Федеральное государственное автономное образовательное

учреждение высшего образования

«CИБИPCKИЙ ФEДEPAЛБHЫЙ УHИBEPCИTET»

Космических и информационных технологий

институт

Информационные системы

кафедра

Утверждаю

Заведующий кафедрой ИC

А.А. Раскина

подпись инициалы, фамилия

« » 2023 г.

**БAKAЛABPCKAЯ PAБOTA**

09.03.02 – «Информационные системы и технологии»

Автоматизация приема и обработки заявок отделом техподдержки

ООО «Максбонус»

Руководитель профессор, к.т.н. Ю. A. Маглинец

подпись, дата инициалы, фамилия

Выпускник П. М. Горбунов

подпись, дата инициалы, фамилия

Нормоконтролер ст.препод. ИС Ю. В. Шмагрис

подпись, дата инициалы, фамилия

Красноярск 2023

Федеральное государственное автономное образовательное

учреждение высшего образования

«CИБИPCKИЙ ФEДEPAЛБHЫЙ УHИBEPCИTET»

Космических и информационных технологий

институт

Информационные системы

кафедра

Утверждаю

Заведующий кафедрой ИC

А.А. Раскина

подпись инициалы, фамилия

« » 2023 г.

**ЗАДАНИЕ**

**НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ**

**В форме бакалаврской работ**

Студенту Горбунову Петру Михайловичу

Группа: КИ19-14Б Направление: 09.03.02 «Информационные системы и технологии»

Тема выпускной квалификационной работы: Автоматизация приема и обработки заявок отделом техподдержки ООО «Максбонус»

Утверждена приказом по университету ………..

Руководитель ВКР: Ю.А. Маглинец, профессор, кандидат технических наук, доцент кафедры Систем искусственного интеллекта ИКИТ СФУ.

Исходные материалы ВКР: …..

Перечень разделов ВКР: …..

Руководитель Ю. A. Маглинец

подпись, дата

Задание принял к исполнению П. М. Горбунов

подпись, дата

« » 2023 г.

**РЕФЕРАТ**

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc135686947)

[1 Выявление и анализ требований к процессу автоматизации приёма и обработки заявок 8](#_Toc135686948)

[1.1 Функциональные требования к системе 8](#_Toc135686949)

[1.2 Описание структуры и функций отдела 10](#_Toc135686950)

[1.3 Анализ потребностей отдела 13](#_Toc135686951)

[1.4 Формализация процессов приема и обработки заявок 14](#_Toc135686952)

[1.5 Архитектура разрабатываемой системы 16](#_Toc135686953)

[2 Разработка технического проекта системы автоматизации приёма и обработки заявок 20](#_Toc135686954)

[2.1 Обзор и сравнение технологий для реализации системы 21](#_Toc135686955)

[2.2 Проектирование сценариев взаимодействия пользователей с системой 32](#_Toc135686956)

[2.3 Описание Прецедентов. Диаграммы деятельности 34](#_Toc135686957)

[2.4 Последовательность выполнения работ в области автоматизации бизнес-процессов 39](#_Toc135686958)

[2.5 Обзор существующих решений 41](#_Toc135686959)

[2.6 Проектирование базы данных 44](#_Toc135686960)

[2.7 Обоснование архитектуры КИС 52](#_Toc135686961)

[3 Разработка и внедрение системы автоматизации приема и обработки заявок в ООО «Максбонус» 56](#_Toc135686962)

[3.1 Разработка серверной части 56](#_Toc135686963)

[3.2 Создание чат-бота телеграмм для оповещения о новых заявках 73](#_Toc135686964)

[3.3 Разработка клиентской части 76](#_Toc135686965)

[3.4 Деплой сервера на удаленный компьютер 80](#_Toc135686966)

[3.5 Подготовка сервера 80](#_Toc135686967)

[3.6 Установка Docker и Git на сервер 81](#_Toc135686968)

[3.7 Клонирование проекта с помощью Git 81](#_Toc135686969)

[3.8 Создание Dockerfile 82](#_Toc135686970)

[3.9 Сборка и развертывание контейнера 84](#_Toc135686971)

[3.10 Мониторинг и управление контейнером 84](#_Toc135686972)

[3.11 Реализация и развертывание системы: набор библиотек, оригинальный код и архитектура сети предприятия 85](#_Toc135686973)

[ЗАКЛЮЧЕНИЕ 95](#_Toc135686974)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 96](#_Toc135686975)

СЛОВАРЬ………………………………………………………………………..101

ПРИЛОЖЕНИЕ А. Акт о внедрении……………………………...…………...102

# ВВЕДЕНИЕ

В современном мире, где бизнес постоянно развивается и усложняется, автоматизация бизнес-процессов играет ключевую роль в повышении эффективности и конкурентоспособности компаний. Особое внимание уделяется автоматизации процессов приема и обработки заявок, поскольку они напрямую влияют на качество обслуживания клиентов и, следовательно, на успех компании в целом.

ООО «Максбонус» является динамично развивающейся компанией, которая стремится к оптимизации своих бизнес-процессов и улучшению качества предоставляемых услуг. В связи с этим возникает потребность в автоматизации приема и обработки заявок отделом технической поддержки.

В нынешнее время, где скорость и эффективность обработки информации имеют первостепенное значение, автоматизация рабочих процессов становится основным направлением развития многих компаний. Таким образом, тема автоматизации приема и обработки заявок отделом техподдержки ООО «Максбонус» является крайне актуальной по следующим причинам:

Увеличение эффективности работы: автоматизация рабочих процессов позволяет сократить время на обработку заявок, что способствует увеличению производительности и снижению времени ожидания для клиентов.

Снижение вероятности ошибок: автоматическая обработка заявок минимизирует вероятность человеческих ошибок, повышая качество предоставляемых услуг.

Рационализация рабочей нагрузки: автоматизация позволяет оптимизировать рабочую нагрузку сотрудников, освобождая их от рутинных задач и давая возможность сконцентрироваться на более важных и творческих аспектах работы.

Контроль и мониторинг: автоматизированные системы предоставляют удобные инструменты для контроля и мониторинга рабочих процессов, что позволяет своевременно выявлять и устранять проблемы, а также собирать статистические данные для анализа эффективности работы отдела.

Улучшение взаимодействия с клиентами: современные автоматизированные системы обработки заявок могут интегрироваться с различными каналами связи (телефон, электронная почта, мессенджеры), что обеспечивает удобство общения с клиентами и ускоряет обработку их запросов.

Экономия ресурсов: автоматизация приема и обработки заявок может значительно снизить затраты на операционные расходы и управление персоналом, что позволяет более эффективно использовать ресурсы компании и сокращать издержки.

В целом, автоматизация приема и обработки заявок отделом техподдержки ООО «Максбонус» является важным шагом в развитии компании, поскольку это позволяет оптимизировать рабочие процессы, повышать эффективность и качество обслуживания клиентов, а также обеспечивать более высокий уровень контроля над внутренними операциями. Внедрение автоматизации в работу отдела будет способствовать росту конкурентоспособности ООО «Максбонус» на рынке, облегчать адаптацию к переменам и инновациям, а также улучшать управление ресурсами и мотивацию сотрудников.

Целью данной работы является разработка и внедрение системы автоматизации приема и обработки заявок для ООО «Максбонус», которая сократит время обработки заявок, уменьшить вероятность ошибок, связанных с человеческим фактором, и повысить удовлетворенность клиентов за счет более быстрого и качественного обслуживания.

Для достижения поставленной цели в работе предполагается выполнить следующие задачи:

1. Выявление и анализ требований к процессу автоматизации приёма и обработки заявок.
2. Разработка технического проекта системы автоматизации приёма и обработки заявок.
3. Разработка и внедрение системы автоматизации приема и обработки заявок в ООО «Максбонус».

# 1 Выявление и анализ требований к процессу автоматизации приёма и обработки заявок

## **1.1 Функциональные требования к системе**

Система автоматизации приема и обработки заявок отделом техподдержки ООО "Максбонус" должна удовлетворять следующим требованиям, основанным на подходе FURPS+:

1. Требования совладельцев:
2. Возможность контроля и анализа работы отдела техподдержки: совладельцы должны иметь возможность получать статистику о количестве обработанных заявок, среднем времени их обработки, а также оценку эффективности работы сотрудников [1].
3. Экономия времени и ресурсов при обработке заявок: автоматизация процесса обработки заявок должна сократить время на выполнение задач и уменьшить нагрузку на сотрудников [1].
4. Рост уровня удовлетворенности клиентов: быстрое и качественное решение проблем клиентов должно повысить их удовлетворенность услугами компании.
5. Требования пользователей:
6. Интуитивно понятный интерфейс: система должна обладать простым и ясным интерфейсом, позволяющим пользователям легко освоить работу с ней [1].
7. Возможность быстрого создания и обработки заявок: пользователи должны иметь возможность без труда создавать новые заявки, а также просматривать и обрабатывать существующие [1].
8. Доступ к истории заявок и их статусам: пользователи должны иметь возможность просматривать историю обработки заявок, а также отслеживать изменение их статусов.
9. Функциональные требования [1]:
10. Регистрация и авторизация пользователей: система должна предоставлять возможность создания учетных записей и идентификации пользователей.
11. Создание, редактирование и удаление заявок: пользователи должны иметь возможность создавать, редактировать и удалять заявки, указывая все необходимые данные и прикрепляя файлы.
12. Назначение исполнителей и изменение статуса заявок: система должна предоставлять возможность назначения исполнителя для каждой заявки и изменения статуса заявки на различных этапах обработки.
13. Возможность комментирования: пользователи должны иметь возможность обмениваться информацией, обеспечивая эффективное взаимодействие между сотрудниками [1].
14. Нефункциональные требования [1]:
15. Быстродействие: система должна быстро обрабатывать запросы и загружать данные, обеспечивая минимальное время ожидания для пользователей.
16. Безопасность: система должна обеспечивать защиту данных пользователей и соблюдение требований по конфиденциальности, включая шифрование данных и использование сильных алгоритмов аутентификации [1].
17. Масштабируемость: система должна быть способна адаптироваться к изменениям, связанным с ростом количества пользователей и объема обрабатываемых заявок, а также к дополнительному функционалу, если потребуется.
18. Надежность: система должна гарантировать непрерывную работу и сохранность данных, предотвращая возникновение сбоев и потерю информации [3].

FURPS+ модель включает следующие аспекты [3]:

* Functionality (функциональность): система должна предоставлять все необходимые функции для управления заявками и контроля работы техподдержки, соответствующие требованиям совладельцев и пользователей.
* Usability (удобство использования): система должна иметь интуитивный интерфейс и предоставлять доступ к обучающим материалам для пользователей, облегчая освоение работы с ней.
* Reliability (надежность): система должна обеспечивать стабильную работу и сохранность данных, предотвращая возникновение сбоев и потерю информации.
* Performance (производительность): быстрое выполнение операций и загрузка данных, а также возможность одновременной работы нескольких пользователей без снижения производительности системы.
* Supportability (поддерживаемость): система должна быть способна к доработке и обновлению в будущем, обеспечивая поддержку изменяющихся потребностей компании и пользователей.

## **1.2 Описание структуры и функций отдела**

Отдел техподдержки ООО «Максбонус» является ключевым звеном, отвечающим за эффективное взаимодействие между клиентами и организацией. Структура и функции отдела могут быть следующими:

* Менеджер по приему заявок: Менеджеры по приему заявок отвечают за первичное общение с клиентами, получение и регистрацию заявок, а также предоставление информации о статусе заявок. Они также могут заниматься консультированием клиентов по продуктам и услугам компании.
* Специалист по обработке заявок: Специалисты по обработке заявок анализируют полученные заявки, определяют их приоритетность и распределяют их между другими сотрудниками отдела или другими подразделениями компании. Они также следят за выполнением заявок и контролируют сроки их исполнения.
* Специалист по координации: Специалист по координации отвечает за организацию взаимодействия между сотрудниками отдела и другими подразделениями компании. Он также может участвовать в процессе принятия решений, связанных с обработкой и выполнением заявок.
* Аналитик: Аналитик отвечает за сбор, анализ и интерпретацию данных о заявках, процессах и результате работы отдела. Он также предлагает рекомендации по оптимизации процессов и повышению эффективности работы отдела.
* Специалист по поддержке и обучению: Специалист по поддержке и обучению проводит тренинги и обучение сотрудников отдела, помогает им развивать профессиональные навыки и знания, а также оказывает техническую поддержку в случае возникновения вопросов или проблем.
* Руководитель отдела: Руководитель отдела координирует работу всех сотрудников, контролирует выполнение задач и сроки их исполнения, принимает решения о распределении ресурсов и приоритетах работы. Руководитель также отвечает за планирование и стратегическое развитие отдела, а также за коммуникацию с вышестоящим руководством компании и другими отделами.
* ИТ-специалист: ИТ-специалист отвечает за техническую поддержку и обслуживание автоматизированной системы приема и обработки заявок. Он также участвует в разработке и оптимизации системы, обеспечивая ее надежность, безопасность и эффективность.

В целом, отдел техподдержки ООО «Максбонус» играет важную роль в обеспечении эффективной работы компании и удовлетворении потребностей клиентов. Создание и поддержание автоматизированной системы для приема и обработки заявок позволяет отделу сосредоточиться на основных задачах и улучшать качество предоставляемых услуг.

Основные функции отдела автоматизации приема и обработки заявок включают:

* Прием и регистрация заявок от клиентов.
* Обработка и распределение заявок между сотрудниками и другими подразделениями.
* Контроль исполнения заявок и сроков их выполнения.
* Поддержание связи с клиентами и предоставление информации о статусе их заявок.
* Анализ и оптимизация процессов обработки заявок.
* Разработка, внедрение и поддержка автоматизированной системы для приема и обработки заявок.
* Обучение и поддержка сотрудников отдела.

Компетентный и эффективный отдел техподдержки является важным фактором успеха компании ООО «Максбонус» и позволяет обеспечить высокий уровень обслуживания и удовлетворенности клиентов.

## **1.3 Анализ потребностей отдела**

Проведение анализа потребностей отдела техподдержки компании ООО «Максбонус» является важным этапом разработки и внедрения автоматизированной системы. Анализ потребностей позволяет выявить ключевые проблемы и требования отдела, которые должны быть учтены при разработке системы. Вот некоторые основные аспекты, которые следует проанализировать:

* Объем и типы заявок: Определение количества и видов заявок, которые обрабатывает отдел, поможет определить масштаб системы и функциональные требования к ней. Это также позволив эффективнее определить возможные узкие места и оптимизировать процессы обработки заявок.
* Роли и обязанности сотрудников: Изучение ролей и обязанностей сотрудников отдела создаст условия для определения, какие функции и возможности должны быть доступны для разных пользователей системы. Это также поможет определить, какие процессы должны быть автоматизированы, и какие должны остаться под контролем сотрудников.
* Безопасность и доступ к данным: Оценка требований к безопасности и контролю доступа к данным поможет определить, какие меры защиты должны быть реализованы для обеспечения надежной и безопасной работы системы.
* Требования к отчетности и аналитике: Определение потребностей отдела в аналитике и отчетности поможет определить, какие инструменты и функции анализа данных должны быть включены в систему, и какие данные должны быть собраны и хранены для создания отчетов.

Проведение тщательного анализа потребностей отдела техподдержки компании ООО «Максбонус» сделает возможным определить и разработать эффективное и гибкое решение, которое удовлетворит требования отдела и обеспечит улучшение его работы. При анализе потребностей следует учитывать не только текущие требования, но и возможное будущее развитие компании и отдела, чтобы система могла масштабироваться и адаптироваться к изменяющимся условиям.

После проведения анализа потребностей отдела автоматизации приема и обработки заявок и определения основных требований к системе, следующим шагом будет разработка технического задания и выбор подходящих технологий для реализации автоматизированной системы. Важно выбирать проверенные и надежные технологии, которые обеспечат стабильную работу системы и возможность ее развития в будущем.

## **Формализация процессов приема и обработки заявок**

Формализация процессов приема и обработки заявок в отделе техподдержки ООО «Максбонус» предполагает создание четких и структурированных процедур, которые будут использоваться сотрудниками для выполнения своих задач. Формализация этих процессов позволяет обеспечить эффективность, снизить вероятность ошибок и упростить автоматизацию. Вот основные этапы формализации процессов:

* Определение основных этапов обработки заявок: выделение ключевых этапов, которые включают прием, регистрацию, обработку, распределение, исполнение и закрытие заявок.
* Разработка процедур для каждого этапа: создание детализированных инструкций для выполнения каждого этапа процесса обработки заявок. Определение критериев и порядок выполнения задач, а также ответственных сотрудников.
* Определение параметров заявок: установите основные атрибуты, которые будут использоваться для хранения информации о заявках, такие как идентификатор, дата создания, статус, тип, приоритет, срок исполнения, контактные данные клиента и дополнительные поля, необходимые для обработки заявок.
* Разработка правил распределения заявок: определите критерии и механизмы распределения заявок между сотрудниками отдела или другими подразделениями компании. Это может включать автоматическое распределение на основе навыков, нагрузки или приоритетов заявок.
* Установление критериев и процедур контроля качества: определите стандарты качества для обработки заявок и разработайте процедуры для контроля их соблюдения. Это может включать регулярные проверки выполненных заявок, анализ причин возникновения ошибок и корректирующие меры для их устранения.
* Создание механизмов отслеживания и анализа процессов: разработайте методы сбора данных о процессах обработки заявок, их продолжительности, эффективности и других параметрах, которые позволят проводить анализ и оптимизацию процессов.
* Документирование процессов: зафиксируйте все процедуры и правила в виде документации, которая будет доступна для сотрудников отдела. Документация должна быть понятной, структурированной и актуализированной. Это сделает возможным сотрудникам легко ознакомиться с процессами, обучиться им и соблюдать требования.
* Обучение сотрудников: проведите обучение сотрудников отдела по новым формализованным процессам. Обучение должно быть комплексным и включать теоретическую часть, практические упражнения и постоянную поддержку в освоении новых процедур.
* Внедрение и мониторинг: внедрите формализованные процессы в рабочую практику отдела и регулярно контролируйте их выполнение.
* Постоянное совершенствование: проведение периодического анализа процессов обработки заявок с целью выявления возможных улучшений и оптимизации.

Формализация процессов приема и обработки заявок в отделе техподдержки компании ООО «Максбонус» сделает возможным определить и улучшить эффективность работы отдела, упростить автоматизацию процессов и обеспечить более высокий уровень удовлетворенности клиентов. Кроме того, это снизит разработку и внедрение автоматизированной системы приема и обработки заявок, так как процессы будут четко структурированы и документированы.

## **1.5 Архитектура разрабатываемой системы**

В рамках дипломной работы по автоматизации приема и обработки заявок отделом техподдержки ООО «Максбонус» предлагается использовать следующие технологии:

* База данных: PostgreSQL - мощная и надежная система управления базами данных с открытым исходным кодом. PostgreSQL поддерживает продвинутые функции, такие как транзакции, хранимые процедуры, индексы и предоставляет высокую производительность и масштабируемость.
* Серверная часть: Go (Golang) - современный язык программирования, разработанный Google, обеспечивающий высокую скорость разработки и производительность приложений. Go хорошо подходит для создания серверных приложений, таких как RESTful API, благодаря своей конкурентности, простоте и отличной поддержке стандартной библиотекой.
* Фреймворк для серверной части: Echo или Gin - популярные и быстрые фреймворки на Go для создания веб-приложений и RESTful API. Оба фреймворка предоставляют удобные маршрутизаторы, middleware и инструменты для работы с запросами и ответами.
* Фронтенд: React - популярная библиотека для разработки пользовательских интерфейсов, созданная Facebook. React позволяет разрабатывать компоненты интерфейса с высокой степенью пере использования и обеспечивает быстрое обновление интерфейса благодаря виртуальному DOM.
* Среда разработки и инструменты: Visual Studio Code, Git, GitHub/GitLab, Node.js, npm/yarn для управления пакетами и зависимостями, а также инструменты для тестирования и сборки проекта, такие как Jest, Webpack или Create React App.

