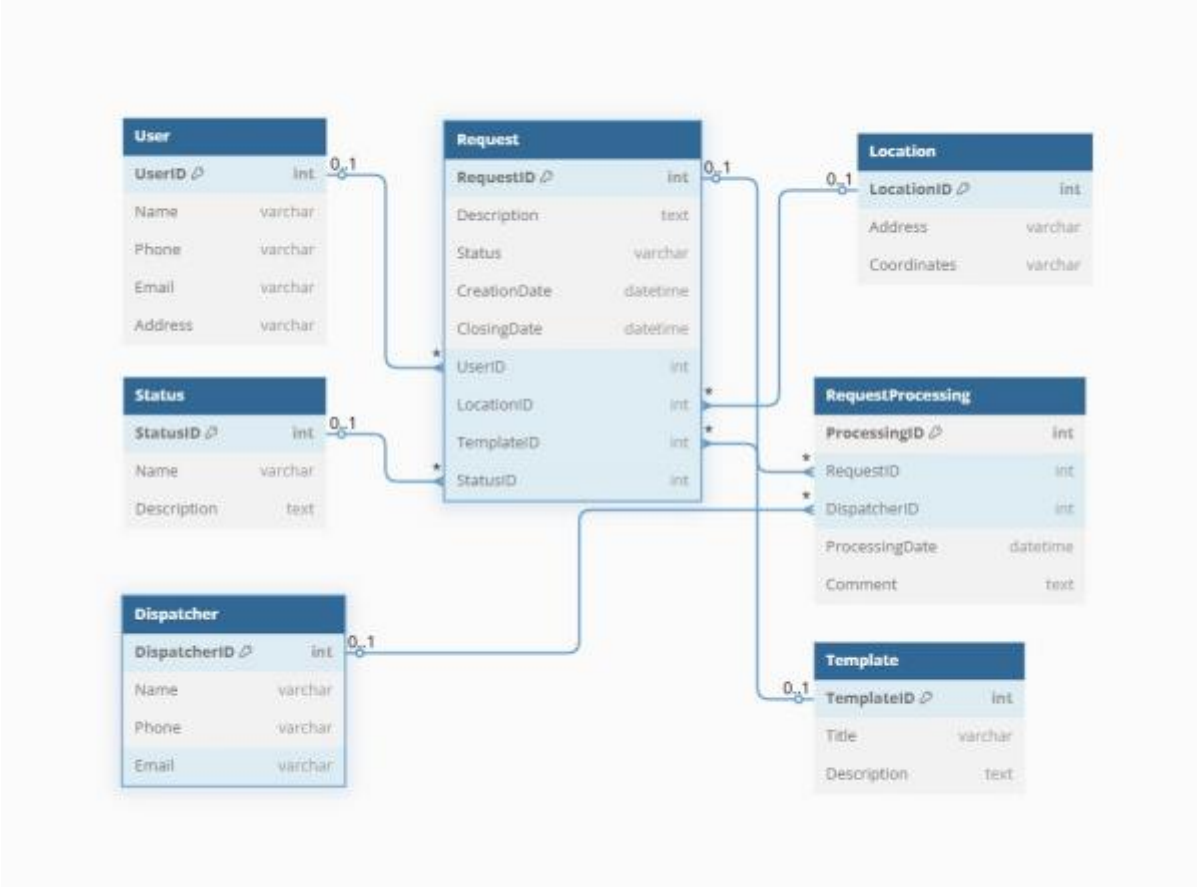


Рисунок 13 – Физическая модель базы данных информационной системы.

Физическая модель базы данных отражает детализированную структуру, включающую таблицы, поля, связи и ключи, которые будут использоваться для хранения данных в базе данных. В данном случае, схема описывает несколько таблиц, которые связаны между собой и соответствуют

сущностям из логической модели.

Таблица User – содержит информацию о пользователях системы, включая их уникальный идентификатор (UserID), имя, телефон, email и адрес. Каждому пользователю может быть присвоено несколько заявок, что отражено связью 1 ко многим с таблицей Request.

Таблица Request – хранит информацию о заявках, таких как уникальный идентификатор заявки (RequestID), описание, даты создания и закрытия, а также ссылки на связанные с ней сущности: пользователя (UserID), локацию (LocationID), статус (StatusID) и шаблон (TemplateID). Эта таблица является центральной в системе, так как связывает все остальные таблицы.

Таблица Status – содержит информацию о статусах заявок (например, «новая», «в обработке», «завершено»), включая уникальный идентификатор (StatusID), название и описание. Каждый статус может быть связан с несколькими заявками через внешний ключ в таблице Request.

Таблица Location – описание локаций, связанных с заявками, включая уникальный идентификатор (LocationID), адрес и координаты. Каждая заявка может быть привязана к определенной локации, что отражается через внешний ключ в таблице Request.

Таблица RequestProcessing – описание обработки заявок, включая уникальный идентификатор обработки (ProcessingID), ссылки на заявку (RequestID), диспетчера (DispatcherID), дату обработки и комментарий. Таблица RequestProcessing связывает заявки с диспетчерами, указывая, какой диспетчер обрабатывает какую заявку и в какой момент времени.

Таблица Dispatcher – хранит информацию о диспетчерах, которые обрабатывают заявки, включая их уникальный идентификатор (DispatcherID), имя, телефон и email.

Таблица Template – хранит информацию о шаблонах заявок, включая уникальный идентификатор шаблона (TemplateID), название и описание. Каждый шаблон может быть использован для создания множества заявок.

Логическая модель описывает структуру данных на абстрактном уровне,

определяя, какие сущности существуют в системе и какие связи между ними существуют. Она позволяет понимать, какие данные должны храниться в базе и как эти данные связаны между собой.

Физическая модель отражает детали реализации, такие как конкретные таблицы, атрибуты и связи, которые будут использоваться для хранения данных. Эта модель определяет, как данные будут реально сохраняться в базе данных, какие типы данных используются для каждого атрибута и как создаются связи между таблицами с помощью внешних ключей.

Вывод по 2 главе:

В данной главе рассмотрены ключевые аспекты проектирования подсистемы визуализации заявок для системы аварийно–диспетчерской службы. Была проведена формализация пользователей (жители, диспетчеры, администраторы), что позволило чётко определить их роли и функциональные потребности.С использованием нотации UML были разработаны диаграммы (развертывания, вариантов использования, последовательности). Эти диаграммы наглядно описали архитектуру системы, взаимодействие компонентов и сценарии использования.Также была спроектирована логическая и физическая модель базы данных, включающая сущности, такие как пользователи, заявки, статусы, шаблоны и локации. Это обеспечило структурированное хранение и обработку данных.Разработан календарный план проекта, охватывающий этапы анализа, проектирования, разработки и тестирования, что позволило организовать процесс создания подсистемы.

Проектирование с использованием UML и структурированный подход к разработке обеспечили чёткое понимание архитектуры системы и её функциональности, создав надёжную основу для реализации подсистемы.

Глава 3 Реализация подсистемы визуализации заявок

3.1 Выбор инструментов реализации

Для реализации проекта подсистемы визуализации заявок для системы аварийно–диспетчерской службы компании «Квартплата 24» были выбраны следующие инструменты и технологии:

HTML был выбран в качестве основного языка для создания структуры веб–приложения. Он позволяет организовать контент на странице, создавая такие элементы, как формы для ввода данных, кнопки, текстовые поля и другие компоненты интерфейса. HTML обеспечивает базовую структуру веб–страницы, которая затем стилизуется и оживляется с помощью CSS и JavaScript.

