

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

Направленность Программная инженерия

БАКАЛАВРСКАЯ РАБОТА

на тему

**Программные средства распределения обращений
клиентов между сотрудниками технической
поддержки**

Студент

(подпись, дата)

Симакин Константин Сергеевич
(ФИО полностью)

Руководитель

(подпись, дата)

Козлов А.Ю.

(фамилия, инициалы)

Нормоконтролёр

(подпись, дата)

Такташкин Д.В.

(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от 03.06.2026 г. № 12)

Заведующий кафедрой

(подпись)

Козлов А.Ю.

(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК

(подпись)

Попова Н.А.

(фамилия, инициалы)

Пенза, 2026

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
А.Ю. Козлов
«26» декабря 2025 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Симакин Константин Сергеевич гр. 22ВП1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Программные средства распределения обращений клиентов между
сотрудниками технической поддержки

Тема утверждена приказом ПГУ №1583/о от "30" декабря 2025 г.

3. Руководитель работы д.т.н., профессор кафедры «МО и ПЭВМ» Козлов А. Ю.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

1) *Назначение: программные средства распределения обращений клиентов между
сотрудниками технической поддержки с возможностью моделирования данных для
демонстрации работоспособности системы*

2) *Среда выполнения: операционные системы Windows, Linux*

Языковые средства разработки: Golang, TypeScript, SQL, UML, Docker

Инструментальные средства: VS Code, Visual Paradigm, PlantUML

Технологии разработки: модульный монолит, асинхронное программирование

3) *Основные функции приложения:*

а) управление данными для моделирования процесса распределения;

б) определение команды сотрудников для обработки обращения;

в) управление механизмом автоматизаций для настройки распределения;

г) распределение обращения на сотрудника из команды обращения;

д) демонстрация актуальных результатов процесса распределения обращений.

5. Перечень подлежащих разработке вопросов

1) <i>Анализ предметной области и требований к системе;</i>
2) <i>Проектирование программных средств;</i>
3) <i>Реализация программных средств;</i>
4) <i>Тестирование программных средств;</i>
5) <i>Оформление пояснительной записки.</i>

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	07.04.2026 – 13.04.2026	
2	Проектирование программных средств	20%	14.04.2026 – 25.04.2026	
3	Реализация программных средств	30%	27.04.2026 – 11.05.2026	
4	Тестирование программных средств	20%	12.05.2026 – 21.05.2026	
5	Оформление пояснительной записки	20%	22.05.2026 – 02.06.2026	

Дата выдачи задания «26» декабря 2025 г.

Руководитель бакалаврской работы _____ 26.12.2025 г.
(подпись, дата)

Задание к исполнению принял студент _____ 26.12.2025 г.
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 100 листов, 59 рисунков, 22 таблицы, 22 использованных источников и 7 приложений.

ПРОГРАММНЫЕ СРЕДСТВА, WEB-ПРИЛОЖЕНИЕ, МОДУЛЬНЫЙ МОНОЛИТ, UML, POSTGRESQL, GOLANG, TYPESCRIPT, DOCKER, АСИНХРОННЫЙ ОБМЕН, API.

Объект исследования – программные средства распределения обращений клиентов между сотрудниками технической поддержки.

Цель выпускной квалификационной работы – разработать программные средства распределения обращений клиентов между сотрудниками технической поддержки с возможностью моделирования данных для демонстрации работоспособности системы.

Технологии разработки – разработка клиентской части приложения проводилась на языке программирования TypeScript с применением библиотеки React в среде разработки Visual Studio Code. Разработка серверной части приложения производилась на языке программирования Golang в среде разработки Visual Studio Code. Разработка базы данных проводилась с использованием СУБД PostgreSQL.

Среда исполнения – операционные системы Windows и Linux, web-браузер.

Результаты работы – в ходе выполнения выпускной квалификационной работы были созданы клиентская и серверная часть программных средств распределения обращений клиентов между сотрудниками технической поддержки с возможностью моделирования данных для демонстрации работоспособности системы. Все требования к разработке приложения соблюдены.

					ПГУ 09.03.04 – 08BP221.19 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Программные средства распределения обращений клиентов между сотрудниками технической поддержки. Пояснительная записка	Лит.	Лист	Листов
Разраб.		Симакин К.С.					4	100
Пров.		Козлов А.Ю.				Группа 22ВП1		
Реценз.								
Н. Контр.		Такташкин Д.В.						
Утв.		Козлов А.Ю.						

Содержание

Введение.....	7
1 Анализ предметной области и требований на разработку.....	9
1.1 Анализ предметной области и постановка задачи	9
1.2 Анализ аналогов.....	10
1.2.1 Коммуникационная платформа Intercom	10
1.2.2 Коммуникационная платформа Carrot Quest	12
1.2.3 Коммуникационная платформа LiveChat.....	13
1.2.4 Результаты сравнительного анализа аналогов.....	15
1.3 Анализ функциональных требований.....	16
1.4 Анализ нефункциональных требований.....	21
1.5 Планирование процесса разработки	21
2 Проектирование программных средств распределения обращений	25
2.1 Выбор технологий и инструментальных средств.....	25
2.2 Проектирование структур данных	26
2.2.1 Концептуальная модель базы данных	26
2.2.2 Логическая модель базы данных.....	27
2.3 Проектирование серверной части приложения	29
2.3.1 Проектирование алгоритма распределения	29
2.3.2 Проектирование механизма работы лимитов сотрудников	32
2.3.3 Проектирование архитектуры серверной части приложения .	33
2.4 Проектирование клиентской части приложения	35
3 Реализация программных средств распределения обращений	37
3.1 Реализация базы данных	37
3.2 Реализация серверной части	44
3.3 Реализация пользовательского интерфейса приложения	46
3.4 Реализация клиентской части приложения	62

4	Тестирование программных средств распределения обращений	64
4.1	Функциональное ручное тестирование	64
4.2	Модульное тестирование	66
4.3	Метрики кода.....	67
4.2.1	Метрики SLOC	67
4.2.2	Метрики Холстеда	68
4.2.3	Цикломатическая сложность	71
5	Развертывание и сопровождение программных средств	72
	Заключение	74
	Список использованных источников	76
	Приложение А (справочное) Глоссарий	78
	Приложение Б (обязательное) Исходный код миграций	80
	Приложение В (обязательное) Исходный код серверной части	84
	Приложение Г (обязательное) Исходный код клиентской части.....	88
	Приложение Д (обязательное) Результаты тестирования.....	91
	Приложение Е (обязательное) Исходный код docker-файлов	97
	Приложение Ж (рекомендуемое) Акт о внедрении	99

Введение

В настоящее время идет активное развитие информационных технологий, которые проникают во все сферы жизни.

Современные компании, которые предоставляют какие-либо услуги, должны иметь механизм связи со своими клиентами. Предприятия активно переходят с коммуникации методом телефонного звонка на формат мессенджеров. Такой подход обеспечивает следующие преимущества:

- возможность асинхронного обмена информацией;
- доступность всей истории общения в рамках текстового чата;
- удобство обмена данными, представленными в виде файлов, например, при передаче документов;
- снижение стресса от общения как у клиентов, так и у сотрудников;
- возможность одновременной обработки нескольких чатов одним сотрудником.

Чтобы сохранить лояльность клиентов и обеспечить высокий уровень сервиса, компания должна оперативно обрабатывать входящие обращения. Это осложняется количеством запросов – в зависимости от размера организации могут создаваться тысячи обращений в день. Кроме того, различные обращения могут требовать участия сотрудников разной квалификации, что должно учитываться при выборе специалиста.

Ручное распределение обращений требует дополнительных человеческих ресурсов, при том в процессе такого распределения будет трудно учитывать все факторы. Таким образом, этот формат работы не является оптимальным.

Исходя из вышесказанного, можно сделать вывод, что для эффективной обработки обращений клиентов требуется автоматизированная система для распределения входящих запросов между сотрудниками предприятия.

Таким образом, целью выпускной квалификационной работы является разработка программных средств распределения обращений клиентов между сотрудниками технической поддержки.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- провести анализ предметной области и определить требования;
- выполнить проектирование структур данных, серверной, а также клиентской частей приложения;
- реализовать систему;
- провести тестирование разработанного приложения;
- настроить развертывание системы для упрощения ее последующего сопровождения.

1 Анализ предметной области и требований на разработку

1.1 Анализ предметной области и постановка задачи

В рамках выполнения задачи выпускной квалификационной работы необходимо разработать систему распределения обращений клиентов между сотрудниками клиентской поддержки.

В качестве предметной области будет рассматриваться онлайн-платформа технической поддержки клиентов. В данной системе существуют клиенты, формирующие обращения, а также сотрудники, которые занимаются их обработкой.

Клиенты могут обращаться по разным вопросам, например, для помощи при оформлении какой-либо услуги или при возникновении технических проблем. Для обработки этих запросов могут понадобиться совершенно разные компетенции, соответственно сотрудники должны быть сгруппированы по определенным предметным областям.

Также в реальной системе каждую секунду может появляться множество обращений. При этом для сохранения качества обслуживания сотрудник должен иметь определенный лимит обращений для одновременной обработки.

Данные особенности предметной области должны быть учтены в разрабатываемой системе.

Функционал распределения обращений будет реализован в рамках серверной части. При этом для проверки работоспособности приложения также понадобится клиентская часть. В ней должна присутствовать возможность управления данными, необходимыми для распределения обращения, также пользовательский интерфейс должен позволять отслеживать результаты работы процесса распределения.

Таким образом, в результате выполнения работы должны быть разработаны программные средства по распределению обращений клиентов между сотрудниками с учетом рассмотренных особенностей предметной

области. Также должны быть реализованы компоненты для демонстрации работоспособности этих системных средств.

1.2 Анализ аналогов

1.2.1 Коммуникационная платформа Intercom

Intercom – платформа, предоставляющая компаниям возможность организовать коммуникацию с их клиентами по средству сообщений [1]. Intercom разрабатывается одноименной американской компанией, предоставляет свои услуги по модели платной подписки. Данная система поддерживает автоматическое распределение обращений клиентов между сотрудниками.

Intercom обладает обширным функционалом. К достоинствам можно отнести следующее:

- возможность группировать сотрудников по предметным областям;
- возможность определять лимит одновременно назначенных на сотрудника обращений;
- возможность приоритезации обращений в зависимости от важности клиента.

Система поддерживает следующие методы распределения обращений:

- ручной режим распределения, когда сотрудники сами назначают себя на выбранные чаты;
- round robin, когда чаты распределяются последовательно по очереди;
- сбалансированный режим, который учитывает множество факторов для обеспечения наиболее оптимального распределения, например, нагрузка сотрудников или важность чата.

Скриншот интерфейса настроек системы распределения Intercom приведен на рисунке 1.

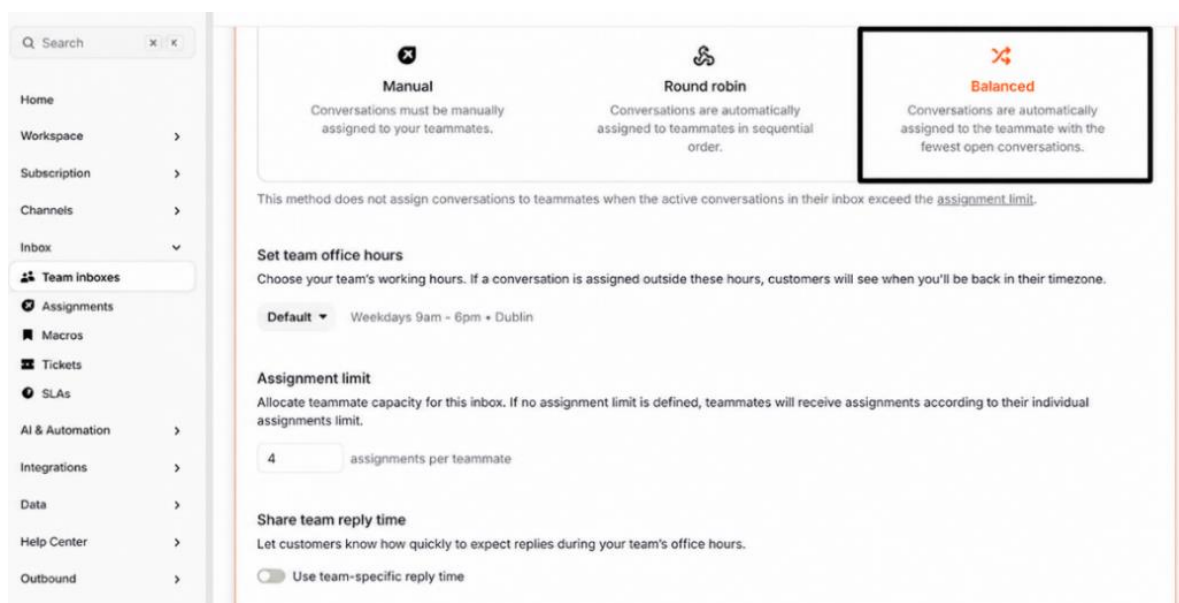


Рисунок 1 – Меню настроек распределения обращений в Intercom

Также в дополнение к распределению обращений поддерживается функционал автоматизаций. Автоматизации при выполнении заданных пользователями условий устанавливают на обращение группу сотрудников, между которыми далее будет распределено обращение. Это позволяет сотрудникам гибко настраивать процесс распределения между командами без внесения изменения в исходный код приложения. Скриншот интерфейса создания автоматизацией приведен на рисунке 2.

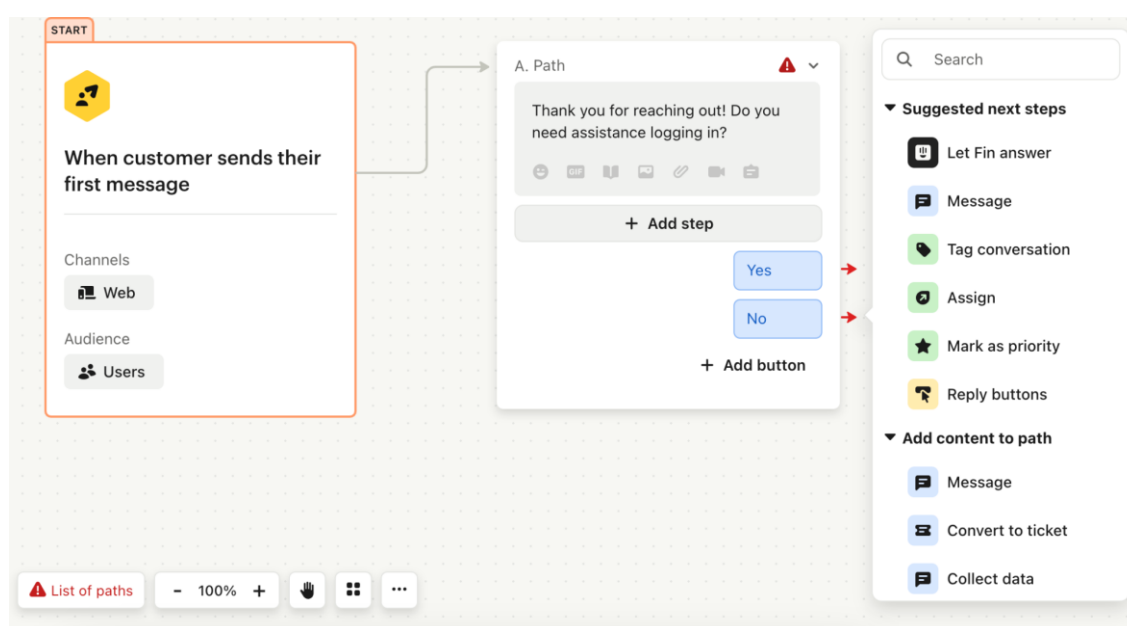


Рисунок 2 – Интерфейс создания автоматизации в Intercom

К минусам Intercom можно отнести следующее:

- автоматическое распределение обращений доступно только в самом дорогом тарифе;
- система официально не распространяется в России.

1.2.2 Коммуникационная платформа Carrot Quest

Carrot Quest – система, разработанная российской компанией, предоставляющая компаниям способ коммуникации со своими клиентами в виде онлайн-чатов [2]. Carrot Quest распространяется по системе платной подписки. Данная система также поддерживает автоматическое распределение обращений клиентов.

К преимуществам Carrot Quest относятся:

- низкая цена подписки по сравнению с аналогами;
- возможность установки лимита одновременно обрабатываемых сотрудником обращений.

Ниже на рисунке 3 приведен скриншот настройки автоматического распределения обращений в Carrot Quest.

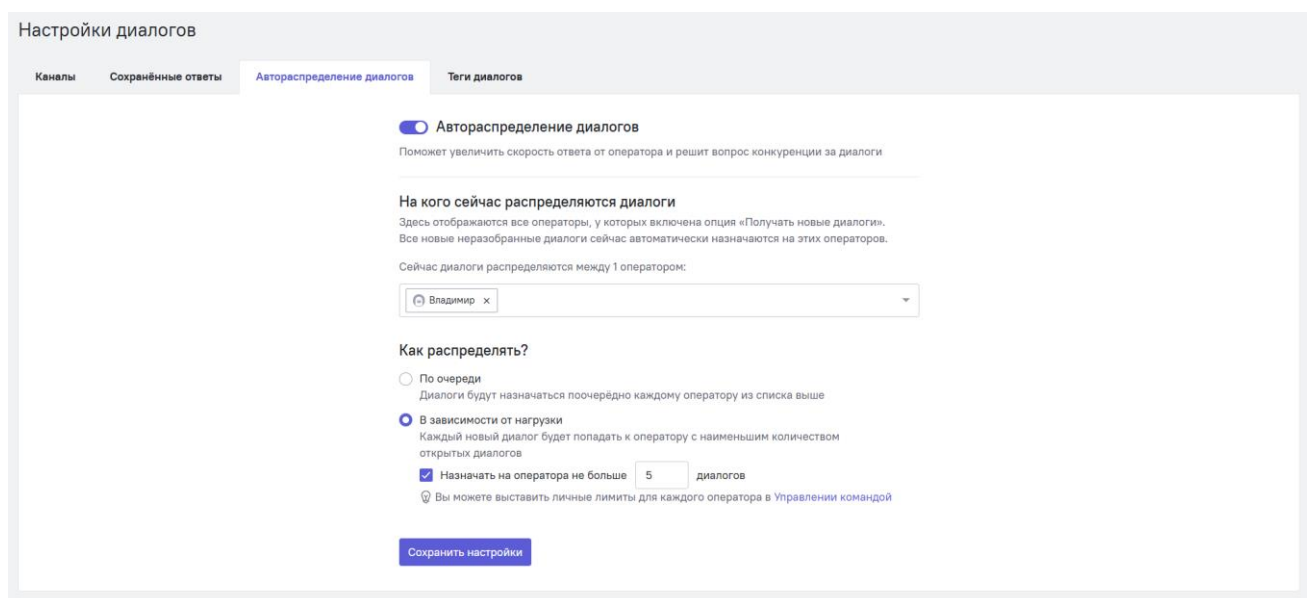


Рисунок 3 – Настройка автоматического распределения диалогов в Carrot Quest

У Carrot Quest есть следующие недостатки:

- отсутствует возможность группировки сотрудников по предметным областям с последующим распределением обращений на сотрудников этих групп;
- отсутствует возможность приоритезации обращения от важных клиентов при распределении.

Также к минусам этой системы можно отнести скромный функционал автоматизаций. Пользователь может создавать автоматизации, скриншот интерфейса по настройке автоматизации приведен на рисунке 4.

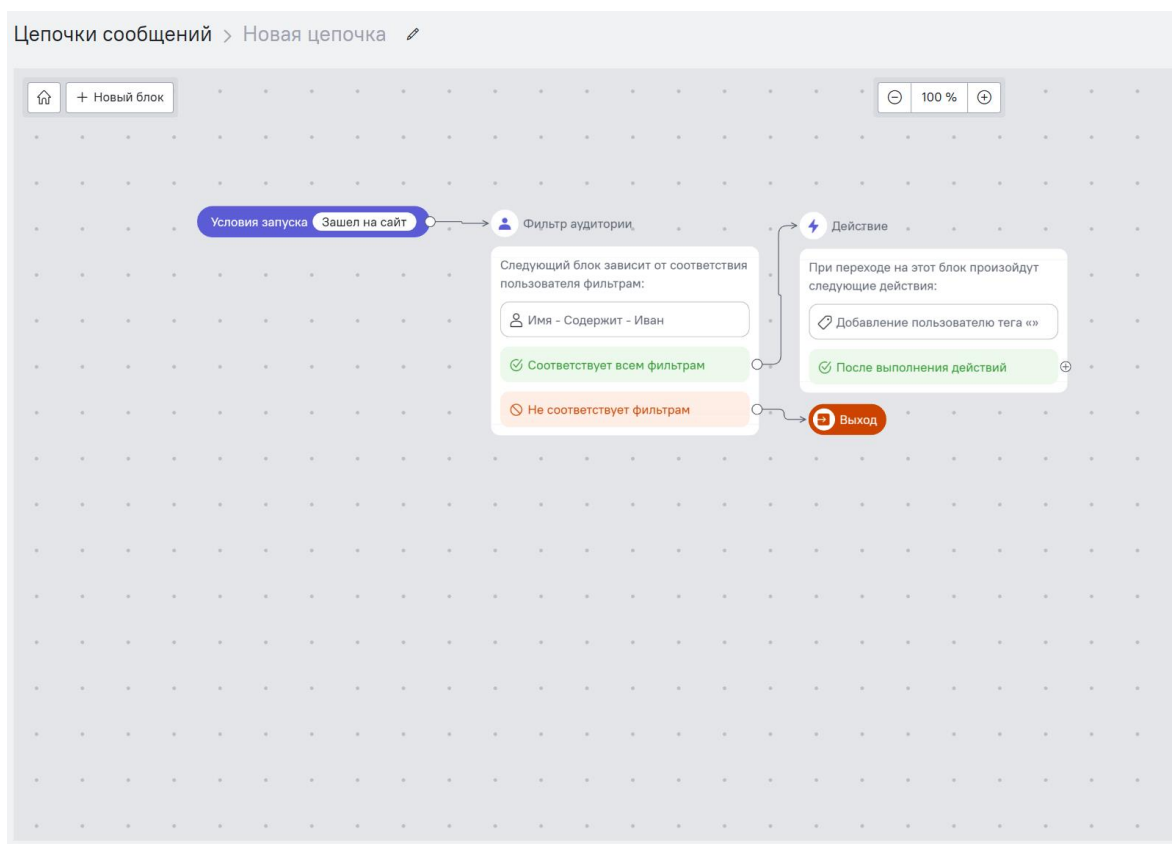


Рисунок 4 – Настройка автоматизации в Carrot Quest

Использование автоматизаций для распределения обращений в Carrot Quest не предусмотрено, что снижает гибкость системы.

1.2.3 Коммуникационная платформа LiveChat

LiveChat – система предоставления компаниям механизма коммуникации с их клиентами в онлайн-чатах [3]. Эта система разрабатывается польской компанией Text и распространяется по подписке.

LiveChat обладает следующими преимуществами:

- присутствует возможность группировки сотрудников с распределением обращения на сотрудника внутри этой группы;
- возможна установка лимита одновременно назначенных на сотрудника обращений.

Ниже на рисунке 5 приведен скриншот интерфейса LiveChat с настройкой сотрудников, обрабатывающих чаты клиентов.

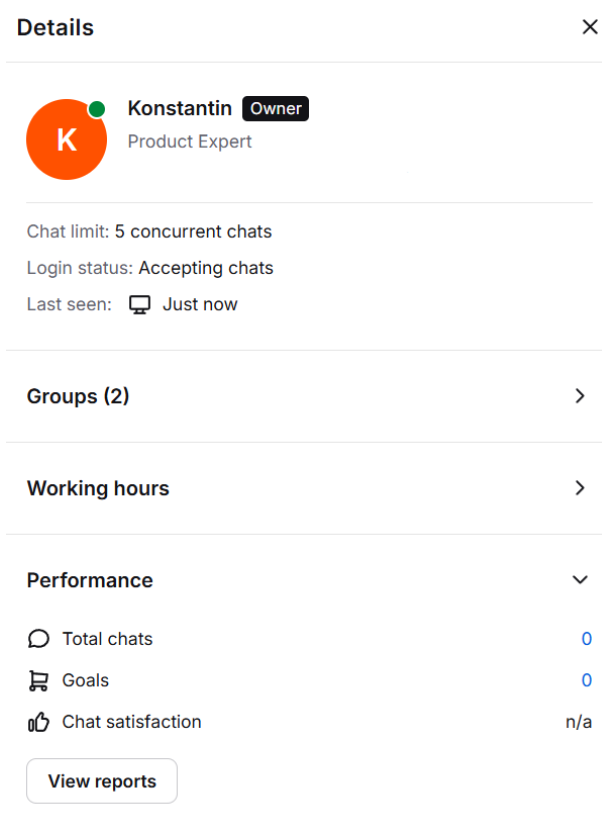


Рисунок 5 – Настройка сотрудников для распределения обращений

Также LiveChat поддерживает механизм автоматизаций для более гибкой настройки механизма распределения. Интерфейс создания автоматизации для распределения обращения на конкретную команду сотрудников приведен на рисунке 6.