Использование указанных технологий баз данных, фронтенда и сервера позволит более четко определить, разработать и внедрить эффективную, масштабируемую и надежную автоматизированную систему для приема и обработки заявок отделом ООО «Максбонус». Комбинация этих технологий обеспечит высокую производительность системы, упрощенное взаимодействие между компонентами и гибкость для будущих изменений и расширения функционала.

* Разработка автоматизированной системы с использованием таких технологий приведет к следующим преимуществам:
* Быстрое время разработки: благодаря мощным библиотекам и фреймворкам, разработчики смогут быстрее создавать и внедрять систему.
* Повышенная надежность: Использование проверенных и надежных технологий обеспечит стабильность и безопасность работы системы.
* Масштабируемость: Технологии, такие как PostgreSQL и Go, обеспечивают высокую масштабируемость и возможность увеличения нагрузки на систему без существенных изменений в архитектуре.
* Простота поддержки и обновления: Четкая разделенность на клиентскую и серверную части, а также использование компонентного подхода в React, облегчает процесс обновления и поддержки системы.
* Переиспользуемость компонентов: React позволяет создавать пере используемые компоненты, что упрощает разработку и сокращает время на создание пользовательских интерфейсов.
* Улучшенное взаимодействие с пользователями: Современные технологии фронтенда, такие как React, позволяют создавать интуитивно понятные и удобные интерфейсы для пользователей, что в свою очередь повышает уровень удовлетворенности сотрудников отдела и клиентов компании.

В результате использования этих технологий баз данных, фронтенда и сервера будет создана автоматизированная система, которая уменьшит процесс приема и обработки заявок, сократит время обработки, уменьшит количество ошибок и повысит общую эффективность работы отдела техподдержки ООО «Максбонус».

**Вывод к главе 1**

В ходе данной главы были определены основные функциональные требования к системе автоматизации приема и обработки заявок отделом техподдержки ООО "Максбонус", включая авторизацию пользователей, управление заявками, назначение исполнителей, комментирование.

Следующим шагом была проанализирована структура и функции отдела техподдержки, что позволило более точно определить требования к системе с учетом особенностей работы отдела и взаимодействия между сотрудниками.

На основе анализа потребностей отдела были выявлены ключевые проблемы и слабые места в текущем процессе обработки заявок. Основные потребности отдела заключаются в ускорении процесса обработки заявок, оптимизации взаимодействия между сотрудниками и повышении уровня удовлетворенности клиентов.

Таким образом, результаты анализа требований и потребностей отдела техподдержки в главе 1 позволят разработать систему автоматизации, которая будет соответствовать выявленным требованиям и обеспечит эффективное выполнение задач отдела. Это является основой для дальнейшего проектирования и разработки системы, внедрения и обучения пользователей.

# 2 Разработка технического проекта системы автоматизации приёма и обработки заявок

При разработке технического проекта необходимо учесть функциональные возможности, графические средства представления сценариев взаимодействия.

Для представления функциональных возможностей применяется диаграмма вариантов использования (Use Cases), которая отражает отношения между акторами и вариантами использования.

В качестве графического средства представления сценариев взаимодействия используются диаграммы деятельности, представляющая собой блок-схему, которая показывает алгоритм действий, выполняющихся в системе. В отличие от традиционной блок-схемы на диаграмме деятельности можно показать параллельно выполняемые действия и распределение ролей.

Актор представляет собой множество связанных ролей, которые взаимодействуют с прецедентами, влияют на систему и используют её функциональные возможности для достижения определенных целей.

Прецедент в свою очередь является особенностью последовательности действий при проектировании программных систем.

Для наглядного представления структуры данных, их взаимосвязи и описания процесса хранения в системе данных с заданиями и ответами на них, необходимо построить модель данных.

В системе необходимо отслеживание заявок и хода выполнения работы. Для заявки, используемой в системе, нужно описать схему маршрута. И схему маршрута относительно пользователей.

## **2.1 Обзор и сравнение технологий для реализации системы**

PostgreSQL — это объектно-реляционная система управления базами данных (ОРСУБД, ORDBMS), основанная на POSTGRES, Version 4.2 — программе, разработанной на факультете компьютерных наук Калифорнийского университета в Беркли. В POSTGRES появилось множество новшеств, которые были реализованы в некоторых коммерческих СУБД гораздо позднее [4].

PostgreSQL — СУБД с открытым исходным кодом, основой которого был код, написанный в Беркли. Она поддерживает большую часть стандарта SQL и предлагает множество современных функций [4].

Функциональные возможности PostgreSQL:

* Транзакции: поддержка ACID-транзакций для надежной обработки данных.
* MVCC: многопользовательский контроль согласованности версий, обеспечивающий параллельный доступ к данным.
* Типы данных: разнообразие встроенных и пользовательских типов данных.
* Индексирование: поддержка различных типов индексов (B-Tree, Hash, GiST, SP-GiST, GIN).
* Расширяемость: возможность добавления новых типов данных, функций, операторов, агрегатных функций и индексных методов.
* Процедурное программирование: поддержка встроенных языков (PL/pgSQL, PL/Tcl, PL/Perl) и возможность использования внешних языков.
* Управление правами доступа: гибкая система управления разрешениями и привилегиями пользователей.
* Представления и материализованные представления: возможность создания виртуальных таблиц и кеширования результатов запросов.
* Триггеры и правила: автоматическое выполнение действий при изменении данных в таблицах.
* Функции окна: поддержка аналитических и агрегатных функций для обработки данных.

Microsoft SQL Server — это РСУБД (система управления реляционными базами данных), которая разработана специалистами из Microsoft. В качестве основного языка запросов используется Transact-SQL (совместная разработка Microsoft и Sybase). При этом Transact-SQL — это реализация стандарта ANSI/ISO по SQL (структурированному языку запросов), но имеющая некоторые расширения. Сегодня СУБД MS SQL широко применяется при работе с базами данных (БД) самых разных размеров, начиная от персональных, заканчивая крупными БД масштаба предприятия [5].

Функциональные возможности Microsoft SQL Server (MSSQL):

* Транзакции: поддержка ACID-транзакций для надежной обработки данных.
* Разнообразие типов данных: встроенные типы данных (числовые, строковые, бинарные, временные, пространственные) и возможность создания пользовательских типов данных.
* Индексирование: поддержка различных типов индексов (кластерные, некластерные, фильтрованные, столбцовые и другие).
* Хранимые процедуры и функции: возможность создания хранимых процедур и функций для обработки данных на стороне сервера.
* Триггеры: автоматическое выполнение действий при изменении данных в таблицах.
* Представления: создание виртуальных таблиц для представления данных из одной или нескольких таблиц.
* Процедурное программирование: поддержка языка T-SQL для создания сложных запросов и процедур.
* Управление доступом: гибкая система управления правами и разрешениями пользователей.
* Функции окна: поддержка аналитических и агрегатных функций для обработки данных.
* Полнотекстовый поиск: инструменты для эффективного поиска текстовых данных.

MySQL - свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию MySQL AB. Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей. Именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации [6].

MySQL является решением для малых и средних приложений. Входит в состав серверов WAMP, AppServ, LAMP и в портативные сборки серверов Денвер, XAMPP, VertrigoServ. Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы [6].

Функциональные возможности MySQL:

* Транзакции: поддержка ACID-транзакций для надежной обработки данных.
* Типы данных: разнообразие встроенных типов данных, таких как числовые, строковые, бинарные, временные и пространственные.
* Индексирование: поддержка различных типов индексов (B-Tree, Hash, Full-text и другие).
* Хранимые процедуры и функции: возможность создания хранимых процедур и функций для обработки данных на стороне сервера.
* Триггеры: автоматическое выполнение действий при изменении данных в таблицах.
* Представления: создание виртуальных таблиц для представления данных из одной или нескольких таблиц.
* Управление доступом: гибкая система управления правами и разрешениями пользователей.
* Функции окна: поддержка аналитических и агрегатных функций для обработки данных.
* Репликация и масштабирование: возможность создания резервных копий и распределенных систем для повышения производительности и надежности.
* Полнотекстовый поиск: инструменты для эффективного поиска текстовых данных.

Go (часто также golang) — компилируемый многопоточный язык программирования, разработанный внутри компании Google. Разработка Go началась в сентябре 2007 года, его непосредственным проектированием занимались Роберт Гризмер, Роб Пайк и Кен Томпсон, занимавшиеся до этого проектом разработки операционной системы Inferno. Официально язык был представлен в ноябре 2009 года. На данный момент поддержка официального компилятора, разрабатываемого создателями языка, осуществляется для операционных систем FreeBSD, OpenBSD, Linux, macOS, Windows, DragonFly BSD, Plan 9, Solaris, Android, AIX.[10]. Также Go поддерживается набором компиляторов gcc, существует несколько независимых реализаций [7].

Функциональные возможности Golang:

* Статическая типизация: строгая система типов для обнаружения ошибок на этапе компиляции.
* Компиляция: быстрая компиляция в машинный код с нативной поддержкой множества платформ.
* Простота синтаксиса: чистый и простой синтаксис, облегчающий чтение и написание кода.
* Быстрая работа: высокая производительность благодаря компиляции в машинный код и оптимизации выполнения.
* Поддержка многопоточности: встроенные функции для управления параллельным выполнением кода (горутины и каналы).
* Сборка мусора: автоматическое управление памятью с минимальными задержками.
* Интерфейсы: гибкая система интерфейсов для упрощения проектирования и повторного использования кода.
* Стандартная библиотека: обширная стандартная библиотека, включающая пакеты для работы с файлами, сетью, криптографией и другими.
* Встроенные инструменты: инструменты для управления зависимостями, тестирования, отладки, форматирования кода и других задач разработки.
* Сообщество и экосистема: активное сообщество разработчиков и обширный набор сторонних библиотек и инструментов.

Node.js - Node или Node.js — программная платформа, основанная на движке V8 (компилирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API, написанный на C++, подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения (при помощи NW.js, AppJS или Electron для Linux, Windows и macOS) и даже программировать микроконтроллеры (например, tessel, low.js и Espruino). В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом [8].

Функциональные возможности Node.js:

* Асинхронное программирование: поддержка асинхронных функций и обратных вызовов для быстрой и эффективной обработки запросов.
* Событийно-ориентированный подход: основа на событиях и обработчиках событий для создания масштабируемых приложений.
* Модульная система: поддержка модулей CommonJS для организации и управления кодом.
* Стандартная библиотека: обширная стандартная библиотека, включающая пакеты для работы с файлами, сетью, потоками и другими.
* NPM: менеджер пакетов Node Package Manager (NPM) для управления зависимостями и установки сторонних библиотек.
* Кросс-платформенность: поддержка различных операционных систем, включая Windows, macOS и Linux.
* Интеграция с базами данных: поддержка различных баз данных через сторонние драйверы и библиотеки (например, MySQL, PostgreSQL, MongoDB и другие).
* Веб-разработка: создание веб-серверов и API с помощью встроенных модулей и популярных фреймворков (Express.js, Koa, Fastify и другие).
* Микросервисы: возможность разработки микросервисных архитектур с использованием сообщений и событий.
* Интеграция с фронтендом: простота интеграции с фронтенд-технологиями, такими как React, Angular и Vue.js.

C# Web API (Интерфейс программирования приложений) представляет собой набор HTTP-сервисов, способствующих созданию веб-приложений, совместимых с разнообразными клиентами, включая браузеры, мобильные устройства и другие серверы. Этот API использует общепринятые протоколы и форматы, например HTTP, JSON и XML, для обмена информацией между сервером и клиентами [9].

Web API является составляющей ASP.NET, фреймворка для разработки веб-приложений, созданного Microsoft. Разработчики могут использовать Web API для создания веб-служб, которые легко масштабируются и интегрируются с другими системами [9].

Функциональные возможности C# Web API:

* RESTful сервисы: поддержка принципов REST для создания масштабируемых веб-API.
* HTTP-маршрутизация: гибкая система маршрутизации для определения URL-шаблонов и привязки методов контроллеров.
* Ответы и запросы в форматах JSON и XML: встроенная поддержка сериализации и десериализации данных в JSON и XML.
* Атрибуты маршрутизации: возможность использования атрибутов для определения маршрутов и ограничений непосредственно в коде контроллеров.
* Модель привязки данных: автоматическое связывание данных запросов с параметрами действий контроллеров.
* Фильтры действий: настраиваемые фильтры для выполнения кода до или после выполнения действий контроллеров (например, для авторизации или обработки ошибок).
* Открытые и связываемые модели данных: поддержка стандарта OData для создания более гибких и настраиваемых API.
* Встроенная поддержка тестирования: инструменты и методы для тестирования веб-API в изоляции.
* Интеграция с ASP.NET Core: возможность использовать ASP.NET Core для создания кроссплатформенных веб-API.
* Расширяемость: множество встроенных и сторонних пакетов для расширения функциональности веб-API, таких как авторизация, кеширование, версионирование и многое другое.

React - это библиотека JavaScript с открытым кодом для создания внешних пользовательских интерфейсов. В отличие от других библиотек JavaScript, предоставляющих полноценную платформу приложений, React ориентируется исключительно на создание представлений приложений через инкапсулированные единицы (называются компонентами), которые сохраняют состояние и генерируют элементы пользовательского интерфейса [10].

Функциональные возможности React:

* Компонентный подход: создание пере используемых компонентов с четко определенными свойствами и состояниями.
* Виртуальный DOM: оптимизация работы с DOM-деревом за счет использования виртуального DOM для эффективного обновления интерфейса.
* JSX: синтаксическое расширение JavaScript, позволяющее объединять HTML и JavaScript код в одном файле.
* Управление состоянием: локальное управление состоянием компонентов и интеграция с внешними решениями, такими как Redux или MobX, для глобального управления состоянием.
* Жизненный цикл компонентов: встроенные методы для управления различными стадиями жизни компонента, включая монтирование, обновление и размонтирование.
* Контекст: обеспечение доступа к глобальным данным приложения без явной передачи пропсов через дерево компонентов.
* React Hooks: функциональность для работы со состоянием и жизненным циклом в функциональных компонентах.
* Оптимизация производительности: возможность использования методов и инструментов, таких как React.memo и useMemo, для предотвращения ненужных обновлений компонентов.
* Серверный рендеринг: поддержка рендеринга React-приложений на сервере для быстрой загрузки страницы и улучшения SEO.
* Экосистема и интеграция: обширная экосистема плагинов, библиотек и инструментов для расширения функциональности, а также интеграция с другими фреймворками и библиотеками, такими как Redux, React Router и Material-UI.

Vue.js — это интерфейсная платформа с открытым кодом для создания пользовательских интерфейсов и одностраничных веб-приложений на JavaScript. Эта платформа была выпущена в 2014 году. Ее поддерживают создатель Эван Ю (Evan You) и члены его основной команды. Во Vue основное внимание уделяется декларативной отрисовке и созданию компонентов, а основная библиотека предлагается только для слоя представления [11].

Функциональные возможности Vue.js:

* Реактивность: автоматическое отслеживание изменений в данных и обновление DOM в соответствии с этими изменениями.
* Компонентный подход: создание переиспользуемых компонентов с локальным состоянием и свойствами.
* Декларативный синтаксис: использование директив и интерполяции для простого описания желаемой структуры DOM и привязки данных.
* Vue CLI: командная строка для быстрого создания и настройки проектов с предустановленными шаблонами и плагинами.
* Управление состоянием: интеграция с Vuex для централизованного управления состоянием приложения.
* Vue Router: официальный маршрутизатор для создания навигации и управления историей браузера в одностраничных приложениях.
* Поддержка серверного рендеринга: использование Nuxt.js или Vue Server Renderer для рендеринга приложений на сервере для улучшения производительности и SEO.
* Поддержка TypeScript: интеграция с TypeScript для статической типизации и улучшения поддержки разработки.
* Переходы и анимации: встроенная поддержка переходов и анимаций для эффектных изменений состояния интерфейса.
* Обширная экосистема: множество сторонних библиотек, плагинов и инструментов для расширения функциональности и интеграции с другими технологиями.

Angular — это популярный открытый фреймворк для создания динамических веб-приложений, разработанный и поддерживаемый Google. Он основан на компонентной архитектуре и предоставляет разработчикам набор инструментов для создания масштабируемых и производительных одностраничных приложений (SPA). Angular использует TypeScript, строго типизированный язык программирования, основанный на JavaScript, что упрощает разработку и облегчает отладку кода [12].

Функциональные возможности Angular в виде списка:

* Компонентный подход: создание модульных и переиспользуемых компонентов для управления представлением и логикой приложения.
* Двусторонняя привязка данных: автоматическое синхронизирование данных между моделью и представлением, обеспечивая быструю и простую реакцию на изменения данных.
* Инъекция зависимостей: механизм для управления зависимостями между различными частями приложения, облегчающий разработку, тестирование и поддержку кода.
* Модульность: организация кода в модули для улучшения читаемости, поддерживаемости и повторного использования.
* Angular CLI: командная строка для создания, настройки и развертывания Angular-приложений с минимальными усилиями.
* Angular Router: мощный маршрутизатор для управления навигацией и состоянием в одностраничных приложениях.
* Формы: инструменты для создания и валидации форм, включая реактивные формы и шаблонные формы.
* Серверный рендеринг: поддержка Angular Universal для рендеринга приложений на сервере для улучшения производительности и SEO.
* Поддержка PWA: инструменты и интеграция для создания Progressive Web Apps с использованием Angular.
* Обширная экосистема: множество сторонних библиотек, плагинов и инструментов для расширения функциональности и интеграции с другими технологиями.

Приоритетные технологии для реализации системы включают PostgreSQL, Go и React. Вот краткое сравнение этих технологий:

* PostgreSQL - мощная и надежная реляционная база данных с широкими возможностями и поддержкой сложных запросов. Она является открытой базой данных, что означает, что она бесплатна и имеет активное сообщество разработчиков, которое обеспечивает ее развитие и поддержку.
* Go - компилируемый язык программирования, который известен своей производительностью, простотой и надежностью. Он предназначен для создания эффективных и распределенных систем, что делает его отличным выбором для разработки серверных приложений и микросервисов.
* React - популярный JavaScript фреймворк для создания пользовательских интерфейсов. Он позволяет разрабатывать компоненты, которые обновляются эффективно и реагируют на изменения данных. React имеет большое сообщество разработчиков и обширную экосистему, что делает его идеальным выбором для разработки современных и интерактивных веб-приложений.

Комбинация PostgreSQL, Go и React предоставляет мощные инструменты для разработки полнофункциональной системы. PostgreSQL обеспечит надежное хранение данных, Go позволит создавать эффективные серверные компоненты, а React обеспечит гибкость и привлекательный пользовательский интерфейс.

## **2.2 Проектирование сценариев взаимодействия пользователей с системой**

Прежде, чем перейти к рассмотрению сценариев, рассмотрим основные функциональные возможности системы (рис. 1)

Диаграмма вариантов использования наглядно показывает с какими вариантами использования сталкивается актор во время того или иного прецедента.