CSS использовался для стилизации веб– приложения, включая оформление интерфейса, цветовую схему, шрифты, отступы и другие визуальные элементы. CSS позволяет создавать адаптивный дизайн, который корректно отображается на различных устройствах, включая мобильные телефоны, планшеты и десктопы. Это особенно важно для обеспечения удобства пользователей, которые могут взаимодействовать с системой через разные устройства.

JavaScript был выбран в качестве основного языка программирования для реализации логики веб–приложения [19]. С его помощью были реализованы такие функции, как обработка данных, отправка заявок, взаимодействие с картой и динамическое обновление интерфейса. JavaScript также позволяет работать с API, что было необходимо для интеграции с внешними сервисами, такими как OpenStreetMap.

Для визуализации заявок на карте была использована библиотека Leaflet.js [24], которая предоставляет удобные инструменты для работы с картами и маркерами. Leaflet.js позволяет отображать карты, добавлять маркеры, группировать их в кластеры и взаимодействовать с пользователем

через интерактивные элементы. Это значительно упрощает процесс визуализации заявок на карте и улучшает производительность приложения.

Для работы с картами и геоданными был использован API OpenStreetMap [25], [26]. Этот сервис предоставляет доступ к картографическим данным, что позволяет отображать заявки на карте города, а также осуществлять поиск адресов и привязку заявок к конкретным координатам. OpenStreetMap является открытым и бесплатным решением, что делает его удобным для использования в проекте.

Для улучшения производительности и удобства отображения большого количества заявок на карте была использована библиотека Leaflet.markercluster. Она позволяет группировать близко расположенные маркеры в кластеры, что упрощает визуализацию и улучшает восприятие карты, особенно при большом количестве заявок.

Для обеспечения функциональности, связанной с геолокацией и визуализацией заявок, были использованы внешние API, такие как OpenStreetMap. Это позволило интегрировать картографические данные в приложение и обеспечить удобное взаимодействие пользователей с системой.

Также для разработки и тестирования приложения были использованы такие инструменты, как Visual Studio и инструменты для отладки и тестирования кода, такие как Chrome DevTools.

Выбор данных инструментов был обусловлен следующими факторами:

– простота и доступность: HTML, CSS и JavaScript являются стандартными технологиями для разработки веб–приложений, что делает их доступными для широкого круга разработчиков и упрощает процесс разработки;
– гибкость и расширяемость: JavaScript и его библиотеки (например, Leaflet.js) позволяют легко расширять функциональность приложения и интегрировать его с другими сервисами;
– производительность – использование библиотеки Leaflet.markercluster позволяет эффективно отображать большое количество заявок на карте, что важно для обеспечения высокой производительности приложения;
– бесплатность и открытость: OpenStreetMap и Leaflet.js являются открытыми и бесплатными решениями, что снижает затраты на разработку и позволяет использовать их без ограничений.

Таким образом, выбранные инструменты и технологии обеспечивают высокую эффективность разработки, удобство использования и возможность дальнейшего расширения функциональности подсистемы визуализации заявок.

3.2 Программная реализация подсистемы

Для начала была разработана структура веб–приложения. Основная страница состоит из нескольких вкладок, каждая из которых отображает разные разделы личного кабинета пользователя, такие как «Главная», «Заявки», «История заявок» и другие. Вкладки помогают организовать информацию, предоставляемую пользователю, и делают интерфейс более удобным. Реализация структуры веб–приложения представлена на рисунке 14.

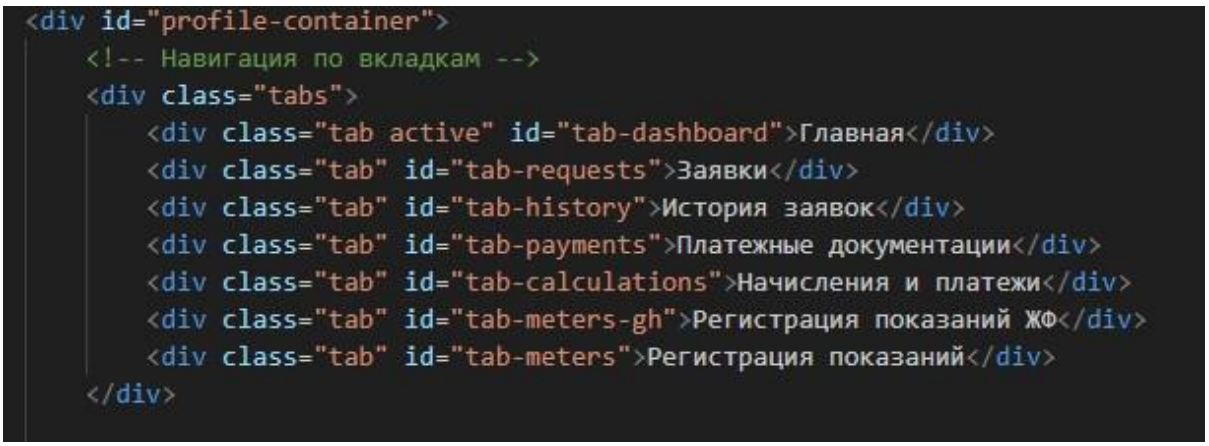


Рисунок 14 – Структура основного интерфейса веб–приложения с вкладками.

Каждая вкладка отображает соответствующий контент. Это реализовано с помощью простого переключателя классов, который скрывает и показывает контент на основе выбора вкладки. Данная реализация обеспечивает быструю и плавную навигацию между разделами интерфейса.

Одним из ключевых улучшений является возможность создания и отправки заявки через форму. В этой форме пользователь может выбрать тип обращения из списка шаблонов, ввести описание проблемы и указать срочность заявки. Для этого был создан блок с шаблонами и полями ввода. Блок с шаблона представлен на рисунке 15.