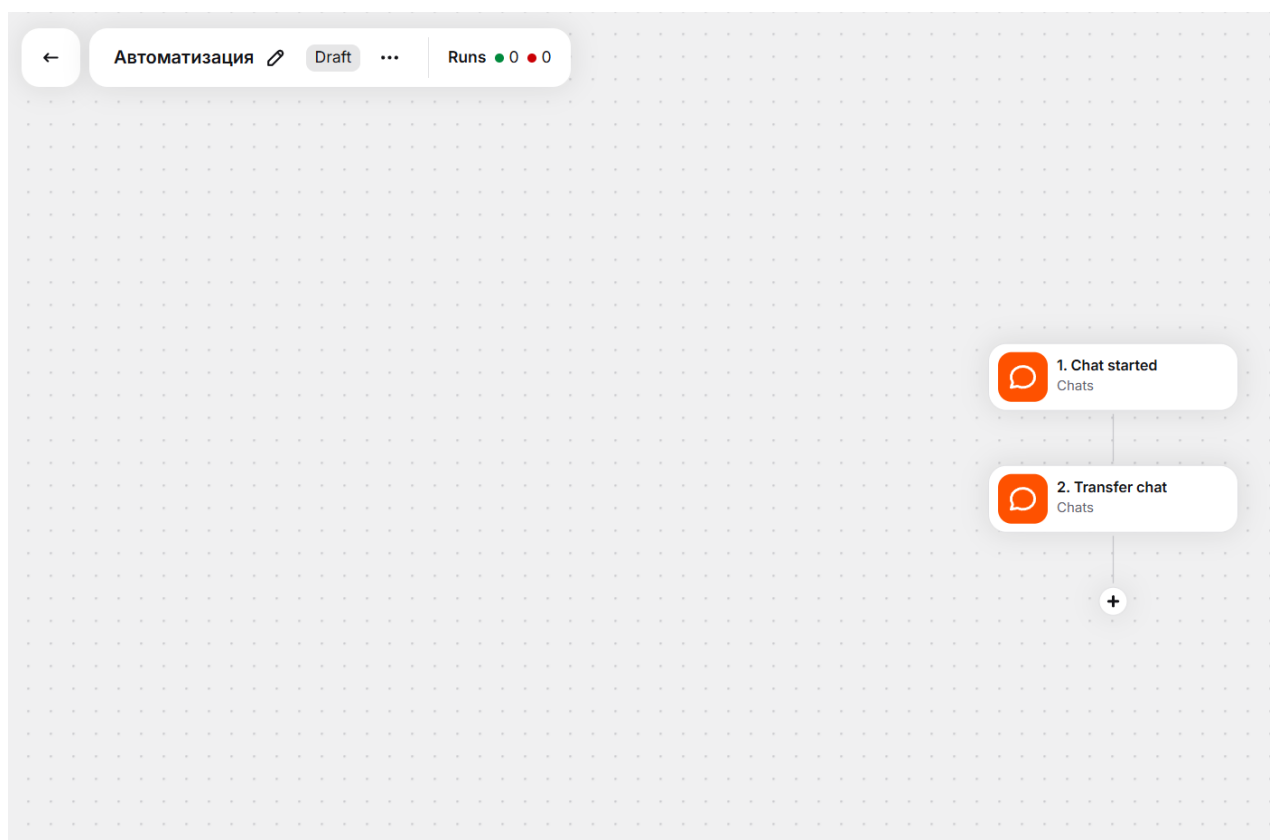


Рисунок 6 – Интерфейс создания автоматизации в LiveChat

К минусам LiveChat можно отнести отсутствие возможности настройки приоритета для распределения обращений от важных клиентов.

1.2.4 Результаты сравнительного анализа аналогов

На основе проведенного анализа аналогов была сформирована таблица 1 со сравнением функционала программных средств.

Таблица 1 – Сравнение систем распределения обращений

№	Критерий	Intercom	Carrot Quest	LiveChat
1	Возможность группировки сотрудников	Да	Нет	Да
2	Возможность создания автоматизаций для распределения	Да	Нет	Да

Продолжение таблицы 1

№	Критерий	Intercom	Carrot Quest	LiveChat
3	Возможность установки лимита количества одновременно назначенных на сотрудника обращений	Да	Да	Да
4	Возможность увеличения приоритета обращений от важных клиентов при распределении	Да	Нет	Нет
5	Доступность на российском рынке	Нет	Да	Да
6	Цена подписки	Высокая	Низкая	Средняя

По результатам сравнения программных средств аналогичного назначения не было найдено ни одной системы, которая удовлетворяла бы всем требованиям. Поэтому было решено разработать собственное программное средство, которое будет удовлетворять всем рассмотренным критериям.

1.3 Анализ функциональных требований

В системе должна присутствовать возможность создания, просмотра, редактирования и удаления клиентов, обращения которых будут распределяться. Клиенты могут быть обычными либо VIP. VIP-статус клиента должен повышать приоритет обращения этого клиента во время распределения.

Должна быть возможность создания, просмотра, редактирования и удаления сотрудников, которые будут назначаться на обращения для обработки.

Каждый сотрудник может либо находиться в статусе «работа», когда на него могут назначаться обращения, либо в статусе «перерыв», когда система не будет назначать на него обращения. Также у каждого сотрудника должен определяться максимальный лимит одновременно назначенных активных

обращений, достигнув которого сотрудник перестает участвовать в процессе распределения.

Должна присутствовать возможность создания, просмотра, редактирования и удаления команд сотрудников. Сначала требуется определить команду, ответственную за обращение, а потом назначить на обращение одного из сотрудников этой команды. Каждый сотрудник может принадлежать нескольким командам.

Должен присутствовать функционал создания, просмотра, редактирования и удаления тем и подтем, с которыми будут связываться команды сотрудников.

Должна поддерживаться возможность создания обращения от лица клиента с указанием темы, подтемы и текста обращения. После создания обращения оно должно распределиться на одного из сотрудников. Необходим функционал, позволяющий закрыть активное обращение. Закрытое обращение не участвует в алгоритме распределения, а также не считается обрабатываемым сотрудником.

Должен быть реализован функционал автоматизаций, которые в зависимости от условий будут устанавливать команду сотрудников обращения. Автоматизации должны поддерживать следующие условия:

- равенство почты клиента указанному значению;
- время создания обращения в указанный временной промежуток;
- содержание указанных ключевых слов в тексте обращения;
- равенство темы обращения указанному значению.

Все условия в рамках автоматизации могут быть объединены логическими «и» или «или». Автоматизация завершается после прохождения всех условий либо при ложном значении одного из условий.

Система должна позволять просматривать созданные обращения пользователей, динамически обновлять состояния обращения для корректного отображения результатов распределения.

Система также должна отображать, сколько еще обращений может быть назначено на определенного сотрудника.

Для демонстрации функциональных требований ниже на рисунке 7 демонстрируется диаграмма вариантов использования [4] системы распределения обращений клиентов между сотрудниками.

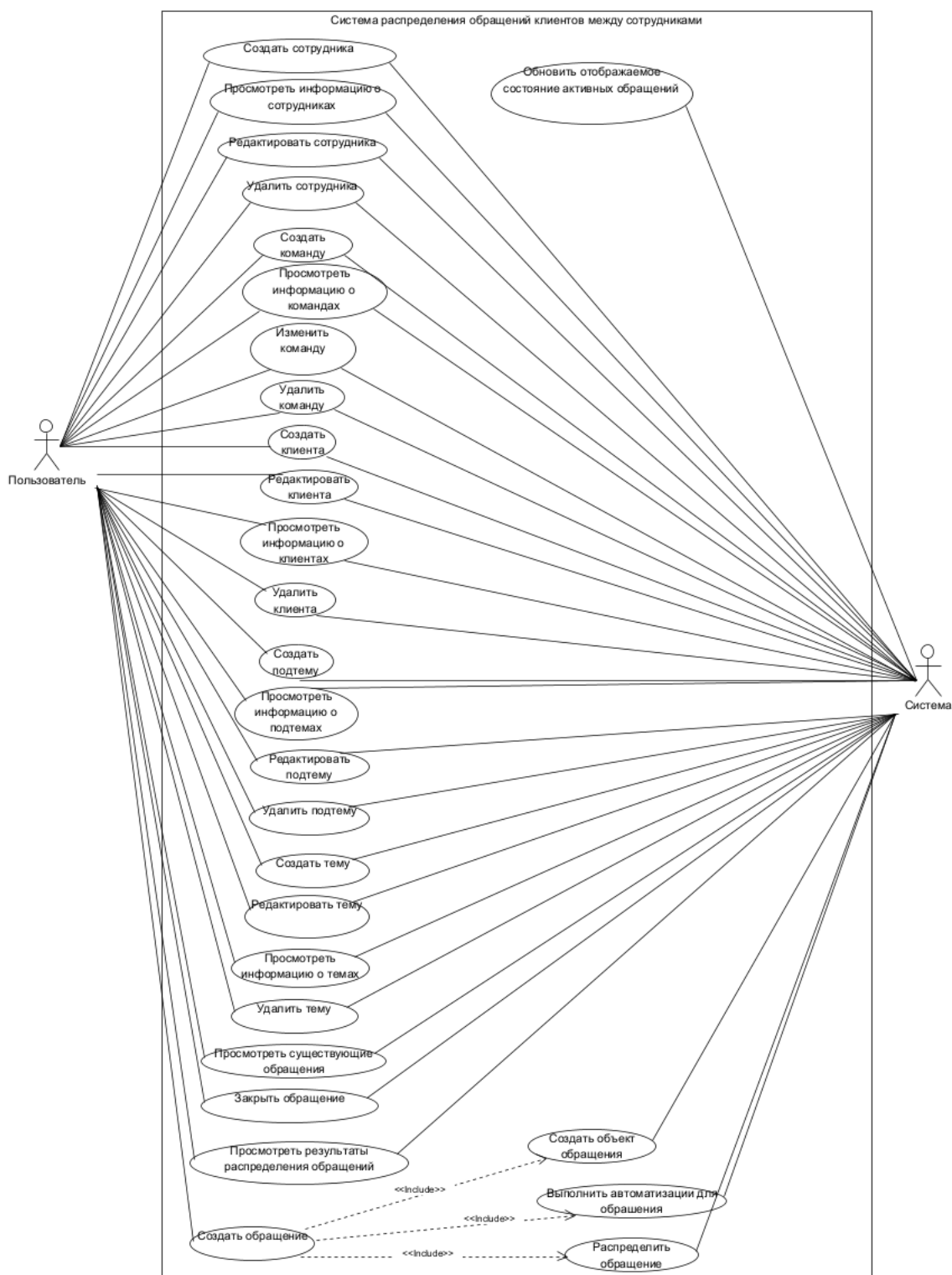


Рисунок 7 – Диаграмма вариантов использования системы распределения обращений клиентов между сотрудниками

На рисунке 8 представлен сценарий варианта использования «Создать обращение».

Название: создать обращение.
ID: 1.
Краткое описание: пользователь создает обращение для распределения.
Действующие лица: пользователь, система.
Предусловие: созданы тема, подтема, клиент, создающий обращение.
Основной поток: 1. Пользователь вводит данные для создания обращения. 2. Исходя из темы, подтемы обращения, VIP-статуса клиента система определяет команду сотрудников для обработки обращения. 3. Система записывает данные обращения в базу данных. 4. Include (Выполнить автоматизации для обращения). 5. Include (Распределить обращение).
Постусловие: создано обращение.

Рисунок 8 – Сценарий варианта использования «Создать обращение»

Далее на рисунке 9 демонстрируется сценарий варианта использования «Распределить обращение».

Название: распределить обращение.
ID: 2
Краткое описание: система распределяет обращение свободному сотруднику.
Предусловие: обращение создано.
Основной поток: 1. Система добавляет обращение в очередь на распределение. 2. Система находит оптимального свободного сотрудника. 3. Система изменяет информацию об обращении в базе данных. 4. Система изменяет информацию о назначенном сотруднике в базе данных. 5. Из очереди на распределение удаляется обращение.
Постусловие: обращению назначен сотрудник для дальнейшей обработки.

Рисунок 9 – Сценарий варианта использования «Распределить обращение»

Ниже на рисунке 10 представлен сценарий варианта использования «Заккрыть обращение».

Название: закрыть обращение.
ID: 3.
Краткое описание: пользователь закрывает активное обращение.
Действующие лица: пользователь, система.
Предусловие: обращение активно.
Основной поток: <ol style="list-style-type: none"> 1. Пользователь изменяет статус обращение на «Закртыо». 2. Система изменяет статус обращения в базе данных.
Постусловие: обращение закрыто.

Рисунок 10 – Сценарий варианта использования «Заккрыть обращение»

Далее на рисунке 11 представлено более детализированное описание варианта использования «Обновить отображаемое состояние активных обращений» в формате диаграммы последовательности.

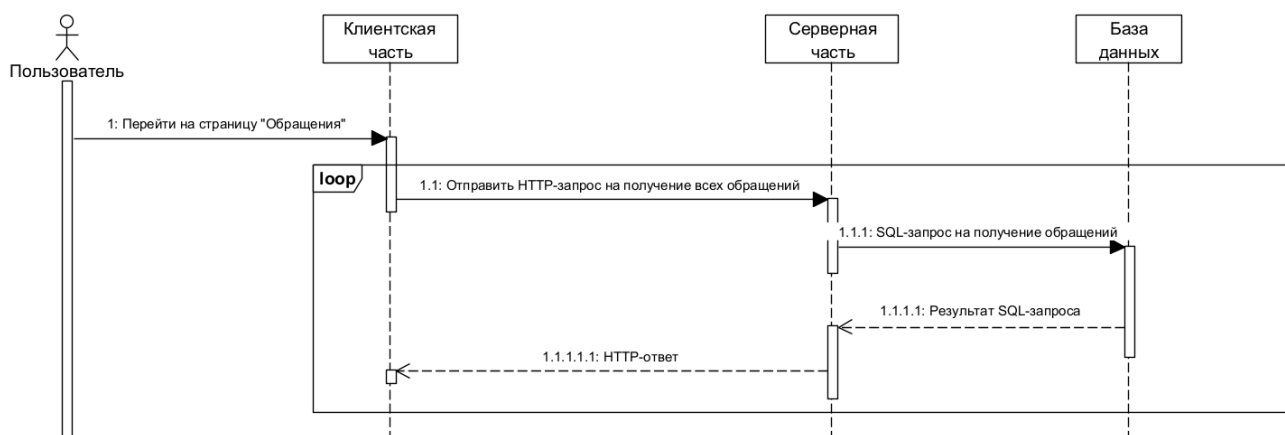


Рисунок 11 – Диаграмма последовательности варианта использования «Обновить отображаемое состояние активных обращений»

Таким образом, система должна циклично получать данные обращения для отображения актуальной информации о статусе распределения обращения.

1.4 Анализ нефункциональных требований

Система должна состоять из клиентской и серверной части, взаимодействующих по HTTP-протоколу.

Серверная часть приложения должна поддерживать асинхронный формат работы для повышения производительности – после создания обращения оно должно асинхронно распределяться.

Клиентская часть должна предоставлять пользователю удобный, простой в использовании интерфейс для взаимодействия с системой.

Система должна быть легко расширяемой, обеспечивать простое добавление нового функционала.

Приложение должно стабильно работать на операционных системах Windows, Linux.

1.5 Планирование процесса разработки

Для эффективной разработки программных средств распределения обращений клиентов между сотрудниками технической поддержки необходимо провести планирование разработки. Планирование является важным этапом выполнения проекта, так как оно позволяет более точно определить задачи разработки, а также снизить риски.

Для планирования процесса разработки использовался Microsoft Project. Были определены все ресурсы проекта, лист ресурсов представлен на рисунке 12.

	Название ресурса	Тип	Единицы измерения материалов	Краткое название	Группа	Макс. единиц	Стандартная ставка	Ставка сверхурочн	Затраты на использ.	Начисление	Базовый календарь
1	Программист	Трудовой		П		100%	450,00 Р/ч	550,00 Р/ч	0,00 Р	Пропорциональн	Стандартный
2	Аналитик	Трудовой		А		100%	300,00 Р/ч	400,00 Р/ч	0,00 Р	Пропорциональн	Стандартный
3	Тестировщик	Трудовой		Т		100%	300,00 Р/ч	400,00 Р/ч	0,00 Р	Пропорциональн	Стандартный

Рисунок 12 – Лист ресурсов проекта

Была разработана диаграмма вариантов разработки, которая демонстрирует задачи каждого участника разработки проекта. Данная диаграмма приведена на рисунке 13.

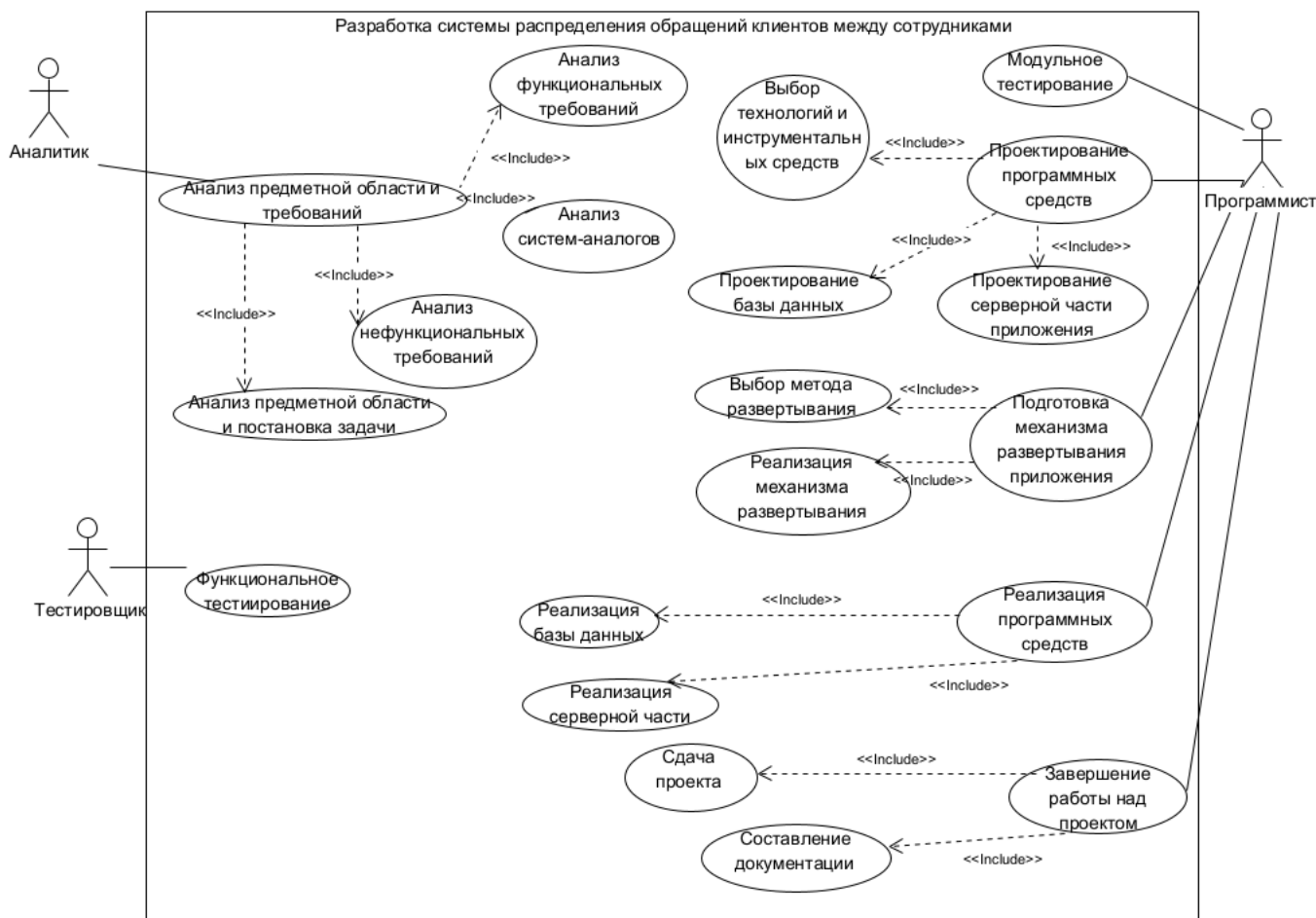


Рисунок 13 – Диаграмма вариантов разработки программных средств

Таким образом, для реализации программных средств распределения обращений будут задействованы следующие роли:

- аналитик;
- тестировщик;
- программист.

Для более детализированного управления задачами была составлена таблица задач проекта, где для каждой задачи определены требуемые ресурсы, а также временные интервалы выполнения. Таблица задач представлена далее на рисунке 14.

	Режим задачи	Название задачи	Длительность	Начало	Окончание	Предшественники	Названия ресурсов
1	🚀	Начало проекта	0 дней	Ср 08.04.26	Ср 08.04.26		
2	🚀	➤ Анализ предметной области и требований	8 дней	Ср 08.04.26	Пн 20.04.26		
3	🚀	Анализ систем-аналогов	3 дней	Ср 08.04.26	Пт 10.04.26	1	Аналитик
4	🚀	Анализ предметной области и постановка зада	2 дней	Пн 13.04.26	Вт 14.04.26	3	Аналитик
5	🚀	Анализ функциональных требований	2 дней	Ср 15.04.26	Чт 16.04.26	4	Аналитик
6	🚀	Анализ нефункциональных требований	1 день	Пт 17.04.26	Пт 17.04.26	5	Аналитик
7	🚀	Анализ предметной области и постановка зада	0 дней	Пн 20.04.26	Пн 20.04.26	6	
8	🚀	➤ Проектирование программных средств	18 дней	Пн 20.04.26	Чт 14.05.26		
9	🚀	Выбор технологий и инструментальных средств	5 дней	Пн 20.04.26	Пт 24.04.26	7	Программист
10	🚀	Проектирование базы данных	3 дней	Пн 27.04.26	Ср 29.04.26	9	Программист
11	🚀	Проектирование серверной части приложения	10 дней	Чт 30.04.26	Ср 13.05.26	10	Программист
12	🚀	Проектирование программных средств заверше	0 дней	Чт 14.05.26	Чт 14.05.26	11	
13	🚀	➤ Подготовка механизма развертывания приложе	3 дней	Чт 21.05.26	Вт 26.05.26		
14	🚀	Выбор метода развертывания	1 день	Чт 21.05.26	Чт 21.05.26	12	Программист
15	🚀	Реализация механизма развертывания	2 дней	Пт 22.05.26	Пн 25.05.26	14	Программист
16	🚀	Подготовка механизма развертывания прилож	0 дней	Вт 26.05.26	Вт 26.05.26	15	
17	🚀	➤ Реализация программных средств	30 дней	Вт 26.05.26	Вт 07.07.26		
18	🚀	Реализация базы данных	5 дней	Вт 26.05.26	Пн 01.06.26	16	Программист
19	🚀	Реализация серверной части приложения	25 дней	Вт 02.06.26	Пн 06.07.26	18	Программист
20	🚀	Реализация программных средств завершена	0 дней	Вт 07.07.26	Вт 07.07.26	19	
21	🚀	➤ Тестирование программных средств	11 дней	Вт 21.07.26	Ср 05.08.26		
22	🚀	Функциональное тестирование	4 дней	Вт 21.07.26	Пт 24.07.26	20	Тестировщик
23	🚀	Модульное тестирование	7 дней	Пн 27.07.26	Вт 04.08.26	22	Программист
24	🚀	Тестирование завершено	0 дней	Ср 05.08.26	Ср 05.08.26	23	
25	🚀	➤ Завершение работы над проектом	8 дней	Ср 05.08.26	Пн 17.08.26		
26	🚀	Составление документации	7 дней	Ср 05.08.26	Чт 13.08.26	24	Программист
27	🚀	Сдача проекта	1 день	Пт 14.08.26	Пт 14.08.26	26	Программист
28	🚀	Работа над проектом завершена	0 дней	Пн 17.08.26	Пн 17.08.26	27	
29	🚀	Конец проекта	0 дней	Пн 17.08.26	Пн 17.08.26	28	

Рисунок 14 – Таблица задач

Для отображения длительности выполнения проекта, а также связей между задачами была составлена диаграмма Ганта [5], которая демонстрируется на рисунке 15.

