**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc134938782)

[1 ОБОСНОВАНИЕ АКТУАЛЬНОСТИ РАЗРАБОТКИ 6](#_Toc134938783)

[1.1 Анализ предметной области 6](#_Toc134938784)

[1.2 Анализ существующих подходов и обоснование актуальности разработки. 8](#_Toc134938785)

[1.3 Требования к программному обеспечению 10](#_Toc134938786)

[1.4 Выбор средств разработки 12](#_Toc134938787)

[1.5 Возможность внедрения 14](#_Toc134938788)

[1.6 Подведение итогов исследования. 15](#_Toc134938789)

[2 Разработка программного обеспечения 16](#_Toc134938790)

[2.1 Анализ требований к системе. 16](#_Toc134938791)

[2.2 Разработка концепции и архитектуры системы. 18](#_Toc134938792)

[2.3 Проектирование процессов системы. 19](#_Toc134938793)

[2.4 Описание алгоритма автоматического назначения заявок на специалиста. 20](#_Toc134938794)

[2.5 Проектирование базы данных. 20](#_Toc134938795)

[2.6 Разработка серверной части. 20](#_Toc134938796)

[2.7 Разработка клиентской части. 20](#_Toc134938797)

[2.8 Интеграция клиентской и серверной части. 20](#_Toc134938798)

[3 Тестирование программного обеспечения 20](#_Toc134938799)

[3.1 Планирование тестирования 20](#_Toc134938800)

[3.2 Описание результатов тестирования 20](#_Toc134938801)

[3.3 Анализ результатов тестирования 20](#_Toc134938802)

[ЗАКЛЮЧЕНИЕ 21](#_Toc134938803)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 22](#_Toc134938804)

[ПРИЛОЖЕНИЕ 24](#_Toc134938805)

ВВЕДЕНИЕ

В современном информационном обществе, где техническая поддержка является важной составляющей успешного функционирования организаций, автоматизация этого процесса играет решающую роль. Разработка программного обеспечения для автоматизации работы службы технической поддержки становится неотъемлемым фактором в повышении эффективности, сокращении времени реагирования на запросы клиентов и улучшении обслуживания.

Целью данной дипломной работы является разработка программного обеспечения, которое обеспечит эффективное функционирование службы технической поддержки. В работе будет осуществлен анализ существующих подходов к автоматизации работы службы технической поддержки, а также рассмотрены современные технологии и методы, необходимые для реализации данного программного решения.

В ходе работы будет предложена концепция системы, которая позволит клиентам, являющимся сотрудниками компании, создавать заявки по проблемам, связанным с техникой и программным обеспечением. Заявки будут разделены по категориям, приоритету и возможности удаленного решения проблемы. Для решения задачи автоматического распределения заявок между специалистами будет разработан алгоритм, учитывающий приоритет, нагрузку и доступность каждого специалиста.

В ходе разработки программного обеспечения будет использован набор современных технологий. Для серверной части будет выбран язык программирования Java, а именно, фреймворк Spring, включая его модули Spring Boot, Spring Data JPA и Spring Security, который будет использоваться для упрощения и ускорения разработки, конфигурации и развертывания веб-приложения, а также для обеспечения безопасности и удобного взаимодействия с базой данных.

Для клиентской части программного обеспечения будет применяться React - популярный и мощный JavaScript-фреймворк, позволяющий создавать современные и отзывчивые пользовательские интерфейсы. React обладает большим сообществом разработчиков и широким выбором инструментов для разработки фронтенда.

В качестве системы управления базами данных будет использован PostgreSQL, что обеспечит надежное хранение и быстрый доступ к данным, а также поддержку расширенных возможностей для масштабирования и обработки больших объемов информации.

В дальнейшем тексте дипломной работы будет детально рассмотрены следующие разделы, необходимые для достижения поставленной задачи:

* Обоснование актуальности разработки:
  + Анализ предметной области.
  + Анализ существующих подходов и обоснование актуальности разработки.
  + Требования к программному обеспечению.
  + Выбор средств для разработки.
  + Возможность внедрения.
  + Подведение итогов исследования.
* Разработка программного обеспечения:
  + Анализ требований к системе.
  + Проектирование архитектуры системы.
  + Разработка алгоритма автоматического назначения заявок на специалистов.
  + Проектирование базы данных.
  + Разработка серверной части.
  + Разработка клиентской части.
  + Интеграция серверной и клиентской частей.
* Тестирование программного обеспечения:
  + Планирование тестирования.
  + Осуществление тестирования.
  + Анализ результатов.
* Составление программной документации:
  + Описание программного продукта.
  + Руководство для сотрудника – клиента системы.
  + Руководство для специалиста технической поддержки.
  + Руководство для администратора системы.