В диаграмме вариантов использования показаны следующие функции реализуемого проекта, такие как: регистрация, авторизация, создание заявки, приём заявок, регистрация заявок, обработка и распределение заявок между сотрудниками и другими подразделениями, контроль исполнения и сроков выполнения заявок.

Для проекта акторами будут являться: Клиент, Менеджер по приему заявок, Специалист по обработке заявок, Специалист по координации, Руководитель отдела.

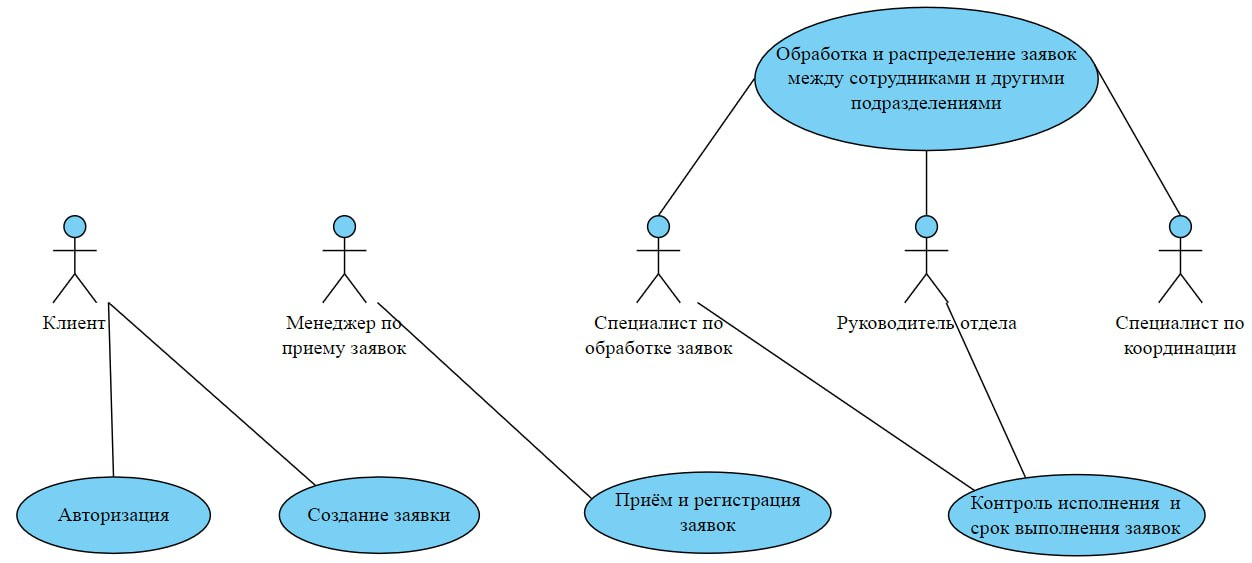


Рисунок 1 – Диаграмма вариантов использования

В таблице 1 отражены все прецеденты, акторы и их формулировка, реализованные в проекте.

Таблица 1 – Акторы и Прецеденты проекта

| Актор | Наименование | Формулировка |
| --- | --- | --- |
| Клиент | Авторизация | Клиент входит в систему под своими учётными данными |
| Клиент | Создание заявки | Клиент создаёт заявку |
| Менеджер по приему заявок | Приём заявок | Менеджер по приему заявок принимает созданную заявку |
| Менеджер по приему заявок | Регистрация заявок | Менеджер по приему заявок регистрирует поданные заявки |
| Специалист по обработке заявок | Обработка и распределение заявок между сотрудниками и другими подразделениями | Специалист по обработке заявок анализирует полученные заявки, определяет их приоритетность и распределяют их между другими сотрудниками отдела или другими подразделениями компании |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| Актор | Наименование | Формулировка |
| Специалист по обработке заявок | Контроль исполнения и сроков выполнения заявок | Специалист по обработке заявок следит за выполнением заявок и сроками их исполнения |
| Специалист по координации | Обработка и распределение заявок между сотрудниками и другими подразделениями | Специалист по координации отвечает за организацию взаимодействия между сотрудниками отдела и другими подразделениями компании |
| Руководитель отдела | Обработка и распределение заявок между сотрудниками и другими подразделениями | Руководитель отдела координирует работу всех сотрудников |
| Руководитель отдела | Контроль исполнения и сроков выполнения заявок | Руководитель отдела контролирует выполнение заявок и сроки их исполнения |

## **2.3 Описание Прецедентов. Диаграммы деятельности**

1. Прецедент – Авторизация
   1. Краткое описание

Клиент проходит авторизацию и входит в систему под своим именем.

Действующие лица Прецедента – Клиент.

* 1. Поток событий

Прецедент начинается, когда Клиент вводит учётные данные для входа в систему.

* 1. Базовый поток – Авторизация
     1. Пользователь вводит учётные данные для входа в систему
     2. Обновляется страница и заходит в систему

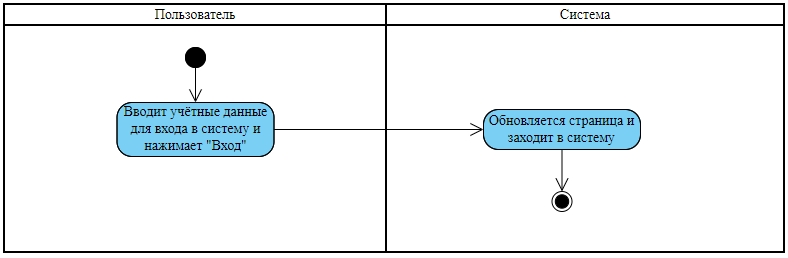


Рисунок 3 – Диаграмма деятельности для прецедента «Авторизация»

1. Прецедент «Создание заявки»
   1. Краткое описание

Клиент создаёт новую заявку.

Действующие лица Прецедента – Клиент

* 1. Предусловия
     1. Авторизация: перед тем, как начать данный Прецедент, Клиент должен авторизоваться в системе
  2. Поток событий

Прецедент начинается, когда Клиент вводит необходимую информацию для создания заявки.

* 1. Базовый поток – Создание заявки
     1. Пользователь вводит необходимую информацию для создания заявки
     2. Пользователь отправляет заполненную форму заявки

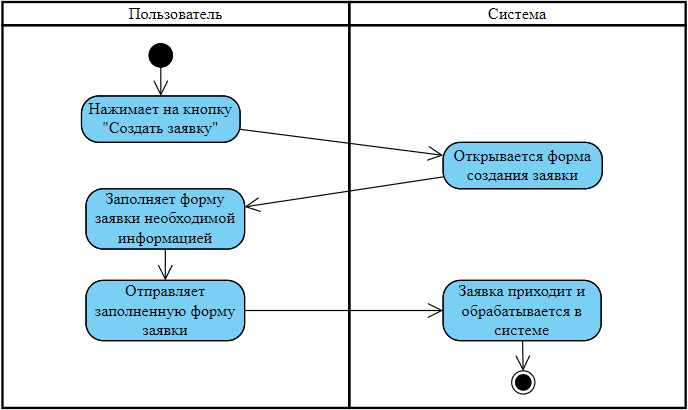


Рисунок 4 – Диаграмма деятельности для прецедента «Создание заявки»

1. Прецедент «Приём и регистрация заявок»
   1. Краткое описание

Менеджер по приему заявок принимает в работу созданную заявку.

Действующие лица Прецедента – Клиент

* 1. Поток событий

Прецедент начинается, когда Менеджер по приему заявок после регистрации заявки в системе принимает в работу созданную заявку.

* 1. Базовый поток – Приём и регистрация заявок
     1. Созданная заявка получает регистрационный номер в системе
     2. Пользователь принимает заявку в работу созданную заявку, проставляя статус заявки на «Принят в работу»

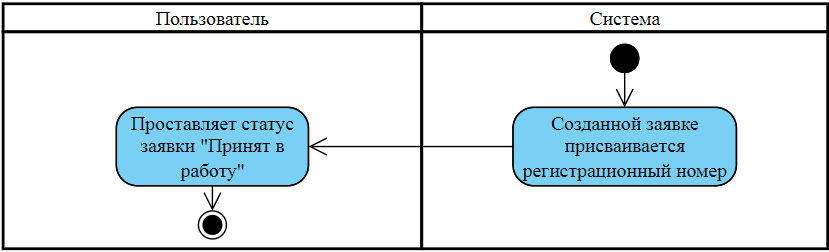


Рисунок 5 – Диаграмма деятельности для прецедента «Приём и регистрация заявок»

1. Прецедент «Обработка и распределение заявок между сотрудниками и другими подразделениями»
   1. Краткое описание

Специалист по обработке заявок распределяет заявки между сотрудниками отдела. Специалист по координации контролирует взаимодействия между сотрудниками отдела. Руководитель отдела координирует работу всех сотрудников.

Действующие лица Прецедента – Специалист по обработке заявок, Специалист по координации, Руководитель отдела.

* 1. Предусловия
     1. Приём и регистрация заявок: перед тем, как начать данный Прецедент, Менеджер по приему заявок должен принять в работу созданную заявку.
  2. Поток событий

Прецедент начинается, когда Специалист по обработке заявок назначит ответственного за исполнение специалиста на принятую заявку.

* 1. Базовый поток – Обработка и распределение заявок между сотрудниками и другими подразделениями
     1. Принятая заявка отображается у Пользователя и назначается ответственный специалист за исполнение
     2. В процессе исполнения ответственного можно поменять, контролировать и координировать работу взаимодействия между сотрудниками

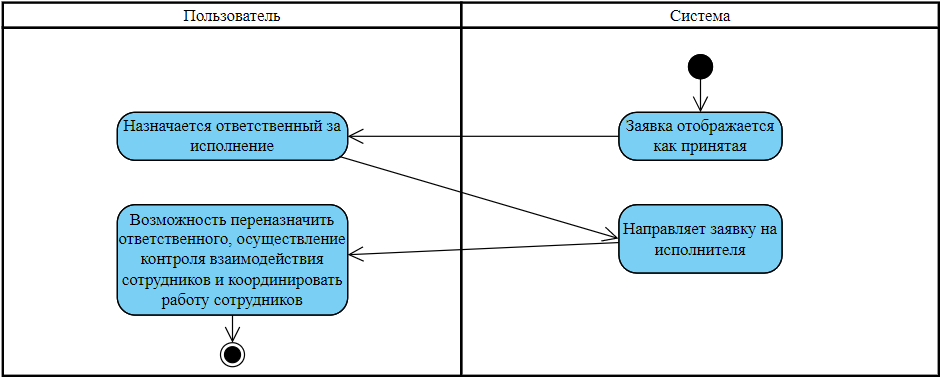


Рисунок 6 – Диаграмма деятельности для прецедента «Обработка и распределение заявок между сотрудниками и другими подразделениями»

1. Прецедент «Контроль исполнения и сроков выполнения заявок»
   1. Краткое описание

Специалист по обработке заявок следит за выполнением заявок и сроками их выполнения. Руководитель отдела контролирует работу сотрудников при выполнении заявок и их сроки исполнения.

Действующие лица Прецедента – Специалист по обработке заявок, Руководитель отдела.

* 1. Предусловия
     1. Обработка и распределение заявок между сотрудниками и другими подразделениями: Перед тем, как начать данный Прецедент, Специалист по обработке заявок должен назначить ответсвенного специалиста за исполнение заявки
  2. Поток событий

Прецедент начинается, когда Специалист по обработке заявок проставляет контроль над заявкой

* 1. Базовый поток – Контроль исполнения и сроков выполнения заявок
     1. Назначенная заявка ответственному находится в одном статусе без обратной связи
     2. Пользователь проставляет контроль и срок исполнения

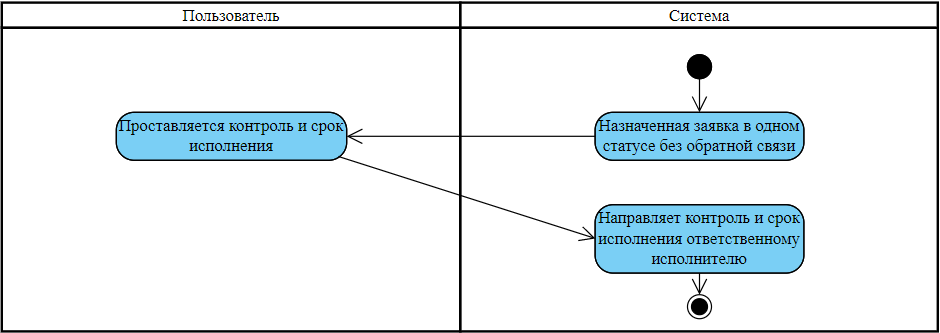


Рисунок 7 – Диаграмма деятельности для прецедента «Контроль исполнения и сроков выполнения заявок»

## **2.4 Последовательность выполнения работ в области автоматизации бизнес-процессов**

Основы автоматизации бизнес-процессов в контексте автоматизации приема и обработки заявок отделом техподдержки ООО «Максбонус» включают следующие аспекты:

Анализ и оптимизация бизнес-процессов: прежде чем приступить к автоматизации, необходимо тщательно проанализировать текущие бизнес-процессы отдела, связанные с приемом и обработкой заявок. Определить их слабые стороны, избыточные шаги и возможности для оптимизации.

Моделирование бизнес-процессов: после анализа и оптимизации процессов, следует разработать модели процессов, которые будут автоматизированы. Моделирование позволяет структурировать процессы, определить роли участников, их взаимодействия и порядок выполнения действий.

Выбор технологий и инструментов: определить подходящие технологии для автоматизации, такие как базы данных, серверные и клиентские технологии, с учетом требований к системе, ресурсов и инфраструктуры компании.

Разработка автоматизированной системы: разработать систему на основе выбранных технологий, реализующую автоматизацию процессов приема и обработки заявок. Система должна быть гибкой и масштабируемой, позволяя внедрять изменения в будущем.

Внедрение системы и обучение пользователей: после разработки системы необходимо провести ее внедрение, настройку и интеграцию с существующими решениями компании. Обучить сотрудников отдела работе с новой системой и предоставить им необходимую поддержку.

Мониторинг и оптимизация: после внедрения системы следует постоянно мониторить ее работу, определять возможные проблемы и предлагать улучшения. Проводить оптимизацию системы с целью повышения ее эффективности и удовлетворения меняющихся потребностей бизнеса.

Оценка результатов автоматизации: провести анализ и оценку экономической эффективности автоматизированной системы, рассчитав затраты на разработку и внедрение системы, а также ожидаемые экономические выгоды, связанные с сокращением времени обработки заявок, уменьшением ошибок и повышением производительности сотрудников.

Контроль и поддержание качества процессов: Важным аспектом автоматизации является контроль и поддержание качества процессов на всех этапах их выполнения. Следует разработать систему метрик и индикаторов, которые помогут отслеживать эффективность процессов и вносить корректировки при необходимости.

Коммуникация и координация: Успешная автоматизация бизнес-процессов требует эффективной коммуникации и координации между различными участниками проекта, включая разработчиков, сотрудников отдела, менеджеров и других заинтересованных сторон.

Безопасность и надежность: Автоматизированная система должна обеспечивать безопасность данных и информации, связанной с приемом и обработкой заявок, а также надежность работы системы, защиту от сбоев и потери данных.

Используя эти основы автоматизации бизнес-процессов, дипломная работа направлена на создание эффективной и надежной системы для автоматизации приема и обработки заявок отделом техподдержки ООО «Максбонус», что в свою очередь повысит уровень удовлетворенности клиентов и оптимизирует работу компании.

## **2.5 Обзор существующих решений**

При разработке автоматизированной системы для приема и обработки заявок отделом техподдержки ООО «Максбонус» важно провести анализ существующих решений на рынке, чтобы определить их преимущества и недостатки, а также выявить возможности для создания уникального и конкурентоспособного продукта. Вот краткий обзор некоторых существующих решений в данной области:

* CRM-системы (Customer Relationship Management): Эти системы предназначены для управления взаимоотношениями с клиентами, включая прием и обработку заявок, отслеживание продаж, учет контактной информации и другие функции. Примеры CRM-систем: Salesforce, Microsoft Dynamics 365, Zoho CRM, iikoCRM, Bitrix24, HubSpot CRM. Преимущества включают гибкость и многофункциональность, недостатки - возможно высокая стоимость и избыточность функций для конкретных нужд ООО «Максбонус».
* BPM-системы (Business Process Management): Эти системы ориентированы на автоматизацию, оптимизацию и мониторинг бизнес-процессов. Примеры BPM-систем: TIBCO, Appian, Pega, Bonita. Преимущества включают возможность моделирования и автоматизации процессов, недостатки - сложность настройки и интеграции с другими системами.
* Helpdesk-системы: Ориентированы на обработку запросов от клиентов и пользователей, таких как заявки на поддержку, жалобы и вопросы. Примеры Helpdesk-систем: Zendesk, Freshdesk, Bitrix24, AmoCRM, Help Scout. Преимущества - простота в использовании и специализация на обработке заявок, недостатки - ограниченные возможности по автоматизации процессов и управлению отношениями с клиентами.
* ERP-системы (Enterprise Resource Planning): Комплексные системы для управления ресурсами компании, включая финансы, закупки, производство и логистику. Примеры ERP-систем: 1С ERP, Галактика, Microsoft Dynamics AX. Преимущества - широкий функционал и интеграция с другими системами, недостатки - сложность внедрения, высокая стоимость и возможная избыточность функций.

В зависимости от специфики и требований ООО «Максбонус», можно выбрать одно из существующих решений или разработать собственную систему автоматизации приема и обработки заявок, учитывая уникальные потребности и особенности компании. Разработка собственной системы может предоставить следующие преимущества:

* Индивидуальный подход: Создание собственной системы позволяет полностью учесть специфику работы отдела и компании, включая особенности бизнес-процессов, требования к отчетности и структуру данных.
* Гибкость и масштабируемость: Разработка собственного решения обеспечивает гибкость в настройке функционала и возможность масштабирования системы в соответствии с ростом и развитием компании.
* Интеграция с существующими системами: Разработанная система может быть легко интегрирована с уже используемыми в компании программами и сервисами, обеспечивая единый процесс обработки заявок и управления информацией.
* Контроль над разработкой и поддержкой: Компания сможет контролировать процесс разработки, внедрения и поддержки системы, что обеспечит быструю реакцию на изменения требований и возникновение проблем.
* Экономическая выгода: В долгосрочной перспективе, создание собственного решения может быть более экономичным вариантом, чем использование сторонних систем с ежемесячной или ежегодной оплатой.

В результате анализа существующих решений для автоматизации приема и обработки заявок отделом техподдержки ООО «Максбонус» может определить наиболее подходящий подход и технологии для создания эффективной системы, которая поможет улучшить работу отдела, сократить время обработки заявок и повысить уровень удовлетворенности клиентов.