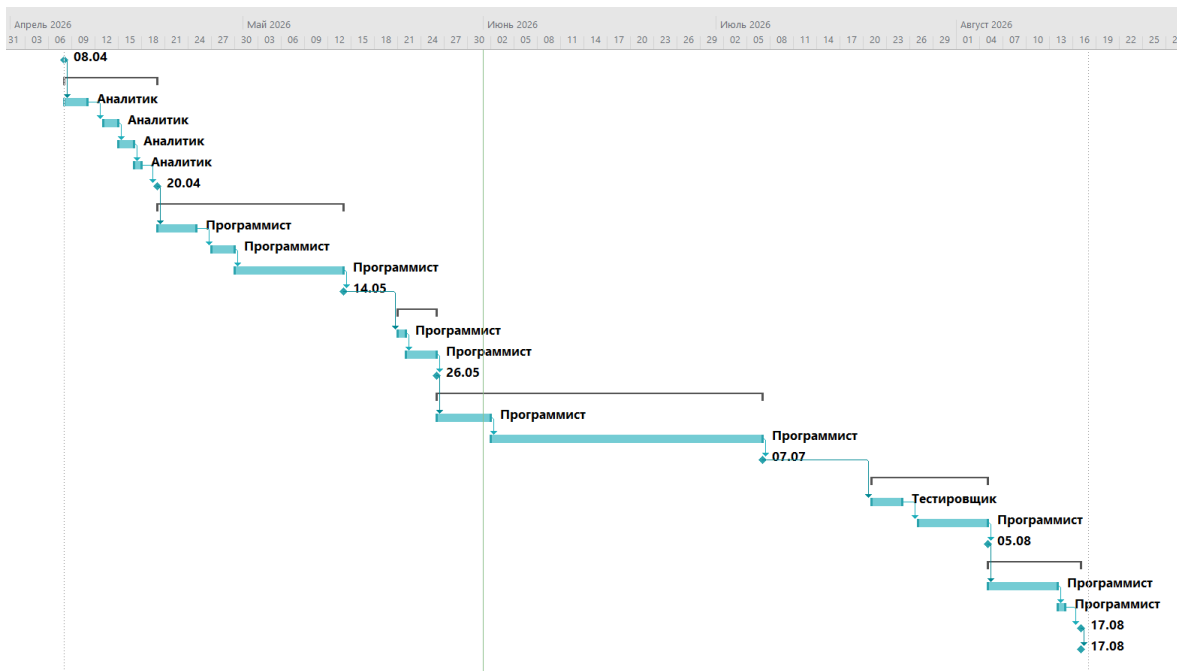


Рисунок 15 – Диаграмма Ганта

Таким образом, общая длительность проекта составляет 78 дней.

Используя встроенный отчет «Обзор затрат» из Microsoft Project был посчитан общий объем затрат для выполнения проекта. График затрат приведен на рисунке 16.

ОБЗОР ЗАТРАТ

СР 08.04.26 - ПН 17.08.26

ЗАТРАТЫ

266 400,00 ₽

ОСТАВШИЕСЯ ЗАТРАТЫ

266 400,00 ₽

% ЗАВЕРШЕНИЯ

0%

СОСТОЯНИЕ ЗАТРАТ

Состояние затрат для задач верхнего уровня.

Название	Фактически затраты	Оставшиеся затраты	Базовые затраты	Затраты	Отклонение по стоимости
Начало проекта	0,00 ₽	0,00 ₽	0,00 ₽	0,00 ₽	0,00 ₽
Анализ предметной области и требований	0,00 ₽	19 200,00 ₽	0,00 ₽	19 200,00 ₽	19 200,00 ₽
Проектирование программных средств	0,00 ₽	64 800,00 ₽	0,00 ₽	64 800,00 ₽	64 800,00 ₽
Подготовка механизма развертывания приложения	0,00 ₽	10 800,00 ₽	0,00 ₽	10 800,00 ₽	10 800,00 ₽
Реализация программных средств	0,00 ₽	108 000,00 ₽	0,00 ₽	108 000,00 ₽	108 000,00 ₽
Тестирование программных средств	0,00 ₽	34 800,00 ₽	0,00 ₽	34 800,00 ₽	34 800,00 ₽
Завершение работы над проектом	0,00 ₽	28 800,00 ₽	0,00 ₽	28 800,00 ₽	28 800,00 ₽
Конец проекта	0,00 ₽	0,00 ₽	0,00 ₽	0,00 ₽	0,00 ₽

ХОД ВЫПОЛНЕНИЯ И ЗАТРАТЫ

Ход выполнения в сравнении с затратами во времени. Превышение значения в строке "Совокупные затраты" над значением в строке "Процент завершения" указывает на возможное превышение бюджета.



СОСТОЯНИЕ ЗАТРАТ

Состояние затрат для всех задач верхнего уровня. Равны ли ваши базовые затраты нулю?

Попытка задания в качестве базового плана

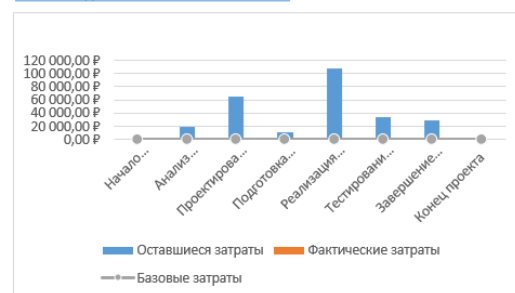


Рисунок 16 – Общие затраты на выполнение проекта

Итоговая сумма на выполнение проекта составляет 266 400.00 рублей.

По итогам проведенного планирования можно сделать вывод, что разработка программных средств распределения обращений клиентов между сотрудниками клиентской поддержки является экономически целесообразной.

2 Проектирование программных средств распределения обращений

Разработка архитектуры является одним из важнейших этапов проектирования системы.

Приложение будет состоять из клиентской и серверной частей, взаимодействие которых показано на уровне контейнера диаграммы C4, приведенной на рисунке 17.

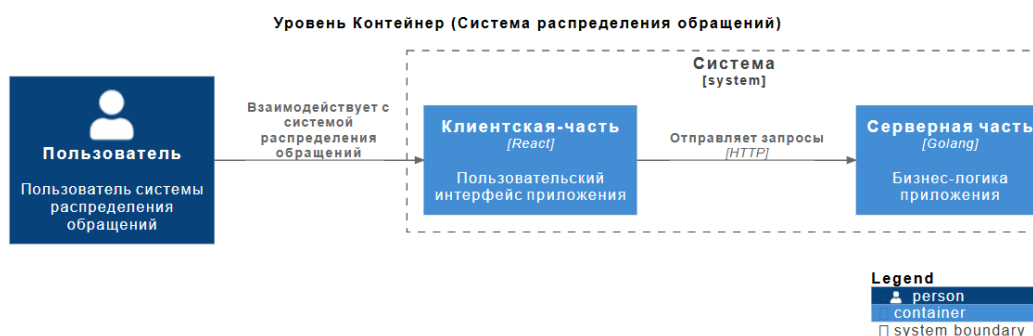


Рисунок 17 – Уровень контейнера системы распределения обращений

Таким образом, пользователь системы напрямую взаимодействует только с интерфейсом, который обращается к серверной части по протоколу HTTP для выполнения операций.

2.1 Выбор технологий и инструментальных средств

Серверная часть приложения будет реализована на языке Golang [6], который обладает широкой поддержкой сторонних библиотек для построения Web-приложений.

Для хранения данных будет использоваться реляционная база данных под управлением PostgreSQL [7], которая обеспечивает необходимый функционал транзакций, поддерживает разнообразное количество типов данных, а также обладает подробной документацией.

Процесс распределения созданных обращений должен работать в фоновом режиме. Для отправки асинхронных сообщений будет использоваться брокер

сообщений RabbitMQ [8], предоставляющий удобный интерфейс взаимодействия, а также легкое развертывание.

Также для обеспечения эффективности модуля распределения обращений будут использоваться асинхронные задачи, управление которыми будет производиться с применением библиотеки `async` [9]. Данная библиотека дает возможность глубоко настраивать обработку задач, например, поддерживает механизм приоритезации и повторного выполнения при ошибках.

Для настройки структуры базы данных будет использоваться механизм миграций, которые будут выполняться с помощью утилиты `goose`.

Клиентская часть системы должны быть реализована с помощью языка программирования TypeScript [10], который был выбран из-за поддержки статической типизации, что позволит снизить количество ошибок на этапе реализации.

Также в процессе реализации клиентской части будет использоваться библиотека React [11], которая обладает подробной документацией и широкими возможностями для построения пользовательских интерфейсов.

Для создания UML-диаграмм будет использоваться редактор Visual Paradigm. Для создания диаграмм C4 [12] будет использоваться PlantUML.

Для написания исходного кода будет применяться IDE Visual Studio Code, которая за счет своей обширной библиотеки плагинов позволяет реализовывать и серверную, и клиентскую часть.

2.2 Проектирование структур данных

2.2.1 Концептуальная модель базы данных

Для определения сущностей системы и связей между ними была разработана концептуальная модель базы данных, которая приведена на рисунке 18.

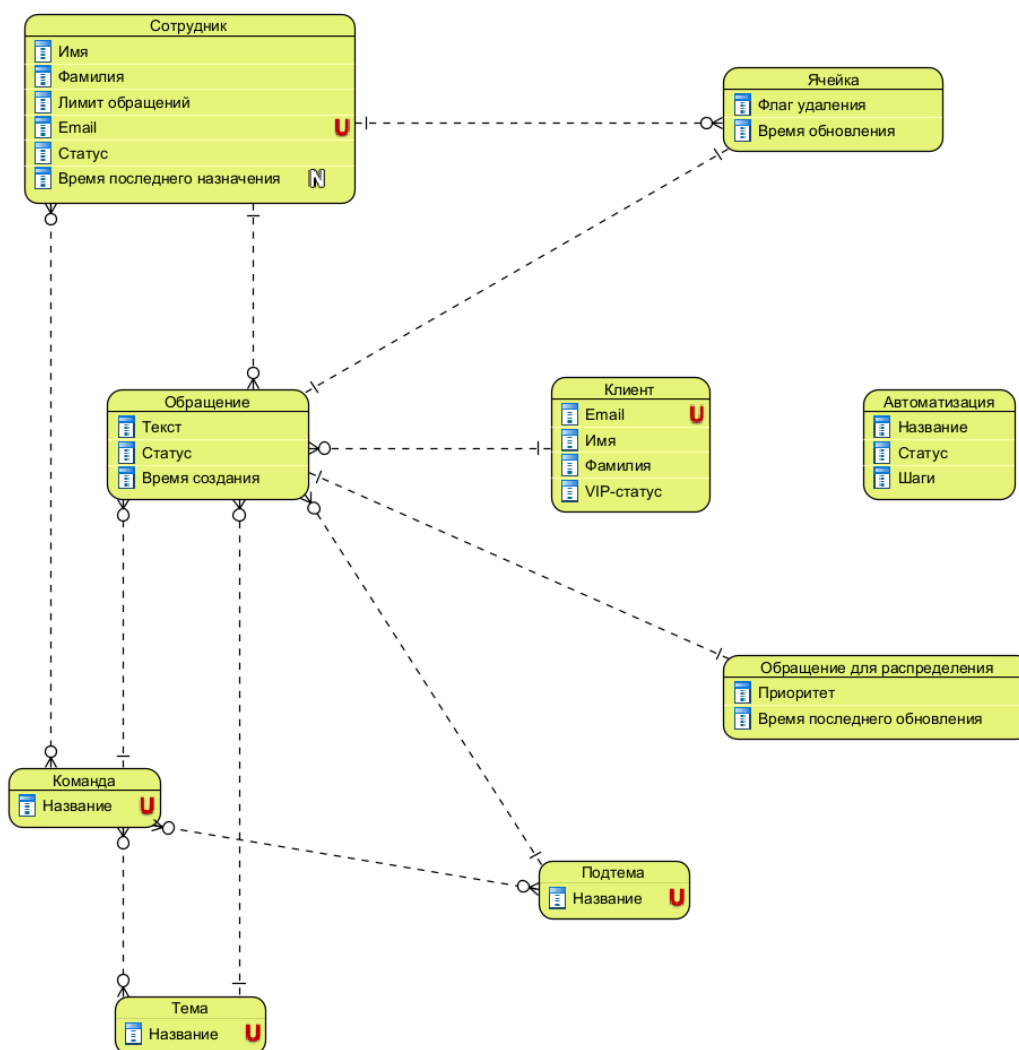


Рисунок 18 – Концептуальная модель базы данных

Данная модель демонстрирует сущности системы и их состав без привязки к типам данных. Также на диаграмме можно увидеть связи между сущностями.

2.2.2 Логическая модель базы данных

В ходе проектирования была разработана логическая модель базы данных. В процессе построения логической модели данных были определены типы данных для колонок таблицы, расставлены первичные и внешние ключи, а также разработаны таблицы для реализации отношения «многие ко многим» [13].

Итоговая логическая модель базы данных представлена на рисунке 19.

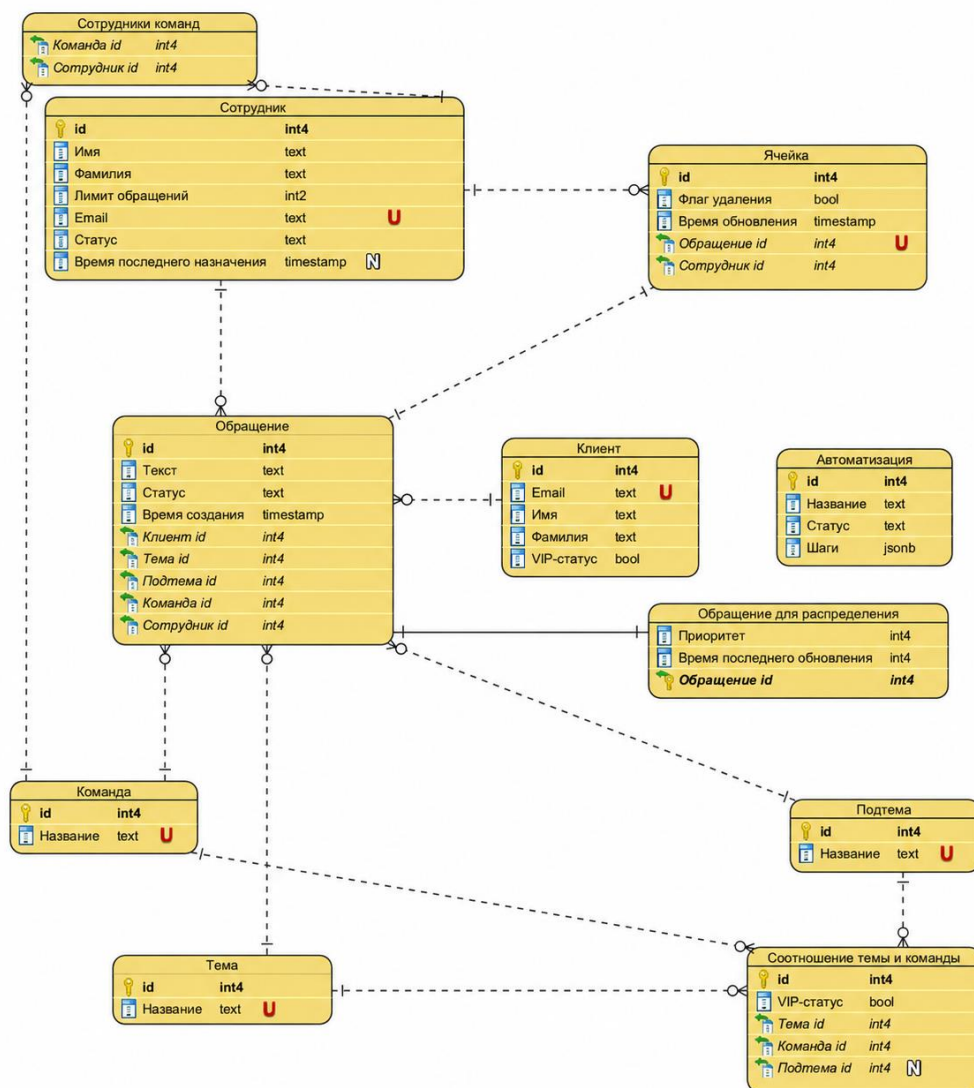


Рисунок 19 – Логическая модель базы данных

Ниже в таблице 2 представлено назначение каждой таблицы.

Таблица 2 – Описание таблиц базы данных

№	Название	Описание
1	Сотрудник	Содержит данные сотрудников, которые занимаются обработкой обращений
2	Клиент	Содержит данные клиентов, от лица которых создаются обращения
3	Обращение	Содержит данные обращений, которые автоматически распределяются между сотрудниками
4	Тема	Содержит доступные темы для создания обращения
5	Подтема	Содержит доступные подтемы для создания обращения

Продолжение таблицы 2

№	Название	Описание
6	Команда	Содержит доступные команды сотрудников
7	Соотношение темы и команды	Содержит правила выбора команды сотрудников на основе темы, подтемы, VIP-статуса клиента
8	Сотрудники команд	Используется для представления связи «многие ко многим». Хранит информацию о том, в какие команды входят сотрудники
9	Обращение для распределения	Содержит список обращений, которые ждут распределения системой
10	Ячейка	Содержит информацию о ячейках сотрудников. Количество ячеек каждого сотрудника определяется его лимитом обращений
11	Автоматизация	Хранит данные об автоматизациях

Таким образом, были определены сущности разрабатываемой системы, а также смоделированы связи между ними.

2.3 Проектирование серверной части приложения

2.3.1 Проектирование алгоритма распределения

На основании ранее определенных требований был спроектирован алгоритм работы серверной части приложения. Серверная часть приложения логически разделена на две части.

Первая – компонент распределения обращений, который отвечает за нахождение оптимального сотрудника для обработки пользовательского обращения.

Вторая – компонент управления сущностями, который отвечает за создание, чтение, обновление удаление объектов системы. Этот компонент ответственен за моделирование данных, которыми будет оперировать система распределения.

Процесс распределения обращения состоит из трех последовательных этапов.

На первом этапе исходя из темы, подтемы, а также VIP-статуса клиента, создавшего обращение, осуществляется поиск команды сотрудников, которая будет обрабатывать обращение. Этот процесс представлен более детально на диаграмме деятельности, приведенной на рисунке 20.

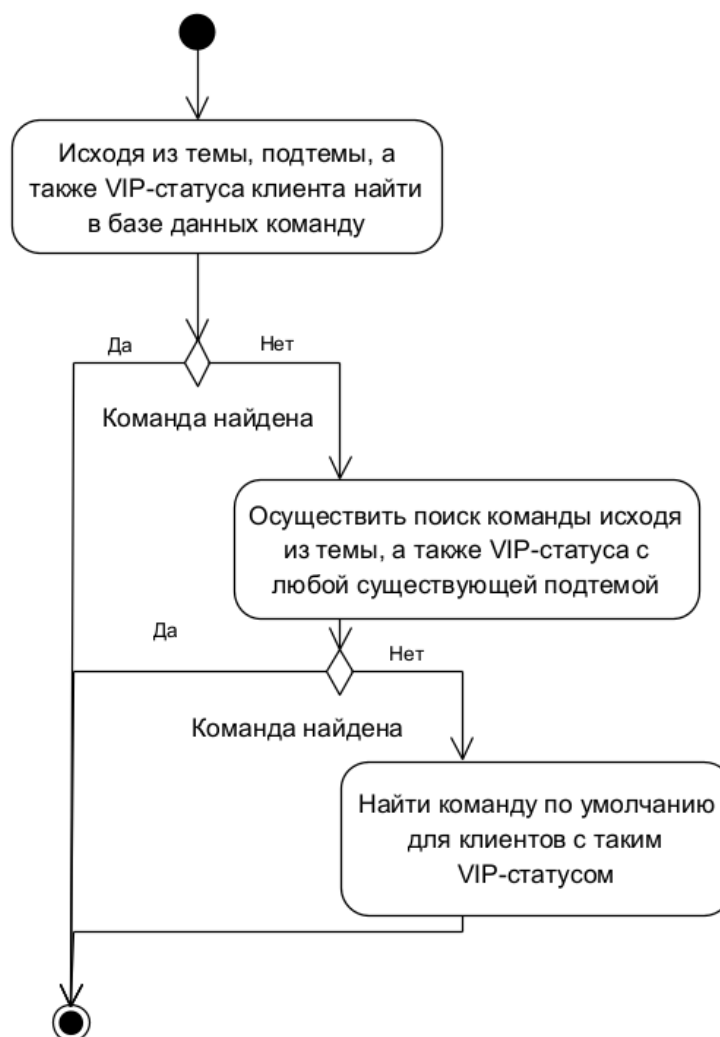


Рисунок 20 – Описание процесса определения команды обращения

На втором этапе для созданного обращения запускаются пользовательские автоматизации. В реальной системе подразумевается, что автоматизации настраивают сами сотрудники технической поддержки.

Автоматизация состоит из последовательных шагов. Каждый шаг может представлять собой либо условие, либо действие по изменению команды обращения.

Поддерживаются следующие условия, проверяющие данные созданного обращения:

- равенство почты клиента указанному значению;
- проверка, что создание обращения произошло в указанный временной промежуток;
- содержание определенных ключевых слов в тексте обращения;
- равенство темы обращения указанному значению.

Условия могут быть составными, тогда каждая часть проверяется отдельно, после чего результаты объединяются с использованием логического «и» или «или».

На третьем этапе из выбранной команды выбирается сотрудник, который и будет обрабатывать обращения.

Данный этап является последним в цепочке распределения, для ускорения работы системы он должен работать в фоновом режиме, запускаться при получении асинхронного сообщения.

Этот этап должен решать три задачи:

- необходимо обработать асинхронное сообщение, рассчитать приоритет обращения и переместить его в очередь на распределение;
- нужно получать обращения из очереди на распределение, для каждого из них требуется находить оптимального сотрудника;
- необходимо обновлять состояние обращения, назначая на него найденного сотрудника.

Для повышения эффективности работы системы все три этих компонента будут выполняться асинхронно.

На рисунке 21 приведен алгоритм выбора оптимального сотрудника для обработки обращения, показанный в виде диаграммы деятельности.

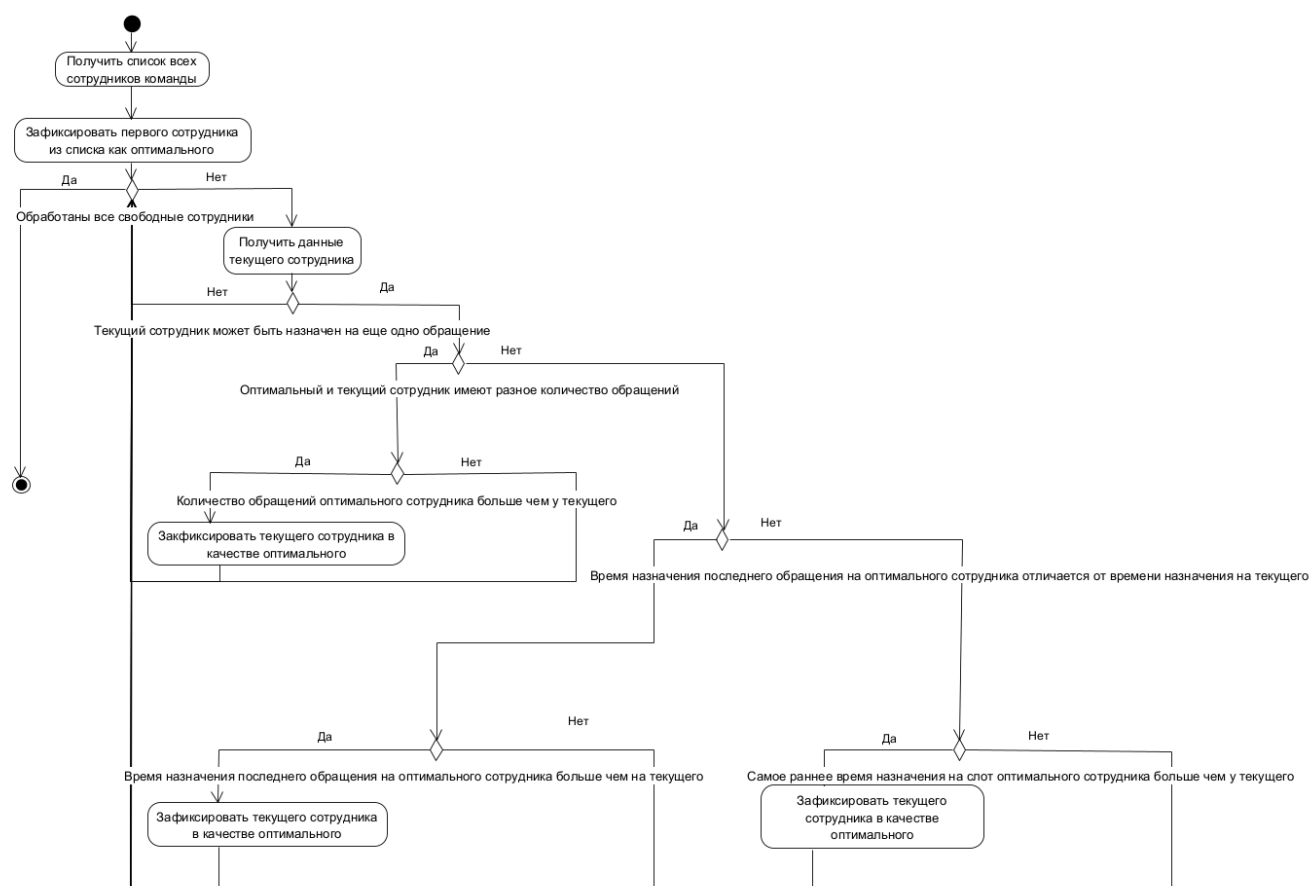


Рисунок 21 – Описание процесса выбора оптимального сотрудника

Таким образом, после обхода всего списка доступных сотрудников будет найден оптимальный для назначения на обращение.