В заключении будут подведены итоги работы, сделан обзор достигнутых результатов и выполнения поставленных задач. Будет оценена эффективность и преимущества разработанного программного обеспечения в контексте автоматизации работы службы технической поддержки. Также будут высказаны рекомендации по дальнейшему развитию и улучшению системы, а также возможности ее применения в реальных условиях.

Таким образом, разработка программного обеспечения с использованием указанных технологий будет способствовать созданию клиент ориентированной системы технической поддержки, обеспечивая эффективное и качественное обслуживание сотрудников компании в их запросах и проблемах, связанных с техникой и программным обеспечением.

# ОБОСНОВАНИЕ АКТУАЛЬНОСТИ РАЗРАБОТКИ

## Анализ предметной области

Анализ предметной области является важным этапом в разработке программного обеспечения для автоматизации работы службы технической поддержки. В данном исследовании мы изучили основные аспекты и особенности предметной области, чтобы лучше понять требования и потребности пользователей, а также определить оптимальные подходы к разработке системы.

Одним из важных аспектов предметной области является эффективное управление заявками и обработка проблем клиентов. В технической поддержке возникает множество запросов от пользователей, связанных с техническими проблемами и неисправностями, а также с запросами на установку, настройку или обновление программного обеспечения. Цель автоматизации работы службы технической поддержки заключается в оптимизации процессов обработки заявок, сокращении времени реакции и улучшении качества обслуживания клиентов.

Также важным аспектом предметной области является взаимодействие между клиентами и специалистами технической поддержки. Исследование позволило выявить необходимость удобных средств коммуникации, таких как система комментариев и возможность прикрепления файлов к заявкам. Это обеспечивает эффективное взаимодействие, обмен информацией и улучшает качество обслуживания клиентов.

Анализ предметной области является важным этапом разработки программного обеспечения для автоматизации работы службы технической поддержки. Он позволяет более глубоко понять особенности и требования данной области, выявить проблемы, с которыми сталкиваются клиенты и специалисты технической поддержки, а также определить функциональные и нефункциональные требования к системе.

В ходе анализа предметной области были проанализированы основные аспекты работы службы технической поддержки. Рассмотрены типичные проблемы, с которыми могут столкнуться клиенты, связанные как с техникой (например, неисправность оборудования, необходимость замены или ремонта), так и с программным обеспечением (сбои, ошибки, проблемы с установкой или удалением программ).

Также были изучены особенности процесса обработки заявок. Были выявлены этапы, которые включают создание заявки, ее регистрацию в системе, распределение на исполнение, выполнение работ, закрытие и отслеживание статуса заявки. Анализ процесса позволил определить ключевые требования к системе, такие как удобный интерфейс для создания и отслеживания заявок, возможность прикрепления файлов и комментариев, а также эффективное распределение заявок между специалистами.

Исходя из проведенного анализа предметной области, были выявлены основные проблемы, требования и потребности, которые должны быть учтены при разработке программного обеспечения для автоматизации работы службы технической поддержки. Некоторые из основных выводов анализа предметной области включают:

1. Комплексность проблем: Клиенты могут сталкиваться с различными проблемами, связанными как с аппаратным обеспечением, так и с программным обеспечением. Это требует разработки системы, способной обрабатывать разнообразные типы заявок и предоставлять соответствующие инструменты для их решения.
2. Необходимость оперативного реагирования: Клиенты ожидают быстрого и эффективного решения своих проблем. Система должна обеспечивать оперативное распределение заявок между специалистами, отслеживание их статуса и своевременное информирование клиентов о прогрессе решения.
3. Управление процессом: эффективное управление процессом работы службы технической поддержки включает регистрацию и классификацию заявок, планирование и отслеживание работ, а также анализ статистики и отчетность. Система должна предоставлять соответствующие инструменты для этих задач, упрощая процесс управления и повышая его прозрачность.
4. Взаимодействие между специалистами и клиентами: Коммуникация является важной частью работы службы технической поддержки. Система должна обеспечивать возможность обмена сообщениями и файлами между специалистами и клиентами, создавая эффективный канал связи и упрощая процесс общения.
5. Гибкость и расширяемость: в современной среде бизнеса требования и потребности могут изменяться со временем. Поэтому система должна быть гибкой и легко расширяемой, чтобы адаптироваться к новым требованиям и интегрироваться с другими системами.