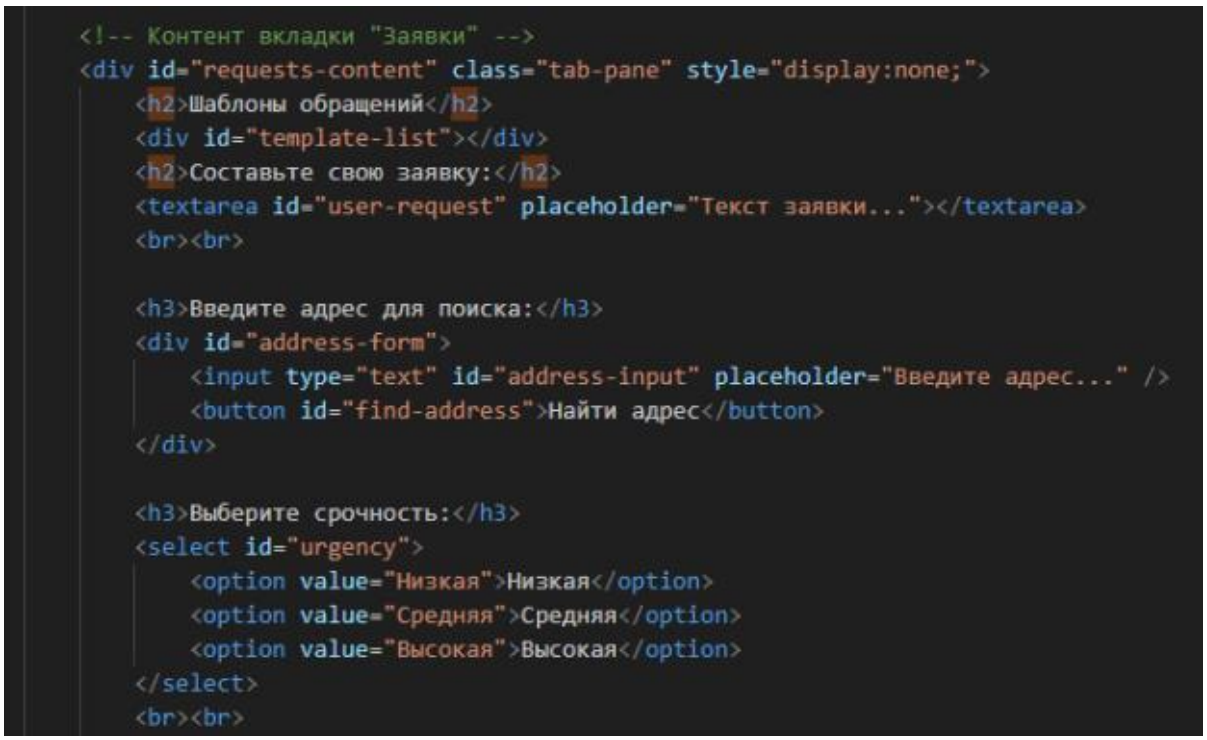


Рисунок 15 – Форма создания и отправки заявки с выбором шаблона и указанием срочности.

В этом разделе пользователи могут выбрать шаблон, который соответствует их проблеме. После выбора шаблона текст заявки автоматически заполняется. Также доступен выбор срочности проблемы (низкая, средняя, высокая), что помогает более эффективно распределять ресурсы. Выбор шаблона и срочности заявки представлен на рисунке 16 и 17.