2.3.2 Проектирование механизма работы лимитов сотрудников

Как ранее описывалось, каждый сотрудник системы должен иметь лимит одновременно назначенных обращений. Для сотрудников будут храниться ячейки, на которые могут быть назначены обращения, притом количество этих ячеек совпадает со значением лимита. Каждая ячейка может быть либо пустой, либо хранить идентификатор обращения, которое ее занимает.

Система должна корректно обрабатывать ситуацию изменения лимитов сотрудников. Описание этого процесса приведено на диаграмме деятельности на рисунке 22.

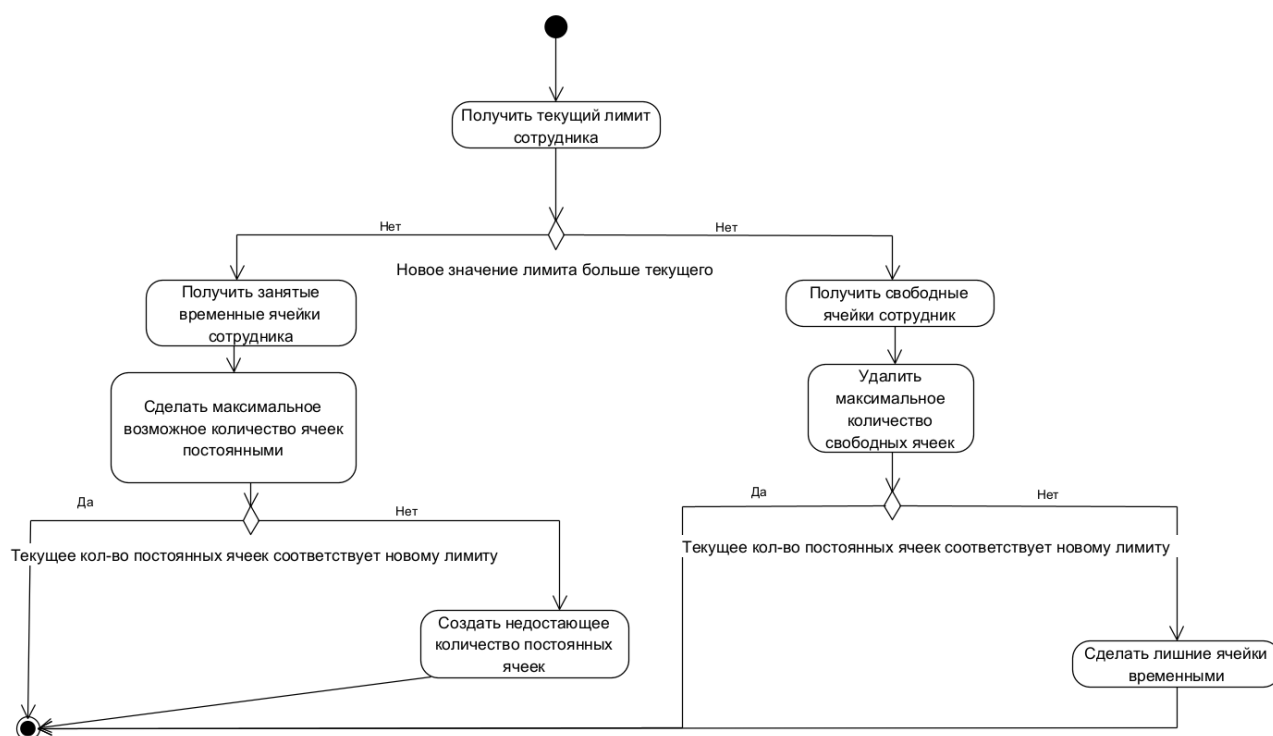


Рисунок 22 – Описание процесса изменения лимита сотрудника

Таким образом ячейки будут разделены на 2 группы: постоянные и временные.

Отличие временных ячеек от постоянных заключается в том, что при закрытии обращения постоянная ячейка освобождается, а временная удаляется. Такая система позволит корректно обрабатывать изменение лимита одновременно назначенных обращений сотрудника.

2.3.3 Проектирование архитектуры серверной части приложения

На основе ранее определенных требований в этом проекте была выбрана архитектура модульного монолита.

Данная архитектура заключается в разделении системы на слабо зависимые модули, работающие вместе в одном исполняемом процессе. Этот архитектурный стиль позволяет на первоначальных этапах жизни системы хранить весь исходный код в одном репозитории, что повышает скорость разработки. При этом данная архитектура в дальнейшем позволит преобразовать систему из монолитной в распределенную за счет низкой связности модулей.

Ниже на рисунке 23 приведен уровень компонента диаграммы С4, который описывает внутреннее взаимодействие модулей.

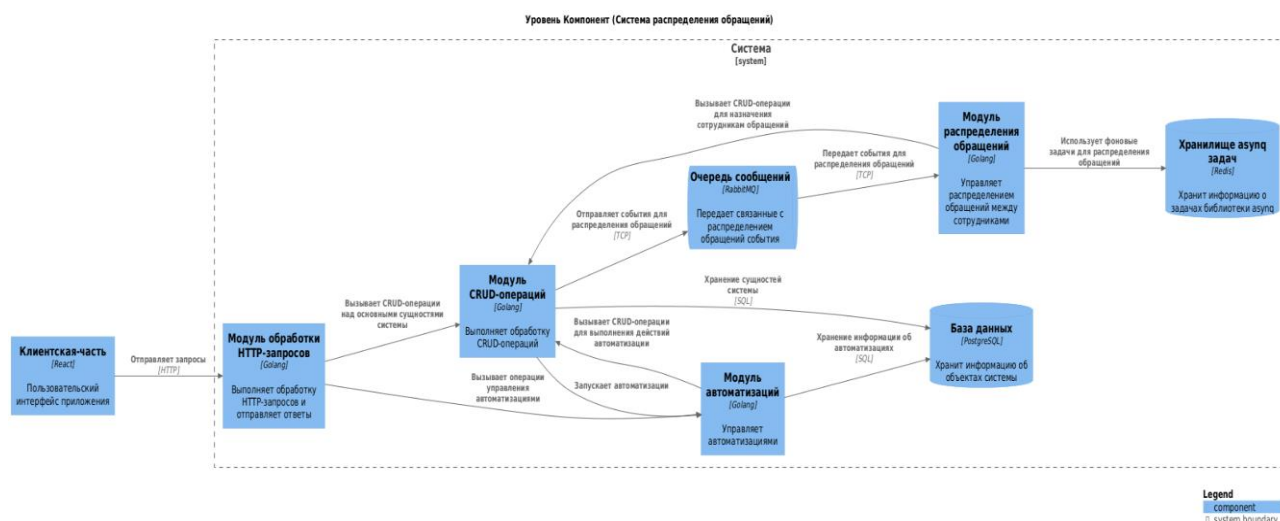


Рисунок 23 – Уровень компонента диаграммы С4 системы распределения обращений

Серверная часть разбита на 4 модуля, подробное описание которых приведено далее.

Модуль обработки HTTP-запросов выполняет обработку внешних HTTP-запросов, которые отправляет клиентская часть. Далее этот модуль вызывает компоненты, которые выполняют саму логику приложения.

Модуль CRUD-операций выполняет операции создания, чтения, обновления, удаления основных сущностей системы. В ходе своей работы этот модуль может вызывать исполнение автоматизаций, а также отправлять асинхронные сообщения для модуля распределения обращений, который описывается далее.

Модуль распределения обращений отвечает за процесс назначения на обращение оптимального сотрудника. Как ранее описывалось, алгоритм распределения обращений будет работать в асинхронно. Для реализации такого подхода будет использоваться библиотека `asynq`, которая позволяет создавать задачи для их отложенного выполнения. Информация об этих задачах хранится

в Redis. Этот модуль получает необходимую для распределения информацию о сущностях системы с помощью модуля CRUD-операций.

Модуль автоматизаций реализует исполнение автоматизация, а также операции их создания, просмотра, удаления, редактирования. Для изменения состояния существующих обращений этот модуль обращается к модулю CRUD-операций.

Таким образом, была выбрана архитектура модульного монолита, которая позволяет быстро реализовывать логику, при этом обладает возможностями для последующего перехода к более сложной распределенной архитектуре.

2.4 Проектирование клиентской части приложения

Клиентская часть приложения должна обеспечивать пользователю единообразный, простой в использовании интерфейс для взаимодействия с системой, который позволяет использовать весь функционал системы.

На основе определенных ранее требований был разработан состав web-страниц приложения. Ниже приведен список страниц приложения:

- страница «Сотрудники», предназначенная для отображения списка сотрудников, а также поддерживающая действия добавления, редактирования и удаления в систему сотрудников;
- страница «Клиенты», отвечающая за отображение списка клиентов, а также поддерживающая действия добавления, редактирования и удаления клиентов;
- страница «Темы», отвечающая за отображение существующих тем, а также добавление, редактирование и удаление тем обращений;
- страница «Подтемы», реализующая отображение существующих подтем обращений, редактирование, удаление подтем;
- страница «Обращения», предназначенная для отображения списка обращений, статуса их распределения, а также добавления, редактирования и удаления обращений;
- страница «Ячейки», реализующая отображение ячеек сотрудников;

- страница «Команды», ответственная за отображение списка команд сотрудников, просмотр и редактирование существующих команд, а также добавление новых;
- страница «Автоматизации», предназначенная для просмотра, модификации, удаления пользовательских автоматизаций.

В каждом из описанных выше разделов в левой части экрана будет находиться меню навигации между страницами, благодаря которому пользователь получит интуитивно понятный способ перемещения по разделам приложения.

Таким образом, в рамках моделирования клиентской части приложения были определены web-страницы приложения, которые будут реализованы, а также описан функционал каждой страницы.

3 Реализация программных средств распределения обращений

3.1 Реализация базы данных

В ходе реализации приложения на основе ранее созданных концептуальной и логической моделей базы данных была создана физическая модель базы данных. Ниже на рисунке 24 представлена физическая модель базы данных.

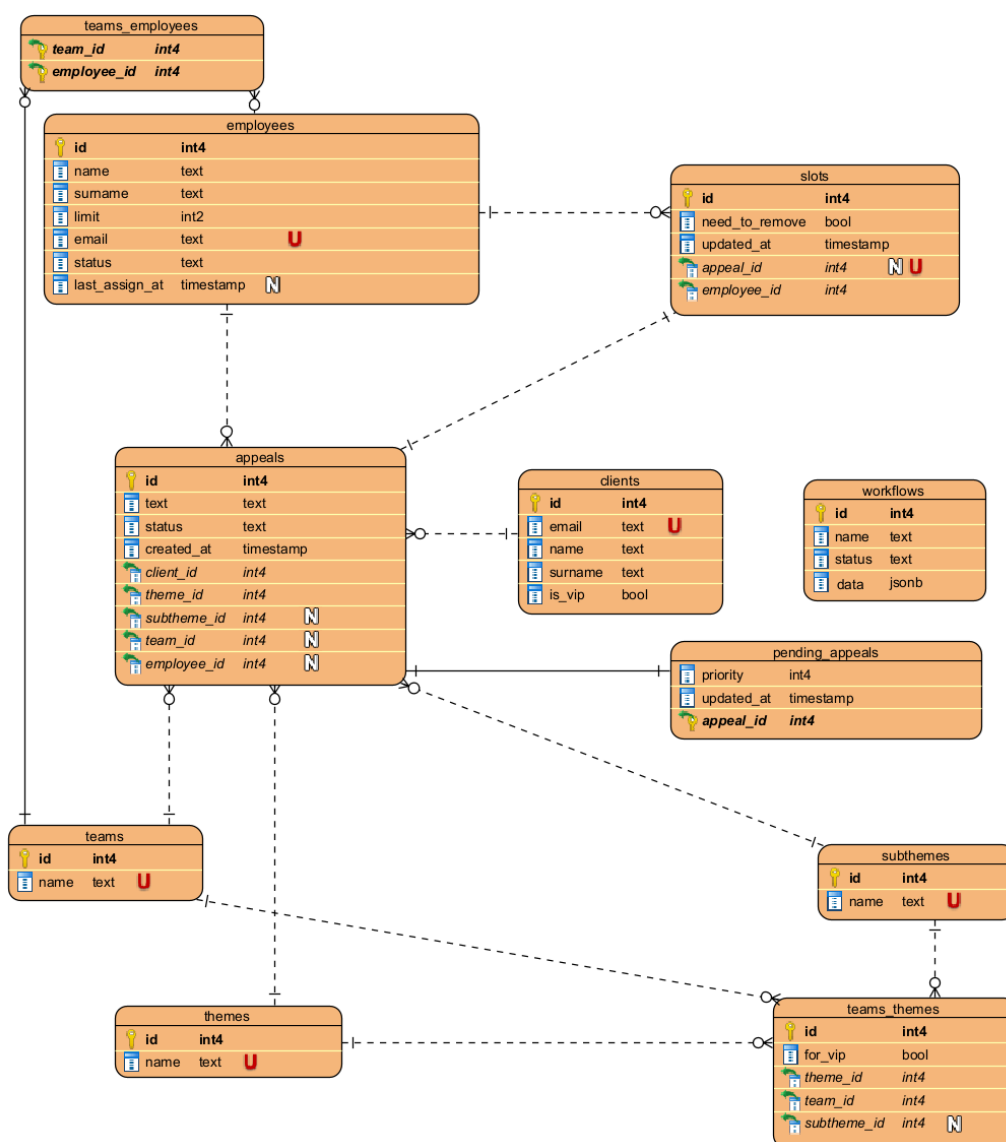


Рисунок 24 – Физическая модель базы данных

Далее в таблице 3 представлено описание сущности appeals.

Таблица 3 – Описание сущности appeals

№	Название	Описание	Тип данных	Возможность быть NULL	Роль
1	id	Уникальный идентификатор	int4	Нет	PK
2	text	Текст обращения	text	Нет	–
3	status	Статус обращения	text	Нет	–
4	created_at	Время создания обращения	timestamp	Нет	–
5	client_id	Идентификатор клиента, создавшего обращение	int4	Нет	FK
6	theme_id	Идентификатор темы обращения	int4	Нет	FK
7	subtheme_id	Идентификатор подтемы обращения	int4	Да	FK
8	team_id	Идентификатор команды сотрудников, которая занимается обработкой обращения	int4	Да	FK
9	employee_id	Идентификатор сотрудника, который занимается обработкой обращения	int4	Да	FK

Далее в таблице 4 расположено описание сущности clients, которая несет информацию о клиентах системы.

Таблица 4 – Описание сущности clients

№	Название	Описание	Тип данных	Возможность быть NULL	Роль
1	id	Уникальный идентификатор	int4	Нет	РК
2	email	Адрес электронной почты	text	Нет	—
3	name	Имя	text	Нет	—
4	surname	Фамилия	text	Нет	—
5	is_vip	Флаг, является ли клиент VIP	bool	Нет	—

В таблице clients столбец email обладает свойством unique. Ниже в таблице 5 находится описание сущности employees.

Таблица 5 – Описание сущности employees

№	Название	Описание	Тип данных	Возможность быть NULL	Роль
1	id	Уникальный идентификатор	int4	Нет	РК
2	name	Имя	text	Нет	—
3	surname	Фамилия	text	Нет	—
4	limit	Лимит обращения, который может одновременно обрабатывать сотрудник	int2	Нет	—
5	email	Адрес электронной почты	text	Нет	—
6	status	Статус сотрудника	text	Нет	—

Продолжение таблицы 5

№	Название	Описание	Тип данных	Возможность быть NULL	Роль
7	last_assign_at	Время последнего назначения обращения на сотрудника	timestamp	Да	—

В таблице employees столбец email обладает свойством unique. Далее в таблице 6 находится описание сущности pending_appeals.

Таблица 6 – Описание сущности pending_appeals

№	Название	Описание	Тип данных	Возможность быть NULL	Роль
1	priority	Приоритет обращения, используемый для распределения	int4	Нет	—
2	updated_at	Время обновления записи в таблице	timestamp	Нет	—
3	appeal_id	Идентификатор обращения	int4	Нет	PK, FK

В таблице pending_appeals appeal_id является как первичным, так и внешним ключом. Колонка ссылается на таблицу appeals, поэтому она – внешний ключ. Для обеспечения связи «один к одному» при условии, что колонка не может иметь значение NULL данная колонка также является и первичным ключом. Ниже в таблице 7 находится описание сущности slots.

Таблица 7 – Описание сущности slots

№	Название	Описание	Тип данных	Возможность быть NULL	Роль
1	id	Уникальный идентификатор	int4	Нет	PK

Продолжение таблицы 7

№	Название	Описание	Тип данных	Возможность быть NULL	Роль
2	need_to_remove	Флаг, что ячейку необходимо удалить при освобождении	bool	Нет	—
3	updated_at	Время последнего обновления ячейки	timestamp	Нет	—
4	appeal_id	Идентификатор обращения	int4	Да	FK
5	employee_id	Идентификатор сотрудника	int4	Нет	FK

В таблице slots значение appeal_id также обладает атрибутом unique. Далее в таблице 8 расположено описание сущности subthemes.

Таблица 8 – Описание сущности subthemes

№	Название	Описание	Тип данных	Возможность быть NULL	Роль
1	id	Уникальный идентификатор	int4	Нет	PK
2	name	Название подтемы	text	Нет	—

В таблице subthemes name также обладает атрибутом unique. Далее в таблице 9 указано описание сущности teams.

Таблица 9 – Описание сущности teams

№	Название	Описание	Тип данных	Возможность быть NULL	Роль
1	id	Уникальный идентификатор	int4	Нет	PK
2	name	Название команды	text	Нет	—

В таблице teams name также обладает атрибутом unique. Ниже в таблице 10 находится описание сущности themes.

Таблица 10 – Описание сущности themes

№	Название	Описание	Тип данных	Возможность быть NULL	Роль
1	id	Уникальный идентификатор	int4	Нет	PK
2	name	Название темы	text	Нет	—

Далее в таблице 11 находится описание сущности teams_employees.

Таблица 11 – Описание сущности teams_employees

№	Название	Описание	Тип данных	Возможность быть NULL	Роль
1	team_id	Идентификатор команды	int4	Нет	Часть PK
2	employee_id	Идентификатор сотрудника	int4	Нет	Часть PK

В таблице teams_employees team_id, employee_id вместе образует составной первичный ключ. Ниже в таблице 12 находится описание сущности teams_themes.

Таблица 12 – Описание сущности teams_themes

№	Название	Описание	Тип данных	Возможность быть NULL	Роль
1	id	Уникальный идентификатор	int4	Нет	PK
2	for_vip	Флаг VIP-статуса	bool	Нет	—
3	theme_id	Идентификатор темы	int4	Нет	FK
4	team_id	Идентификатор команды	int4	Нет	FK

Продолжение таблицы 12

№	Название	Описание	Тип данных	Возможность быть NULL	Роль
5	subtheme_id	Идентификатор подтемы	int4	Да	FK

В таблице teams_themes колонки theme_id, team_id, subtheme_id вместе образуют составное уникальное значение. Так как subtheme_id может иметь NULL значение, составное уникальное значение создано с опцией NULLS NOT DISTINCT [14], которая позволяет рассматривать два NULL значения, как одинаковые. Ниже в таблице 13 представлено описание сущности workflows.

Таблица 13 – Описание сущности workflows

№	Название	Описание	Тип данных	Возможность быть NULL	Роль
1	id	Уникальный идентификатор	int4	Нет	PK
2	name	Название автоматизации	text	Нет	—
3	status	Статус автоматизации	text	Нет	—
4	data	Структура автоматизации	jsonb	Нет	—

В таблице workflows поле data хранится в формате JSONB, так как workflow в приложении представляет собой иерархическую структуру, которая читается и записывается целиком как единый документ.

В логике приложения не требуется доступ к отдельным элементам data, поэтому хранение единым документом уменьшает сложность программного кода. Нормализация этой структуры потребовала бы введения дополнительных таблиц и более сложных запросов без существенной выгоды для текущего сценария.

Таким образом была разработана физическая структура базы данных. Добавление новых сущностей производилось с помощью механизма миграций, которые автоматически выполняются перед запуском приложения. Исходный код миграций приведен в приложении Б.

3.2 Реализация серверной части

На основе проведенного ранее проектирования была разработана серверная часть приложения. Исходный код серверной части приведен в приложении В.

Серверная часть содержит всю логику распределения обращения, отвечает за взаимодействие с базой данных и клиентской частью, с которой взаимодействие происходит по HTTP API.

Ниже в таблице 14 демонстрируются доступные HTTP-запросы.

Таблица 14 – Поддерживаемые HTTP-запросы

№	URL	HTTP-метод	Описание
1	/health	GET	Возвращает ответ с HTTP-кодом 200, что говорит, о том, что приложение готово к работе. Используется при разворачивании системы.
2	/employees	GET	Получение данных всех сотрудников.
3	/employees	POST	Создание сотрудника.
4	/employees/{id}	GET	Получение данных определенного сотрудника.
5	/employees/{id}	PUT	Обновление данных сотрудника.
6	/employees/{id}	DELETE	Удаление данных сотрудника.
7	/slots	GET	Получение данных всех ячеек сотрудников.
8	/slots/{id}	DELETE	Удаление данных ячейки.
9	/appeals	GET	Получение данных всех обращений.
10	/appeals	POST	Добавление нового обращения.
11	/appeals/{id}/close	POST	Закрытие обращения.
12	/appeals/{id}	GET	Получение данных определенного обращения.

Продолжение таблицы 14

№	URL	HTTP-метод	Описание
13	/appeals/{id}	PUT	Обновление данных обращения.
14	/appeals/{id}	DELETE	Удаление данных обращения.
15	/subthemes	GET	Получение данных всех подтем.
16	/subthemes	POST	Добавление новой подтемы.
17	/subthemes/{id}	GET	Получение данных определенной подтемы.
18	/subthemes/{id}	PUT	Обновление данных подтемы.
19	/subthemes/{id}	DELETE	Удаление данных подтемы.
20	/clients	GET	Получение данных всех клиентов.
21	/clients	POST	Добавление нового клиента.
22	/clients/{id}	GET	Получение данных определенного клиента.
23	/clients/{id}	PUT	Обновление данных клиента.
24	/clients/{id}	DELETE	Удаление данных клиента.
25	/clients/emails	GET	Получение электронных адресов всех клиентов.
26	/themes	GET	Получение всех данных всех тем.
27	/themes	POST	Добавление новой темы.
28	/themes/{id}	GET	Получение данных определенной темы.
29	/themes/{id}	PUT	Обновление данных темы.
30	/themes/{id}	DELETE	Удаление данных темы.
31	/teams	GET	Получение данных всех команд сотрудников.
32	/teams	POST	Добавление команды сотрудников.
33	/teams/{id}	GET	Получение данных определенной команды сотрудников.
34	/teams/{id}	PUT	Обновление данных команды сотрудников.
35	/teams/{id}	DELETE	Удаление данных команды сотрудников.
36	/workflows	GET	Получение данных всех автоматизаций.
37	/workflows	POST	Добавление новой автоматизации.
38	/workflows/{id}	GET	Получение данных определенной автоматизации.
39	/workflows/{id}	PUT	Обновление данных автоматизации.
40	/workflows/{id}	DELETE	Удаление автоматизации.

Как описывалось ранее, система поддерживает распределение клиентских обращений между сотрудниками с учетом приоритетов. В процессе реализации была выбрана следующая формула вычисления приоритета:

$$p=i \times 100+a \times 10,$$

где i – важность обращение, если клиент является VIP равно 2, иначе – 1;
 a – время жизни обращения в минутах.

Таким образом, приоритет распределения зависит от VIP-статуса создавшего его клиента, а также времени жизни обращения. При распределении из очереди берутся выбираются обращения с максимальным приоритетом.

Точное количество обращений с максимальным приоритетом, которые выбираются для распределения в каждой итерации зафиксирована в виде константы, которую в дальнейшем можно будет изменить для более тонкой настройки распределения.

3.3 Реализация пользовательского интерфейса приложения

На основе смоделированных ранее web-страниц был разработан пользовательский интерфейс. Далее на рисунке 25 находится страница просмотра списков сотрудников.