## **2.6 Проектирование базы данных**

Проектирование базы данных для автоматизации приема и обработки заявок отделом техподдержки ООО «Максбонус» включает определение таблиц, полей, связей между ними, а также разработку схемы базы данных. В данном случае, используется база данных PostgreSQL [18]. Рассмотрим основные таблицы и их поля:

1. Таблица «board»

* id (INTEGER PRIMARY KEY) – идентификатор доски
* userid (INTEGER, FOREIGN KEY)– идентификатор пользователя
* title (TEXT) – заголовок доски
* description (TEXT) – описание доски

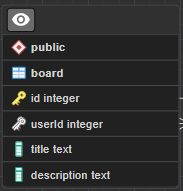


Рисунок 8 – Созданная таблица board

1. Таблица «reason»

* id (INTEGER PRIMARY KEY) – идентификатор причины обращения
* nameReason (TEXT) – название причины обращения

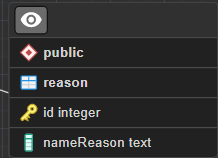


Рисунок 9 – Созданная таблица reason

1. Таблица «status»

* id (INTEGER PRIMARY KEY) – идентификатор статуса
* nameStatus (TEXT) – название статуса

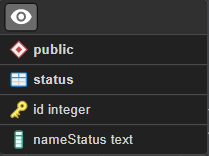


Рисунок 10 – Созданная таблица status

1. Таблица «task»

* id (INTEGER PRIMARY KEY) – идентификатор задачи
* title (TEXT) – описание задачи
* userid (INTEGER, FOREIGN KEY) – идентификатор пользователя
* reasonid (INTEGER, FOREIGN KEY) – идентификатор причины обращения
* boardid (INTEGER, FOREIGN KEY) – идентификатор доски
* statusid (INTEGER, FOREIGN KEY) – идентификатор статуса
* icon (TEXT) – изображение задачи
* phone (VARCHAR (10)) – номер телефона
* email (TEXT) – почта
* companyname - (TEXT) – название компании

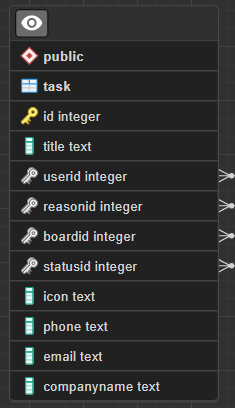


Рисунок 11 – Созданная таблица task

1. Таблица «user»

* id (INTEGER PRIMARY KEY) – идентификатор пользователя
* login (TEXT) – логин пользователя
* password (TEXT) – пароль пользователя
* FIO (TEXT) – ФИО пользователя

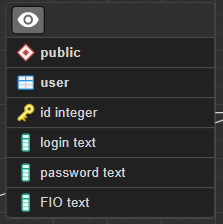


Рисунок 12 – Созданная таблица user

6 Таблица «reasonDescription»

* id (INTEGER PRIMARY KEY) – идентификатор причины
* description (TEXT) – полное описание причины



Рисунок 13 – Созданная таблица reasonDescription

7 Таблица «reason\_description»

* id (INTEGER PRIMARY KEY) – идентификатор связки между таблицами
* id\_reason (INTEGER) – идентификатор причины
* id\_description (INTEGER) – идентификатор описания причины

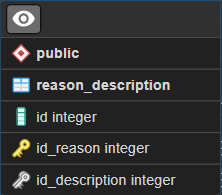


Рисунок 14 - Созданная таблица reason\_description

8 Таблица «roles»

* id (INTEGER PRIMARY KEY) – идентификатор роли
* description (TEXT) – описание роли
* full\_access (INTEGER) – идентификатор доступа

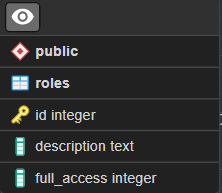


Рисунок 15 - Созданная таблица roles

9 Таблица «user\_roles»

* id (INTEGER PRIMARY KEY) – идентификатор связки между таблицами
* id\_user (INTEGER) – идентификатор пользователя
* id\_roles (INTEGER) – идентификатор роли

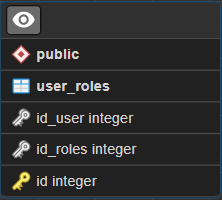


Рисунок 16 - Созданная таблица user\_roles

10 Таблица «statusDescription»

* id (INTEGER PRIMARY KEY) – идентификатор описания статуса
* description (TEXT) – описание статуса

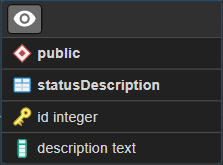


Рисунок 17 - Созданная таблица statusDescription

11 Таблица «status\_description»

* id (INTEGER PRIMARY KEY) – идентификатор связки между таблицами
* id\_status (INTEGER) – идентификатор статуса
* id\_description (INTEGER) – идентификатор описания статуса

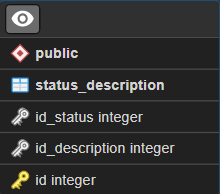


Рисунок 18 - Созданная таблица status\_description

Связи между таблицами:

- в таблице «task» поле userid ссылается на поле id таблицы «user»

- в таблице «task» поле reasonid ссылается на поле id таблицы «reason»

- в таблице «task» поле board ссылается на поле id таблицы «board»

- в таблице «task» поле status ссылается на поле id таблицы «status»

- в таблице «board» поле userid ссылается на поле id таблицы «user»

- в таблице «status\_description» поле id\_status ссылается на поле id таблицы «status»

- в таблице «status\_description» поле id\_description ссылается на поле id таблицы «statusDescription»

- в таблице «user\_roles» поле id\_user ссылается на поле id таблицы «user»

- в таблице «user\_roles» поле id\_roles ссылается на поле id таблицы «roles»

- в таблице «reason\_description» поле id\_reason ссылается на поле id таблицы «reason»

- в таблице «reason\_description» поле id\_description ссылается на поле id таблицы «reasonDescription»

Ниже на рисунке 19 представлена ER диаграмма базы данных

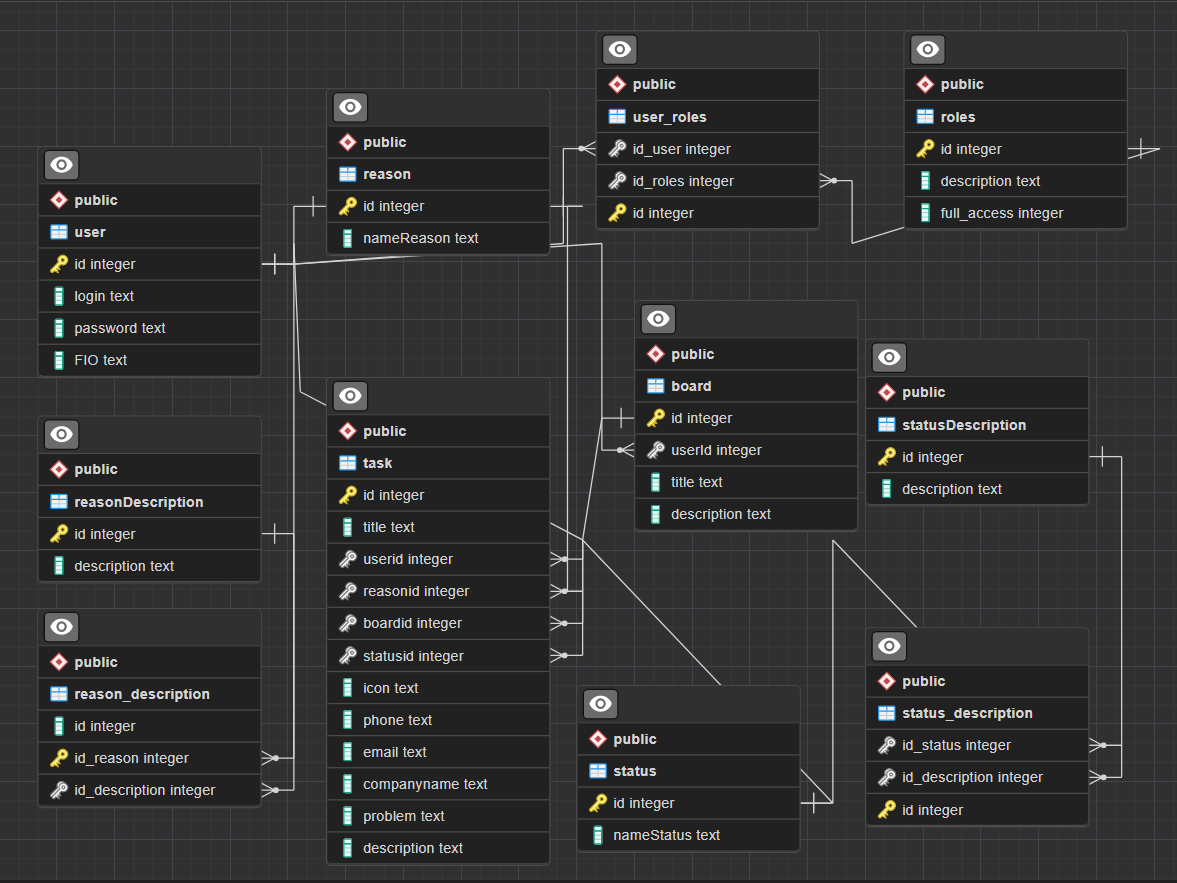


Рисунок 19 – ER диаграмма базы данных

Для обеспечения безопасности при работе с базой данных в автоматизации приема и обработки заявок отделом техподдержки ООО «Максбонус», пароли пользователей хранятся в виде хеш-суммы, сгенерированной с использованием алгоритма SHA-256. Это является стандартной практикой и позволяет защитить пароли даже в случае утечки данных.

При создании нового пользователя или изменении пароля, вместо хранения пароля в открытом виде, система будет генерировать хеш-сумму с использованием алгоритма SHA-256 и сохранять этот хеш в поле "password" таблицы "user". При аутентификации пользователя, система также будет генерировать хеш-сумму введенного пароля и сравнивать ее с хешем, хранящимся в базе данных, чтобы определить, является ли пароль верным.

Для реализации этой функциональности, можно воспользоваться встроенными функциями и библиотеками языка программирования серверной части (Go) или использовать внешние пакеты, предоставляющие реализацию алгоритма SHA-256.

Для обеспечения безопасности хранения паролей пользователей, в проекте используется алгоритм SHA-256, который является рекомендованным стандартом безопасности, согласно NIST [19]. Реализация алгоритма SHA-256 для языка программирования Go представлена в пакете crypto/sha256 [20]. Такой подход соответствует рекомендациям OWASP по безопасному хранению паролей [21]. При создании нового пользователя или изменении пароля, система генерирует хеш-сумму пароля с использованием алгоритма SHA-256 и сохраняет его в базе данных. В процессе аутентификации пользователя, система также генерирует хеш-сумму введенного пароля и сравнивает ее с хранящимся хешем для проверки соответствия пароля.

## **2.7 Обоснование архитектуры КИС**

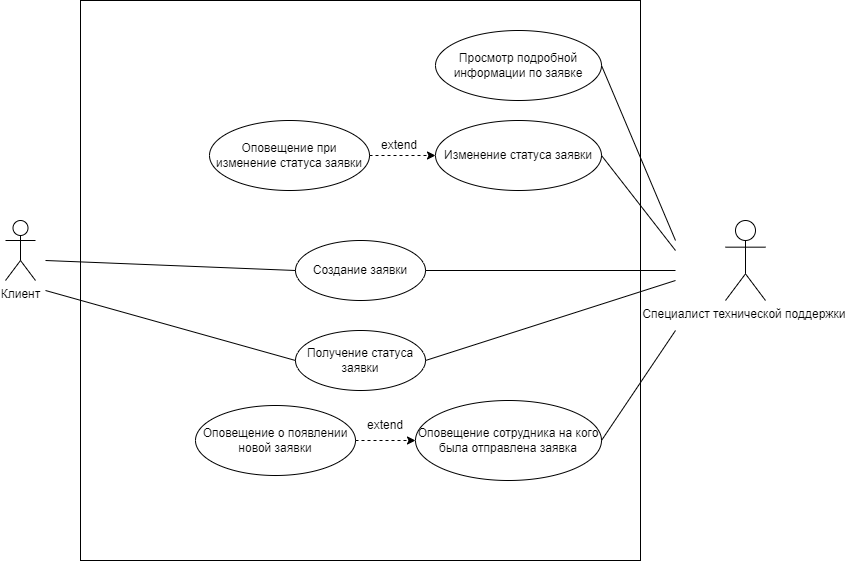
Для информационной системы техподдержки, основанной на модели Iterative and Incremental Lifecycle Model, подходящей архитектурой может быть сервисно-ориентированная архитектура (SOA) [22]. Сервисно-ориентированная архитектура подразумевает разделение системы на независимые компоненты (сервисы), каждый из которых выполняет определенную функцию. Каждый сервис может быть разработан и поддерживаться отдельно, что облегчает разработку, управление и масштабирование системы.

Почему SOA подходит для информационной системы техподдержки, основанной на модели Iterative and Incremental Lifecycle Model? Данный подход предусматривает поэтапную разработку системы, где каждый этап содержит набор функциональных требований, которые необходимо реализовать. SOA позволяет разрабатывать и тестировать каждый сервис по отдельности, что упрощает процесс интеграции и снижает риск возникновения ошибок в процессе разработки [23]. Кроме того, SOA может обеспечить гибкость системы и возможность ее дальнейшей модернизации [24].

Для проекта КИС (корпоративной информационной системы) можно использовать CASE-средства (Computer-Aided Software Engineering), такие как Rational Rose, Enterprise Architect, MagicDraw и др. [25]. Эти средства позволяют разработчикам создавать диаграммы, описывающие различные аспекты системы (структуру базы данных, архитектуру приложения, диаграммы классов и пр.), автоматизировать процессы разработки, управлять изменениями и повторно использовать код [26].

Например, в проекте КИС можно использовать CASE-средство Enterprise Architect для создания диаграммы классов, описывающей структуру объектов в системе, диаграмму последовательности для описания процесса обработки запросов техподдержки и диаграмму компонентов для описания

архитектуры системы [27]. Эти диаграммы могут помочь разработчикам лучше понимать структуру и логику системы, а также облегчить коммуникацию между членами команды. Кроме того, CASE-средство может автоматизировать генерацию кода, что может ускорить процесс разработки. На рисунке 20 представлена Use Case диаграмма.

Рисунок 20 – Use Case диаграмма

На рисунке 21 представлена Activity Diagram

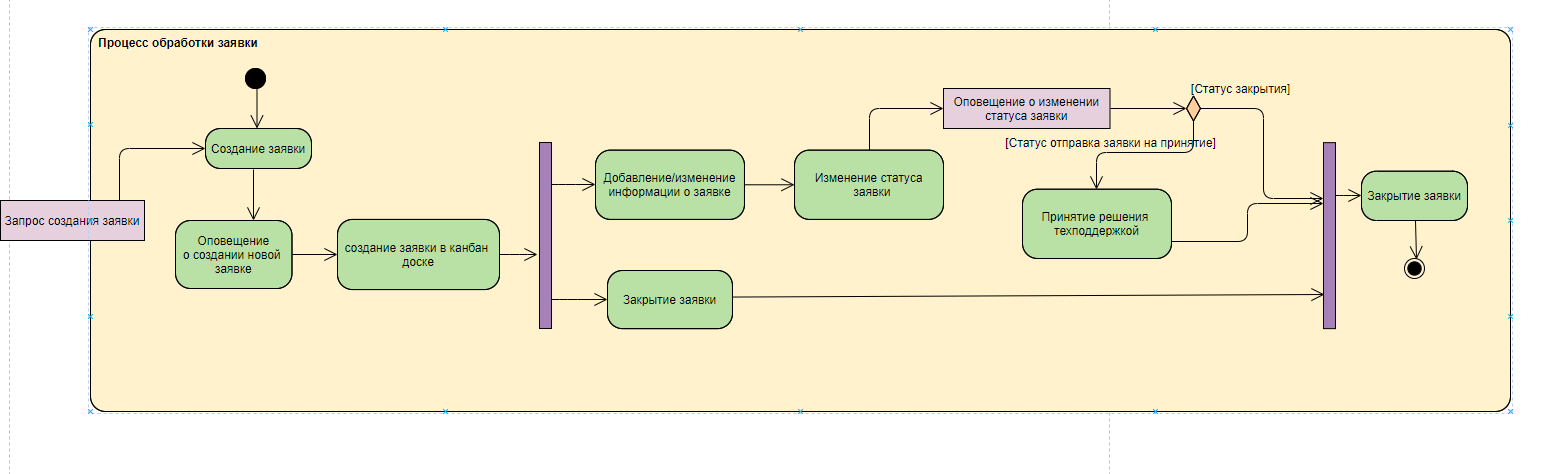


Рисунок 21 – Activity diagram

**Вывод к главе 2**

Во второй главе был разработан технический проект системы автоматизации приема и обработки заявок. В ходе работы определены основные компоненты системы, архитектура и выбраны инструменты для реализации. Также была разработана структура базы данных и определены основные функциональные требования к системе. Это позволило создать основу для дальнейшей разработки и внедрения системы автоматизации приема и обработки заявок отделом техподдержки ООО «Максбонус».

# 3 Разработка и внедрение системы автоматизации приема и обработки заявок в ООО «Максбонус»

## **3.1 Разработка серверной части**

Разработка серверной части системы автоматизации приема и обработки заявок отделом техподдержки ООО «Максбонус» включает использование языка программирования Go и ряда библиотек и технологий для обеспечения производительности, безопасности и удобства разработки. В частности, сервер будет использовать следующие компоненты:

* Gin: это высокопроизводительный HTTP-фреймворк для Go, HTTP-запросов, управление роутингом [3] и мидлварами [1]. Gin позволяет создавать RESTful API [2] для обработки запросов от клиентской части системы [28].
* DI Container: Dependency Injection (DI) контейнеры используются для управления зависимостями между компонентами приложения. В Go есть несколько библиотек для DI, таких как Uber's Dig [29], Google's Wire [30] или Facebook's Inject [31]. Использование DI контейнера облегчает разработку и тестирование, делая код более модульным и расширяемым.
* JWT Validation: JSON Web Tokens (JWT) являются стандартом для безопасной передачи информации между клиентом и сервером. JWT используется для аутентификации пользователей и предоставления доступа к защищенным ресурсам. В Go существуют библиотеки, такие как github.com/dgrijalva/jwt-go [32] или github.com/golang-jwt/jwt [33], которые упрощают работу с JWT, включая создание, проверку подписи и валидацию токенов.

На рисунке 20 представлены реализованные роуты [3] для https запросов



Рисунок 20 – Реализованные роуты [3] для сервера

Роутер «/api/login» необходим для аутентификации пользователей в системе автоматизации приема и обработки заявок отделом техподдержки ООО «Максбонус». Этот эндпоинт [4] отвечает за проверку учетных данных пользователя, таких как логин и пароль, и в случае успешной аутентификации генерирует и выдает JSON Web Token (JWT), который будет использоваться для доступа к защищенным ресурсам системы.

При отправке POST-запроса на «/api/login» с корректными учетными данными (логином и паролем) в теле запроса, серверная часть на Go с использованием фреймворка Gin обрабатывает запрос, хэширует [5] в sha-256, выполняет проверку учетных данных пользователя, сверяя их с сохраненными данными в базе данных PostgreSQL. Если данные верны, сервер генерирует JWT, который содержит информацию о пользователе и его роли в системе, а также подписывает его секретным ключом. Затем сервер отправляет ответ с JWT в формате JSON, который клиентская часть будет использовать для дальнейшей авторизации и доступа к ресурсам системы.

Важно отметить, что при неудачной аутентификации (например, если предоставлены неверные учетные данные) сервер возвращает соответствующий код ошибки и сообщение, информируя пользователя о проблеме.

Использование роутера «/api/login» для аутентификации пользователей является стандартным подходом при разработке веб-приложений и обеспечивает безопасность и контроль доступа к ресурсам системы автоматизации приема и обработки заявок отделом техподдержки ООО «Максбонус».