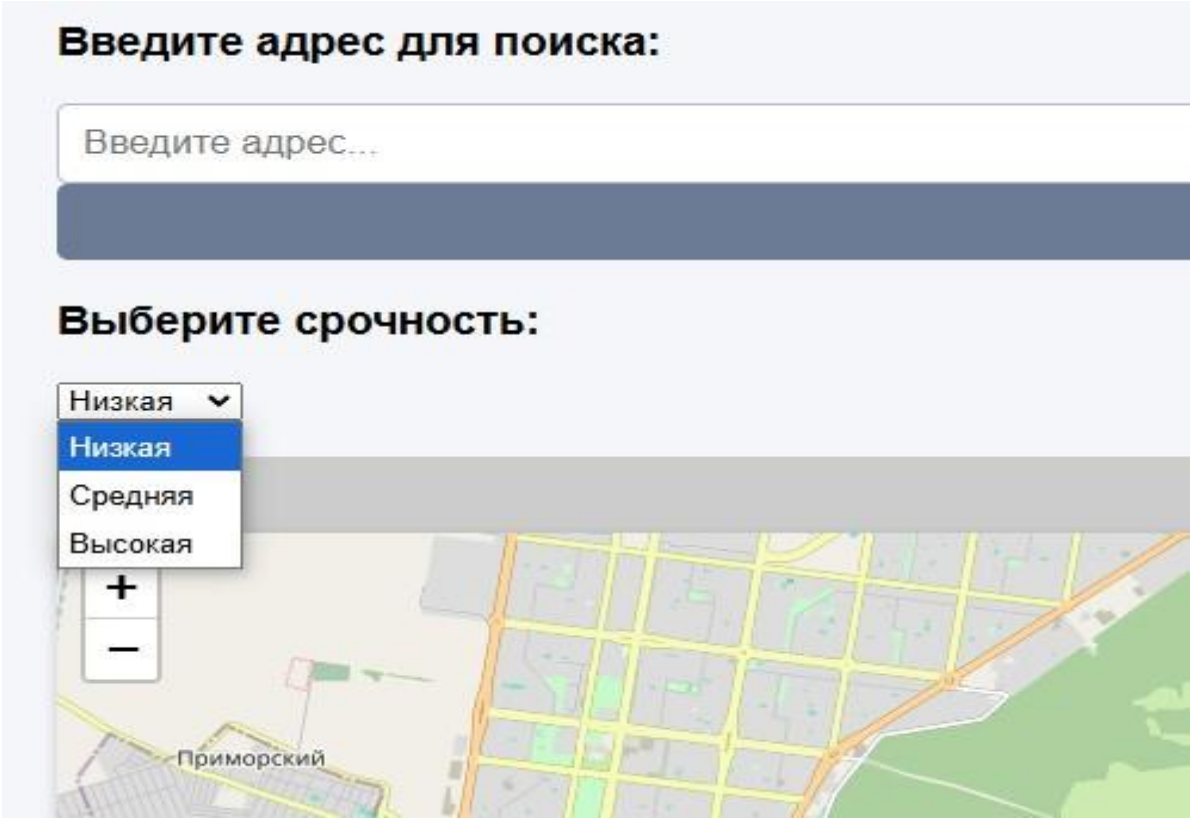


Рисунок 16 – Выбор срочности заявки пользователем

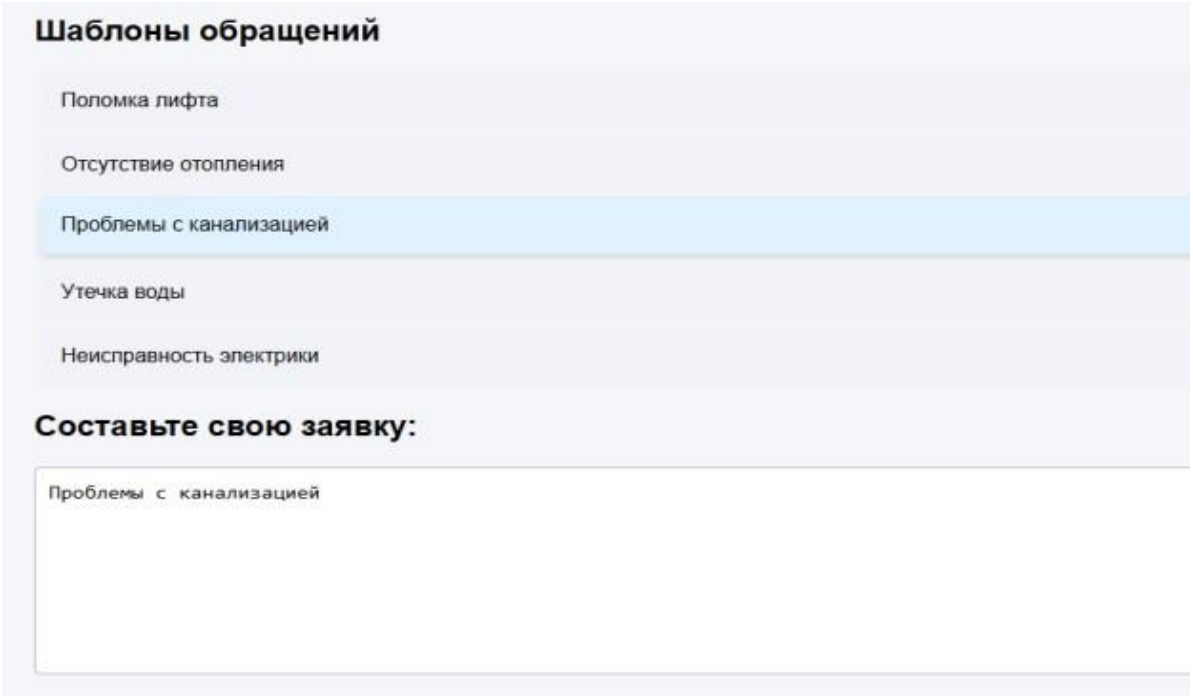


Рисунок 17 – Выбор шаблона заявки пользователем

Так же важной функцией является визуализация заявок на карте. Для

этого использовалась библиотека Leaflet, которая позволяет отображать карты и маркеры на веб–страницах. Для удобства отображения меток использовалась библиотека Leaflet.markercluster, которая объединяет близко расположенные метки в кластеры, что значительно улучшает производительность карты. Подключение библиотеки Leaflet к проекту представлено на рисунке 18.

```
// Инициализация карты
function initMap() {
    map = L.map('map-test').setView([53.498899, 49.398329], 12);

    L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
        attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
    }).addTo(map);

    updateRequests();
    displayTemplates();
}
```

Рисунок 18 – Подключение библиотеки Leaflet

После инициализации карты, заявкам назначаются маркеры, и эти маркеры группируются в кластеры, если они расположены близко друг к другу. Это значительно улучшает восприятие карты, особенно в случае большого количества заявок. Группировка маркеров заявок в кластеры представлена на рисунке 19.

Рисунок 19 – Группировка маркеров заявок в кластеры.

Для улучшения работы диспетчерской службы была внедрена система подсчета заявок по адресу. Если количество однотипных заявок на одном адресе превышает 5, такая заявка становится общедоступной и видимой для всех жильцов дома. Это позволяет оперативно выявлять и решать наиболее острые проблемы. Система подсчета заявок по адресу представлена на рисунке 20.

```
Object.keys(requestCounts).forEach(address => {
    const requestsAtAddress = requests.filter(req => req.address === address);
    const coords = requestsAtAddress[0]?.coords; // Берём координаты из первой заявки
    const count = requestCounts[address];
    let icon;
```

Рисунок 20 – Система подсчета заявок по адресу.

В данном фрагменте кода реализован подсчет количества заявок, поступивших по одному адресу, и обновление статуса заявки, если количество

заявок превышает пороговое значение (5 заявок). Подсчет заявок и обновление статуса при превышении порога представлено на рисунке 21.

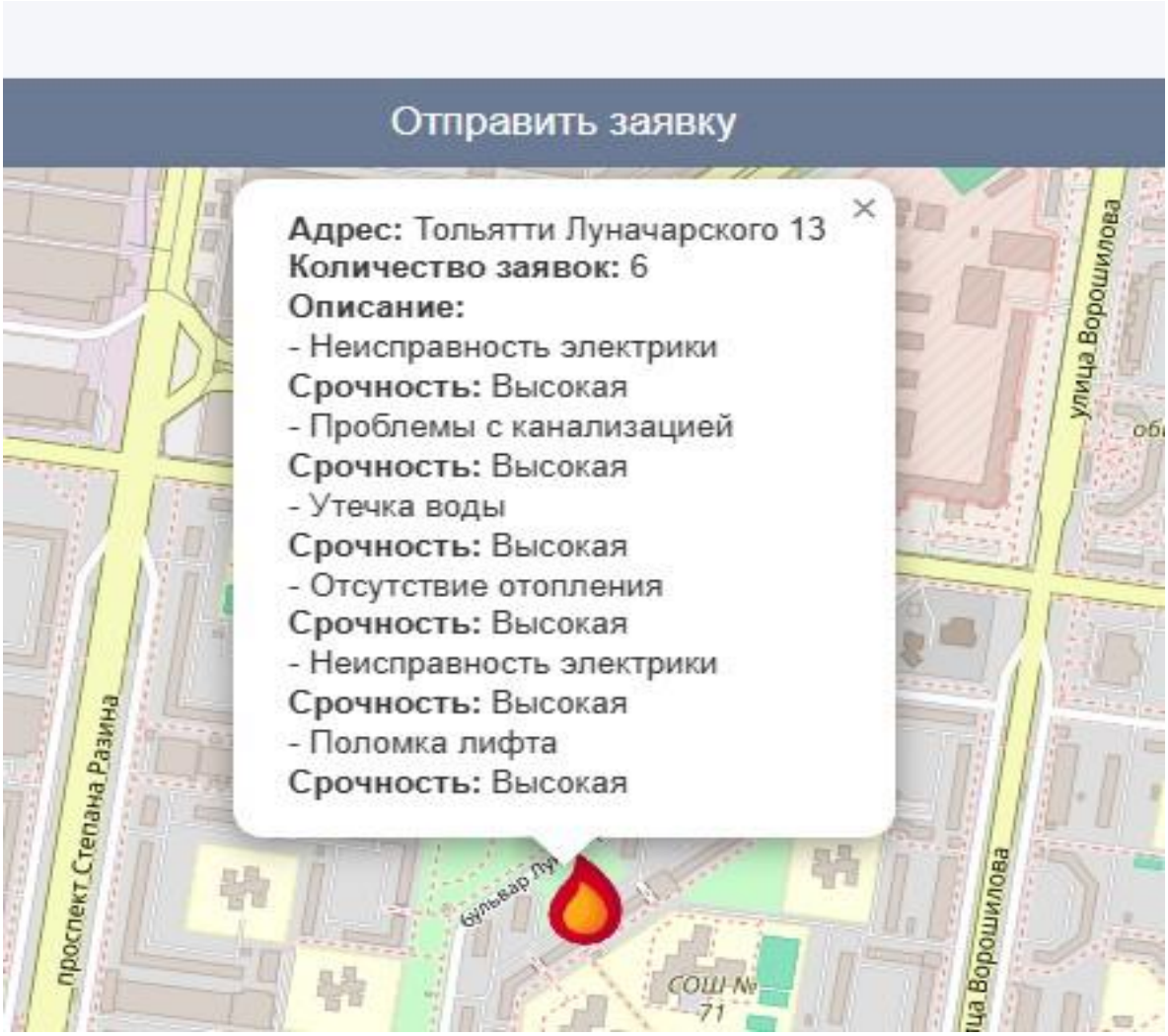
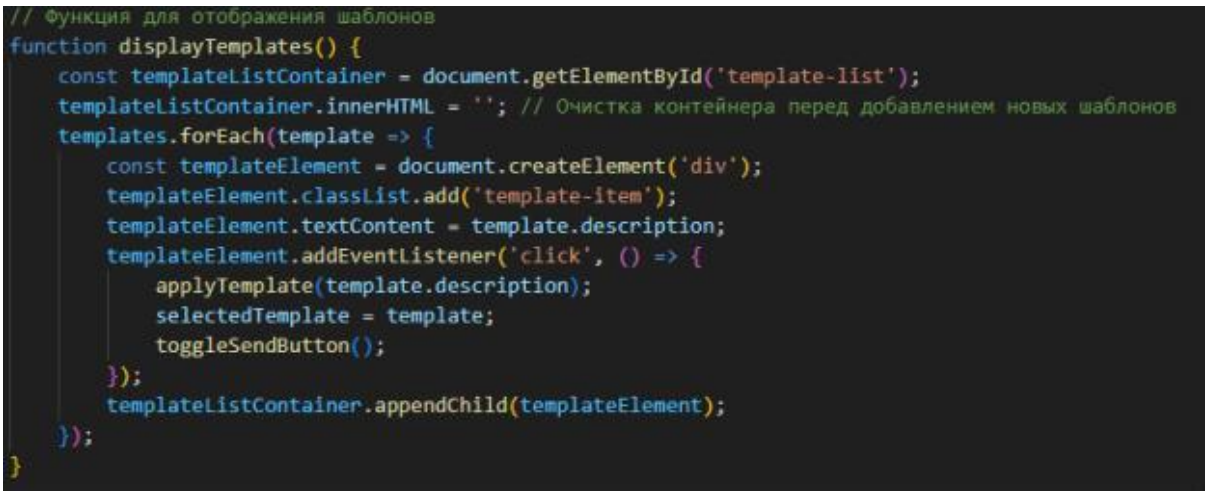