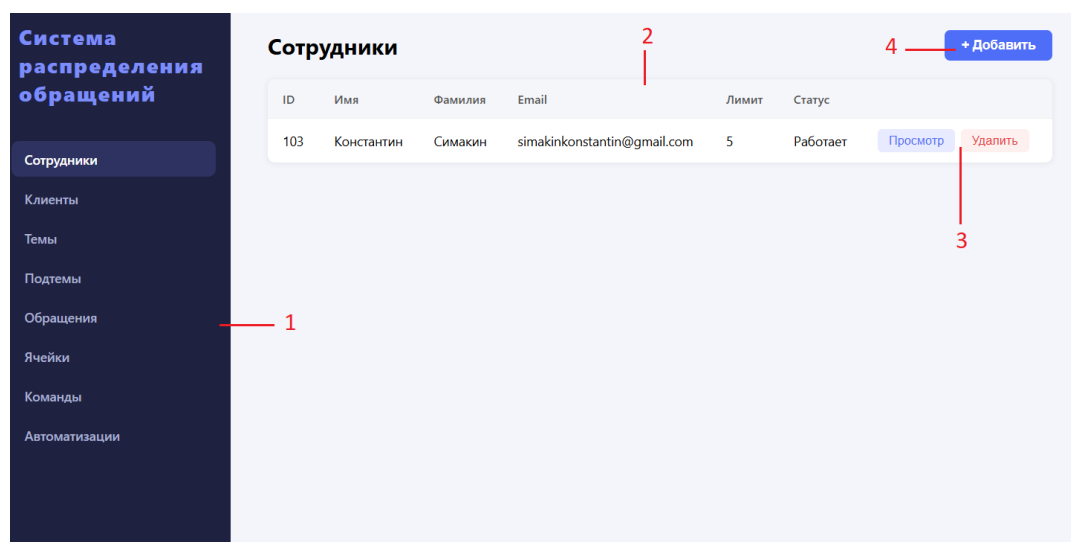


Рисунок 25 – Страница просмотра списков сотрудников

На странице просмотра списка сотрудников есть следующие элементы:

- 1 – меню перехода на другие страницы;
- 2 – список с информацией о существующих сотрудниках в системе;
- 3 – кнопки для перехода к окну редактирования или удаления сотрудника;
- 4 – кнопка добавления сотрудника.

Ниже на рисунке 26 находится страница создания сотрудника.

Сотрудники

ID	Имя	Фамилия	Email	Лимит	Статус	
103	Константин	Симакин	simakinkonstantin@gmail.com	3	Работает	Просмотр Удалить

Новый сотрудник

Имя 1

Фамилия 2

Email 3

Лимит 4 5

Команды 5

☐ (1) Не распределенные ☐ (2) Не распределенные VIP

☐ (3) Авиа VIP

Статус 6 Работает

7 Сохранить

Рисунок 26 – Страница создания сотрудника

На странице создания сотрудника есть следующие элементы:

- 1 – окно ввода имени сотрудника;
- 2 – окно ввода фамилии сотрудника;
- 3 – окно ввода электронного адреса сотрудника;
- 4 – окно ввода максимального количества одновременно назначенных на сотрудника обращений;
- 5 – окно ввода команд сотрудника;
- 6 – окно выбора статуса сотрудника;
- 7 – кнопка сохранения данных сотрудника.

Ниже на рисунке 27 находится страница редактирования существующего сотрудника.

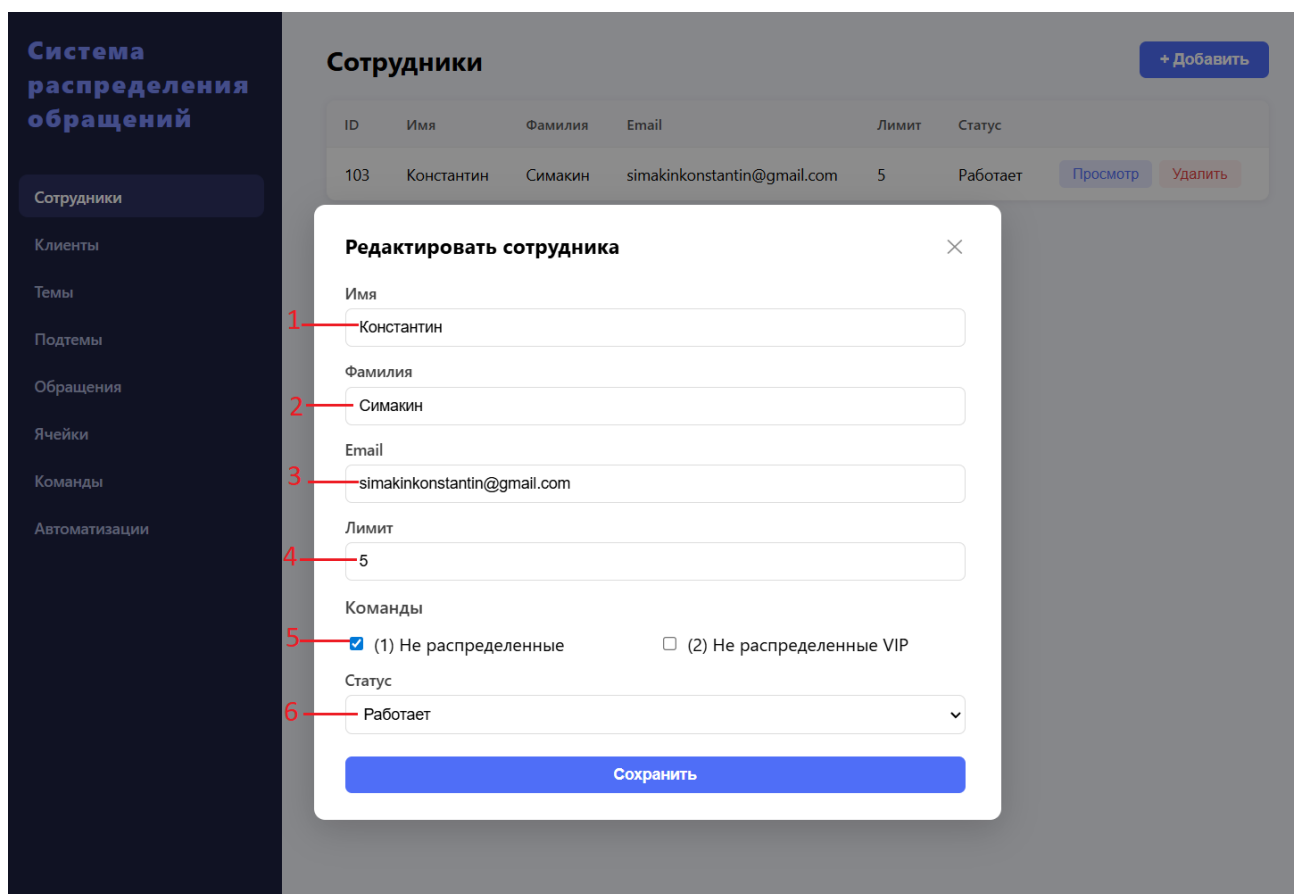


Рисунок 27 – Страница редактирования сотрудника

На странице редактирования сотрудников есть следующие элементы:

- 1 – окно ввода нового имени сотрудника;
- 2 – окно ввода новой фамилии сотрудника;
- 3 – окно ввода нового электронного адреса сотрудника;
- 4 – окно ввода нового максимального количества одновременно назначенных на сотрудника обращений;
- 5 – окно отображения команд, куда входит сотрудник;
- 6 – окно выбора статуса сотрудника;
- 7 – кнопка сохранения изменения.

Далее на рисунке 28 расположена страница просмотра списка существующих клиентов.

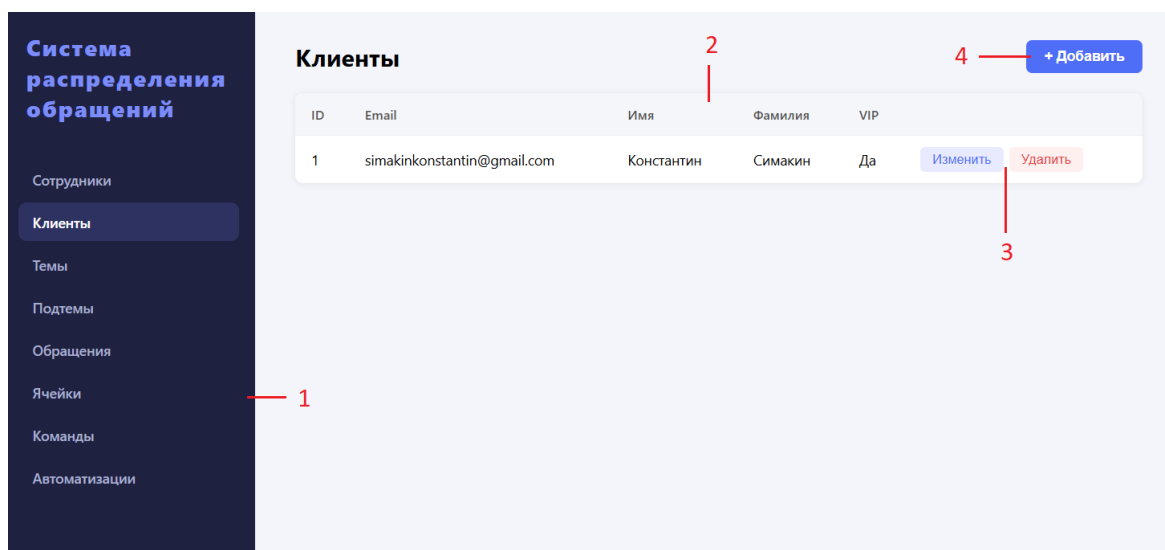


Рисунок 28 – Страница просмотра списка клиентов

На странице просмотра списка клиентов есть следующие элементы:

- 1 – меню перехода на другие страницы;
- 2 – список с информацией о существующих в системе клиентах;
- 3 – кнопки для перехода к окну редактирования или удаления клиента;
- 4 – кнопка добавления клиента.

Ниже на рисунке 29 находится страница создания клиента.

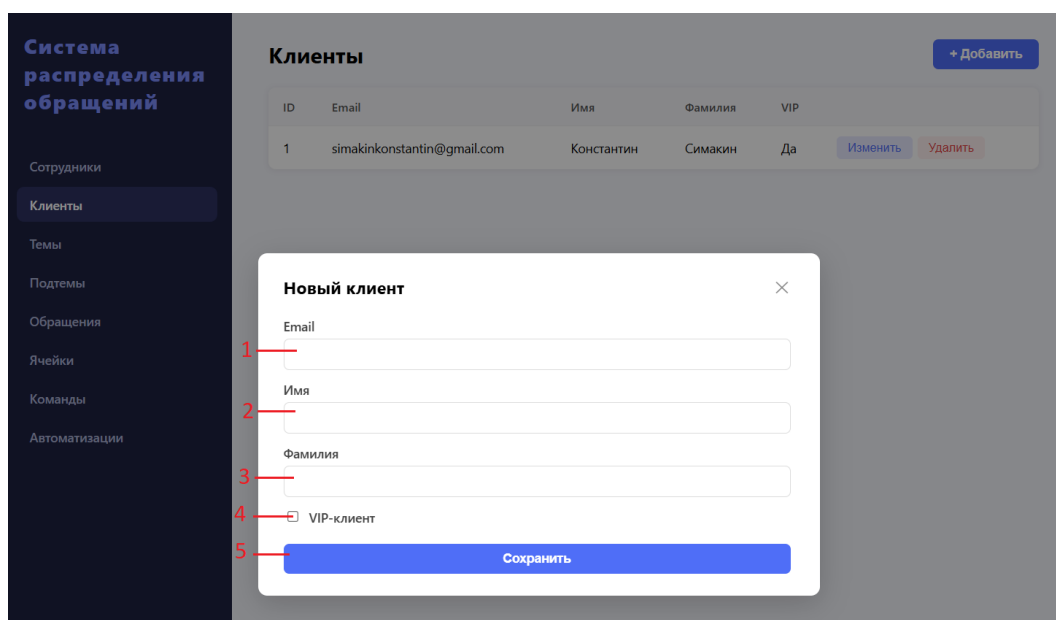


Рисунок 29 – Страница создания клиента

На странице создания сотрудника есть следующие элементы:

- 1 – поле ввода адреса электронной почты;
- 2 – поле ввода имени;
- 3 – поле ввода фамилии;
- 4 – флаг VIP-статуса клиента;
- 5 – кнопка подтверждения создания клиента.

Далее на рисунке 30 находится страница редактирования данных клиента.

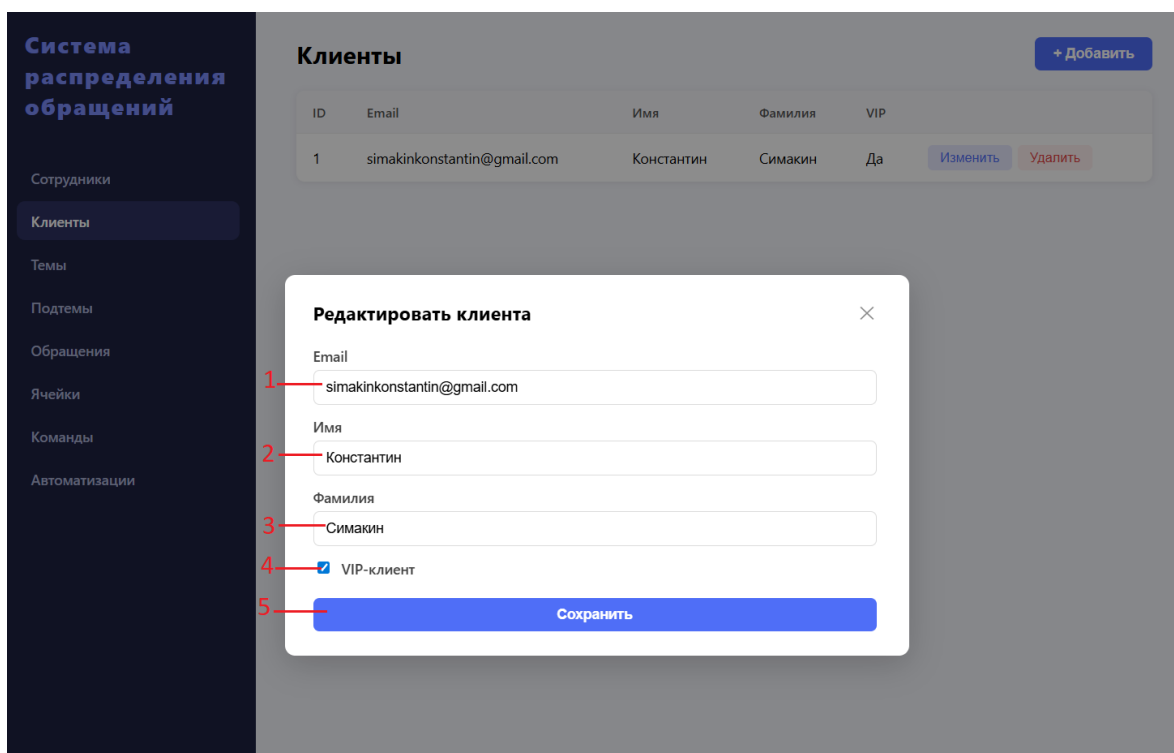


Рисунок 30 – Страница редактирования данных клиента

На странице редактирования данных клиента присутствуют следующие элементы:

- 1 – поле ввода нового значения адреса электронной почты;
- 2 – поле ввода нового значения имени;
- 3 – поле ввода нового значения фамилии;
- 4 – новое значение флага VIP-статуса клиента;
- 5 – кнопка сохранения изменений.

Ниже на рисунке 31 находится страница со списком доступных тем обращений.

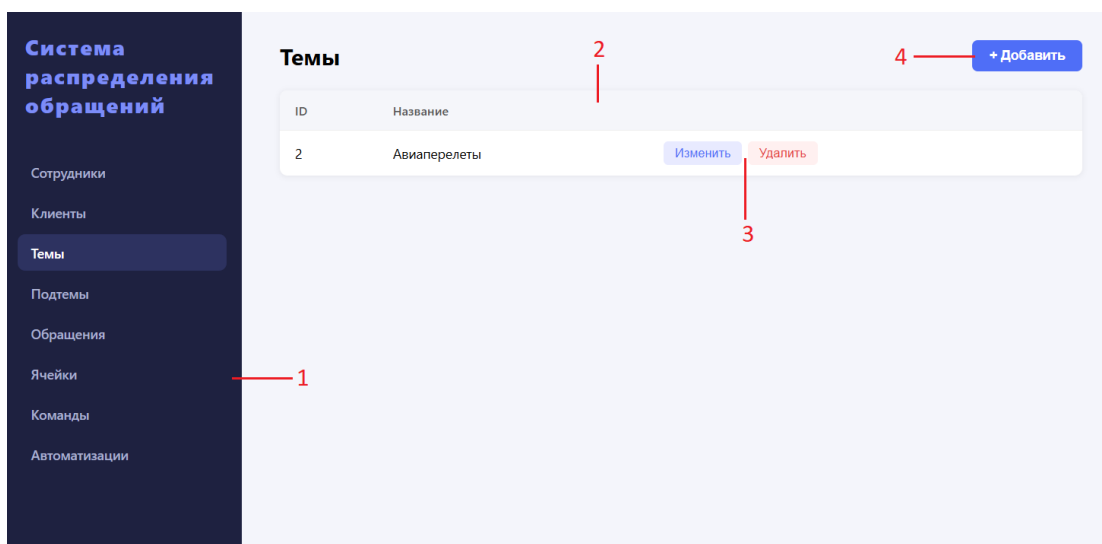


Рисунок 31 – Страница со списком тем обращений

На данной страницы есть следующие элементы:

- 1 – меню перехода на другие страницы;
- 2 – список существующих тем обращений;
- 3 – кнопки изменения, редактирования тем обращений;
- 4 – кнопка добавления новой темы обращения.

Далее на рисунке 32 находится страница создания темы обращения.

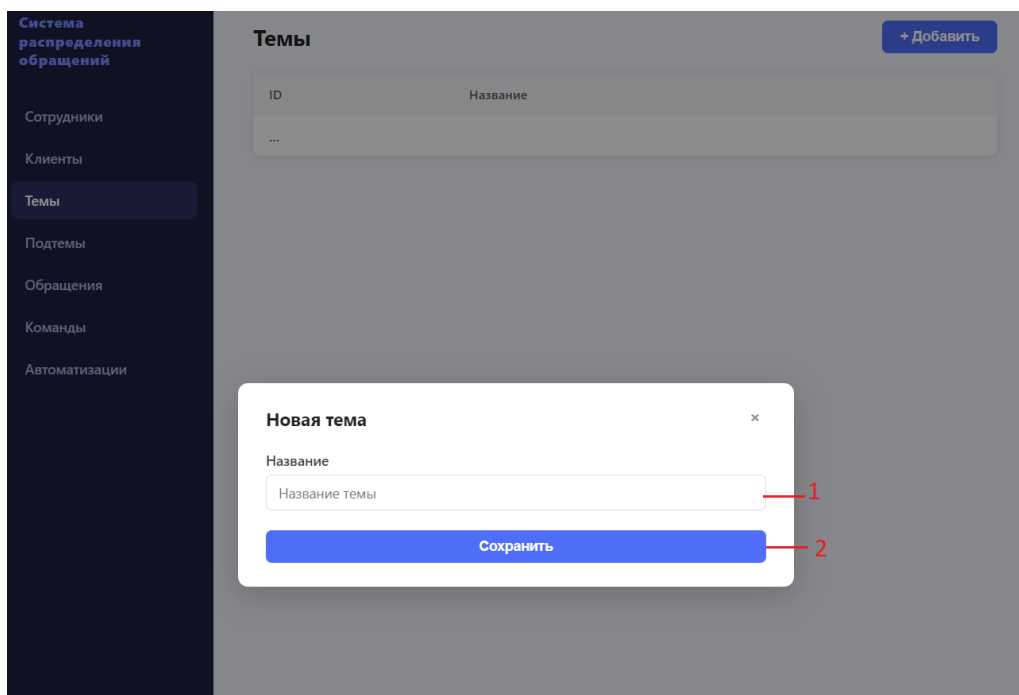


Рисунок 32 – Страница создания темы обращения

На странице создания темы обращения расположены следующие элементы:

- 1 – поле ввода названия темы;
- 2 – кнопка подтверждения создания темы.

Ниже на рисунке 33 находится страница создания команды.

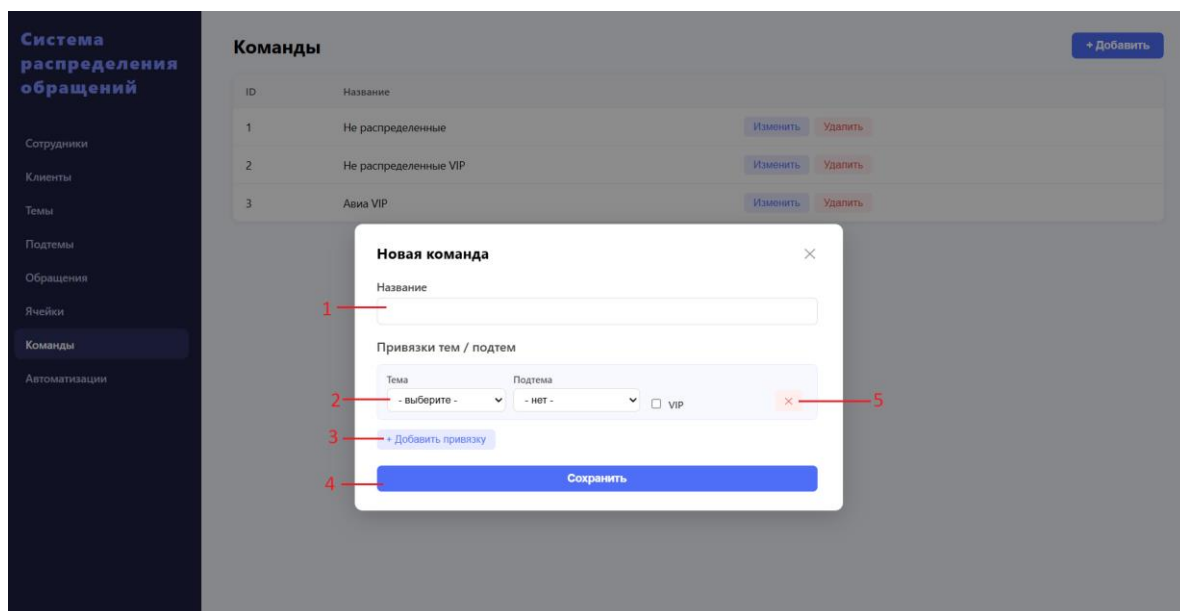


Рисунок 33 – Страница создания новой команды

На данной странице находятся следующие элементы:

- 1 – поле ввода названия команды;
- 2 – поля ввода темы, подтемы, а также указания VIP-флага для связывания с создаваемой командой;
- 3 – кнопка для добавления новой связи темы, подтемы, VIP-флага и создаваемой команды;
- 4 – кнопка подтверждения создания команды;
- 5 – кнопка удаления связи темы, подтемы, VIP-флага и создаваемой команды.

Далее на рисунке 34 расположена страница со списком подтем доступных подтем обращения.

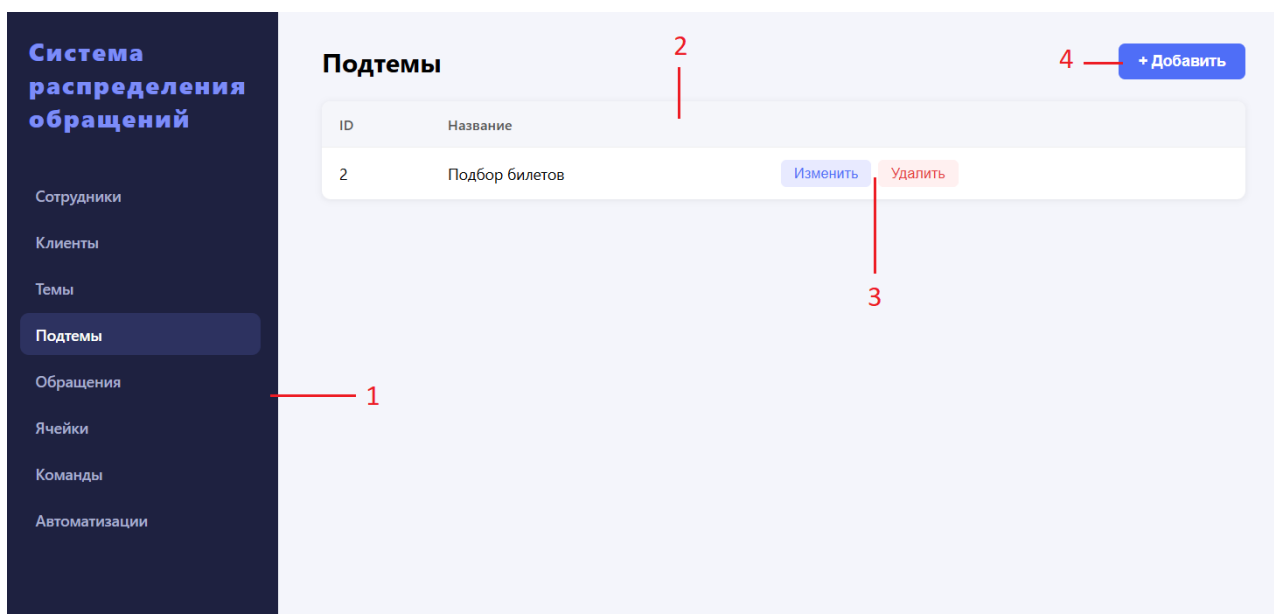


Рисунок 34 – Страница со списком подтем обращений

На данной странице присутствуют следующие элементы:

- 1 – меню перехода к другим страницам;
- 2 – список доступных подтем обращений;
- 3 – кнопки изменения и удаления обращений;
- 4 – кнопка добавления подтемы.