Тестирование эндопоинта [4] будет происходить при помощи запросов в Postman. Необходимо ввести в строку адрес сервиса <https://help-maxbonus.ru/api/login> .

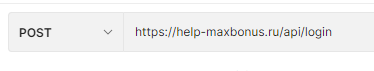


Рисунок 21 – Ввод адрес эндпоинта

После того, как адрес сервера введён необходимо отправить JSON в тело запроса.

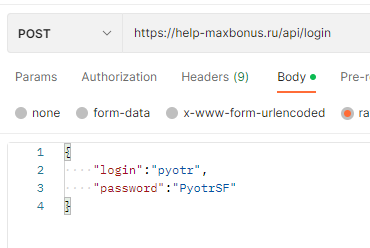


Рисунок 22 – заполненный JSON в соответствии с RESTful API [2]

После того, как JSON заполнен необходимо отправить запрос, затем, сервер отправит «Status: 200 OK» и код авторизации если логин и пароль совпадает в базе данных.

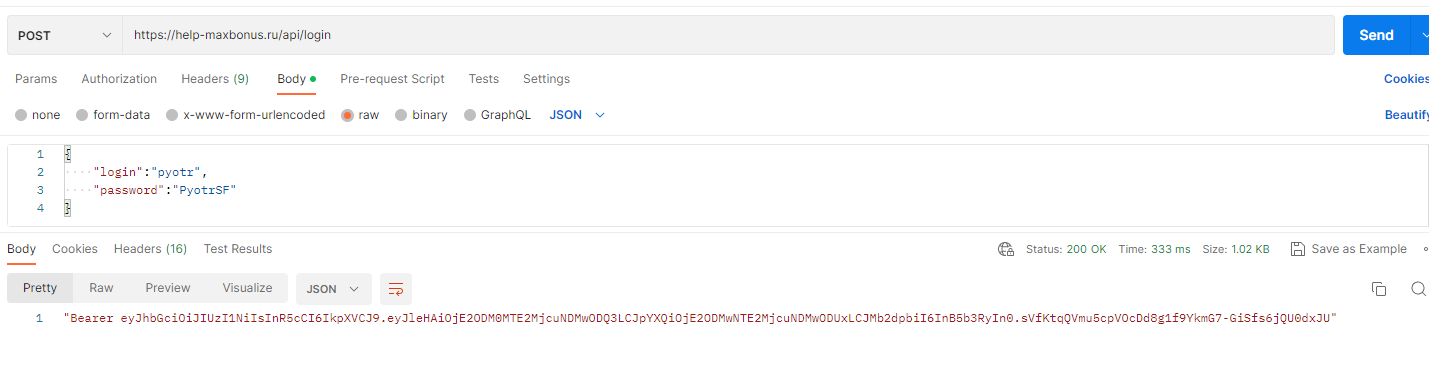


Рисунок 23 – Успешная авторизация

Если данные не совпадают, то сервер вернёт «Status: 401 Unauthorized».

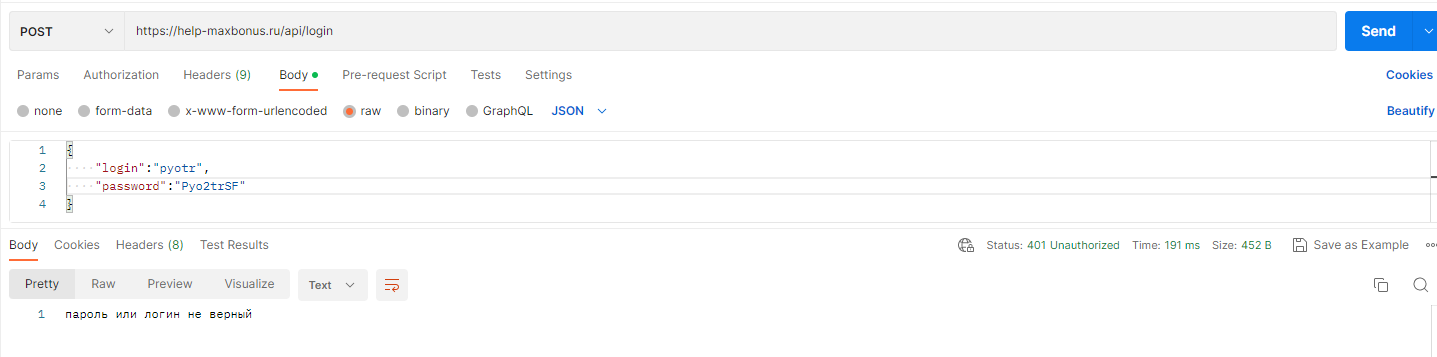


Рисунок 24 – Пароль или логин не верный

При неправильном заполнение JSON, сервер возвращает «400 Bad Request»

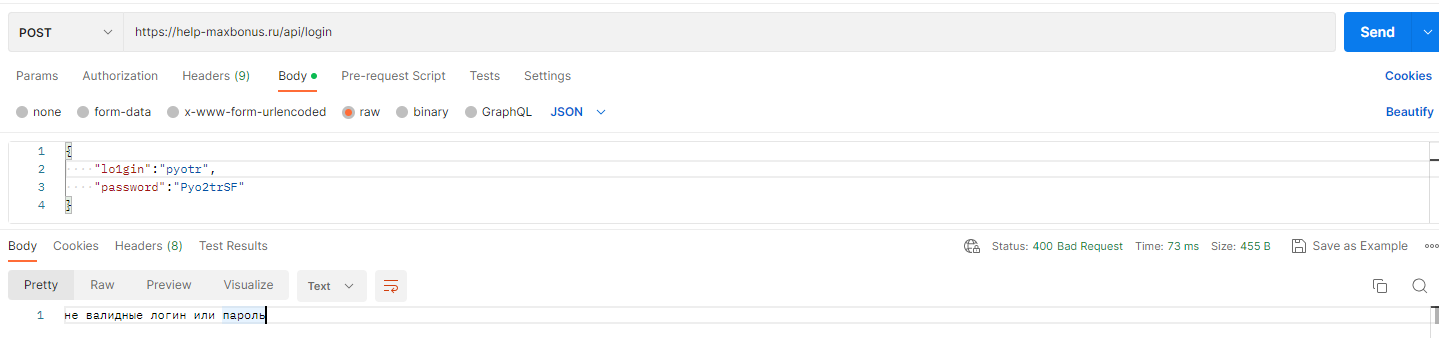


Рисунок 25 - Не валидные логин или пароль

Роутер «/api/task/add» необходим для добавления новых заявок в систему автоматизации приема и обработки заявок отделом техподдержки ООО «Максбонус». Этот эндпоинт [4] предназначен для создания новых задач, связанных с обработкой заявок, и сохранения их в базе данных для дальнейшей обработки и мониторинга.

При отправке POST-запроса на «/api/task/add» с корректными данными о задаче в теле запроса, серверная часть на Go с использованием фреймворка Gin обрабатывает запрос и выполняет проверку наличия необходимых полей и правильности их формата. При успешной проверке сервер создает новую запись о задаче в базе данных PostgreSQL и возвращает ответ с идентификатором созданной задачи и другой информацией.

Для доступа к этому эндпоинту [4] пользователь должен быть аутентифицирован, и его роль в системе должна предоставлять разрешение на создание новых задач. В этом случае, JWT, полученный при успешной аутентификации, должен быть передан в заголовке запроса (в поле "Authorization") для подтверждения прав доступа пользователя.

Использование роутера «/api/task/add» для добавления новых заявок позволяет упростить процесс создания и обработки заявок, делая его более удобным и эффективным для сотрудников отдела. Это также обеспечивает структурированное хранение информации о задачах и их обработке, что способствует повышению производительности работы отдела и улучшению качества предоставляемых услуг.

Тестирование эндопоинта [4] будет происходить при помощи запросов в Postman. Необходимо ввести в строку адрес сервиса https://help-maxbonus.ru/api/task/add.

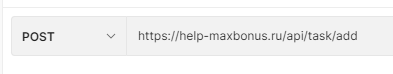


Рисунок 26 - Ввод эндпоинта [4]

После того, как адрес сервера введён необходимо отправить JSON в тело запроса.

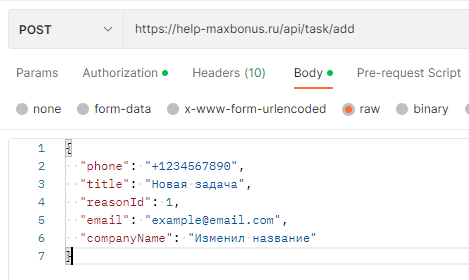


Рисунок 27 – Заполненный JSON в соответствии RESTful API [2]

При успешном заполнение тела запроса и введены корректные поля, сервер вернёт «201 Created» и номер заявки и отправит сообщение в телеграм.

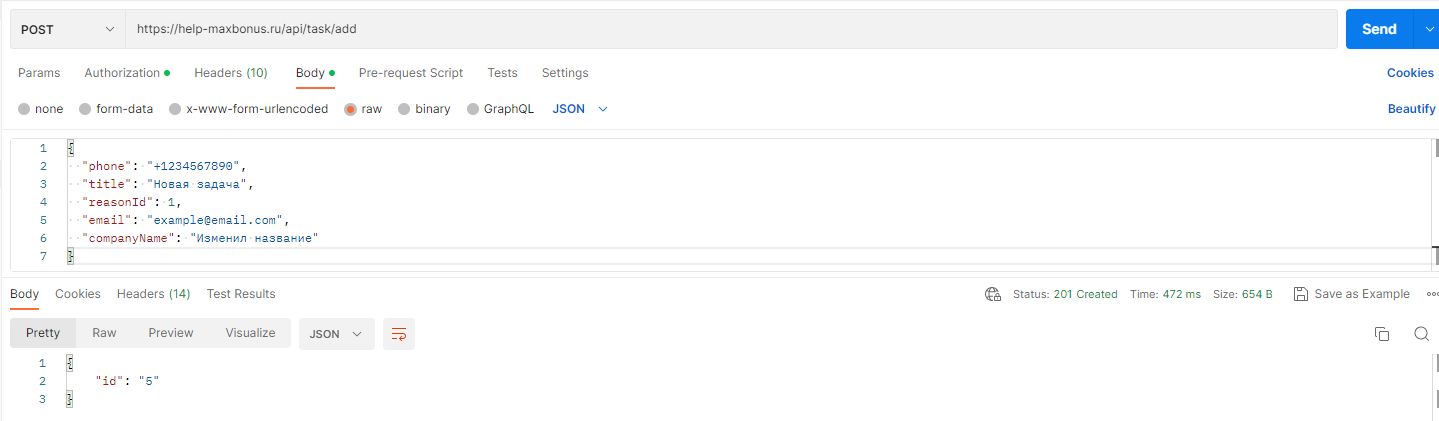


Рисунок 28 – Успешное добавление задачи

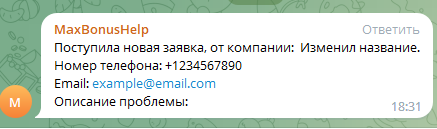


Рисунок 29 – Отправленное сообщение в телеграм

Роутер «/api/task/status/:id» необходим для обновления статуса заявок и отслеживания их выполнения в системе автоматизации приема и обработки заявок отделом техподдержки ООО «Максбонус». Этот эндпоинт [4] позволяет сотрудникам отдела изменять статус задачи на основе ее выполнения и контролировать прогресс выполнения всех заявок.

При отправке PUT-запроса на «/api/task/status/:id» с новым статусом задачи (например, "выполнено" или "в процессе") в теле запроса, серверная часть на Go с использованием фреймворка Gin обрабатывает запрос, извлекает идентификатор задачи из параметра URL и выполняет проверку наличия задачи с указанным идентификатором в базе данных PostgreSQL. Если задача существует, сервер обновляет ее статус и возвращает ответ с подтверждением об успешном обновлении статуса задачи.

Доступ к этому эндпоинту [4] также должен быть ограничен аутентифицированными пользователями, чья роль в системе предоставляет разрешение на изменение статуса задач. JWT, полученный при успешной аутентификации, должен быть передан в заголовке запроса (например, в поле "Authorization") для подтверждения прав доступа пользователя.

Использование роутера «/api/task/status/:id» для обновления статуса заявок обеспечивает эффективное отслеживание выполнения задач и позволяет менеджерам и сотрудникам отдела оценивать прогресс в обработке заявок. Это также позволяет своевременно определять возможные проблемы и задержки, что в свою очередь помогает оптимизировать рабочие процессы и повышать уровень удовлетворенности клиентов.

Тестирование эндопоинта [4] будет происходить при помощи запросов в Postman. Необходимо ввести в строку адрес сервиса <https://help-maxbonus.ru/api/task/status/8>

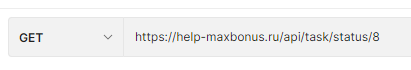


Рисунок 30 – Ввод адрес эндпоинта [4]

Сервер вернёт статус заявки с кодом «200 OK»

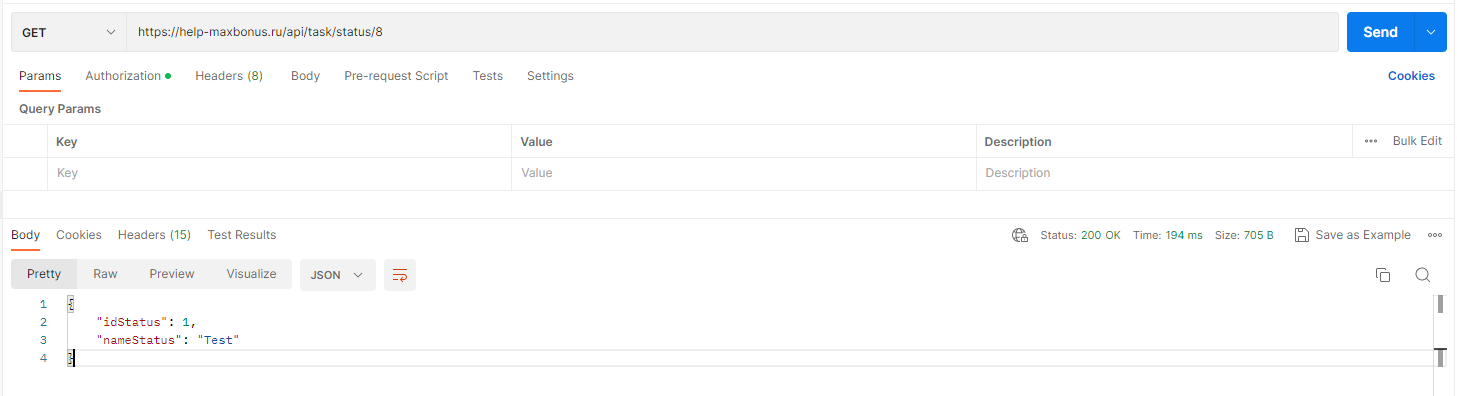


Рисунок 31 – Получение данных о заявке

GET-роутер «/api/status» необходим для проверки статуса заявки в системе автоматизации приема и обработки заявок отделом техподдержки ООО «Максбонус». Этот эндпоинт [4] позволяет пользователям и сотрудникам отдела получать информацию о текущем состоянии и статусе обработки заявок.

При отправке GET-запроса на «/api/status» с идентификатором заявки в параметрах запроса, серверная часть на Go с использованием фреймворка Gin обрабатывает запрос и осуществляет поиск заявки с указанным идентификатором в базе данных PostgreSQL. Если заявка найдена, сервер возвращает ответ с информацией о статусе заявки, такой как "новая", "в процессе" или "выполнена", а также другие сведения, связанные с заявкой.

Доступ к этому эндпоинту [4] может быть ограничен аутентифицированными пользователями, чья роль в системе предоставляет разрешение на просмотр статуса заявок. В этом случае, JWT, полученный при успешной аутентификации, должен быть передан в заголовке запроса (например, в поле "Authorization") для подтверждения прав доступа пользователя.

Использование роутера «/api/status» для проверки статуса заявки позволяет пользователям и сотрудникам отдела отслеживать ход выполнения заявок и контролировать их обработку. Это обеспечивает прозрачность рабочего процесса и позволяет клиентам быть в курсе состояния своих заявок, что, в свою очередь, способствует повышению уровня удовлетворенности клиентов и эффективности работы отдела.

Тестирование эндопоинта [4] будет происходить при помощи запросов в Postman. Необходимо ввести в строку адрес сервиса [https://help-maxbonus.ru/api/task/status](https://help-maxbonus.ru/api/task/status/8)

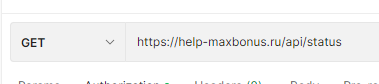


Рисунок 32 – Ввод адрес эндпоинта [4]

Сервер вернёт статус заявки с кодом «200 OK»

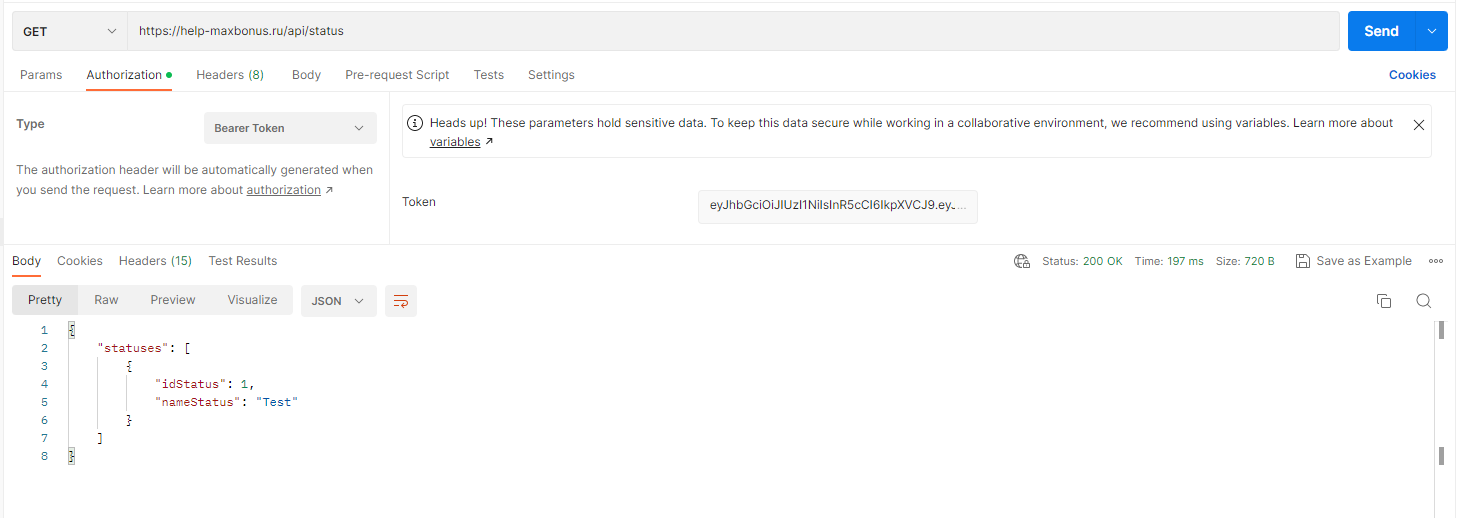


Рисунок 33 – Получение данных о заявках

GET-роутер «/api/users» необходим для получения списка пользователей системы автоматизации приема и обработки заявок отделом техподдержки ООО «Максбонус». Этот эндпоинт [4] позволяет менеджерам и сотрудникам отдела просматривать список доступных пользователей, чтобы назначать их на заявки, а также для обеспечения правильного распределения рабочей нагрузки между сотрудниками.