Рисунок 21 – Подсчет заявок и обновление статуса при превышении порога в 5 заявок

Для удобства пользователей был реализован список часто используемых шаблонов заявок [21]. При клике на шаблон его описание автоматически подставляется в текстовое поле, что ускоряет процесс создания заявки. Код, реализующий авто подстановку описания шаблона заявки представлен на рисунке 22.



```
// Функция для отображения шаблонов
function displayTemplates() {
    const templateListContainer = document.getElementById('template-list');
    templateListContainer.innerHTML = ''; // Очистка контейнера перед добавлением новых шаблонов
    templates.forEach(template => {
        const templateElement = document.createElement('div');
        templateElement.classList.add('template-item');
        templateElement.textContent = template.description;
        templateElement.addEventListener('click', () => {
            applyTemplate(template.description);
            selectedTemplate = template;
            toggleSendButton();
        });
        templateListContainer.appendChild(templateElement);
    });
}
```

Рисунок 22 – Код, реализующий авто подстановку описания шаблона заявки

На данном этапе была разработана функциональность, значительно улучшившая взаимодействие пользователей с аварийно–диспетчерской службой через личный кабинет [7]. Все ключевые задачи, поставленные на этапе реализации, успешно выполнены. Это позволило обеспечить удобство подачи заявок, повысить их прозрачность и ускорить обработку.

3.3 Контрольный пример

Данный контрольный пример иллюстрирует процесс подачи заявки на отсутствие отопления через личный кабинет пользователя на платформе «Квартплата 24». Он демонстрирует удобство использования стандартных шаблонов, интерактивной карты и механизма выбора срочности, что делает процесс подачи заявок более быстрым и эффективным.

Описание процесса: Жилец дома по адресу г. Тольятти, ул. Ленина, д. 10, Даниил Нестеренко, открывает сайт «Квартплата 24» и вводит свои учетные данные для входа в личный кабинет. После успешной авторизации он переходит в раздел «Заявки», где представлены функции для подачи и отслеживания обращений. В данном разделе можно создать новую заявку, просмотреть историю обращений и следить за их статусом. Пример

авторизованного жильца на сайте представлен на рисунке 23.

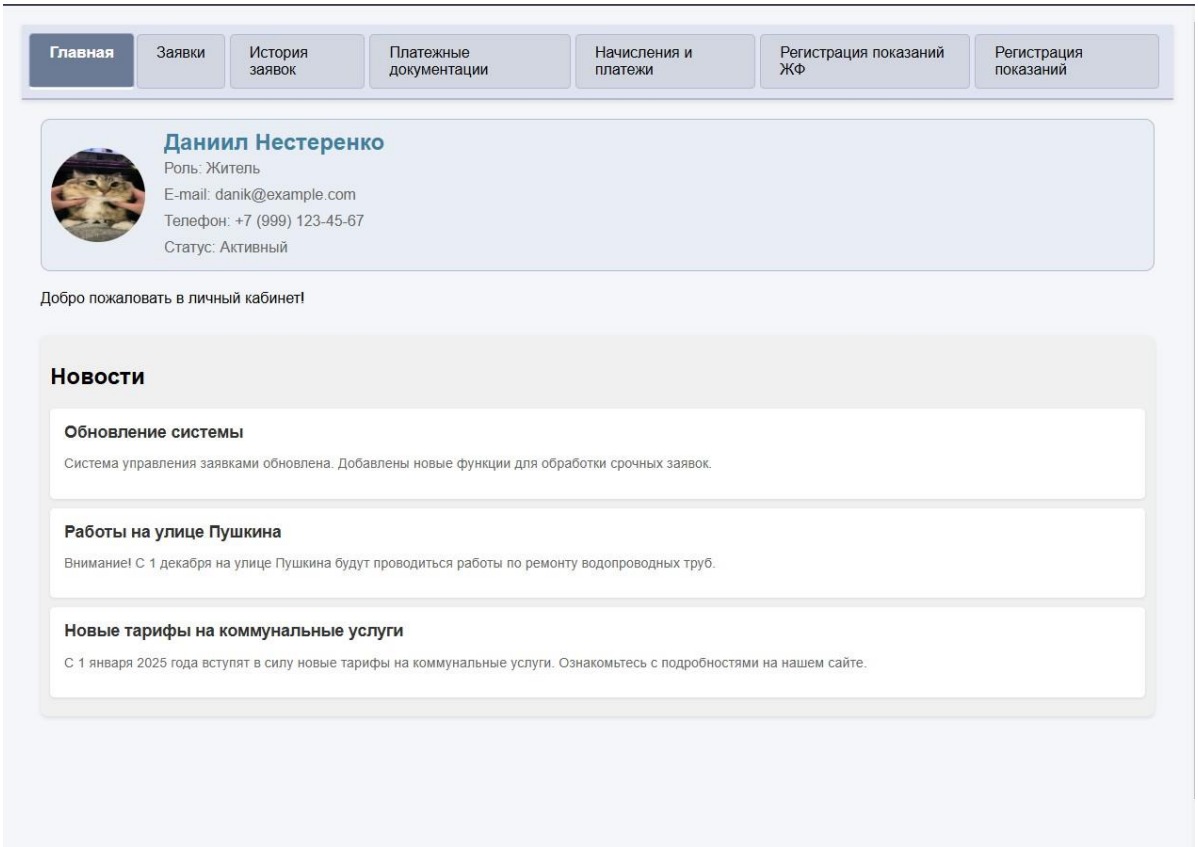


Рисунок 23 – Авторизация на сайте

В разделе «Заявки» Даниил видит перечень стандартных шаблонов для наиболее распространённых проблем, среди которых «Поломка лифта», «Протечка воды» и «Отсутствие отопления». Выбрав шаблон «Отсутствие отопления», он автоматически получает стандартное описание проблемы, но добавляет в текстовое поле дополнительную информацию: «Отопление отсутствует уже третий день, температура в квартире опустилась до 15 градусов». Это уточнение позволяет диспетчерской службе быстрее оценить срочность ситуации. Выбор шаблона обращений и уточнение запроса представлен на рисунке 24.