Исследование предметной области позволяет лучше понять основные аспекты работы службы технической поддержки и определить ключевые требования, которые должны быть учтены при разработке программного обеспечения. Это обеспечивает основу для успешной разработки системы, способной повысить эффективность и качество работы службы технической поддержки, улучшить взаимодействие между специалистами и клиентами, а также обеспечить более оперативное и надежное решение проблем.

## Анализ существующих подходов и обоснование актуальности разработки.

Анализ существующих подходов к автоматизации работы службы технической поддержки позволяет выявить сильные и слабые стороны различных решений и обосновать актуальность разработки программного обеспечения, описанного выше.

Существующие подходы к автоматизации работы службы технической поддержки включают использование тикет-систем, электронной почты, чат-ботов и специализированных программных решений. Каждый из этих подходов имеет свои преимущества и недостатки.

Тикет-системы и электронная почта позволяют организовать процесс обработки запросов клиентов, однако они часто требуют ручной обработки и недостаточно эффективны при большом объеме заявок. Кроме того, в них отсутствует возможность автоматического назначения заявок на специалистов и контроля времени реагирования.

Чат-боты становятся все более популярными и предлагают быстрый и интерактивный способ взаимодействия с клиентами. Однако они не всегда способны предоставить полноценное решение проблемы клиента и требуют наличия заранее подготовленных ответов и сценариев.

Разработка программного обеспечения, описанного выше, предлагает решение, объединяющее преимущества различных подходов и обеспечивающее более эффективную автоматизацию работы службы технической поддержки. Возможность создания заявок, разделение их по категориям, приоритету и возможности удаленного решения проблемы позволяет структурировать процесс обработки запросов клиентов и повысить эффективность работы специалистов.

Особенно актуальной является функциональность автоматического назначения заявок на специалистов. Это позволяет оптимизировать процесс распределения работы между сотрудниками и гарантировать быстрое реагирование на запросы клиентов. Алгоритм автоматического назначения заявок, зависящий от приоритета и времени реагирования, обеспечивает справедливое распределение нагрузки и минимизацию простоев в обработке заявок.

Для обеспечения эффективного взаимодействия между специалистами технической поддержки и клиентами в разрабатываемой системе будет предусмотрена функциональность комментирования и прикладывания файлов к заявкам.

Такой функционал общения и обмена файлами существенно улучшит качество и эффективность работы службы технической поддержки. Специалисты смогут получать более полную информацию о проблемах клиентов, а клиенты смогут быстрее и удобнее предоставлять необходимые материалы для анализа и решения проблемы.

Кроме того, разработанное программное обеспечение позволяет иметь централизованную систему управления заявками, что способствует лучшей организации и отслеживанию процесса обработки. Специалисты технической поддержки могут видеть все активные заявки, их приоритеты и статусы, что позволяет им эффективно планировать свою работу и сосредотачиваться на наиболее срочных и важных задачах.

Актуальность разработки программного обеспечения для автоматизации работы службы технической поддержки обосновывается не только повышением эффективности и качества обслуживания клиентов, но и сокращением затрат на техническую поддержку. Автоматизация позволяет сократить время на обработку и решение проблем, уменьшить необходимость в привлечении большого количества сотрудников и минимизировать человеческий фактор, который может привести к ошибкам или пропуску заявок.