При отправке GET-запроса на «/api/users», серверная часть на Go с использованием фреймворка Gin обрабатывает запрос и получает список всех пользователей из базы данных PostgreSQL. Затем сервер возвращает ответ, содержащий информацию о каждом пользователе, такую как имя, фамилия, должность и контактные данные.

Доступ к этому эндпоинту [4], как правило, ограничен аутентифицированными пользователями с определенными ролями и разрешениями, такими как менеджеры или администраторы системы. JWT, полученный при успешной аутентификации, должен быть передан в заголовке запроса (например, в поле "Authorization") для подтверждения прав доступа пользователя.

Использование роутера «/api/users» для получения списка пользователей для назначения на заявку облегчает процесс распределения задач между сотрудниками и обеспечивает более эффективное управление рабочим процессом. Это также способствует улучшению коммуникации и координации между сотрудниками, что, в свою очередь, ведет к повышению производительности отдела и качества предоставляемых услуг.

Тестирование эндопоинта [4] будет происходить при помощи запросов в Postman. Необходимо ввести в строку адрес сервиса <https://help-maxbonus.ru/api/task/users>

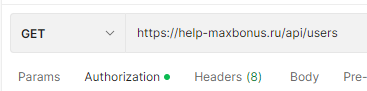


Рисунок 34 – Ввод адрес эндпоинта [4]

Сервер вернёт статус заявки с кодом «200 OK»

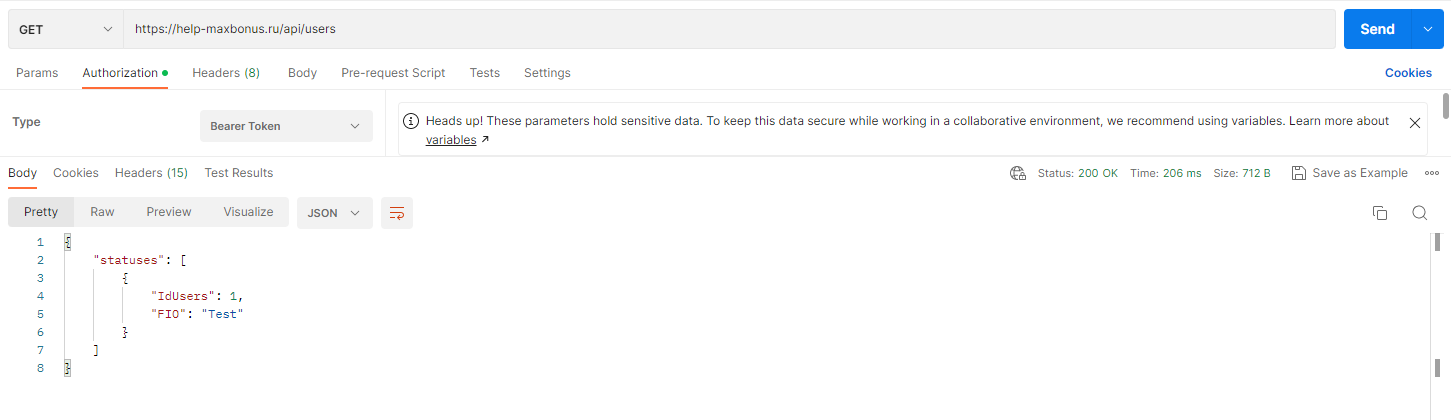


Рисунок 35 – Получение данных о заявках

DELETE-роутер «/api/task/:id» необходим для удаления заявок в системе автоматизации приема и обработки заявок отделом техподдержки ООО «Максбонус». Этот эндпоинт [4] позволяет менеджерам и сотрудникам отдела удалять заявки, которые больше не актуальны или были завершены, что помогает поддерживать актуальность информации и оптимизировать рабочий процесс.

При отправке DELETE-запроса на «/api/task/:id» с идентификатором заявки в параметрах URL, серверная часть на Go с использованием фреймворка Gin обрабатывает запрос и выполняет проверку наличия заявки с указанным идентификатором в базе данных PostgreSQL. Если заявка существует, сервер удаляет ее из базы данных и возвращает ответ с подтверждением успешного удаления заявки.

Доступ к этому эндпоинту [4] должен быть ограничен аутентифицированными пользователями с определенными ролями и разрешениями, такими как менеджеры или администраторы системы. JWT, полученный при успешной аутентификации, должен быть передан в заголовке запроса (например, в поле "Authorization") для подтверждения прав доступа пользователя.

Использование роутера «/api/task/:id» для удаления заявок обеспечивает гибкость и контроль над рабочим процессом, что позволяет удалять устаревшие или завершенные заявки и освобождать ресурсы для работы с актуальными задачами. Это также содействует улучшению общей организации работы отдела и повышению уровня удовлетворенности клиентов.

Тестирование эндопоинта [4] будет происходить при помощи запросов в Postman. Необходимо ввести в строку адрес сервиса https://help-maxbonus.ru/api/task/4

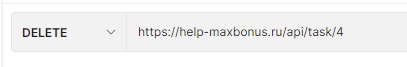


Рисунок 36 – Ввод адрес эндпоинта [4]

Сервер вернёт статус заявки с кодом «200 OK»

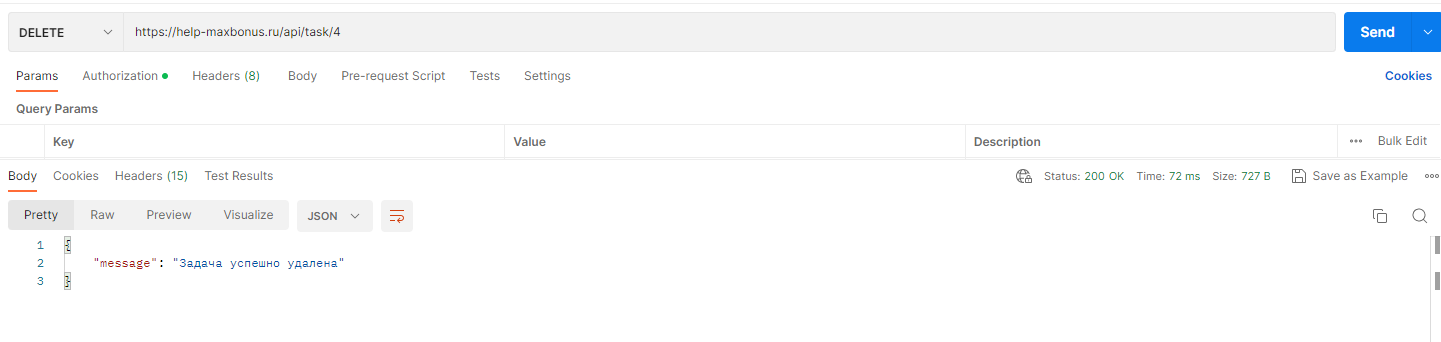


Рисунок 37 – Получение данных о заявках

PUT-роутер «/api/task/:id» необходим для изменения заявок в системе автоматизации приема и обработки заявок отделом техподдержки ООО «Максбонус». Этот эндпоинт [4] позволяет менеджерам и сотрудникам отдела вносить изменения в существующие заявки, такие как обновление статуса, изменение сроков исполнения, назначение нового исполнителя или корректировка деталей заявки.

При отправке PUT-запроса на «/api/task/:id» с идентификатором заявки в параметрах URL и данными для обновления в теле запроса, серверная часть на Go с использованием фреймворка Gin обрабатывает запрос и выполняет проверку наличия заявки с указанным идентификатором в базе данных PostgreSQL. Если заявка существует, сервер обновляет ее сведения в соответствии с переданными данными и возвращает ответ с подтверждением успешного обновления заявки.

Доступ к этому эндпоинту [4] должен быть ограничен аутентифицированными пользователями с определенными ролями и разрешениями, такими как менеджеры или администраторы системы. JWT, полученный при успешной аутентификации, должен быть передан в заголовке запроса (например, в поле "Authorization") для подтверждения прав доступа пользователя.

Использование роутера «/api/task/:id» для изменения заявок предоставляет гибкий и управляемый рабочий процесс, позволяющий адаптироваться к изменяющимся условиям и потребностям клиентов. Это также способствует улучшению коммуникации и координации между сотрудниками отдела, что, в свою очередь, ведет к повышению производительности и качества предоставляемых услуг.

Тестирование эндопоинта [4] будет происходить при помощи запросов в Postman. Необходимо ввести в строку адрес сервиса https://help-maxbonus.ru/api/task/8

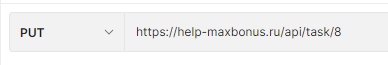


Рисунок 38 - Ввод эндпоинта [4]

После того, как адрес сервера введён необходимо отправить JSON в тело запроса.

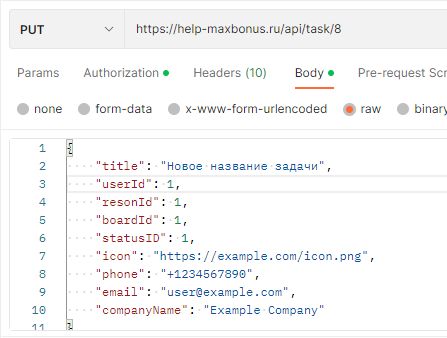


Рисунок 39 – Заполненный JSON в соответствии RESTful API [2]

При успешном заполнение тела запроса и введены корректные поля, сервер вернёт «200 OK» и отправит сообщение в телеграм.

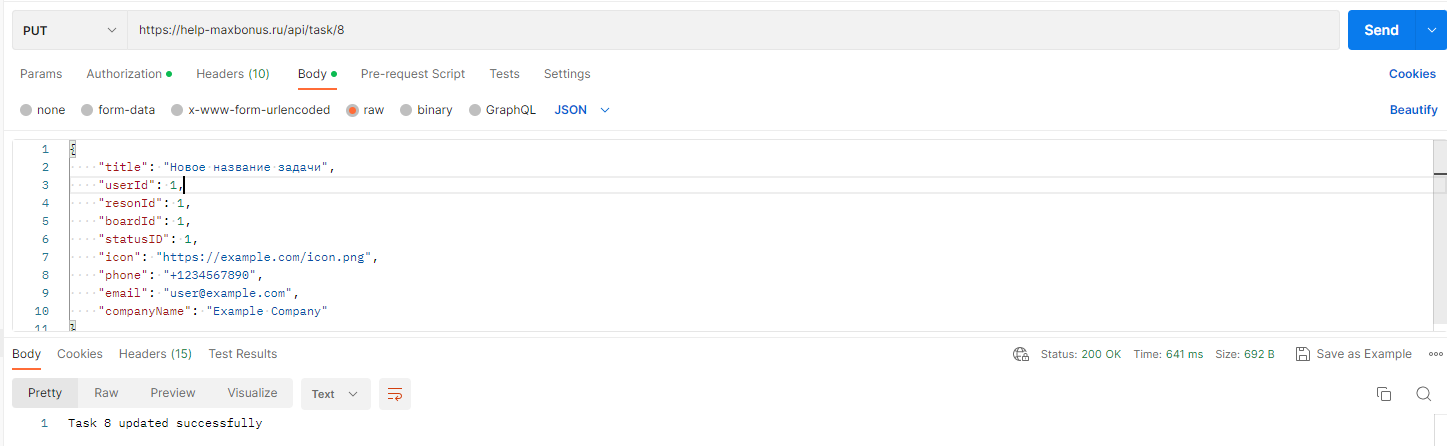


Рисунок 40 – Успешное изменение задачи

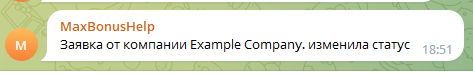


Рисунок 41 – Сообщение в телеграм о изменение статуса заявки

На рисунке 42 представлена uml-диаграмма компонентов сервера.

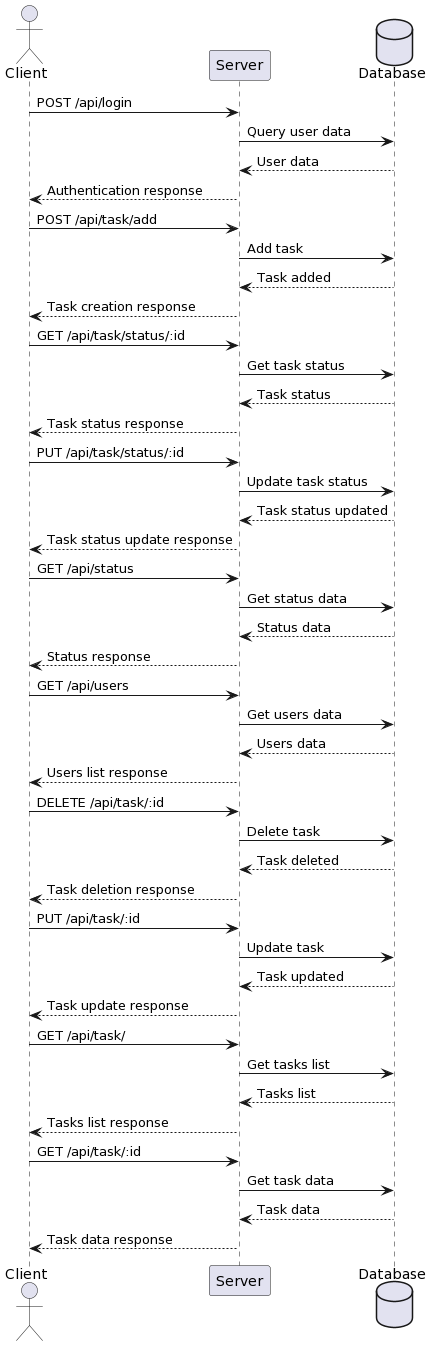


Рисунок 42 – Uml-диаграмма компонентов сервера

## **3.2 Создание чат-бота телеграмм для оповещения о новых заявках**

Создание чат-бота в Telegram для оповещения о новых заявках является важным дополнительным функционалом системы автоматизации приема и обработки заявок отделом техподдержки ООО «Максбонус». Чат-бот будет автоматически отправлять уведомления сотрудникам отдела о поступлении новых заявок, что упростит процесс определения оперативно реагировать на запросы клиентов и повысит эффективность работы отдела.

Для создания и разработки чат-бота в Telegram предполагается выполнить следующие этапы:

* Создание и настройка бота в Telegram через BotFather: с помощью специального бота Telegram - BotFather, необходимо зарегистрировать нового бота, получить его токен и настроить основные параметры, такие как имя описание [34]. На рисунке 37 представлено результат создания и настройки бота

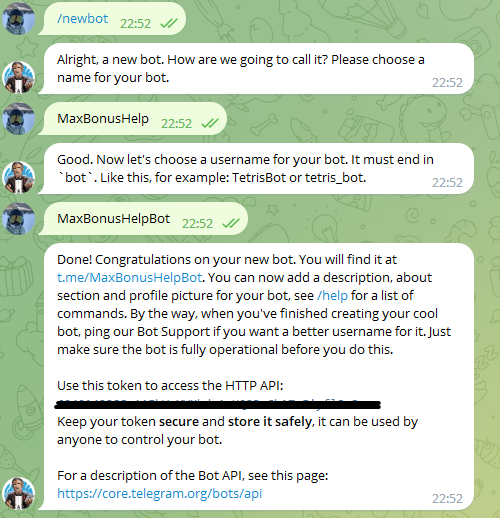


Рисунок 43 – Cоздание бота

* Изучение Telegram Bot API и выбор библиотеки для языка программирования Go. Для упрощения работы с API на языке Go можно использовать библиотеку go-telegram-bot-api [35].
* Реализация функционала для отправки уведомлений о новых заявках: после настройки бота и изучения API, реализован функционал для отправки уведомлений о новых заявках.

Для реализации отправки было принято решение добавить функцию sendTelegramNotification в handler «/api/task/add». Функция принимает токен чат-бота, чат id и сообщение. На рисунке 38 представлена функция sendTelegramNotification.



Рисунок 44 – Функция sendTelegramNotification

После того, как заявка поступает, вызывается функция sendTelegramNotification и отправляет сообщение в группу. Ниже на рисунке 45 представлен результат работы.

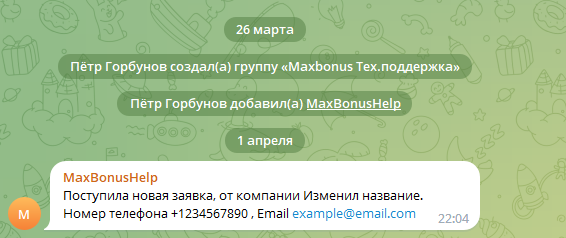


Рисунок 45 – Уведомление о поступлении заявки

Также чат-бот уведомляет о изменении статуса заявки.

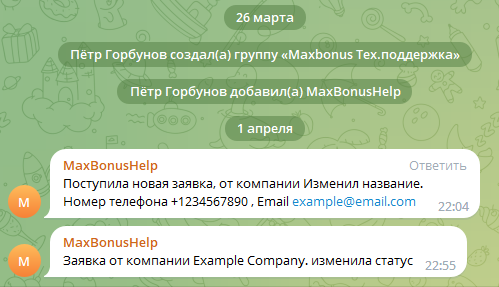


Рисунок 46 – Изменение статуса заявки

При создании и настройке бота через BotFather, были получены подробные инструкции по регистрации нового бота и его настройке. Изучение Telegram Bot API позволило получить общее представление о возможностях API и способах работы с ним. С использованием библиотеки go-telegram-bot-api была упрощена разработка функционала чат-бота, так как библиотека предоставляет готовые методы и структуры для работы с API [36]. Статья "Telegram Bots: An introduction for developers" [37] также оказала помощь в общем понимании принципов работы и создания ботов в Telegram.

## **3.3 Разработка клиентской части**

В процессе разработки дипломного проекта на тему "Автоматизация приема и обработки заявок отделом техподдержки ООО «Максбонус»" был использован популярный фреймворк React для создания клиентской части веб-приложения [38]. React является открытым исходным кодом и поддерживается Facebook, что обеспечивает его актуальность и надежность [39].

React использует компонентный подход, который позволяет создавать модульные и переиспользуемые блоки кода, что упрощает процесс разработки и облегчает сопровождение проекта [40]. Ключевой особенностью React является его виртуальный DOM (Virtual DOM), что позволяет оптимизировать обновление пользовательского интерфейса, сокращая время на перерисовку компонентов и улучшая производительность приложения [41].

Для управления состоянием приложения был использован Redux, который обеспечивает однонаправленный поток данных и предоставляет простой способ синхронизации состояния между компонентами [42]. Такой подход упрощает архитектуру приложения и улучшает его масштабируемость [43].

Для стилизации интерфейса использовалась библиотека Material-UI, основанная на принципах Google Material Design, что позволяет создать современный и интуитивно понятный интерфейс для пользователей [44].

В процессе разработки клиентской части были использованы также следующие инструменты и технологии: React Router для организации навигации и маршрутизации между страницами приложения [45], Axios для выполнения HTTP-запросов к серверной части и обмена данными [46] и JWT (JSON Web Token) для аутентификации и авторизации пользователей [47].

Для упрощения разработки и повышения качества кода были применены инструменты ESLint для статического анализа кода [48] и Prettier для автоматического форматирования кода [49]. В качестве системы контроля версий использовался Git [50], что обеспечивало надежность и прозрачность процесса разработки.