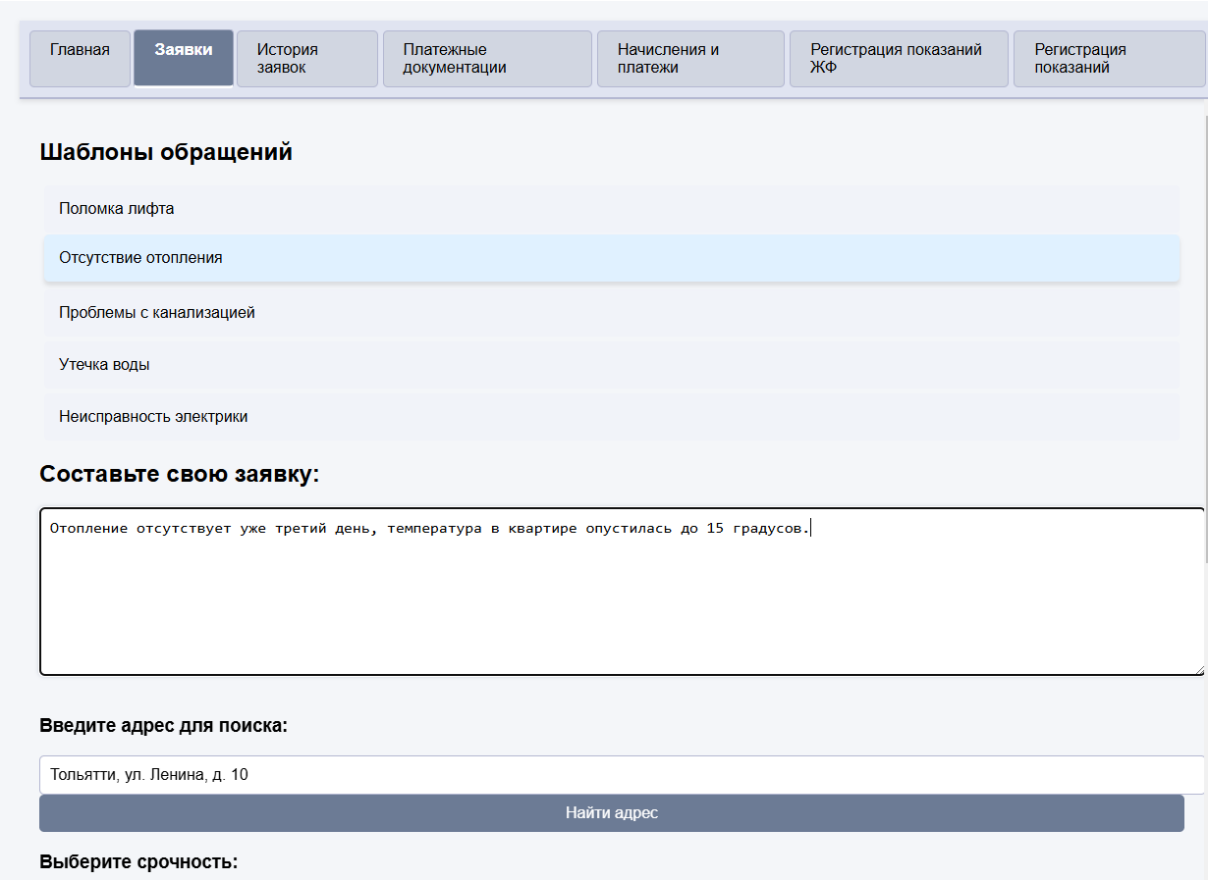


Рисунок 24 – Выбор шаблона обращений и уточнение запроса

Следующий этап – указание адреса. Система предлагает два варианта: ввести адрес вручную или отметить его на интерактивной карте. Даниил вводит «Тольятти, ул. Ленина, д. 10» и нажимает кнопку «Найти адрес». Система автоматически определяет местоположение и отображает его на карте. Убедившись в правильности данных, Даниил подтверждает выбор. Поиск и выбор адреса заявки пользователем представлен на рисунке 25.

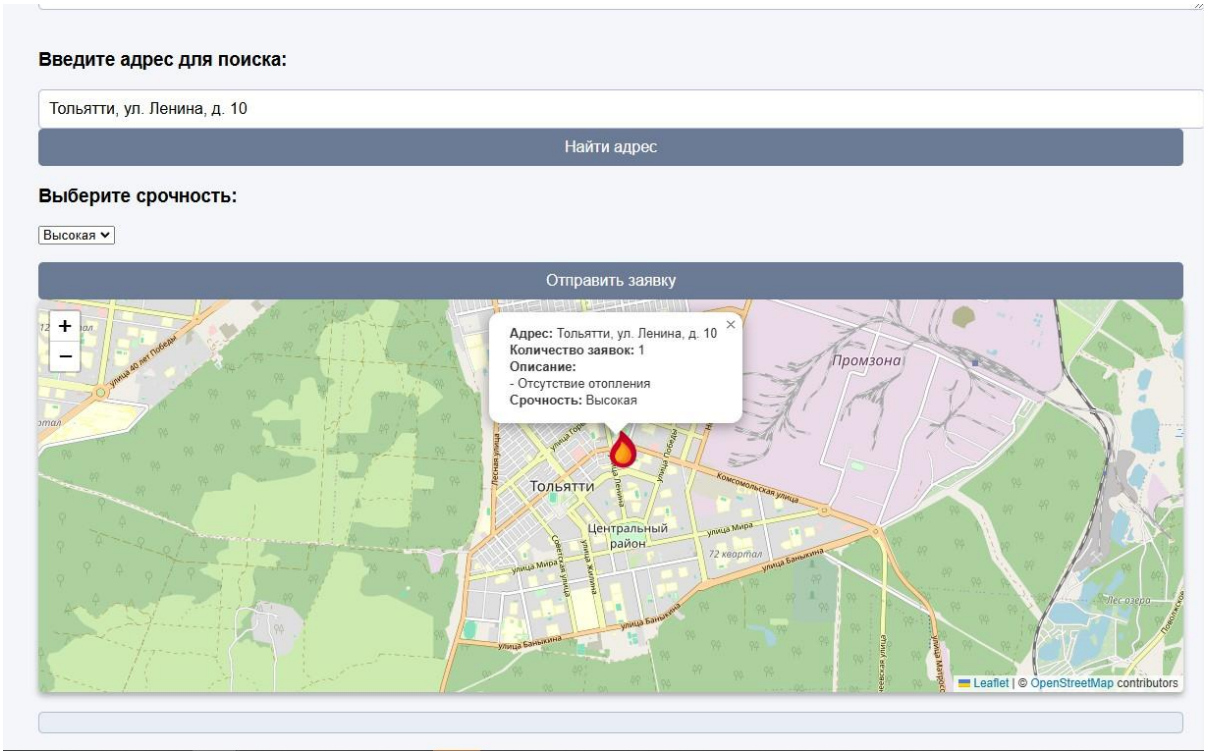


Рисунок 25 – Поиск и выбор адреса заявки пользователем

Далее система предлагает выбрать срочность заявки. Понимая, что отсутствие отопления требует немедленного вмешательства, Даниил выбирает вариант «Высокая». После этого он нажимает кнопку «Отправить заявку». Уведомление пользователя об отправке заявки представлен на рисунке 26.