## Требования к программному обеспечению

Базируясь на особенностях и анализе предметной области, вот некоторые требования к разрабатываемому веб-приложению для автоматизации работы службы технической поддержки:

* Авторизация и аутентификация: Пользователи должны иметь возможность регистрации, аутентификации и управления своими учетными записями. Необходимо обеспечить безопасное хранение и передачу пользовательских данных.
* Управление заявками: приложение должно предоставлять возможность создания, отслеживания, редактирования и удаления заявок. Каждая заявка должна иметь уникальный идентификатор, описание проблемы, приоритет, статус, отметку о назначенном ответственном лице, фиксировать время и дату создания и решения проблемы.
* Учет состояние и владельцев техники: система должна иметь возможность отслеживать состояние техники, такое как новое, списанное, ранее использованное или имеющее дефекты. Это позволит эффективно управлять техническими ресурсами компании. Каждая единица техники должна быть связана с конкретным владельцем, который может быть сотрудником компании. Также должна быть возможность указать, что техника находится на складе офиса и не закреплена за каким-либо сотрудником.
* Автоматическое распределение заявок: система должна автоматически назначать специалиста по специальному алгоритму, который будет описан в дальнейшем, в случае если превышен срок ожидания принятие заявки в работу.
* Комментарии и обсуждение: пользователи должны иметь возможность добавлять комментарии к заявкам, обсуждать проблему со специалистами и прикладывать дополнительные файлы, необходимые для решения проблемы. Комментарии должны быть связаны с конкретными заявками и отображаться в хронологическом порядке.
* Фильтрация и сортировка: Пользователи должны иметь возможность фильтровать заявки по различным параметрам, таким как статус, приоритет, дата создания, категории, возможности онлайн-решения и т.д. Также следует предусмотреть возможность сортировки заявок по заданным критериям, например, по приоритету или категории.
* Многопользовательская поддержка: система должна поддерживать работу нескольких пользователей одновременно, позволяя им работать с заявками и обмениваться информацией в реальном времени. Разграничение прав доступа должно быть реализовано, чтобы пользователи могли получать доступ только к необходимым им данным и функциональности.
* Безопасность: безопасность данных является важным аспектом разработки программного продукта. Приложение должно обеспечивать защиту данных пользователя, использовать шифрование при передаче данных по сети и иметь механизмы аутентификации и авторизации для предотвращения несанкционированного доступа.
* Документация и руководство пользователя: разработка должна сопровождаться документацией и руководством пользователя, которые помогут пользователям быстро освоить приложение и эффективно использовать его функциональность.
* Поддержка различных браузеров: приложение должно быть совместимо с различными популярными веб-браузерами (например, Chrome, Firefox, Safari, Edge) и обеспечивать одинаковое качество работы в любом из них.

## Выбор средств разработки

Определение выбранных технологий для разработки программного обеспечения базируется на особенностях разрабатываемой системы, а также учете преимуществ open-source продуктов. Основываясь на этих критериях, были выбраны следующие технологии:

* Spring: является одним из наиболее популярных и широко используемых фреймворков для разработки приложений на языке Java. Он обеспечивает мощный инструментарий для создания эффективных и надежных приложений, включая модули Spring Boot, Spring Data JPA и Spring Security. Spring Boot облегчает настройку и развертывание приложения, а Spring Data JPA предоставляет удобный способ взаимодействия с базой данных. Spring Security обеспечивает безопасность приложения и управление доступом пользователей.
* React: является популярной библиотекой JavaScript для создания пользовательских интерфейсов. Он позволяет разрабатывать мощные и интерактивные веб-приложения, основанные на компонентном подходе. Благодаря виртуальной DOM и эффективному управлению состоянием, React обеспечивает быструю отрисовку интерфейса и удобное взаимодействие с пользователем.
* PostgreSQL: предлагает мощные возможности хранения и управления данными. PostgreSQL отличается высокой надежностью, производительностью и поддержкой различных средств обеспечения целостности данных. Это позволяет эффективно работать с большим объемом данных, необходимых для работы службы технической поддержки.
* Docker: предоставляет инструменты для развертывания и управления контейнерами, что упрощает процесс развертывания и обеспечивает единообразие окружения между разработкой, тестированием и производством. Docker обеспечивает изоляцию приложения, упрощает масштабирование и повышает гибкость разработки и развертывания программного обеспечения.

Таким образом, выбор open-source технологий, таких как Spring, React, PostgreSQL, Docker, позволяет компании разрабатывать программное обеспечение с использованием мощных инструментов, обеспечивает гибкость, надежность, безопасность и снижает затраты на лицензии.