Таким образом, использование React в сочетании с другими современными технологиями и инструментами позволило создать надежное, масштабируемое и производительное веб-приложение для автоматизации приема и обработки заявок отделом техподдержки ООО «Максбонус».

В ходе разработки клиентской части были созданы компоненты для авторизации, создания и обработки заявок, а также компоненты для администрирования пользователей и статистики. Результатом является веб-приложение, которое обеспечивает эффективную работу отдела техподдержки и улучшает взаимодействие с клиентами.

На рисунках 47 и 48 представлена главная страница для заполнения заявки

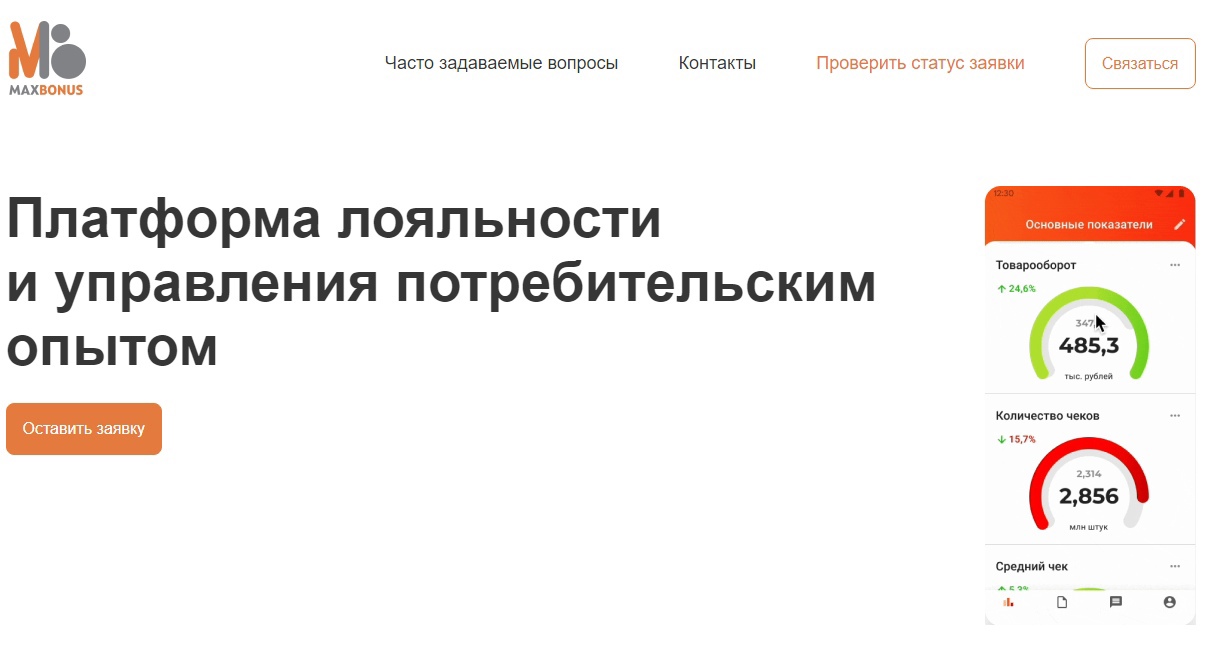


Рисунок 47 – Главная страница

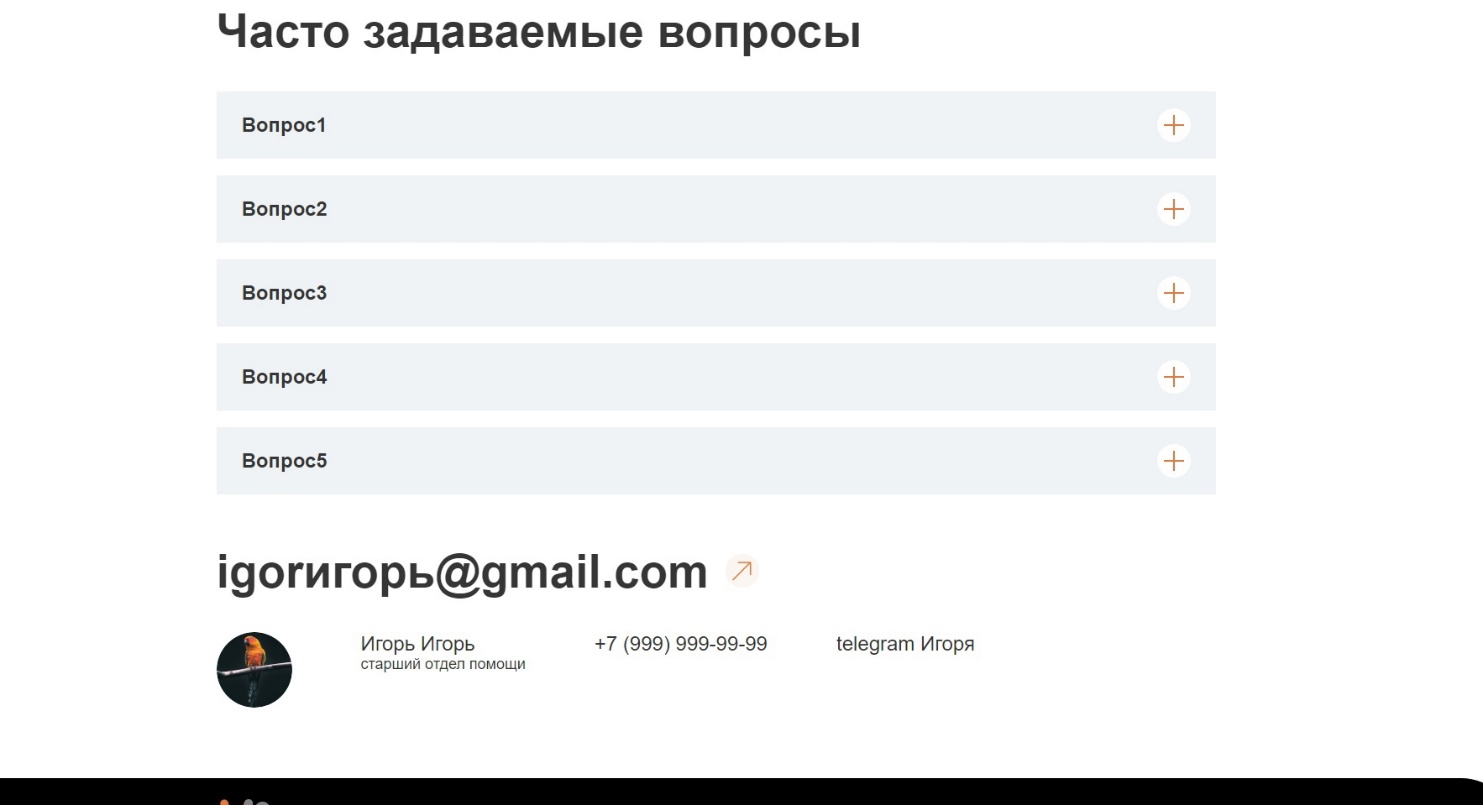


Рисунок 48 – Главная страница

На рисунке 49 представлена форма для заполнения заявки

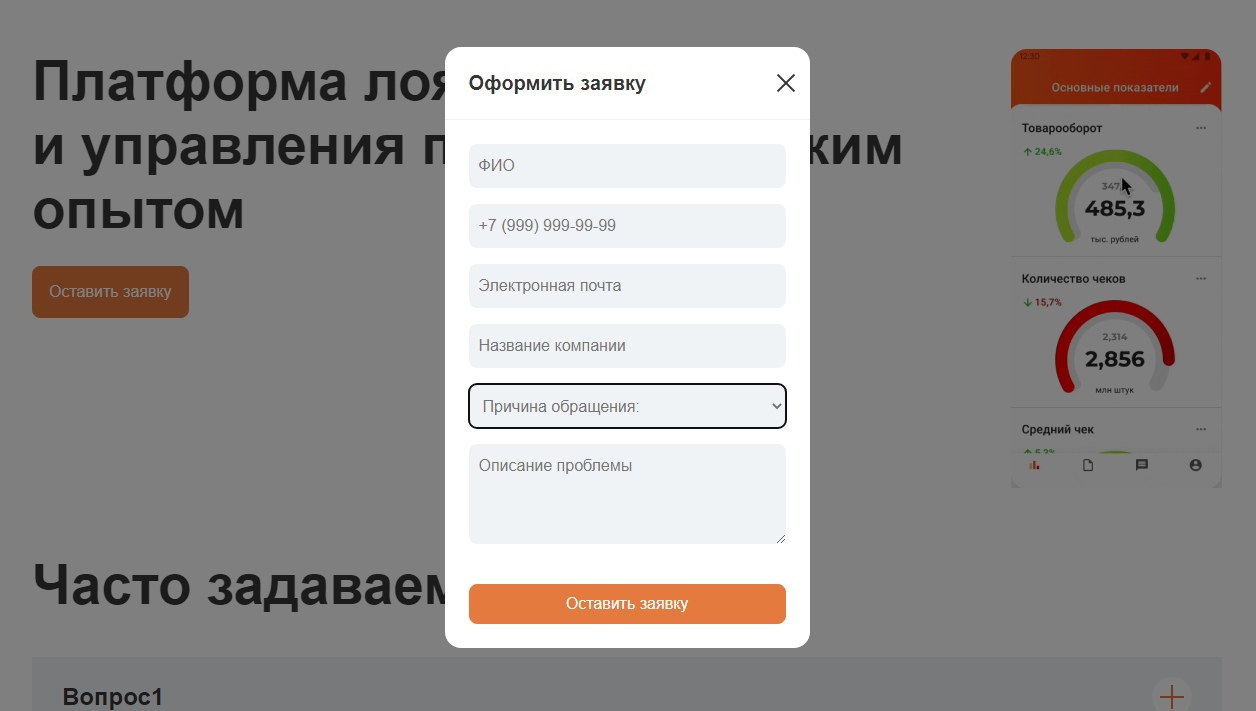


Рисунок 49 – Форма для заполнения заявки

На рисунке 50 представлена форма для проверки статуса заявки

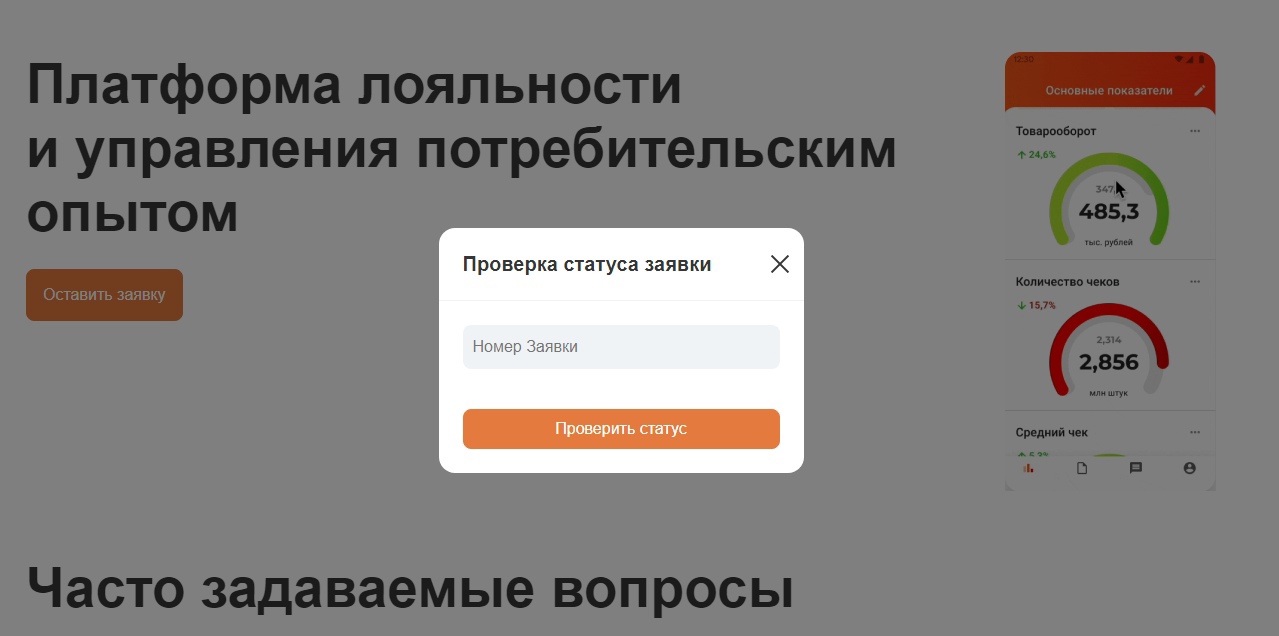


Рисунок 50 – Форма для проверки статуса заявки

## **3.4 Деплой сервера на удаленный компьютер**

В данном разделе рассмотрен процесс развертывания сервера на удаленном компьютере с использованием Docker и Git. Docker обеспечивает удобную среду для упаковки, распределения и управления приложениями, помогая обеспечить их однородность и надежность. Git служит системой контроля версий, что позволяет быстро и легко получать последние версии проекта и управлять ими.

## **3.5 Подготовка сервера**

Сначала проведена подготовка удаленного компьютера (далее - сервер). Был выбран хостинг-провайдер, который предоставляет услуги виртуальных серверов (VPS) или физических серверов. Затем был создан аккаунт, зарегистрирован у хостинг-провайдера и заказан сервер с необходимыми параметрами (оперативная память, жесткий диск, процессор и т.д.). На сервер была установлена операционная система (например, Ubuntu, CentOS, Debian и т.д.).

## **3.6 Установка Docker и Git на сервер**

После подготовки сервера были установлены Docker и Git. Процесс установки зависит от выбранной операционной системы, но в целом состоит из следующих шагов:

* Установка необходимых пакетов и зависимостей для Docker и Git.
* Добавление репозитория Docker.
* Установка Docker и Git.

## **3.7 Клонирование проекта с помощью Git**

Для получения последней версии проекта и его файлов на сервере, был выполнен следующий процесс:

* Создана рабочая директория на сервере.
* Командой git clone был склонирован репозиторий с сервером в рабочую директорию: git clone https://github.com/username/repository.git.

## **3.8 Создание Dockerfile**

Для развертывания сервера с помощью Docker, был создан Dockerfile - инструкция для сборки образа контейнера. В данном файле указаны основа образа, копирование файлов приложения, установка зависимостей и конфигурация сервера. На рисунке 51 представлен файл докера.



Рисунок 51 – Docker файл

## **3.9 Сборка и развертывание контейнера**

После создания Dockerfile, были выполнены следующие шаги для сборки и развертывания контейнера:

* В терминале сервера была выполнена команда сборки образа контейнера, указав имя образа и рабочую директорию: docker build -t server-image .
* Затем был запущен контейнер с помощью команды `docker run`, указав имя образа, созданного на предыдущем шаге, и опции для развертывания: docker run -d -p 8000:8000 --name server-container server-image

В этой команде:

- `-d` означает запуск контейнера в фоновом режиме (detached mode).

- `-p 8000:8000` указывает на проброс порта 8000 контейнера на порт 8000 хоста.

- `--name server-container` задает имя для контейнера.

## **3.10 Мониторинг и управление контейнером**

После развертывания контейнера было выполнено его мониторинг и управление с помощью следующих команд:

docker ps: Команда отображает список всех запущенных контейнеров.

docker logs server-container: Команда показывает логи контейнера с указанным именем.

docker stop server-container: Команда останавливает контейнер с указанным именем.

docker start server-container: Команда запускает контейнер с указанным именем.

docker rm server-container: Команда удаляет контейнер с указанным именем.

Сервер успешно развернут на удаленном компьютере с использованием Docker и Git, что обеспечивает надежность, масштабируемость и удобство управления приложением. В дальнейшем можно обновлять и модифицировать сервер, а также развертывать его на других платформах, сохраняя при этом однородность среды.

## **3.11 Реализация и развертывание системы: набор библиотек, оригинальный код и архитектура сети предприятия**

В данном параграфе будут описаны основные детали имплементации и развёртывания системы, включая использованный набор библиотек и оригинальный код. В серверной части проекта были использованы следующие библиотеки:

* Gin [22]: является легковесным фреймворком для разработки веб-приложений на языке Go. Он обеспечивает высокую производительность и удобство использования. Gin предоставляет роутинг, обработку запросов и ответов, а также другие функции для разработки серверной части приложения.
* go.uber.org/zap[23]: представляет собой библиотеку логирования для языка Go, разработанную компанией Uber. Она обладает высокой производительностью и гибкими возможностями настройки. Zap позволяет регистрировать различные уровни логов и может быть интегрирован с различными хранилищами логов.
* github.com/caarlos0/env/v6[51]: Эта библиотека предоставляет возможность удобной работы с переменными окружения в Go. Она позволяет определить структуру конфигурационных параметров, которые могут быть загружены из переменных окружения или файлов конфигурации.
* github.com/dgrijalva/jwt-go/v4: Jwt-go [26] - является библиотекой для работы с JSON Web Token (JWT) в языке Go. Она предоставляет функции для создания, валидации и обработки JWT. JWT используется для обмена информацией между клиентом и сервером с использованием цифровой подписи.
* github.com/sarulabs/di: DI (Dependency Injection) [52] — это паттерн проектирования, который позволяет управлять зависимостями между компонентами системы. Библиотека di предоставляет удобный способ реализации DI в языке Go. Она помогает организовать инверсию управления и обеспечить гибкость и расширяемость кода.

В клиентской части проекта была использована технология "Drag and Drop" в React. Эта функциональность позволяет пользователям перетаскивать и упускать элементы интерфейса в пределах веб-приложения. Использование данной функциональности может быть реализовано с помощью соответствующих библиотек, таких как react-dnd или react-beautiful-dnd. Они предоставляют компоненты и хуки, которые упрощают разработку и обработку перетаскивания элементов интерфейса в React.