Далее на рисунке 35 расположена страница, которая предоставляет функционал редактирования существующей темы обращения.

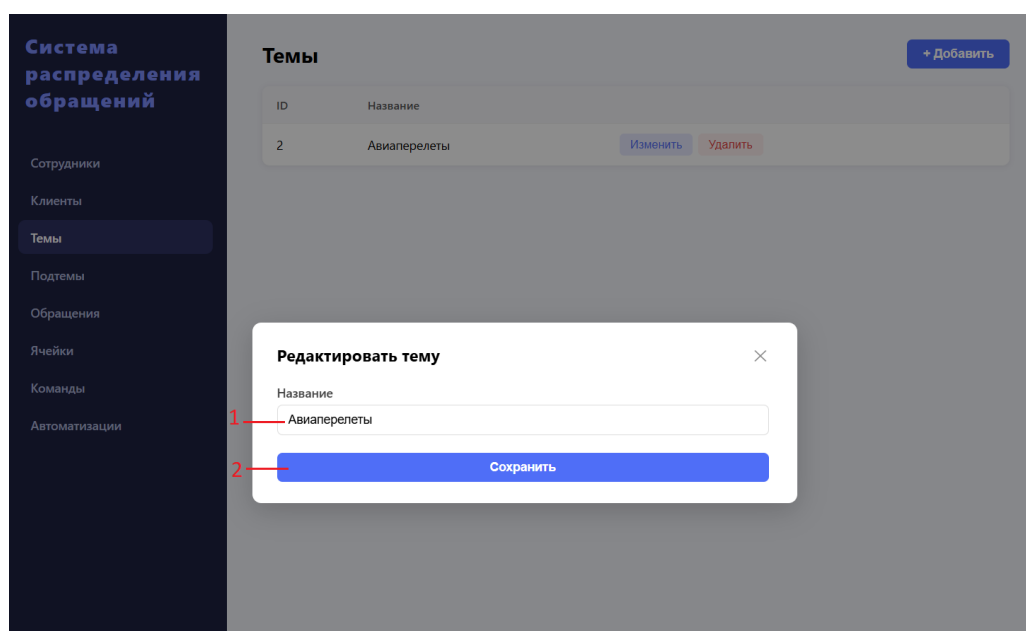


Рисунок 35 – Страница редактирования темы обращения

На странице редактирования темы обращения расположены следующие элементы:

- 1 – поле ввода нового названия темы;
- 2 – кнопка подтверждения сохранения изменений.

Далее на рисунке 36 приведена страница добавления новой подтемы обращения.

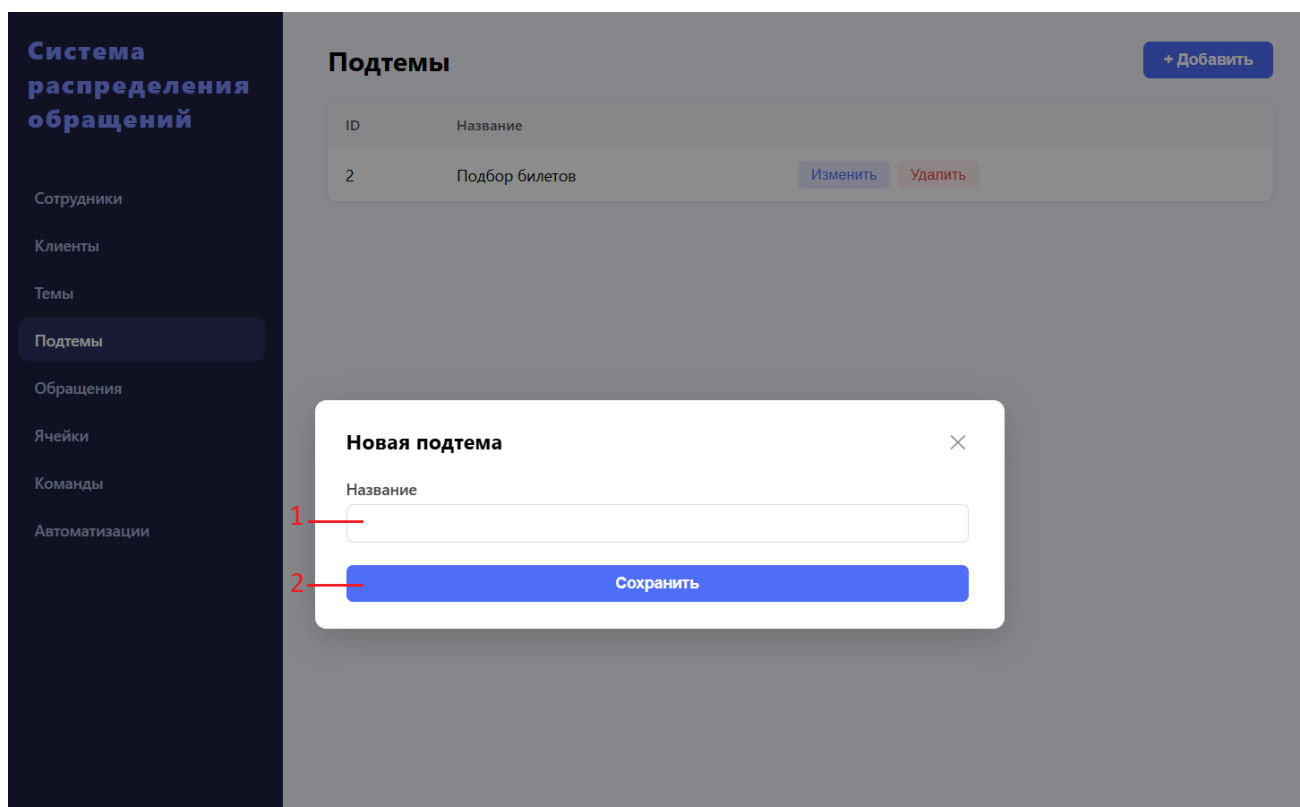


Рисунок 36 – Страница добавления подтемы обращений

На странице добавления подтемы обращения есть элементы:

- 1 – поле ввода названия обращения;
- 2 – кнопка подтверждения создания.

Ниже на рисунке 37 находится страница редактирования подтемы обращения, которая позволяет изменить данные ранее созданной в системе подтемы.

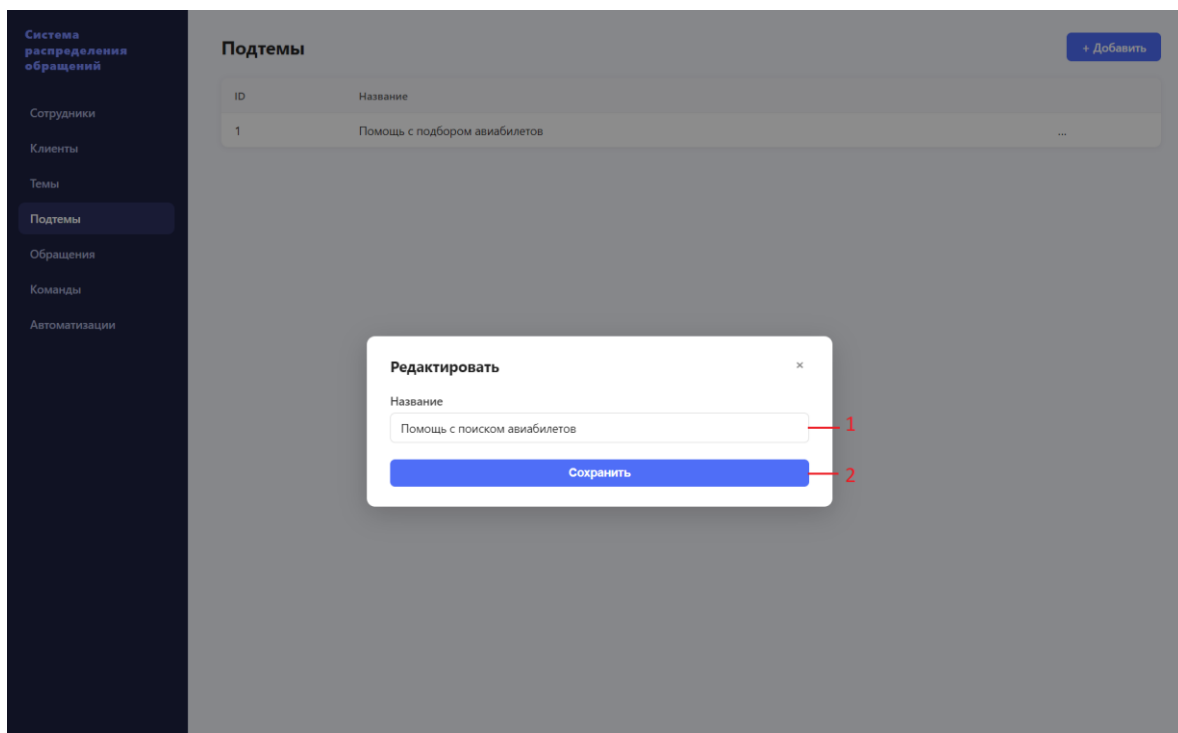


Рисунок 37 – Страница редактирования подтемы обращений

На странице редактирования подтемы обращения присутствуют следующие элементы:

- 1 – поле ввода нового названия обращения;
- 2 – кнопка сохранения изменений.

Далее на рисунке 38 приведена страница со списком ранее созданных клиентских обращений.

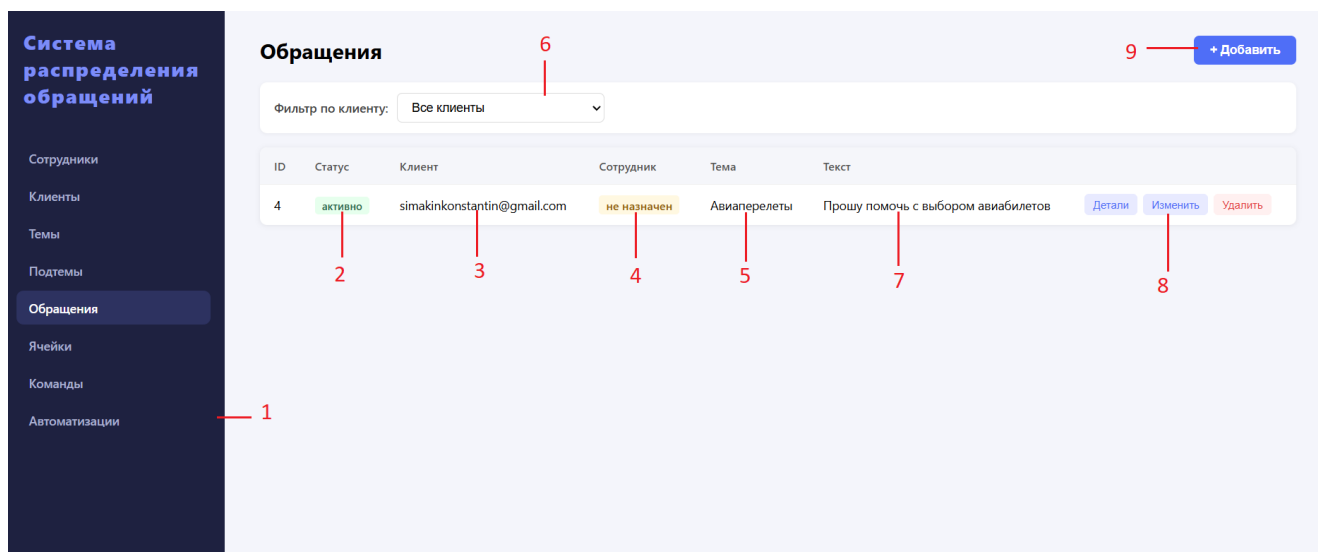


Рисунок 38 – Страница со списком обращений

На странице со списком обращений представлена информация обо всех существующих обращениях системы.

На данной странице отображаются следующие элементы:

- 1 – меню для перехода на другие страницы;
- 2 – текущий статус обращения;
- 3 – электронный адрес клиента, который инициировал обращение;
- 4 – статус распределения обращения между сотрудниками;
- 5 – тема обращения;
- 6 – список для фильтрации обращений определенного клиента;
- 7 – сокращенный текст обращения;
- 8 – кнопки удаления и открытия окна с подробной информацией об обращении;
- 9 – кнопка создания обращения.

Для отображения актуальной информации о результате распределения данная страница обновляется раз в три секунды.

Ниже на рисунке 39 находится страница детальной информации об обращении.

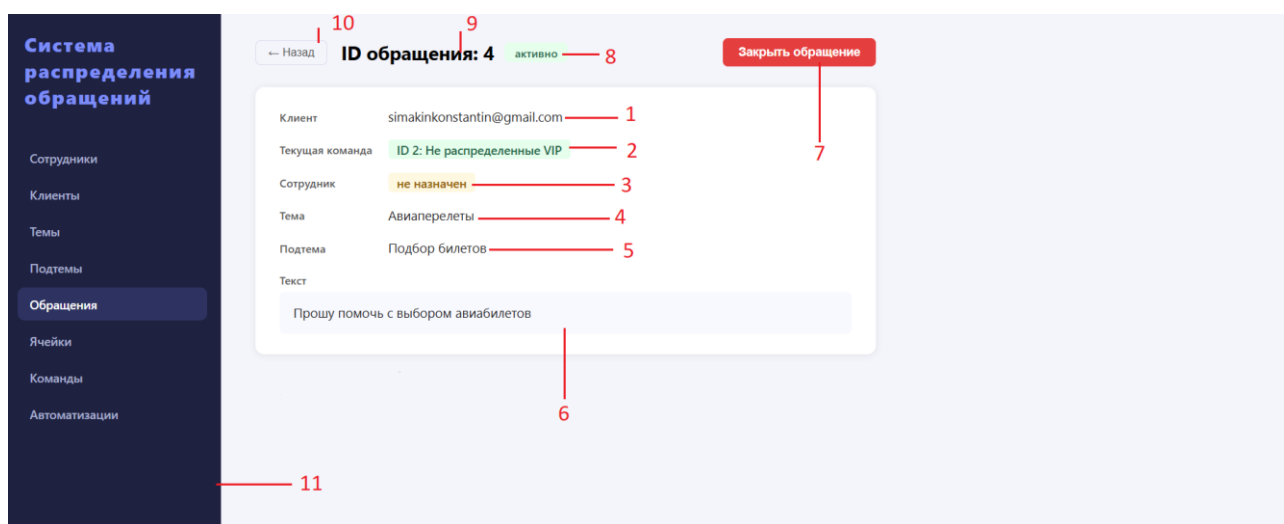


Рисунок 39 – Страница детальной информации об обращении

На странице просмотра детальной информации обращения есть следующие элементы:

- 1 – поле, отображающее электронный адрес клиента;
- 2 – поле, отображающее команду сотрудников, которая обрабатывает обращение;
- 3 – поле, отображающее сотрудника, который обрабатывает обращение;
- 4 – поле, отображающее тему обращения;
- 5 – поле, отображающее подтему обращения;
- 6 – поле, отображающее текст обращения;
- 7 – кнопка закрытия обращения;
- 8 – поле, отображающее статус обращения;
- 9 – идентификатор обращения;
- 10 – кнопка возврата к странице со списком обращений;
- 11 – меню для перехода на другую страницу;

Данная страница, так же, как и страница, отображающая список обращений, автоматически выполняет обновление раз в три секунды для отображения актуальной информации о состоянии просматриваемого обращения.

Далее на рисунке 40 находится страница со списком ячеек.

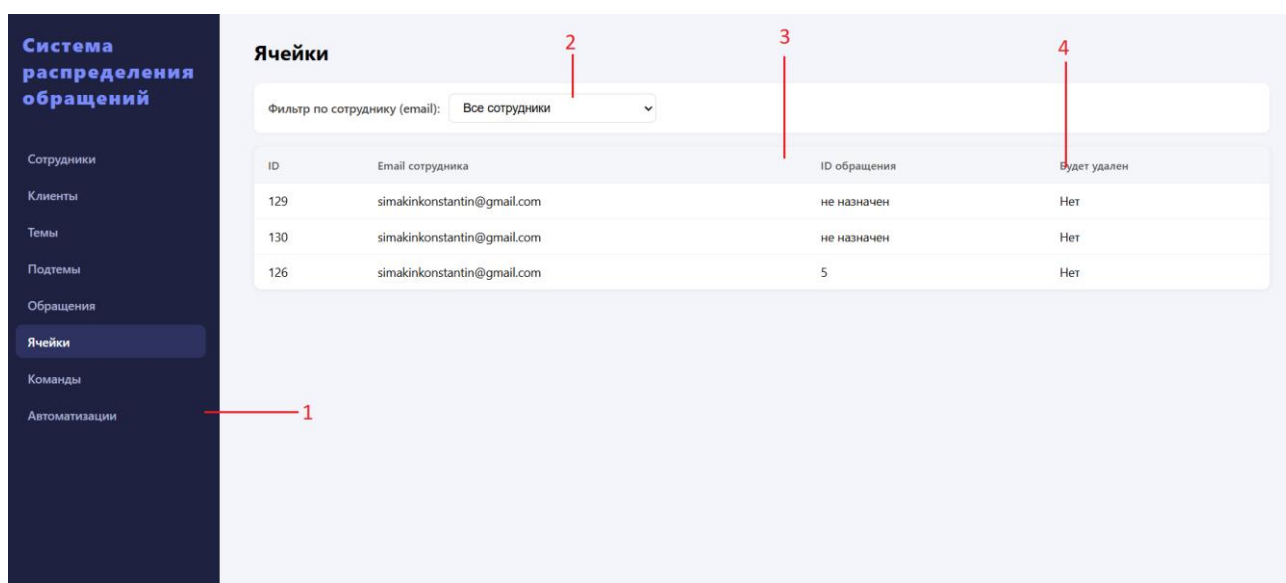


Рисунок 40 – Страница списка ячеек

На странице списка ячеек находятся следующие элементы:

- 1 – меню перехода к другим страницам;
- 2 – фильтр по электронному адресу сотрудника;
- 3 – список ячеек сотрудников;
- 4 – флаг, что соответствующий слот будет удален вместо освобождения.

Данная страница обновляется раз в три секунды для демонстрации актуальных данных.

Далее на рисунке 41 расположена страница редактирования команды.

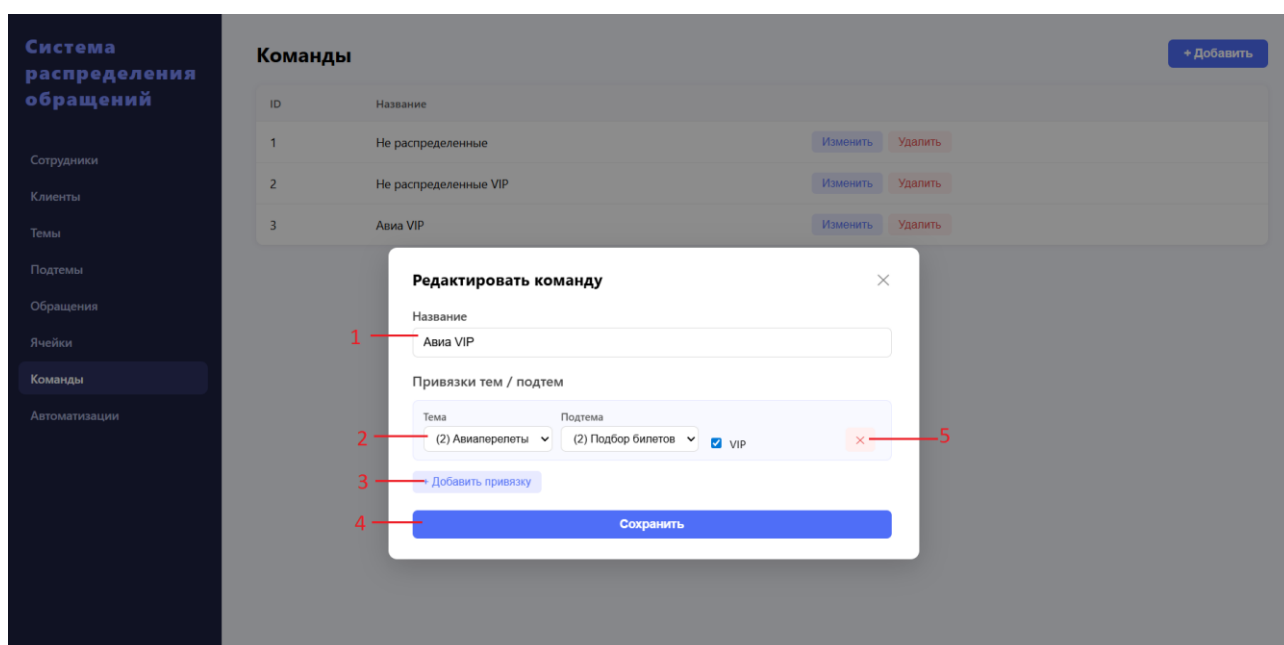


Рисунок 41 – Страница редактирования команды

На данной странице содержатся следующие объекты:

- 1 – поле ввода нового названия команды;
- 2 – поля ввода новой комбинации темы, подтемы, VIP-флага для связывания с редактируемой командой;
- 3 – кнопка для добавления новой связи темы, подтемы, VIP-флага и создаваемой команды;
- 4 – кнопка сохранения измененной команды;
- 5 – кнопка удаления связи темы, подтемы, VIP-флага и редактируемой команды.

Ниже на рисунке 42 приведена страница создания обращения.

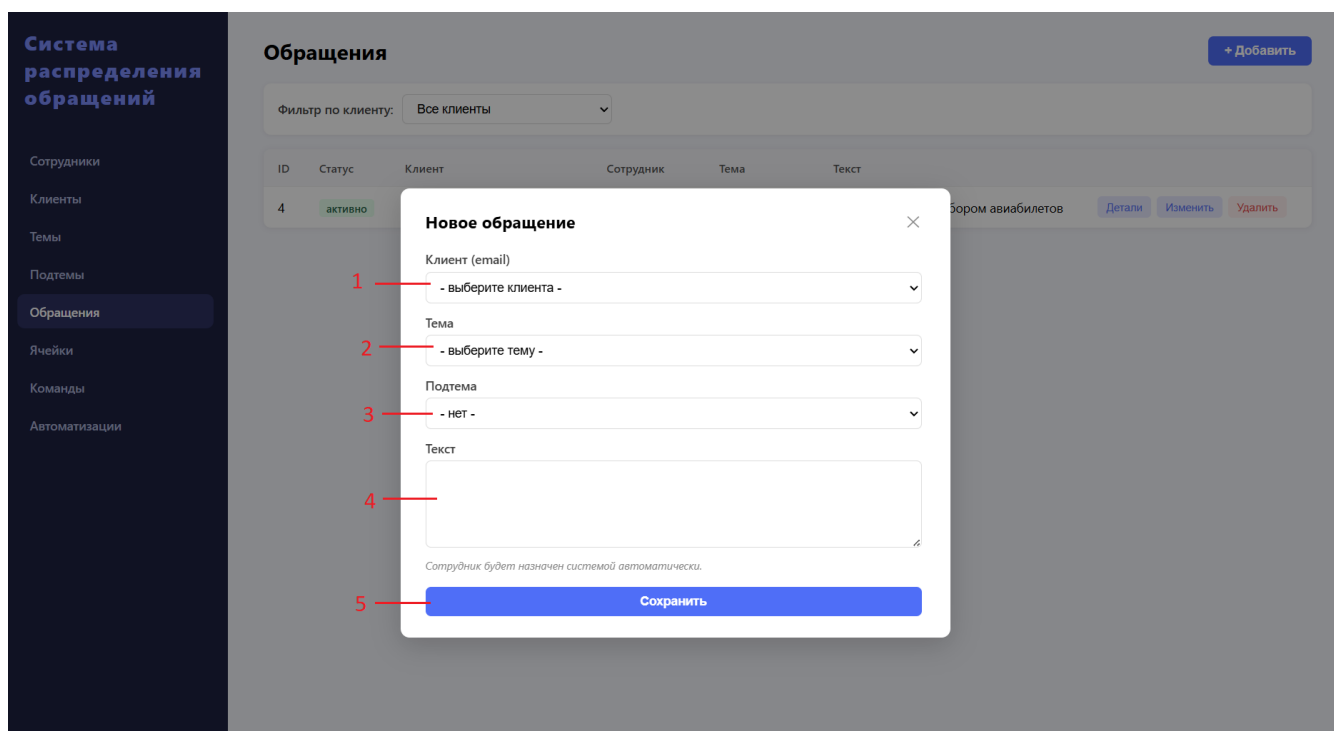


Рисунок 42 – Страница создания обращения

На приведенной выше странице создания обращения есть следующие элементы:

- 1 – поле ввода электронного адреса пользователя, от лица которого будет создано обращение;
- 2 – поле ввода темы обращения;
- 3 – поле ввода подтемы обращения, которое является необязательным для заполнения;
- 4 – поле ввода текста обращения;
- 5 – кнопка подтверждения создания обращения.