В качестве среды разработки выбрана IntelliJ IDEA является одной из ведущих интегрированных сред разработки (IDE). Одним из основных факторов при выборе IDE является удобство для команды разработчиков и возможность работать со всем выбранным набором технологий в одном месте. Исходя из этих критериев IntelliJ IDEA выбрана в качестве отличного инструмента для работы с исходным кодом при разработке программного обеспечения для автоматизации работы службы технической поддержки.

## Возможность внедрения

Возможность внедрения системы технической поддержки может принести несколько значительных преимуществ:

* Улучшение обслуживания клиентов: Внедрение системы технической поддержки позволит компании оперативно реагировать на запросы и проблемы клиентов. Это повысит качество обслуживания, ускорит процесс решения проблем и улучшит общую удовлетворенность клиентов.
* Эффективное управление запросами и проблемами: система технической поддержки позволит эффективно управлять запросами и проблемами клиентов. Все обращения будут централизованно отслеживаться, адекватно распределяться между специалистами и контролироваться в рамках установленных SLA (уровней обслуживания).
* Улучшение коммуникации и взаимодействия: система технической поддержки обеспечит эффективное взаимодействие между специалистами и клиентами. Будет возможность обмена сообщениями, прикрепления файлов и отслеживания истории коммуникации. Это сократит время и усилия, затрачиваемые на общение и уточнение информации.
* Анализ и отчетность: система будет предоставлять инструменты для анализа данных, отчетности и мониторинга производительности. Это поможет компании выявить общие тенденции, узкие места в процессе обслуживания и принимать информированные решения для улучшения работы службы технической поддержки.
* Повышение производительности и эффективности: внедрение системы технической поддержки позволит автоматизировать и оптимизировать многие рутинные задачи, такие как маршрутизация запросов, отслеживание статусов и приоритетов, управление временем реагирования и т.д. Это сэкономит время и ресурсы, повысит производительность и эффективность работы службы технической поддержки.

В целом, внедрение системы технической поддержки приведет к повышению качества обслуживания клиентов, оптимизации внутренних процессов и повышению эффективности работы службы технической поддержки. Это позволит компании повысить уровень удовлетворенности сотрудников и конкурентоспособность бизнеса.

## Подведение итогов исследования.

Анализ предметной области помог определить основные цели и задачи разрабатываемого программного обеспечения, а также выбрать подходящие технологии и инструменты для его разработки. На основе проведенного исследования предметной области мы можем приступить к следующему этапу - проектированию и разработке программного обеспечения для автоматизации работы службы технической поддержки.

Также был проведен обзор существующих решений на рынке. Исследование показало, что существует ряд программных продуктов, предназначенных для автоматизации работы службы технической поддержки. Однако, большинство из них имеют ограниченный функционал или не удовлетворяют полностью требованиям и потребностям конкретных компаний. Это подтверждает актуальность разработки собственной системы, специально адаптированной к нуждам и особенностям конкретной службы технической поддержки.

# Разработка программного обеспечения

## Анализ требований к системе.

Анализ требований к программному обеспечению подразумевает описание функциональности системы, выделение функциональных и нефункциональных требований.

### Диаграмма вариантов использования

Диаграмма вариантов использования (Use Case Diagram) является графическим инструментом моделирования, который используется для описания функциональности системы с точки зрения ее акторов (пользователей или внешних систем) [1]. Она помогает понять, как различные акторы взаимодействуют с системой и как система отвечает на их запросы.

Основная цель диаграммы вариантов использования заключается в идентификации и описании функциональных требований системы. Она помогает разработчикам и заинтересованным сторонам получить ясное представление о том, как система должна взаимодействовать с ее окружением.

Диаграмма вариантов использования, разрабатываемой системы показана на рисунке 1.

****

Рисунок - Диаграмма вариантов использования, разрабатываемого программного продукта

### Функциональные требования

Функциональные требования определяют, какая функциональность и возможности должны быть реализованы в разрабатываемой системе. Они описывают, как система должна взаимодействовать с пользователями и выполнять определенные действия и операции. Функциональные требования определяют конкретные задачи, которые система должна выполнять, и описывают ожидаемые результаты и поведение системы в различных сценариях использования.