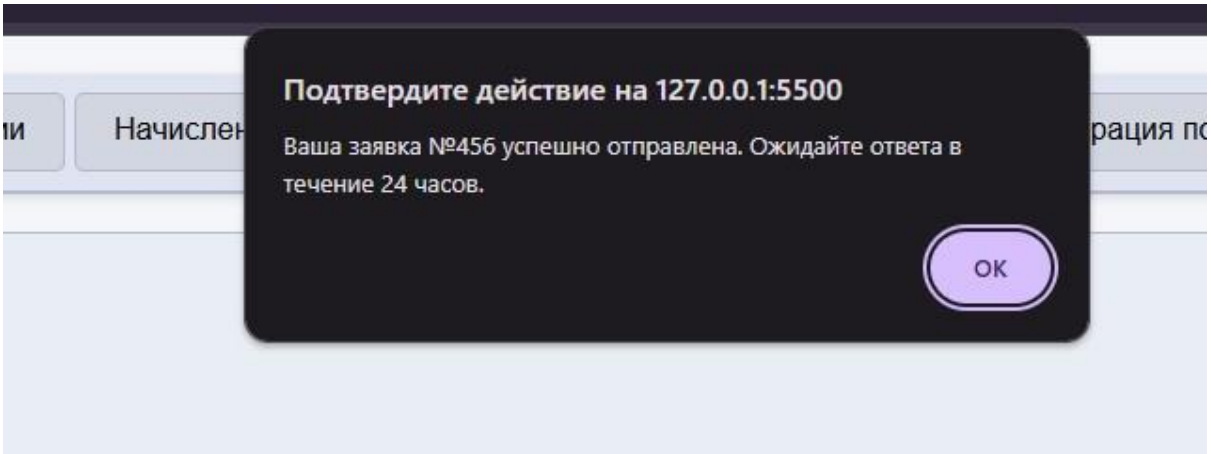


Рисунок 26 – Уведомление пользователя об отправке заявки

После успешной отправки заявки на экране появляется сообщение: «Ваша заявка №456 успешно отправлена. Ожидайте ответа в течение 24 часов». (Рисунок 26). Даниил может закрыть окно и вернуться в личный кабинет, где в разделе «История заявок» отображается созданное им обращение (Рисунок 27).

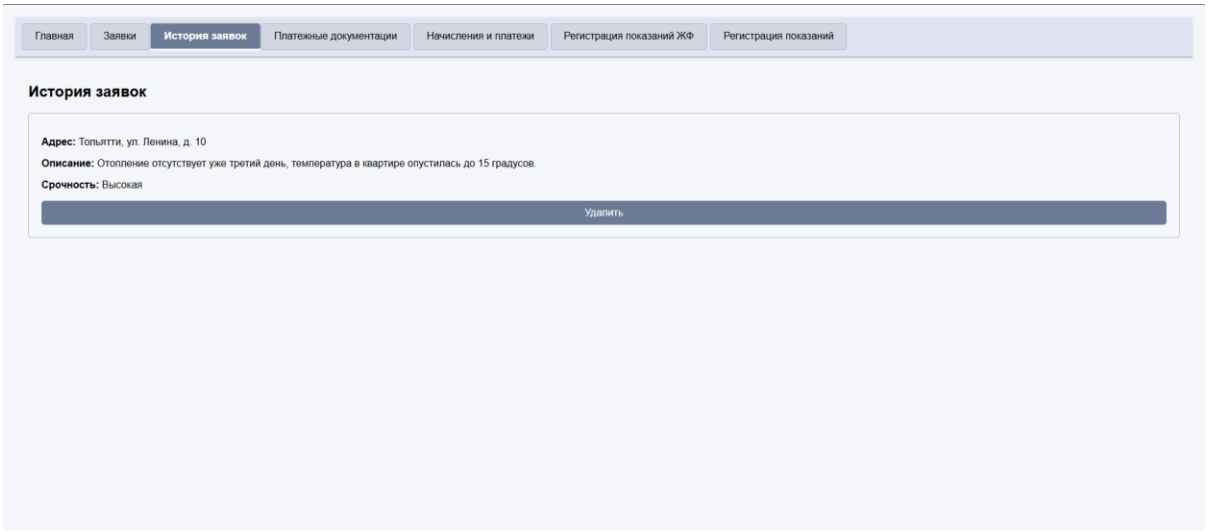


Рисунок 27 – Список оставленных заявок пользователем

Данный процесс показывает, как использование предустановленных шаблонов, карты для уточнения адреса и механизма выбора срочности упрощает и ускоряет подачу заявок [13]. Благодаря этому диспетчерская служба быстрее получает необходимую информацию, что повышает оперативность реагирования. Это подтверждает эффективность предложенных улучшений в модели TO–BE и способствует улучшению качества обслуживания жильцов.

3.4 Расчет экономического эффекта

В рамках расчёта экономического эффекта реализации приложения для компании «Квартплата 24» были определены следующие категории затрат:

– оплата труда специалистов, задействованных в разработке приложения;
– амортизация компьютерного оборудования, используемого в процессе разработки;
– расходы на электроэнергию для работы оборудования.

Детальный расчет затрат на оплату труда специалистов, задействованных в разработке, приведен в таблице 4. Общая сумма затрат на оплату труда составила 506 000 руб., что включает расходы на программиста, системного аналитика, UI/UX дизайнера, тестировщика и менеджера проекта.

Таблица 4 – Расчет сумм оплаты труда специалистов, привлеченных к разработке приложения для компании «Квартплата 24»

Должность	Величина оплаты труда специалиста за 1 час, руб.	Величина трудозатрат	Величина заработной платы, руб.
Программист	1200	150	180 000
Системный аналитик	900	100	90 000
UI/UX дизайнер	800	80	64 000
Тестировщик	600	120	72 000
Менеджер проекта	1000	60	60 000
Итого	–	–	506 000

В структуре издержек основными статьями выступают оплата труда специалистов, амортизационные отчисления, а также расходы на электроэнергию, потреблённую в процессе разработки.

Затраты на оплату труда можно выразить через формулу:

$$З = \sum_{i=1}^{n} (Т_i \times Н_i), \quad (1)$$

где $З$ – суммарные затраты на оплату труда, T_i – часовая ставка i–го специалиста, H_i – затраченное количество часов. Для расчёта берутся показатели по пяти категориям работников: программист (180 000 руб.), системный аналитик (90 000 руб.), UI/UX дизайнер (64 000 руб.), тестировщик (72 000 руб.) и менеджер проекта (60 000 руб.). В итоге общая сумма затрат составляет 506 000 руб.