На рисунках 42 представлен обработчик добавления заявки

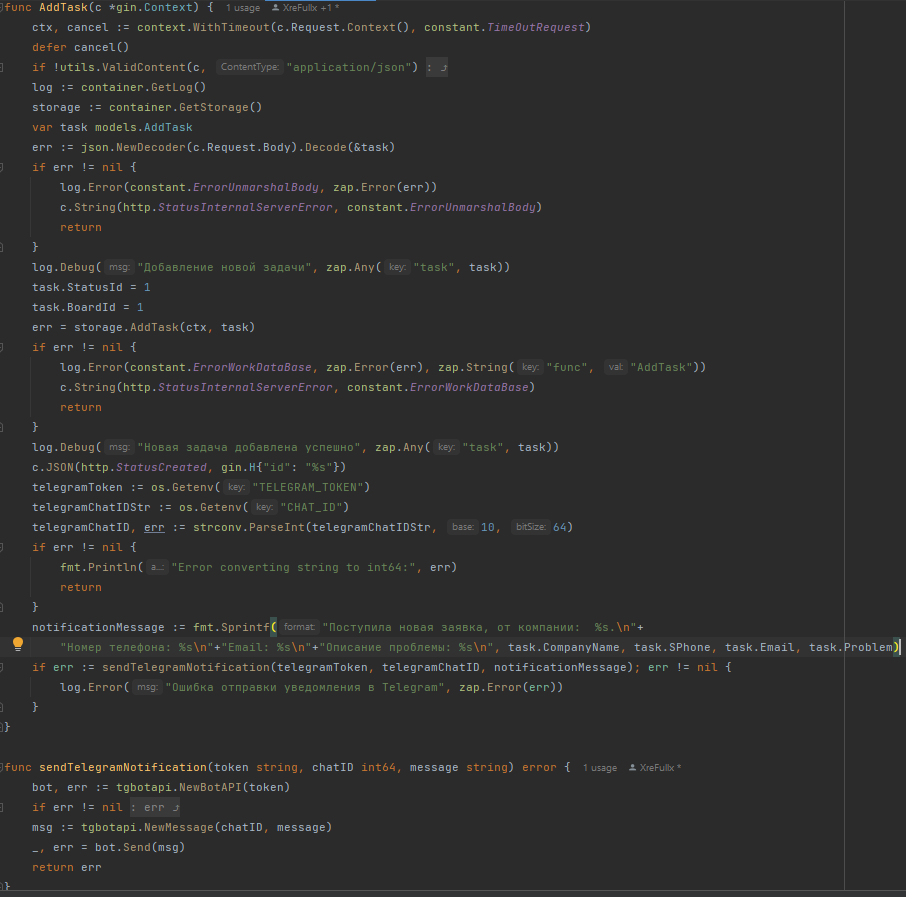


Рисунок 52 – Код добавления задачи

Данный код представляет собой функцию AddTask, которая обрабатывает запрос на добавление новой задачи в систему. Разберём его поэтапно:

* Сначала создаётся контекст и функция cancel, которая будет вызвана после выполнения функции AddTask или в случае превышения времени ожидания запроса. Это позволяет контролировать время выполнения запроса и предотвращает его зависание.
* Затем происходит проверка типа содержимого запроса с помощью функции ValidContent из пакета utils. Если тип содержимого не соответствует ожидаемому application/json, функция прерывает выполнение.
* Из контейнера container получаются логгер и хранилище данных.
* Создаётся переменная task типа models.AddTask, в которую декодируется JSON-тело запроса.
* Если при декодировании возникает ошибка, записывается соответствующее сообщение об ошибке в лог и возвращается HTTP-ответ с кодом http.StatusInternalServerError.
* Затем в лог записывается сообщение о добавлении новой задачи.
* Задаются значения StatusId и BoardId для задачи.
* Вызывается функция AddTask у хранилища данных storage для сохранения задачи в базе данных. Если возникает ошибка, записывается соответствующее сообщение об ошибке в лог и возвращается HTTP-ответ с кодом http.StatusInternalServerError.
* В лог записывается сообщение об успешном добавлении задачи.
* Возвращается HTTP-ответ со статусом http.StatusCreated и JSON-объектом, содержащим только идентификатор добавленной задачи.
* Получаются значения переменных TELEGRAM\_TOKEN и CHAT\_ID из переменных окружения.
* Значение CHAT\_ID преобразуется из строки в целое число.
* Формируется сообщение для отправки в Telegram, содержащее информацию о новой задаче.
* Вызывается функция sendTelegramNotification для отправки уведомления в Telegram с помощью бота, используя заданный токен и идентификатор чата.
* Если возникает ошибка при отправке уведомления, она записывается в лог.
* Функция sendTelegramNotification создаёт экземпляр бота с помощью заданного токена и отправляет сообщение в чат с указанным идентификатором.
* Если при создании бота возникает ошибка, она возвращается из функции.

В итоге, функция AddTask обрабатывает запрос на добавление новой задачи, сохраняет её в базе данных,

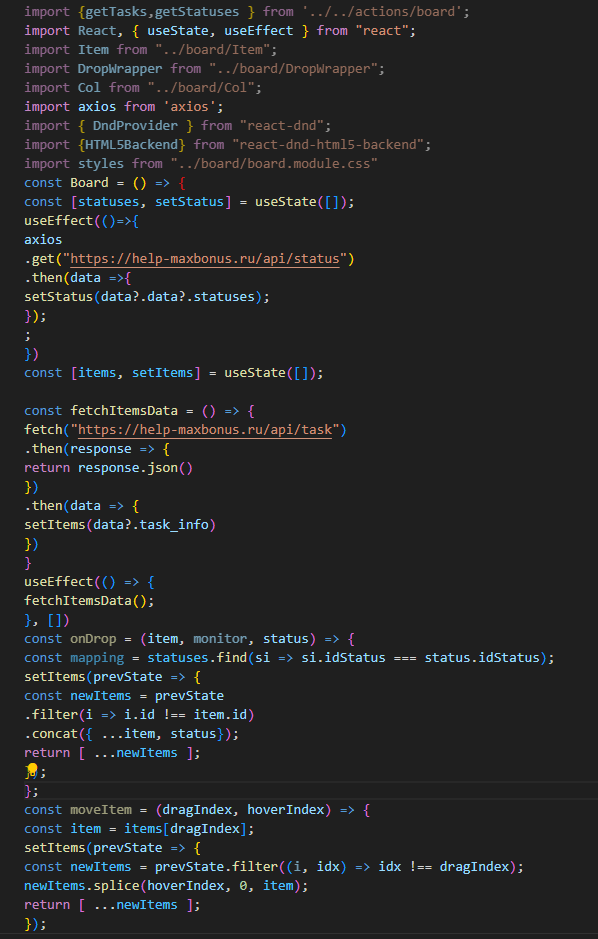
На рисунке 52 представлен код для перемещения задач. 

Рисунок 52 – Разметка страницы

Данный код представляет собой компонент "Board" в React, который отображает доску с задачами и позволяет перетаскивать их между статусами.

В начале импортируются необходимые зависимости, такие как функции getTasks и getStatuses из файла "actions/board", а также компоненты и стили для доски.

Затем объявляется компонент "Board" в виде функционального компонента.

Внутри компонента используются хуки состояния useState и useEffect для хранения и обновления данных.

* statuses - состояние, содержащее массив статусов задач.
* items - состояние, содержащее массив задач.

С помощью хука useEffect выполняется запрос к API для получения списка статусов задач. Результат полученного ответа сохраняется в состояние statuses с помощью функции setStatus.

Затем с помощью хука useEffect вызывается функция fetchItemsData, которая выполняет запрос к API для получения списка задач. Результат полученного ответа сохраняется в состояние items с помощью функции setItems.

Далее определены функции onDrop и moveItem.

* onDrop вызывается при перетаскивании задачи на другой статус на доске. Он обновляет состояние items, чтобы переместить задачу на новый статус.
* moveItem вызывается при перемещении задачи внутри одного статуса. Он обновляет состояние items, чтобы поменять порядок задач.

Затем компонент возвращает JSX-разметку, которая отображает доску с задачами. Используется компонент DndProvider из библиотеки "react-dnd" для обеспечения функциональности перетаскивания элементов. Внутри DndProvider находится компонент Col, который представляет статусные колонки на доске. Каждая колонка содержит компонент DropWrapper, в котором отображается список задач с помощью компонента Item.

В итоге, компонент "Board" позволяет отображать и управлять задачами на доске, а также осуществлять перетаскивание задач между статусами.

Для развёртывания системы в сети предприятия был использован гипервизор Proxmox, который предоставляет виртуальные машины (ВМ). В рамках этой системы вам была выделена виртуальная машина в качестве сервера. У этой ВМ были выделены следующие ресурсы: 4 ГБ оперативной памяти и 2 ядра процессора.

На гипервизоре Proxmox также был настроен и развернут веб-сервер Nginx. Nginx выполняет функцию проксирования запросов, и в зависимости от доменного имени и пути запроса, он направляет их на различные порты. Если запрос приходит с доменным именем "help-maxbonus.ru" и имеет путь "/api", то он перенаправляется на порт 8080. В противном случае, если путь не содержит "/api", запрос перенаправляется на порт 3000.

Виртуальная машина, которая была выделена для сервера, содержит среду Docker. Внутри Docker-контейнеров развернуты компоненты системы, включая фронтенд, бэкенд и базу данных. Docker позволяет упаковать приложение и его зависимости в контейнеры, которые могут быть легко развернуты и масштабированы.

Для подключения к виртуальной машине, используется протокол SSH (Secure Shell). SSH обеспечивает защищённое удалённое подключение и выполнение команд на удалённой машине. Учётные данные необходимы для аутентификации при подключении к виртуальной машине по SSH, такие как имя пользователя и пароль или SSH-ключ.

На рисунке 53 представлена диаграмма размещения сервера.

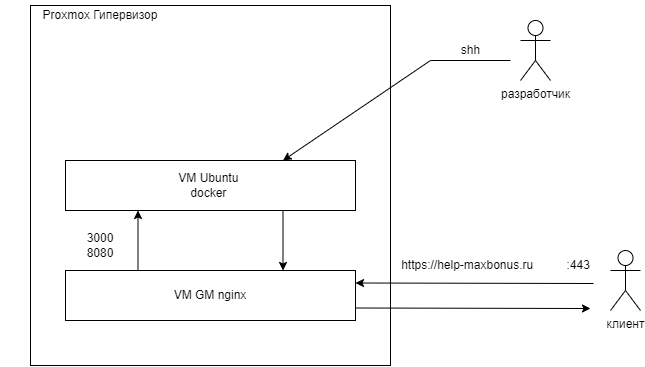


Рисунок 53 – Диаграмма размещения сервера

Общая архитектура системы включает в себя следующие компоненты:

* Гипервизор Proxmox, который управляет виртуальными машинами.
* Виртуальная машина с выделенными ресурсами (4 ГБ ОЗУ и 2 ядрами процессора) для развёртывания сервера.
* Веб-сервер Nginx, выполняющий функцию проксирования запросов.
* Docker для развёртывания контейнеров с фронтендом, бэкендом и базой данных.
* SSH для удалённого подключения к виртуальной машине.

Эта архитектура позволяет разделить компоненты системы, обеспечивая изоляцию и масштабируемость каждого из них.

**Вывод к главе 3**

В третьей главе была выполнена разработка и внедрение системы автоматизации приема и обработки заявок в ООО «Максбонус». В процессе реализации были использованы выбранные инструменты и технологии, основанные на техническом проекте, разработанном во второй главе. Успешное внедрение системы позволило оптимизировать рабочие процессы отдела, сократить время обработки заявок и минимизировать возможные ошибки. Таким образом, было доказано, что разработанная система является эффективным инструментом для автоматизации приема и обработки заявок отделом техподдержки ООО «Максбонус».

# ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломной работы "Автоматизация приема и обработки заявок отделом техподдержки ООО «Максбонус»" были успешно решены поставленные задачи:

Были выявлены и проанализированы требования к процессу автоматизации приема и обработки заявок, что позволило определить основные аспекты и потребности в данной области. Проведенный анализ позволил выявить ключевые узкие места и возможности для оптимизации процессов отделом техподдержки ООО «Максбонус».

Разработан технический проект системы автоматизации приема и обработки заявок, включая архитектуру, структуру базы данных, выбор инструментов и технологий для разработки системы. В результате был создан детальный план для дальнейшей реализации и внедрения системы.

Разработана и внедрена система автоматизации приема и обработки заявок отделом техподдержки ООО «Максбонус». Это позволило существенно ускорить и оптимизировать рабочие процессы отдела, снизить нагрузку на сотрудников и повысить уровень удовлетворенности клиентов компании.

Таким образом, автоматизация приема и обработки заявок отделом техподдержки ООО «Максбонус» позволила повысить эффективность работы отдела, сократить время на обработку заявок, уменьшить вероятность ошибок и улучшить качество предоставляемых услуг. В долгосрочной перспективе, внедрение данной системы автоматизации будет способствовать росту экономических показателей компании и успешному развитию бизнеса.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Wiegers K. E., Beatty J. Software Requirements. Microsoft Press, 2013.
2. IEEE Computer Society. ISO/IEC/IEEE 29148:2011, 2014.
3. Grady R. B. Practical Software Metrics for Project Management and Process Improvement. Prentice Hall, 1992.
4. PostgreSQL: Что такое PostgreSQL? [Электронный ресурс]. URL: https://postgrespro.ru/docs/postgresql/15/intro-whatis (дата обращения: 08.05.2023).
5. Типы индексов в PostgreSQL [Электронный ресурс]. URL: https://otus.ru/nest/post/1566/ (дата обращения: 08.05.2023).
6. MySQL [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/MySQL (дата обращения: 08.05.2023).
7. Go [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Go (дата обращения: 08.05.2023).
8. Node.js [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Node.js (дата обращения: 08.05.2023).
9. ASP.NET Web API [Электронный ресурс]. URL: https://docs.microsoft.com/en-us/aspnet/web-api/ (дата обращения: 08.05.2023).
10. React [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/React (дата обращения: 08.05.2023).
11. Vue.js: Обзор [Электронный ресурс]. URL: https://learn.microsoft.com/ru-ru/windows/dev-environment/javascript/vue-overview (дата обращения: 08.05.2023).
12. Angular [Электронный ресурс]. URL: https://angular.io/ (дата обращения: 08.05.2023).
13. Box G. E. P., Draper N. R. Empirical model-building and response surfaces. John Wiley & Sons, 1987.
14. Ross S. M. Introduction to probability models. Academic Press, 2014.
15. Jeston J., Nelis J. Business Process Management: Practical Guidelines to Successful Implementations. Routledge, 2018.
16. Dumas M., La Rosa M., Mendling J., Reijers H. A. Process Aware Information Systems: Bridging People and Software Through Process Technology. Wiley, 2013.
17. Winter K., Rinderle-Ma S. Improving Business Process Automation by Combining Workflow Management and Machine Learning. Business & Information Systems Engineering, 63(3), 205-219, 2021.
18. PostgreSQL Documentation [Электронный ресурс]. URL: https://www.postgresql.org/docs/ (дата обращения: 08.05.2023).
19. NIST: Secure Hash Standard (SHS) [Электронный ресурс]. URL: <https://nvlpubs.nist.gov/nistpubs/FIPS> NIST.FIPS.180-4.pdf (дата обращения: 08.05.2023).
20. Go Documentation: crypto/sha256 package [Электронный ресурс]. URL: https://golang.org/pkg/crypto/sha256/ (дата обращения: 08.05.2023).
21. OWASP: Password Storage Cheat Sheet [Электронный ресурс]. URL: https://cheatsheetseries.owasp.org/cheatsheets/Password\_Storage\_Cheat\_Sheet.html (дата обращения: 08.05.2023).
22. Erl, T. (2005). Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall PTR.
23. Boehm, B., & Turner, R. (2004). Balancing Agility and Discipline: A Guide for the Perplexed. Addison-Wesley Professional.
24. Papazoglou, M. P., & van den Heuvel, W. J. (2007). Service oriented architectures: approaches, technologies and research issues. The VLDB Journal, 16(3), 389-415.
25. Chikofsky, E. J., & Cross, J. H. (1990). Reverse engineering and design recovery: a taxonomy. IEEE Software, 7(1), 13-17.
26. Rumbaugh, J., Jacobson, I., & Booch, G. (2004). The Unified Modeling Language Reference Manual, Second Edition. Addison-Wesley Professional.
27. Sparx Systems. (n.d.). Enterprise Architect. Retrieved from https://www.sparxsystems.com/products/ea/index.html.
28. Gin [Электронный ресурс]. URL: https://github.com/gin-gonic/gin (дата обращения: 08.05.2023).
29. Dig [Электронный ресурс]. URL: https://github.com/uber-go/dig (дата обращения: 08.05.2023).
30. Wire [Электронный ресурс]. URL: https://github.com/google/wire (дата обращения: 08.05.2023).
31. Inject [Электронный ресурс]. URL: https://github.com/facebookarchive/inject (дата обращения: 08.05.2023).
32. jwt-go [Электронный ресурс]. URL: https://github.com/dgrijalva/jwt-go (дата обращения: 08.05.2023).
33. Golang JWT [Электронный ресурс]. URL: https://github.com/golang-jwt/jwt (дата обращения: 08.05.2023).
34. BotFather [Электронный ресурс]. Telegram. URL: https://core.telegram.org/bots#botfather (дата обращения: 08.05.2023).
35. Telegram Bot API [Электронный ресурс]. Telegram. URL: https://core.telegram.org/bots/api (дата обращения: 08.05.2023).
36. go-telegram-bot-api [Электронный ресурс]. GitHub. URL: https://github.com/go-telegram-bot-api/telegram-bot-api (дата обращения: 08.05.2023).
37. Church J. Telegram Bots: An introduction for developers [Электронный ресурс]. Medium, 24 апреля 2016. URL: https://medium.com/@jonchurch/telegram-bots-an-introduction-for-developers-2ff2dfc86b6f (дата обращения: 08.05.2023).
38. Abramov, D., & Zlatkov, A. (2017). The Complete Redux Book: Designing, developing, and debugging maintainable and scalable Redux applications. Leanpub.
39. Facebook Inc. (2021). React - A JavaScript library for building user interfaces. [Электронный ресурс]. <URL:https://reactjs.org/> (дата обращения: 08.05.2023).
40. Geary, D. M. (2018). Core React: A hands-on guide to building maintainable and high-performance Web UIs. Pearson Education.
41. Krohn, M. React.js - A guide book on how to create, deploy and maintain highly-scalable and high-performance web applications. Packt Publishing.
42. Mater, R. (2019). Redux in Action. Manning Publications.
43. Noring, J. (2020). Full-Stack React, TypeScript, and Node: Build cloud-ready web applications using React 17 with Hooks and GraphQL. Packt Publishing.
44. Oliveira, D. (2018). Material-UI Cookbook: Build modern day application by implementing Material Design principles in React applications using Material-UI. Packt Publishing.
45. Banks, A., & Porcello, E. (2020). Learning React: Modern Patterns for Developing React Apps. O'Reilly Media.
46. Kulkarni, A. (2019). Mastering Axios: A hands-on guide to making HTTP requests using Axios. Packt Publishing.
47. Bares, B. (2018). Secure Your Node.js Web Application: Keep Attackers Out and Users Happy. Pragmatic Bookshelf.
48. Lindesay, J. (2019). Beginning Node.js, Express & MongoDB Development. Apress.
49. Haverbeke, M. (2018). Eloquent JavaScript: A Modern Introduction to Programming. No Starch Press.
50. Chacon, S., & Straub, B. (2021). Pro Git: Everything You Need to Know About Git. Apress.
51. caarlos0/env/v6 [Электронный ресурс]. URL: https://github.com/ caarlos0/env/v6 (дата обращения: 08.05.2023).
52. DI [Электронный ресурс]. URL: https://github.com/ sarulabs/di (дата обращения: 08.05.2023).
53. Стандарт университета «Общие требования к построению, изложению и оформлению документов учебной деятельности» // Сибирский федеральный университет: официальный сайт. – URL: https://about.sfu-kras.ru/docs/8127/pdf (дата обращения: 08.05.2023).

# СЛОВАРЬ

1. Мидлвары (middlewares) - программные компоненты, которые находятся между клиентским запросом и серверным обработчиком, выполняя дополнительные функции, такие как проверка аутентификации или логирование.
2. RESTful API (Representational State Transfer API) - архитектурный стиль интерфейсов для взаимодействия между клиентом и сервером, основанный на принципах стандартных HTTP-методов и предоставляющий унифицированный доступ к ресурсам.
3. Роуты (routes) - пути, по которым клиентские запросы направляются к определенным обработчикам в веб-приложении.
4. Эндпоинт (endpoint) - уникальный URL-адрес, по которому клиентские запросы отправляются для получения или отправки данных в веб-приложении.
5. Хэширование (hashing) - процесс преобразования данных в уникальное значение фиксированной длины, использующийся для обеспечения безопасности и целостности данных.

# ПРИЛОЖЕНИЕ А. Акт о внедрении