Исходя из построенной диаграммы можно выделить следующие функциональные требования:

* Администратор должен иметь возможность создавать, удалять и редактировать информацию о пользователях системы.
* Администратор должен иметь возможность добавления, редактирования, удаления категорий техники и программного обеспечения.
* Администратор должен иметь возможность добавления, редактирования, удаления категорий заявок на технику и программное обеспечение.
* Специалист ТХП должен иметь возможность обрабатывать заявки различных категорий.
* Специалист ТХП должен иметь возможность отслеживать состояние техники, включая информацию о владельце и статусе (новое, списанное, ранее использованное, с дефектами) – вести учет.
* Специалист ТХП должен иметь возможность просматривать архив заявок.
* Специалист ТХП должен иметь возможность комментирования и приложения дополнительных файлов к заявке для уточнения и получение актуальной информации от клиента.
* Клиент должен иметь возможность создания заявок на решение проблем с техникой или программным обеспечением.
* Клиент должен иметь возможность отслеживания состояние своих заявок.
* Клиент должен иметь возможность отслеживать состояние, закрепленной за ним техники.
* Клиент должен иметь возможность комментирования и приложения дополнительных файлов к заявке для объяснения своей проблемы и коммуникации со специалистом ТХП.
* Система должна обеспечивать фильтрацию и сортировку элементов.
* Система должна обеспечивать автоматическое распределение заявок между специалистами технической поддержки, в том случае, если заявка не была взята в работу.
* Система должна обеспечивать авторизацию и аутентификацию пользователей.

### Нефункциональные требования

Нефункциональные требования определяют атрибуты и характеристики системы, которые не связаны непосредственно с ее функциональностью, но оказывают влияние на ее общую производительность, качество и эффективность. Они описывают такие аспекты, как надежность, производительность, безопасность, масштабируемость, удобство использования и другие атрибуты системы.

Для разрабатываемого программного продукта можно выделить следующие нефункциональные требования:

* Система должна быть способна масштабироваться горизонтально, чтобы справляться с увеличением нагрузки.
* Система должна обеспечивать аутентификацию и авторизацию пользователе.
* Взаимодействие между клиентской и серверной частями должно быть защищено с помощью протокола HTTPS.
* Доступ к конфиденциальным данным, таким как личная информация клиентов, должен быть ограничен и защищен от несанкционированного доступа.
* Веб-интерфейс системы должен быть адаптивным, одинаково хорошо работать на различных браузерах и устройствах.

## Проектирование архитектуры системы.

### Проектирование архитектуры приложения.

Проектирование системы — это процесс создания архитектурного и технического решения для разрабатываемой системы. Оно включает в себя определение структуры системы, ее компонентов, интерфейсов, а также выбор подходящих технологий, архитектурных шаблонов и принципов, которые будут использоваться при разработке [2].

При проектировании разрабатываемой программной системы было принято решение использовать классическую трехзвенную архитектуру, состоящую из клиентского, серверного уровней и базы данных. Этот выбор обоснован рядом преимуществ, которые такая архитектура предоставляет:

1. Распределение ответственности: позволяет четко разделить функциональность и ответственность между различными уровнями системы. Клиентский уровень отвечает за пользовательский интерфейс и взаимодействие с пользователем, серверный уровень обрабатывает бизнес-логику и взаимодействует с базой данных, а базовый уровень отвечает за хранение и обработку данных. Это способствует более четкому распределению задач и обеспечивает модульность и гибкость системы.
2. Масштабируемость и гибкость: позволяет гибко масштабировать и расширять систему. Каждый уровень может масштабироваться независимо от других, что обеспечивает горизонтальное масштабирование и легкую добавление новых функциональных модулей. Это позволяет системе эффективно реагировать на изменения в требованиях и нагрузке.
3. Удобство сопровождения и разработки: Трехзвенная архитектура обеспечивает четкое разделение функций и ответственности между уровнями, что упрощает сопровождение и разработку системы. Каждый уровень может быть разработан и тестирован независимо, что улучшает модульность и повторное использование кода. Это также способствует параллельной разработке и позволяет командам специализироваться на определенных уровнях.
4. Безопасность и контроль доступа: позволяет централизованно управлять безопасностью и контролем доступа к системе. Серверный уровень может обеспечивать аутентификацию и авторизацию пользователей, а также применять механизмы шифрования для защиты данных.
5. Интеграция и повторное использование: облегчает интеграцию системы с другими внешними сервисами и компонентами. Каждый уровень может быть независимо интегрирован с внешними системами, обеспечивая гибкую и модульную интеграцию. Более того, благодаря четкому разделению функций и модульности каждого уровня, компоненты системы могут быть повторно использованы в других проектах или модулях.