Однако при определении реальных расходов необходимо учитывать обязательные налоговые отчисления. Они составляют 30,2 % от фонда оплаты труда, что даёт следующую формулу:

$$З_{общ} = З \times (1 + Н) = 506\,000 \times 1{,}302 = 659\,012 \text{руб.}, \quad (2)$$

Следующим элементом затрат является амортизация оборудования, использованного в проекте. Стоимость вычислительной техники составляет 200 000 руб., срок её службы – 5 лет, или 60 месяцев. Период участия оборудования в проекте составляет два месяца. Тогда расчёт амортизационных отчислений выглядит следующим образом:

$$А = \frac{С}{S} \times Р = \frac{200\,000}{60} \times 2 = 8\,000 \text{руб.}, \quad (3)$$

Дополнительным расходом являются затраты на электроэнергию, связанные с использованием вычислительной техники. При мощности 0,7 кВт, времени работы 180 часов и тарифе 6 руб./кВт·ч получаем:

$$Э = М \times В \times Т = 0{,}7 \times 180 \times 6 = 756 \text{руб.}, \quad (4)$$

Таким образом, суммарные расходы на реализацию подсистемы составляют:

$$З_{итог} = З_{общ} + А + Э = 659\,012 + 8\,000 + 756 = 667\,768 \text{руб.}, \quad (5)$$

Для окончательной оценки эффективности внедрения необходимо рассчитать срок окупаемости проекта. Ожидаемый годовой экономический эффект от автоматизации бизнес–процессов оценивается в 350 000 руб., что позволяет определить срок возврата инвестиций по следующей формуле:

$$Т = \frac{З_{итог}}{Э_{год}} = \frac{667\,768}{350\,000} \approx 1{,}91 \text{года}, \quad (6)$$

Таким образом, проведённые расчёты свидетельствуют о том, что разработка и внедрение рассматриваемой подсистемы представляют собой экономически обоснованный шаг. Проект окупится в течение 1,91 года или примерно 22,9 месяцев, что является приемлемым сроком для информационных решений в сфере ЖКХ

Вывод по 3 главе:

В данной главе рассмотрена реализация подсистемы визуализации заявок с использованием HTML, CSS, JavaScript, Leaflet.js и OpenStreetMap для создания удобного интерфейса и интерактивной карты. Реализованы функции: создание заявок с шаблонами, визуализация на карте с кластеризацией, автоизменение статуса при 5+ заявках по адресу. Контрольный пример показал удобство подачи заявок через личный кабинет. Расчет экономического эффекта выявил затраты 664 812 руб. и окупаемость 22,9 месяца. Подсистема обеспечивает автоматизацию, прозрачность и масштабируемость..

Заключение

В ходе выполнения выпускной квалификационной работы была разработана подсистема визуализации заявок для системы аварийно–диспетчерской службы компании «Квартплата 24». Основное внимание было уделено созданию функционала, который улучшает эффективность работы диспетчерской службы и облегчает взаимодействие пользователей с системой.

В процессе работы выполнены следующие задачи:

- изучена предметная область и бизнес–процессы системы аварийно– диспетчерской службы. Для более детального понимания был проведён анализ существующих процессов и потребностей компании;
- построены модели текущих и целевых процессов формирования заявок (AS–IS и TO–BE);
- разработан прототип подсистемы визуализации с функциональностью отображения заявок на карте города, автоматического изменения их статусов и упрощения их обработки;
- проведена оценка затрат на разработку подсистемы и определён срок её окупаемости;
- проанализированы существующие аналоги решений для аварийно– диспетчерских служб, что позволило учесть лучшие практики и повысить эффективность разрабатываемой подсистемы.

Полученные результаты подтверждают актуальность выбранной темы и практическую значимость работы. Подсистема визуализации заявок эффективна, адаптируема для ЖКХ, обеспечивает автоматизацию, оперативность и наглядность. Цели работы достигнуты, задачи выполнены полностью, результаты имеют высокую ценность.

Список используемой литературы и используемых источников

1. Андреев А. В. Автоматизация бизнес-процессов: Учебное пособие. – М.: Инфра-М, 2020. – 320 с.
2. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – М.: Питер, 2020. – 368 с.
3. Дёмин С. А., Кузнецова Е. В. Информационные технологии в управлении: Практическое руководство. – СПб.: Питер, 2021. – 280 с.
4. Дубейковский В. И. Практика функционального моделирования с AllFusion Process Modeler. – М.: ДИАЛОГ-МИФИ, 2021. – 464 с.
5. Исаев Г. Проектирование информационных систем. – М.: Омега-Л, 2012. – 431 с.
6. Концептуальная модель базы данных – диаграмма связи между объектами. [Электронный ресурс]. URL: https://webonto.ru/kontseptualnaya-model-bazyi-dannyih/ (дата обращения: 22.04.2025).
7. Куликов С. С. Тестирование программного обеспечения. Базовый курс. – 3-е изд. – Минск: Четыре четверти, 2020. – 312 с.
8. Лаптев В. И. Информационные системы в управлении ЖКХ: Практическое пособие. – М.: Альпина Паблишер, 2019. – 256 с.
9. Макконнелл С. Совершенный код. – М.: Русская редакция, 2021. – 896 с.
10. Мартин Р. Чистая архитектура. – М.: Питер, 2020. – 352 с.
11. Никифоров А. С. Разработка и оформление заявок. – М.: СибКАП, Москва-Петербургский, 2020. – 424 с.
12. Новиков Б. А., Горшкова Е. А., Графеева Н. Основы технологий баз данных: учебное пособие / под ред. Е. В. Рогова. – 2-е изд. – М.: ДМК Пресс, 2020. – 582 с.

13. Нотации моделирования бизнес-процессов. [Электронный ресурс]. URL: https://leanvector.ru/blog-eksperta/notatsii-modelirovaniya-biznes-protsessov/ (дата обращения: 15.04.2025).

14. Официальный сайт компании «Квартплата 24». [Электронный ресурс]. URL: https://kvartplata24.ru/ (дата обращения: 05.04.2025).

15. Попов В. А. Информационные технологии в ЖКХ: Теория и практика. – СПб.: Лань, 2022. – 304 с.

16. Смирнов С. А. Анализ и моделирование бизнес-процессов с использованием BPMN 2.0. – М.: Диалектика, 2020. – 288 с.

17. Требования к системе: классификация FURPS+. [Электронный ресурс]. URL: https://sysana.wordpress.com/2010/09/16/furps/ (дата обращения: 08.04.2025).

18. Фаулер М. UML. Основы. – М.: Символ-Плюс, 2018. – 224 с.

19. Фримен Э., Робсон Э. Изучаем программирование на JavaScript. – СПб.: Питер, 2021. – 640 с.

20. Харченко В. С. Проектирование и разработка программных систем. – Киев: Логос, 2019. – 412 с.

21. Шаблонные системы управления заявками в ЖКХ. [Электронный ресурс]. URL: https://template-zhkh.ru/ (дата обращения: 25.04.2025).

22. Dumas M., La Rosa M., Mendling J., Reijers H. A. Fundamentals of Business Process Management. – Springer, 2018. – 546 с.

23. ISO 9001:2015: Quality Management Systems – Requirements. – Geneva: International Organization for Standardization, 2015. – 29 с.

24. Leaflet.js: Полное руководство по использованию. [Электронный ресурс]. URL: https://leafletjs.com/ (дата обращения: 19.04.2025).

25. OpenStreetMap Documentation. [Электронный ресурс]. URL: https://wiki.openstreetmap.org/ (дата обращения: 12.04.2025).

26. OpenStreetMap OpenAPI Specification. [Электронный ресурс]. URL: https://github.com/sparkfabrik/nominatim-openapi (дата обращения: 12.04.2025).