После создания обращения автоматически запустится процесс его распределения.

Ниже на рисунке 43 находится страница отображения списка автоматизаций.

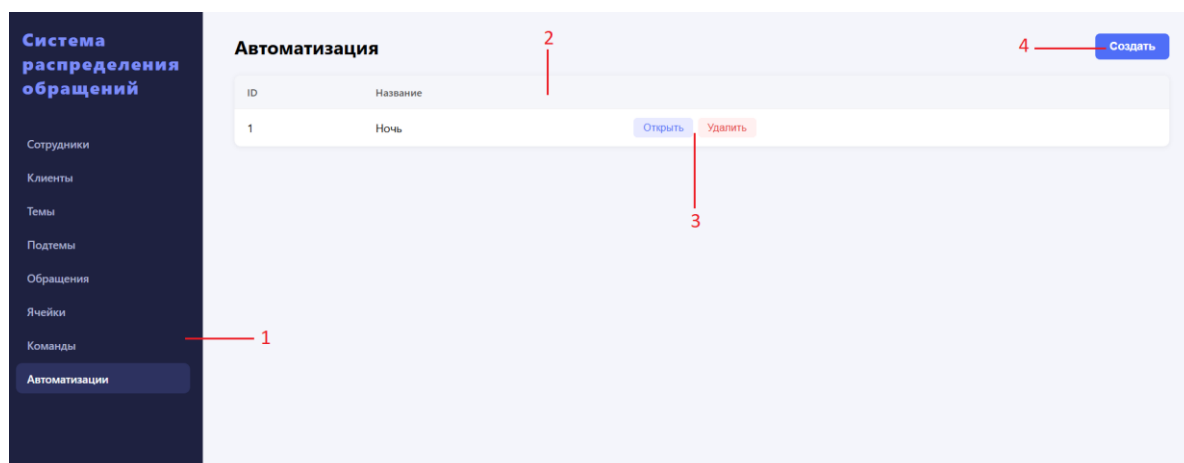


Рисунок 43 – Страница отображения списка автоматизаций

На странице отображения списка автоматизаций присутствуют элементы:

- 1 – меню для перехода на другие страницы;
- 2 – список автоматизаций;
- 3 – кнопка открытия и удаления автоматизации;
- 4 – кнопка добавления автоматизации.

Ниже на рисунке 44 приведена страница списка команд сотрудников.

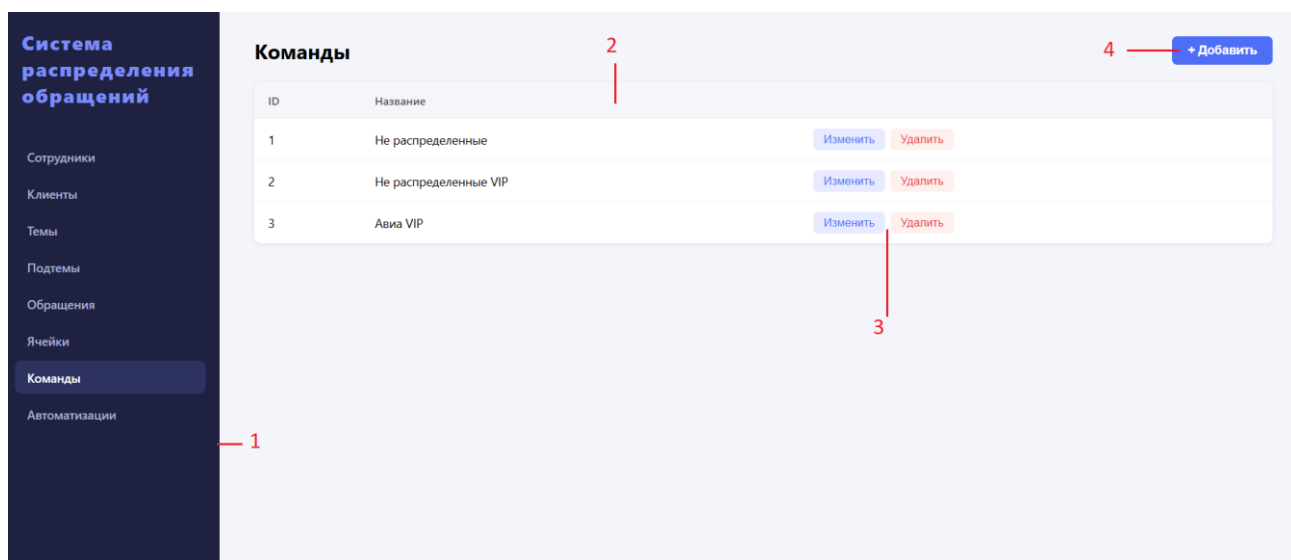


Рисунок 44 – Страница списка команд сотрудников

На странице списка команд сотрудников расположены следующие элементы:

- 1 – меню для перехода на другие страницы;

2 – список, отображающий команды сотрудников;

3 – кнопки изменения и удаления команд;

4 – кнопка добавления новой команды.

Ниже на рисунке 45 находится страница создания автоматизаций.

Рисунок 45 – Страница создания автоматизации

На странице создания автоматизации присутствуют объекты:

1 – поле ввода названия автоматизации;

2 – поле ввода статуса автоматизации;

3 – поле ввода режима автоматизации;

4 – кнопки добавления нового элемента автоматизации: условия или действия;

5 – номер шага автоматизации;

6 – кнопка удаления шага автоматизации;

7 – булевский оператор, декларирующий логику объединения нескольких предикатов в рамках одного шага автоматизации;

8 – кнопка удаления предиката из условия;

9 – атрибут, по которому будет происходить проверка предиката;

10 – операция, которая будет проверяться в предикате, это значение специфично для конкретного атрибута;

11 – значения, которые будут проверяться в предикате, это значение специфично для конкретного атрибута;

12 – добавление предиката в условие;

13 – кнопка, подтверждающая создание автоматизации.

3.4 Реализация клиентской части приложения

Была разработана клиентская часть приложения, которая реализует описанный ранее пользовательский интерфейс. Исходный код клиентской части представлен в приложении Г.

Структура реализованной клиентской части приложения с указанием зависимостей между компонентами приведена в виде диаграммы компонентов на рисунке 46.

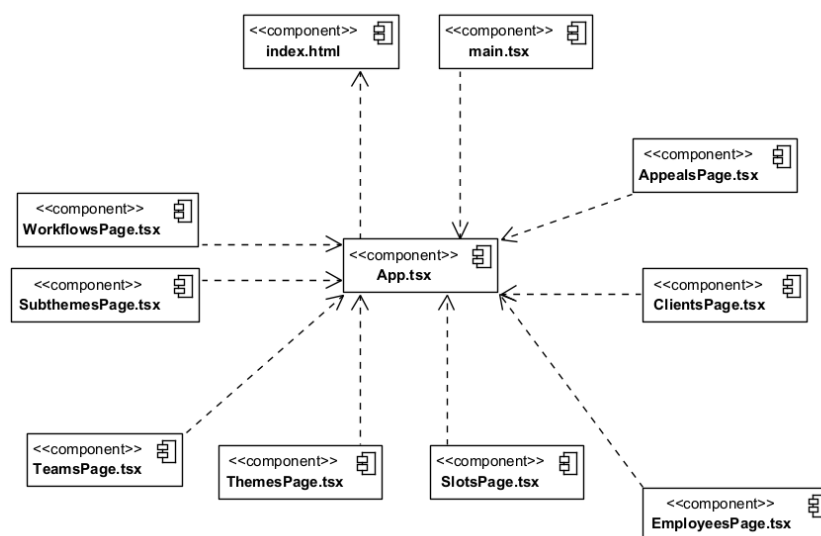


Рисунок 46 – Диаграмма компонентов клиентской части приложения

Описание компонентов приведено в таблице 15.

Таблица 15 – Описание компонентов клиентской части приложения

№	Компонент	Описание
1	index.html	Входная точка приложения для работы в браузере
2	main.tsx	Входная точка React-приложения
3	App.tsx	Компонент, реализующий меню навигации между страницами приложения
4	AppealsPage.tsx	Компонент, реализующий страницу управления обращениями
5	ClientsPage.tsx	Компонент, реализующий страницу управления клиентами
6	EmployeePage.tsx	Компонент, реализующий страницу управления сотрудниками
7	SlotsPage.tsx	Компонент, реализующий страницу управления ячейками сотрудников
8	ThemePage.tsx	Компонент, реализующий страницу управления темами обращений
9	TeamsPage.tsx	Компонент, реализующий страницу управления командами сотрудников
10	SubthemesPage.tsx	Компонент, реализующий страницу управления подтемами обращений
11	WorkflowsPage.tsx	Компонент, реализующий страницу управления автоматизациями

Таким образом, была разработана клиентская часть приложения, которая обладает простым, удобным для использования графическим интерфейсом. При этом исходный код клиентской части разбит на отдельные компоненты, что упрощает дальнейшее сопровождение приложения.

4 Тестирование программных средств распределения обращений

4.1 Функциональное ручное тестирование

Рассмотрим тестирование подсистемы управления сотрудниками ручным способом методом черного ящика на примере добавления нового сотрудника.

Исходными данными для данной функции являются:

- фамилия;
- имя;
- email-адрес;
- команда;
- лимит количества назначенных обращений;
- статус сотрудника.

Рассмотрим составление тест-кейсов на примере поля «Фамилия сотрудника». Фамилия сотрудника может содержать только буквы русского алфавита и дефис, который не может быть указан в виде первого и последнего символа. Длина фамилии должна быть больше, чем 1 символ, но не более чем 50 символов. Классы эквивалентности [15], приведенные в таблице 16.

Таблица 16 – Классы эквивалентности поля «Фамилия»

№	Название	Правильные КЭ	Неправильные КЭ
1	Длина фамилии	2 – 50 (1)	0 – 1(2) 51 – ∞ (3)
2	Структура	Начинается с буквы (4) Заканчивается буквой (5)	Начинается не с буквы (6) Заканчивается не буквой (7)
3	Содержание	Русские буквы (8) Дефис (9)	Остальное (10)

На основе этих классов эквивалентности были составлены тест-кейсы, приведенные в таблице 17, скриншоты с результатами представлены в приложении Д.

Таблица 17 – Тест-кейсы поля «Фамилия»

№	КЭ	Входные данные	Выходные данные	Фактический результат
1	1, 4, 5, 8, 9	Иванов	Успешно	Тест пройден. Рисунок Д.1
2	2, 4, 5, 8, 9	И	Ошибка: значение не может состоять из одного символа	Тест пройден. Рисунок Д.2
3	3, 4, 5, 8, 9	Иииииииииииииииииии ииииииииииииииииии ииииииииии	Ошибка: неверный формат	Тест пройден. Рисунок Д.3
4	1, 6, 5, 8, 9	-Иванов	Ошибка: неверный формат	Тест пройден. Рисунок Д.4
5	1, 4, 7, 8, 9	Иванов-	Ошибка: неверный формат	Тест пройден. Рисунок Д.5
6	1, 4, 5, 10	Иван123ов	Ошибка: неверный формат	Тест пройден. Рисунок Д.6

Также были составлены тесты на граничные условия. Тесты приведены в таблице 18.

Таблица 18 – Граничные тест-кейсы поля «Фамилия»

№	Граница	Входные данные	Выходные данные	Фактический результат
1	Физическая	Ъиванов	Ошибка: недопустимый первый символ	Тест пройден. Рисунок Д.7

Продолжение таблицы 18

№	Граница	Входные данные	Выходные данные	Фактический результат
2	Логическая	А	Ошибка: значение не может состоять из одного символа	Тест пройден. Рисунок Д.8
3	Произвольная	Ivanov	Ошибка: неверный формат	Тест пройден. Рисунок Д.9

Проведенное тестирование методом черного ящика позволяет сделать выводы о корректной работе приложения.

4.2 Модульное тестирование

Для тестирования отдельных функций приложения использовалось модульное тестирование [16].

Модульное тестирование – способ тестирования, при котором происходит проверка отдельных компонентов системы. В рамках тестирования каждый тестируемый компонент должен быть изолирован от внешних зависимостей.

В качестве примера использования модульных тестов рассмотрим тест функции «InInterval», которая определяет, лежит ли время в заданном интервале, и возвращает логическое значение.

Ниже на рисунке 47 демонстрируется результат работы тестов.

```

✓ Tests passed: 48 of 48 tests – 354 ms

--- PASS: TestEventHandlerHandleDelivery (0.00s)
--- PASS: TestEventHandlerHandleDelivery/delivery_handled (0.00s)
=== RUN    TestMatcherHandleDistributionTickEmpty
=== RUN    TestMatcherHandleDistributionTickEmpty/empty_sources_produce_no_error
2026/06/02 17:41:30 Entered distribution tick
2026/06/02 17:41:30 FetchPendingAppeals: []
2026/06/02 17:41:30 Fetched available employees: []
--- PASS: TestMatcherHandleDistributionTickEmpty (0.00s)
--- PASS: TestMatcherHandleDistributionTickEmpty/empty_sources_produce_no_error (0.00s)
=== RUN    TestCountFreeSlots
=== RUN    TestCountFreeSlots/counts_only_unused
--- PASS: TestCountFreeSlots (0.00s)
--- PASS: TestCountFreeSlots/counts_only_unused (0.00s)
=== RUN    TestOldestFreeSlotTime
=== RUN    TestOldestFreeSlotTime/returns_oldest_updated_at
--- PASS: TestOldestFreeSlotTime (0.00s)
--- PASS: TestOldestFreeSlotTime/returns_oldest_updated_at (0.00s)
PASS

Process finished with the exit code 0

```

Рисунок 47 – Результат работы тестов

Таким образом, все модульные тесты были пройдены успешно, из чего можно сделать вывод о корректной работе разработанных программных средств.

Модульными тестами покрыта как серверная, так и клиентская части, что повышает общую надежность приложения.

4.3 Метрики кода

Для анализа характеристик кода были собраны метрики исходного текста приложения. Рассмотрим функцию «calculateAppealPriority», которая используется для вычисления приоритета обращения перед помещением ее в очередь для последующего распределения. Описание этой функции приведено на листинге 1.

Листинг 1 – Функция «calculateAppealPriority»

```
func calculateAppealPriority(isImportant bool, createdAt
time.Time) int64 {
    now := time.Now()

    // base рассчитывается в зависимости от статуса пользователя.
    var base int64
    if isImportant {
        base = 2
    } else {
        base = 1
    }

    priority := int64(base * 100)

    // Расчет времени жизни обращения в минутах.
    ageMinutes := (int64)(now.Sub(createdAt).Minutes())
    priority += ageMinutes * 10

    return priority
}
```

Далее демонстрируются полученные метрики этой функции.

4.2.1 Метрики SLOC

Данные метрики заключаются в оценке количества строк исходного кода программы. Результаты приведены в таблице 19.

Таблица 19 – SLOC-метрики

№	Описание	Количество
1	Физическое количество строк	20
2	Логическое количество строк	13

Из таблицы 19 можно сделать вывод, что количество физических строк кода превышает число логических. Это связано с наличием комментариев и пустых строк для разграничения программных блоков, благодаря которым повышается читаемость исходного кода.

4.2.2 Метрики Холстеда

Для вычисления метрик Холстеда [17] были определены все операнды и операторы исходного кода. Операторы представлены в таблице 20.

Таблица 20 – Операторы функции «calculateAppealPriority»

№	Оператор	Количество
1	Now	1
2	int64	4
3	:=	3
4	=	2
5	if	1
6	else	1
7	+=	1
8	*	2
9	return	1
10	{	3
11	}	3
12	func	1
13	(	6
14	)	6
15	,	1
16	.	4

Продолжение таблицы 20

№	Оператор	Количество
17	Sub	1
18	Minutes	1
19	calculateAppealPriority	1
20	bool	1
21	Time	1
22	var	1

Операнды функции представлены в таблице 21.

Таблица 21 – Операнды функции «calculateAppealPriority»

№	Операнд	Количество
1	isImportant	2
2	createdAt	2
3	now	2
4	base	3
5	priority	3
6	ageMinutes	2
7	2	1
8	1	1
9	100	1
10	10	1

Таким образом, количество уникальных операторов – n_1 : 22; количество уникальных операндов – n_2 : 10; общее число операторов – N_1 : 46; общее число операндов: N_2 : 18.

Благодаря полученным значениям были вычислены метрики, описанные далее.

Словарь программы рассчитывается по формуле:

$$n = n_1 + n_2 = 22 + 10 = 32.$$

Длина программы рассчитывается по формуле:

$$N=N_1+N_2=46+18=64.$$

Теоретическая длина программы вычисляется по формуле:

$$N'=n_1 \times \log_2(n_1)+n_2 \times \log_2(n_2)=22 \times \log_2(22)+10 \times \log_2(10)=131.32.$$

Объем программы рассчитывается по формуле:

$$V=N \times \log_2(n)=64 \times \log_2(32)=320.$$

Трудоемкость кодирования программы вычисляется по формуле:

$$D=\frac{n_1}{2} \times \frac{N_2}{n_2}=\frac{22}{2} \times \frac{18}{10}=19.8.$$

Усилия на написания кода рассчитываются по формуле:

$$E=D \times V=19.8 \times 320=6336.$$

Время на программирование вычисляется по формуле:

$$T=\frac{E}{18}=\frac{6336}{18}=352 \text{ секунд.}$$

Примерное количество ошибок рассчитываются по формуле:

$$B=\frac{E^{\frac{2}{3}}}{3000}=\frac{6336^{\frac{2}{3}}}{3000}=0.114.$$

Выводы по полученным метрикам приведены в таблице 22.

Таблица 22 – Объяснение значений метрик Холстеда

№	Метрика	Значение	Диапазон допустимых значений	Вывод
1	Словарь программы	32	10 – 150	Значение в норме
2	Длина программы	64	10 – 120	Значение в норме
3	Теоретическая длина программы	131.32	20 – 250	Значение в норме
4	Объем программы	320	30 – 450	Значение в норме
5	Трудоемкость кодирования программы	19.8	5 – 75	Значение в норме
6	Усилия на написания кода	6336	100 – 9000	Значение в норме
7	Время на программирование	352 секунд	20 – 700 секунд	Значение в норме

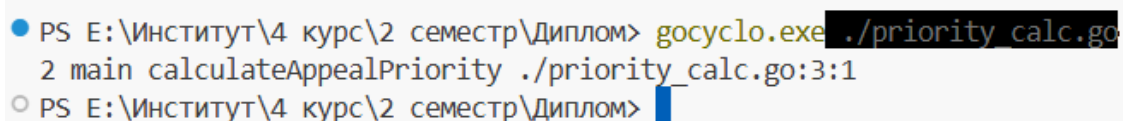
Продолжение таблицы 22

№	Метрика	Значение	Диапазон допустимых значений	Вывод
8	Примерное количество ошибок	0.114	0 – 1	Ожидается малое количество ошибок при работе с функцией

Таким образом, рассмотренная функция обладает небольшим объемом исходного кода. Это повышает читаемость кода программного обеспечения, а также упрощает его дальнейшее сопровождение. От функции ожидается низкое количество ошибок, а для ее написания требуется небольшое количество времени. Рассчитанные метрики позволяют сделать вывод о высоком качестве разработанных программных средств.

4.2.3 Цикломатическая сложность

Для оценки простоты кода была подсчитана цикломатическая сложность кода. Данная метрика была вычислена утилитой gocyclo [18]. Результат расчета цикломатической сложности кода приведен на рисунке 48.



```

PS E:\Институт\4 курс\2 семестр\Диплом> gocyclo.exe ./priority_calc.go
2 main calculateAppealPriority ./priority_calc.go:3:1
PS E:\Институт\4 курс\2 семестр\Диплом>

```

Рисунок 48 – Результат вычисления цикломатической сложности

Таким образом, исходный код обладает цикломатической сложностью, равной двум, что меньше предельного значения равного десяти, а значит код прост для понимания и расширения.

5 Развертывание и сопровождение программных средств

Система разворачивается в виде docker compose приложения [19]. Его декларативное описание приведено в приложении Е.

За счет использования контейнеризации приложение можно запускать на разных вычислительных узлах в идентичном окружении без дополнительной настройки. Все компоненты системы: клиентская часть, серверная часть, хранилища данных, брокер сообщений, могут быть запущены одной командой.

В docker compose приложении определены следующие сервисы:

frontend – клиентская часть приложения;

backend – серверная часть приложения;

postgresql – СУБД PostgreSQL;

redis – СУБД Redis;

rabbitmq – брокер сообщений RabbitMQ.

Хранение постоянных данных реализовано с помощью механизма томов, благодаря чему при пересоздании контейнеров данные приложения сохраняются.

Для настройки программного обеспечения используются переменные окружения, которые также описаны в docker compose файле. Такой подход делает конфигурацию системы более прозрачной, так как все параметры приведены в одном месте.

Компоненты разработанной системы должны запускаться в определенной последовательности, например, серверная часть приложения для корректной работы требует подключения к базе данных. Таким образом, запуск элементов docker compose приложения происходит в определенном порядке, компоненты запускаются после того, как все их зависимости начнут функционировать.

Ниже на рисунке 49 приведена схема зависимостей компонентов системы.

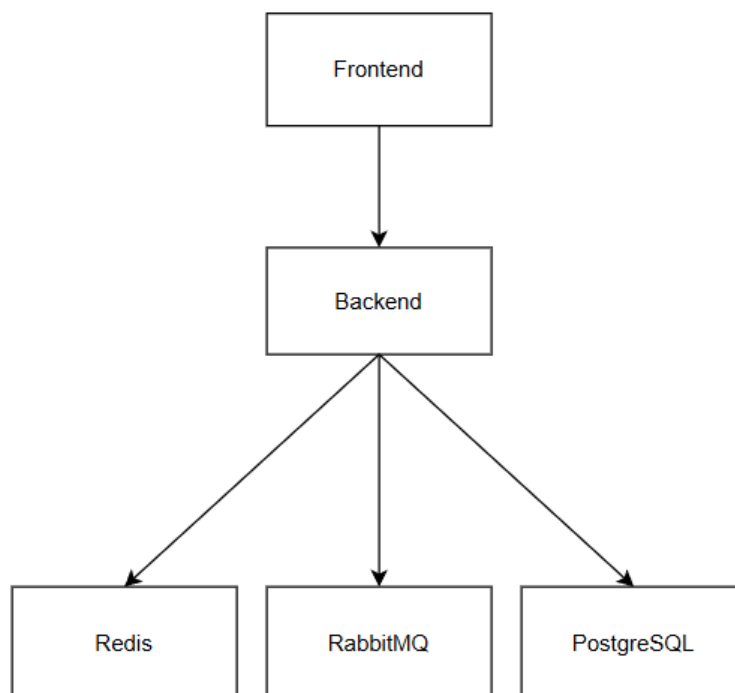


Рисунок 49 – Зависимости между компонентами приложения

Согласно приведенной схеме зависимостей, клиентская часть запускается после полного запуска серверной части и всех ее зависимостей.

Контейнеризация также понадобится при дальнейшем развитии системы. Описанную схему работы можно будет преобразовать к масштабируемой распределенной системе, используя docker swarm [20] или kubernetes [21] вместо docker compose.

Таким образом, использование механизма контейнеризации упрощает и ускоряет процесс разработки системы, а также помогает при ее дальнейшем сопровождении.

Заключение

В ходе выполнения задания выпускной квалификационной работы были разработаны программные средства распределения обращений клиентов между сотрудниками технической поддержки.

На этапе анализа предметной области и постановки задачи были проанализированы нюансы предметной области технической поддержки пользователей. Также на этом этапе был проведен анализ систем аналогичного назначения. В итоге были определены функциональные, а также нефункциональные требования к разрабатываемым программным средствам.

На этом этапе было проведено планирования процесса разработки. В результате планирования было вычислено, что для его выполнения проекта потребуется 78 дней, а также 266 400.00 рублей.

На этапе проектирования были выбраны технологии и инструментальные средства для разработки приложения. На этом этапе была определена структура базы данных, разработаны концептуальная, а также логическая модели.

При проектировании было произведено моделирование алгоритма распределения обращения клиентов между сотрудниками технической поддержки, механизма поддержки лимита одновременно назначенного количества обращений на сотрудника, а также была определена архитектура серверной части приложения – модульный монолит.

На этом этапе была спроектирована клиентская часть приложения, описан состав web-страниц для реализации.

На этапе реализации была разработана итоговая структура базы данных, составлены файлы миграций. Была разработана серверная часть, реализующая механизм распределения обращения клиентов, а также функционал для моделирования процесса работы системы. Также была реализована клиентская часть, которая предоставляет пользователю графический интерфейс для взаимодействия с программными средствами. Полный исходный приложения размещен на платформе GitHub [22].

На этапе программных средства было проведено функциональное тестирование разработанного приложения. Также исходный код был покрыт модульными тестами. В результате все тесты были успешно пройдены, на основании чего можно сделать вывод о корректной работе приложения.

На этом этапе также были рассчитаны метрики SLOC, Холстеда, а также цикломатическая сложность.

На этапе развертывания и сопровождения был реализован запуск программных средств в виде docker compose приложения, что повышает удобство разработки и последующего сопровождения программного обеспечения.

Таким образом, задача выпускной квалификационной работы была успешно выполнена, были достигнуты все поставленные цели.

Список использованных источников

1. What is Intercom? // Intercom – 2026 [Электронный ресурс]. URL: <https://www.intercom.com/help/en/articles/294-what-is-intercom> (дата обращения: 01.05.2026)
2. Carrot Quest Чат // Carrot Quest – 2026 [Электронный ресурс]. URL: <https://www.carrotquest.io/chat/> (дата обращения: 01.05.2026)
3. Обзор LiveChat – Тип по продукту // LiveChat – 2026 [Электронный ресурс]. URL: <https://www.livechat.com/ru/obzor/> (дата обращения: 01.05.2026)
4. Арлоу Д. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование / Арлоу Д., Нейштадт А. – Пер. с англ. – СПб.: Символ-Плюс, 2007. – 624 с.
5. Что такое Диаграмма Ганта // Яндекс Практикум – 2025 [Электронный ресурс]. URL: <https://practicum.yandex.ru/blog/diagramma-ganta/> (дата обращения: 07.05.2026)
6. The Go Programming Language Specification // Go – 2026 [Электронный ресурс]. URL: <https://go.dev/ref/сpec> (дата обращения 07.05.2026)
7. PostgreSQL: The World's Most Advanced Open Source Relational Database // PostgreSQL – 2026 [Электронный ресурс]. URL: <https://www.postgresql.org/> (дата обращения 07.05.2026)
8. One broker to queue them all // RabbitMQ – 2026 [Электронный ресурс]. URL: <https://www.rabbitmq.com/> (дата обращения 07.05.2026)
9. Simple, reliable & efficient distributed task queue in Go // GitHub – 2026 [Электронный ресурс]. URL: <https://github.com/hibiken/asyncq#simple-reliable--efficient-distributed-task-queue-in-go> (дата обращения 07.05.2026)
10. TypeScript is JavaScript with syntax for types // TypeScript – 2026 [Электронный ресурс]. URL: <https://www.typescriptlang.org/> (дата обращения 07.05.2026)
11. The library for web and native user interfaces // React – 2026 [Электронный ресурс]. URL: <https://react.dev/> (дата обращения 08.05.2026)

12. Модель С4 архитектуры программного обеспечения // Cinimex – 2026 [Электронный ресурс]. URL: <https://cinimex.ru/solutions/model-s4-arkhitektury-programmnogo-obespecheniya/>
13. Руководство по связям типа «многие ко многим» // Microsoft Learn – 2026 [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/power-bi/guidance/relationships-many-to-many> (дата обращения 08.05.2026)
14. Release 15 // PostgreSQL – 2022 [Электронный ресурс]. URL: <https://www.postgresql.org/docs/15/release-15.html#id-1.11.6.5.5.3.4> (дата обращения 14.05.2026)
15. Классы эквивалентности в тестировании // Точка качества – 2025 [Электронный ресурс]. URL: <https://tquality.ru/blog/klassy-ekvivalentnosti-v-testirovanii/> (дата обращения 17.05.2026)
16. Юнит-тесты: что это такое, зачем они нужны и как их проводят // Яндекс Практикум – 2025 [Электронный ресурс]. URL: <https://practicum.yandex.ru/blog/chto-takoe-modulnoe-yunit-testirovanie/> (дата обращения 18.05.2026)
17. Метрики кода программного обеспечения // PVS-Studio – 2025 [Электронный ресурс]. (дата обращения 19.05.2026)
18. Gocyclo // GitHub – 2026 [Электронный ресурс]. URL: <https://github.com/fzipp/gocyclo#gocyclo> (дата обращения 18.05.2026)
19. Деплой из Docker Compose // Timeweb Cloud – 2025 [Электронный ресурс]. URL: <https://timeweb.cloud/docs/apps/deploying-with-docker-compose> (дата обращения 19.05.2026)
20. Swarm Mode // Docker docs – 2026 [Электронный ресурс]. URL: <https://docs.docker.com/engine/swarm/> (дата обращения 25.05.2026)
21. Оркестрация контейнеров промышленного уровня // Kubernetes – 2026 [Электронный ресурс]. URL: <https://kubernetes.io/ru/> (дата обращения 26.05.2026)
22. Исходный код разработанных программных средств // Github – 2026 [Электронный ресурс]. URL: <https://github.com/SimakinKonstantin/diploma> (дата обращения 01.06.2026)

Приложение А
(справочное)
Глоссарий

VIP (Very Important Person) – обозначение важного лица, которое обладает особыми привилегиями.

HTTP (Hypertext Transfer Protocol) – сетевой протокол прикладного уровня, взаимодействие по которому осуществляется по схеме «запрос–ответ».

Брокер сообщений – программное обеспечение, которое позволяет обмениваться информацией по компьютерной сети в асинхронном режиме.

Миграция базы данных – преобразование базы данных, направленная на изменение схемы данных.

UML (Unified Modeling Language) – графический язык, используемый в сфере разработки программного обеспечения, который применяется для моделирования бизнес-процессов, проектирования систем.

IDE (Integrated Development Environment) – программные средства, которые используются для разработки программного обеспечения.

CRUD (Create Read Update Delete) – обозначение четырех базовых операций над данными в вычислительной техники: создание, чтение, обновление, удаления.

API (Application Programming Interface) – набор инструкций, который позволяет взаимодействовать нескольким программам друг с другом.

Email (Electronic Mail) – технология отправки и получения электронных сообщений по компьютерной сети.

SLOC (Source Lines of Code) – метрика программного кода, используемая для оценки его объема путем подсчета количества строк кода.

Контейнеризация – механизм, при котором приложение, а также все компоненты, от которых оно зависит помещаются в изолированную среду выполнения.

СУБД (Система Управления Базами Данных) – программное обеспечение, которое позволяет создавать базы данных, выполнять операции над данными этих баз данных.

Приложение Б
(обязательное)
Исходный код миграций

Листинг Б.1 – Файл backend/migrations/00001_create_subthemes.sql

```
-- +goose Up
CREATE TABLE IF NOT EXISTS subthemes (
    id    SERIAL          PRIMARY KEY,
    name  TEXT NOT NULL UNIQUE
);

-- +goose Down
DROP TABLE IF EXISTS subthemes;
```

Листинг Б.2 – Файл backend/migrations/00002_create_themes.sql

```
-- +goose Up
-- +goose StatementBegin
CREATE TABLE IF NOT EXISTS themes (
    id    SERIAL PRIMARY KEY,
    name  TEXT NOT NULL UNIQUE
);
-- +goose StatementEnd

-- +goose Down
-- +goose StatementBegin
DROP TABLE IF EXISTS themes;
-- +goose StatementEnd
```

Листинг Б.3 – Файл backend/migrations/00003_create_employees.sql

```
-- +goose Up
CREATE TABLE IF NOT EXISTS employees (
    id        SERIAL          PRIMARY KEY,
    name      TEXT NOT NULL,
    surname   TEXT NOT NULL,
    "limit"   INTEGER         NOT NULL,
    email     TEXT UNIQUE NOT NULL,
    status    TEXT NOT NULL
);

-- +goose Down
DROP TABLE IF EXISTS employees;
```

Листинг Б.4 – Файл backend/migrations/00004_create_clients.sql

```
-- +goose Up
-- +goose StatementBegin
CREATE TABLE IF NOT EXISTS clients (
    id        SERIAL PRIMARY KEY,
    email     TEXT NOT NULL UNIQUE,
    name      TEXT NOT NULL,
```

```

        surname TEXT NOT NULL,
        is_vip BOOLEAN NOT NULL
    );
-- +goose StatementEnd

-- +goose Down
-- +goose StatementBegin
DROP TABLE IF EXISTS clients;
-- +goose StatementEnd

```

Листинг Б.5 – Файл backend/migrations/00005_create_teams.sql

```

-- +goose Up
-- +goose StatementBegin
CREATE TABLE IF NOT EXISTS teams (
    id SERIAL PRIMARY KEY,
    name TEXT UNIQUE NOT NULL
);
-- +goose StatementEnd

-- +goose Down
-- +goose StatementBegin
DROP TABLE IF EXISTS teams;
-- +goose StatementEnd

```

Листинг Б.6 – Файл backend/migrations/00006_create_appeals.sql

```

-- +goose Up
CREATE TABLE IF NOT EXISTS appeals (
    id SERIAL PRIMARY KEY,
    client_id INTEGER NOT NULL REFERENCES clients(id) ON DELETE
    CASCADE,
    employee_id INTEGER REFERENCES employees(id) ON DELETE
    CASCADE,
    theme_id INTEGER NOT NULL REFERENCES themes(id) ON DELETE
    CASCADE,
    subtheme_id INTEGER REFERENCES subthemes(id) ON DELETE
    CASCADE,
    text TEXT NOT NULL,
    status TEXT NOT NULL DEFAULT 'active' CHECK (status IN
    ('active', 'closed')),
    team_id INTEGER REFERENCES teams(id) ON DELETE CASCADE
);

CREATE INDEX IF NOT EXISTS idx_appeals_employee_id ON
appeals(employee_id);
CREATE INDEX IF NOT EXISTS idx_appeals_subtheme_id ON
appeals(subtheme_id);
CREATE INDEX IF NOT EXISTS idx_appeals_client_id ON
appeals(client_id);

```

```
-- +goose Down
DROP TABLE IF EXISTS appeals;
```

Листинг Б.7 – Файл backend/migrations/00007_create_slots.sql

```
-- +goose Up
CREATE TABLE IF NOT EXISTS slots (
    id SERIAL PRIMARY KEY,
    employee_id INTEGER NOT NULL REFERENCES employees(id) ON
DELETE CASCADE,
    appeal_id INTEGER REFERENCES appeals(id) ON DELETE SET NULL
    UNIQUE,
    need_to_remove BOOLEAN NOT NULL DEFAULT FALSE,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX IF NOT EXISTS idx_slots_employee_id ON
slots(employee_id);

-- +goose Down
DROP INDEX IF EXISTS idx_slots_employee_id;
DROP TABLE IF EXISTS slots;
```

Листинг Б.8 – Файл backend/migrations/00008_create_teams_employees.sql

```
-- +goose Up
-- +goose StatementBegin
CREATE TABLE IF NOT EXISTS teams_employees (
    team_id INTEGER NOT NULL REFERENCES teams(id) ON DELETE
    CASCADE,
    employee_id INTEGER NOT NULL REFERENCES employees(id) ON
    DELETE CASCADE,
    PRIMARY KEY (team_id, employee_id)
);
-- +goose StatementEnd

-- +goose Down
-- +goose StatementBegin
DROP TABLE IF EXISTS teams_employees;
-- +goose StatementEnd
```

Приложение В
(обязательное)
Исходный код серверной части

Листинг В.1 – Файл backend/cmd/main.go

```

package main

import (
    "context"
    "log"
    "net/http"
    "os"
    "os/signal"
    "syscall"

    "crud-service/internal/balancer"
    "crud-service/internal/config"
    "crud-service/internal/crud/repository"
    "crud-service/internal/crud/service"
    "crud-service/internal/db"
    "crud-service/internal/handler"
    "crud-service/internal/workflow"
)

func main() {
    cfg := config.Load()

    ctx, cancel := context.WithCancel(context.Background())
    defer cancel()

    sig := make(chan os.Signal, 1)
    signal.Notify(sig, syscall.SIGINT, syscall.SIGTERM)
    go func() {
        <-sig
        log.Println("shutdown signal received")
        cancel()
    }()

    database, err := db.New(cfg.ConnectionString)
    if err != nil {
        log.Fatalf("Failed to connect to database: %v", err)
    }
    defer database.Close()

    employeeRepo := repository.NewEmployeeRepository(database)
    slotRepo := repository.NewSlotRepository(database)
    appealRepo := repository.NewAppealRepository(database)
    subthemeRepo := repository.NewSubthemeRepository(database)
    clientRepo := repository.NewClientRepository(database)
    themeRepo := repository.NewThemeRepository(database)
    teamRepo := repository.NewTeamRepository(database)
    pendingAppealRepo :=
repository.NewPendingAppealRepository(database)
    workflowRepo := workflow.NewWorkflowRepository(database)

```

```

    employeeSvc := service.NewEmployeeService(database,
employeeRepo, slotRepo)
    slotSvc := service.NewSlotService(database, slotRepo)
    subthemeSvc := service.NewSubthemeService(database,
subthemeRepo)
    clientSvc := service.NewClientService(database, clientRepo)
    themeSvc := service.NewThemeService(database, themeRepo)
    teamSvc := service.NewTeamService(database, teamRepo)
    workflowSvc := workflow.NewWorkflowService(workflowRepo,
teamSvc)
    var appealEventPub service.BalancerEventPublisher
    if cfg.RabbitURL != "" {
        appealEventPub =
balancer.NewRabbitPublisher(cfg.RabbitURL, cfg.RabbitQueue)
    }
    appealSvc := service.NewAppealService(database, appealRepo,
teamRepo, clientRepo, slotRepo, pendingAppealRepo, &workflowSvc,
teamSvc, appealEventPub)

    h := handler.New(employeeSvc, slotSvc, appealSvc, subthemeSvc,
clientSvc, themeSvc, teamSvc, workflowSvc)
    router := h.InitRoutes()

    balancerSvc := balancer.Services{
        AppealService: appealSvc,
        SlotService:    slotSvc,
        EmployeeRepo:   employeeRepo,
    }
    balancerErrCh := balancer.StartInBackground(ctx, *cfg,
balancerSvc)
    go func() {
        if err := <-balancerErrCh; err != nil && err !=
context.Canceled {
            log.Fatalf("balancer stopped with error: %v", err)
        }
    }()
    log.Printf("balancer: started (role=%s, queue=%s)",
cfg.BalancerRole, cfg.RabbitQueue)

    log.Printf("Starting server on %s", cfg.ServerAddr)
    if err = http.ListenAndServe(cfg.ServerAddr, router); err !=
nil {
        log.Fatalf("Failed to start server: %v", err)
    }
}

```

Листинг В.2 – Файл backend/internal/balancer/assigner.go

```

package balancer

import (
    "context"
    "crud-service/internal/crud/service"
    "database/sql"
    "encoding/json"
    "errors"
    "log"

    "github.com/hibiken/asynq"
)

type Assigner struct {
    appealService service.AppealService
}

func NewAssigner(appealService service.AppealService) *Assigner {
    return &Assigner{appealService: appealService}
}

func (a *Assigner) HandleAssignTask(ctx context.Context, t
*asynq.Task) error {
    var p AssignPayload
    if err := json.Unmarshal(t.Payload(), &p); err != nil {
        return err
    }

    err := a.appealService.Assign(p.AppealID, p.ManagerID,
p.SlotID)
    if errors.Is(err, service.ErrAppealAlreadyAssigned) {
        return nil
    }
    if errors.Is(err, sql.ErrNoRows) {
        // Слот уже занят или недоступен – повтор задачи не
        // поможет, иначе asynq делает MaxRetry попыток и спамит WARN.
        log.Printf("assigner: no free slot for manager=%d slot=%d
        appeal=%d (dropping task)", p.ManagerID, p.SlotID, p.AppealID)
        return nil
    }
    if err != nil {
        return err
    }

    log.Printf("assigner: assigned appeal=%d manager=%d slot=%d
    team=%d", p.AppealID, p.ManagerID, p.SlotID, p.TeamID)
    return nil
}

```

Приложение Г
(обязательное)
Исходный код клиентской части

Листинг Г.1 – Файл frontend/src/App.tsx

```
import { NavLink, Route, Routes } from 'react-router-dom'
import EmployeesPage from '../pages/EmployeesPage'
import ClientsPage from '../pages/ClientsPage'
import ThemesPage from '../pages/ThemesPage'
import AppealsPage from '../pages/AppealsPage'
import AppealDetailPage from '../pages/AppealDetailPage'
import SlotsPage from '../pages/SlotsPage'
import SubthemesPage from '../pages/SubthemesPage'
import TeamsPage from '../pages/TeamsPage'
import WorkflowsPage from '../pages/WorkflowsPage'

const navItems = [
  { to: '/employees', label: 'Сотрудники' },
  { to: '/clients', label: 'Клиенты' },
  { to: '/themes', label: 'Темы' },
  { to: '/subthemes', label: 'Подтемы' },
  { to: '/appeals', label: 'Обращения' },
  { to: '/slots', label: 'Ячейки' },
  { to: '/teams', label: 'Команды' },
  { to: '/workflows', label: 'Автоматизации' },
]

export default function App() {
  return (
    <div style={layout}>
      <aside style={sidebar}>
        <div style={logo}>Система распределения обращений</div>
        <nav style={{ display: 'flex', flexDirection: 'column',
gap: 4 }}>
          {navItems.map(({ to, label }) => (
            <NavLink key={to} to={to} style={({ isActive }) => ({
...navLink, ...(isActive ? navActive : {})) }}>
              {label}
            </NavLink>
          ))}
        </nav>
      </aside>

      <main style={main}>
        <Routes>
          <Route path="/" element={<EmployeesPage />} />
          <Route path="/employees" element={<EmployeesPage />} />
          <Route path="/clients" element={<ClientsPage />} />
          <Route path="/themes" element={<ThemesPage />} />
          <Route path="/appeals" element={<AppealsPage />} />
          <Route path="/appeals/:id" element={<AppealDetailPage />} />
          <Route path="/slots" element={<SlotsPage />} />
          <Route path="/subthemes" element={<SubthemesPage />} />
          <Route path="/teams" element={<TeamsPage />} />
        </Routes>
      </main>
    </div>
  )
}
```

```

        <Route path="/workflows" element={<WorkflowsPage />} />
      </Routes>
    </main>
  </div>
)
}

const layout: React.CSSProperties = {
  display: 'flex', minHeight: '100vh', fontFamily: 'system-ui,
  sans-serif', background: '#f4f5fb',
}
const sidebar: React.CSSProperties = {
  width: 220, background: '#1e2140', color: '#fff',
  display: 'flex', flexDirection: 'column', padding: '24px 16px',
  position: 'sticky', top: 0, height: '100vh',
}
const logo: React.CSSProperties = {
  fontSize: 22, fontWeight: 800, letterSpacing: 2,
  color: '#7c8cf8', marginBottom: 36, paddingLeft: 8,
}
const navLink: React.CSSProperties = {
  display: 'block', padding: '10px 14px', borderRadius: 8,
  color: '#aab0d0', textDecoration: 'none', fontSize: 14,
  fontWeight: 500,
  transition: 'background .15s, color .15s',
}
const navActive: React.CSSProperties = {
  background: '#2d3260', color: '#fff',
}
const main: React.CSSProperties = {
  flex: 1, padding: '32px 36px', overflowY: 'auto',
}
}

```

Листинг Г.2 – Файл frontend/src/main.tsx

```

import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import { BrowserRouter } from 'react-router-dom'
import App from './App'

const root = document.getElementById('root')!
createRoot(root).render(
  <StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </StrictMode>,
)

```

Приложение Д
(обязательное)
Результаты тестирования

Сотрудники						+ Добавить
ID	Имя	Фамилия	Email	Лимит	Статус	
35	Иван	Иванов	ivanov@mail.com	0	Работает	<div>Просмотр</div> <div>Удалить</div>

Рисунок Д.1 – Результат выполнения теста №1

Новый сотрудник

×

Имя

Иван

Фамилия

И

Email

ivanov@mail.com

! Значение не может состоять из одного символа

Лимит

5

Команды

☒ (1) Не распределенные ☐ (2) Не распределенные VIP

Статус

Работает

▼

Сохранить

Рисунок Д.2 – Результат выполнения теста №2

Новый сотрудник ×

Имя

Иван

Фамилия

Иии

Email

ivanov@mail.com

 Значение не должно превышать 50 символов

Лимит

5

Команды

☒ (1) Не распределенные ☐ (2) Не распределенные VIP

Статус

Работает ▼

Сохранить

Рисунок Д.3 – Результат выполнения теста №3

Новый сотрудник ×

Имя

Иван

Фамилия

-Иванов

Email

ivanov@gmail

 Введите данные в указанном формате.
Только русские буквы и дефис, дефис не может быть первым/последним

Лимит

5

Команды

☒ (1) Не распределенные ☐ (2) Не распределенные VIP

Статус

Работает ▼

Сохранить

Рисунок Д.4 – Результат выполнения теста №4

Новый сотрудник



Имя

Иван

Фамилия

Иванов-

Email

ivanov@gmail



Введите данные в указанном формате.

Только русские буквы и дефис, дефис не может быть первым/последним

Лимит

5

Команды



(1) Не распределенные



(2) Не распределенные VIP

Статус

Работает



Сохранить

Рисунок Д.5 – Результат выполнения теста №5

Новый сотрудник



Имя

Иван

Фамилия

Иван123ов

Email

ivanov@gmail



Введите данные в указанном формате.

Только русские буквы и дефис, дефис не может быть первым/последним

Лимит

5

Команды



(1) Не распределенные



(2) Не распределенные VIP

Статус

Работает



Сохранить

Рисунок Д.6 – Результат выполнения теста №6

Новый сотрудник

Имя

Иванов

Фамилия

Ъван

Email

ivanov@mail.com



Недопустимый первый символ

Лимит

5

Команды



(1) Не распределенные



(2) Не распределенные VIP

Статус

Работает



Сохранить

Рисунок Д.7 – Результат выполнения теста №7

Новый сотрудник

Имя

Иванов

Фамилия

А

Email

ivanov@mail.com



Значение не может состоять из одного символа

Лимит

5

Команды



(1) Не распределенные



(2) Не распределенные VIP

Статус

Работает



Сохранить

Рисунок Д.8 – Результат выполнения теста №8

Новый сотрудник ×

Имя

Иванов

Фамилия

Ivanov

Email

ivan

Лимит

5

Команды

☒ (1) Не распределенные

☐ (2) Не распределенные VIP

Статус

Работает

Сохранить

Введите данные в указанном формате.
Только русские буквы и дефис, дефис не может быть первым/последним

Рисунок Д.9 – Результат выполнения теста №9

Приложение Е
(обязательное)
Исходный код docker-файлов

Листинг Е.1 – Файл frontend/Dockerfile

```

FROM node:20-alpine AS builder

WORKDIR /app
ENV NODE_ENV=development \
    NPM_CONFIG_PRODUCTION=false \
    NPM_CONFIG_OMIT=

COPY package.json package-lock.json* ./
RUN npm ci

COPY . .

ARG VITE_API_URL=http://localhost:8080
ENV VITE_API_URL=$VITE_API_URL

RUN npm ls typescript vite && npm run build

FROM node:20-alpine

RUN npm install -g serve

COPY --from=builder /app/dist /app/dist

EXPOSE 3000
CMD ["serve", "-s", "/app/dist", "--listen", "tcp://0.0.0.0:3000"]

```

Листинг Е.2 – Файл backend/Dockerfile

```

FROM golang:1.26.1

WORKDIR /app

COPY go.mod go.sum ./
RUN go mod download

RUN go install github.com/pressly/goose/v3/cmd/goose@latest

COPY . .

RUN CGO_ENABLED=0 GOOS=linux go build -o main ./cmd/server/

EXPOSE 8080

CMD goose -dir ./migrations postgres "$DATABASE_URL" up && ./main

```

Приложение Ж
(рекомендуемое)
Акт о внедрении

SAWADY

Общество с ограниченной ответственностью «Савади»

(ООО «Савади»)

ИНН: 7725809338 КПП: 7725010011

115093, г. Москва, муниципальный округ Даниловский,
пер. Партийный д. 1, к. 57, стр. 3

02.06.2026 № 12

На № _____ от _____

СПРАВКА

о внедрении результатов выпускной квалификационной работы

Результаты выпускной квалификационной работы студента ФГБОУ ВО «Пензенский государственный университет» группы 22ВП1 очной формы обучения Симакина Константина Сергеевича на тему «Программные средства распределения обращений клиентов между сотрудниками технической поддержки» рассмотрены и одобрены, как имеющие практическую значимость. Разработанный автором выпускной квалификационной работы механизм распределения обращений используется в работе продукта Smartchat.

Специалист по кадровому делопроизводству



А.А. Краснова