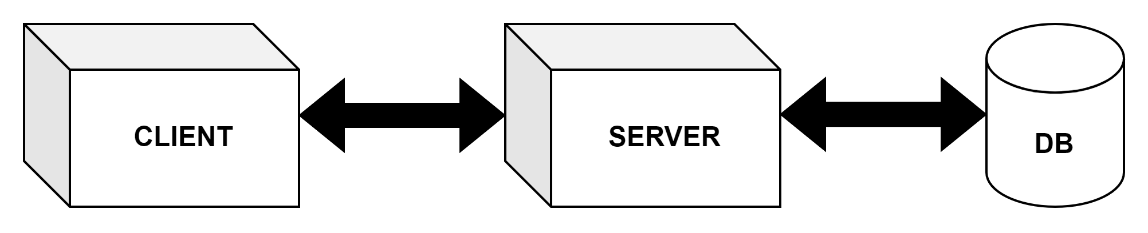


Рисунок - Клиент-серверная трехзвенная архитектура

### Проектирование архитектуры программного кода.

### Проектирование архитектуры взаимодействия между компонентами системы.

## Проектирование процессов системы.

## Описание алгоритма автоматического назначения заявок на специалиста.

## Проектирование базы данных

### Концептуальная модель данных

Концептуальное проектирование базы данных — это процесс разработки абстрактной модели данных, которая представляет собой высокоуровневое описание сущностей, атрибутов и связей между ними в информационной системе. Оно основывается на анализе требований и целей системы, и направлено на создание структуры данных, независимой от конкретных технологий и реализаций.

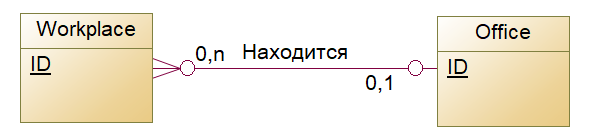
Основная цель концептуального проектирования базы данных заключается в создании ясной и понятной модели данных, которая будет служить основой для дальнейшего разработки физической базы данных и ее реализации с использованием конкретных технологий и СУБД [3].

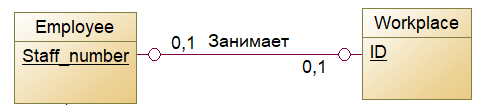
В выбранной предметной области можно выделить следующие сущности:

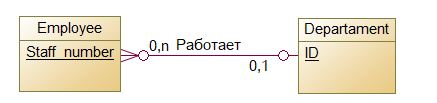
* Employee – хранит общую информацию о сотрудниках компании. Первичный ключ: Staff\_number.
* Client, Admin, Technical\_support\_specialist – отражают роли сотрудников компании в системе: клиент, администратор, специалист ТХП, соответственно. Первичный ключ: Staff\_number.
* Office – хранит информацию об офисах компании, в которых работают сотрудники. Первичный ключ: ID.
* Workplace – хранит информацию о рабочем месте сотрудников в офисе. Первичный ключ: ID.
* Departament – хранит информацию об отделах, в которых работают сотрудники. Первичный ключ: ID.
* Post – хранит информацию о должностях сотрудников. Первичный ключ: ID.
* Device – хранит информацию о технике. Первичный ключ: ID.
* Device\_type – хранит информацию о типах техники. Первичный ключ: ID.
* Device\_condition – хранит информацию о возможных состояниях техники. Первичный ключ: ID.
* Software – хранит информацию о программном обеспечении. Первичный ключ: ID.
* Sofware\_type – хранит информацию о типах программного обеспечения. Первичный ключ: ID.
* Application – хранит общую информацию о заявках. Первичный ключ: ID.
* Status – хранит информацию о возможных состояниях заявки. Первичный ключ: ID.
* Priority – хранит информацию о возможных значениях приоритета заявки. Первичный ключ: ID.
* Application\_device – хранит информацию о заявках, связанных с техникой. Первичный ключ: ID.
* Application\_software – хранит информацию о заявках, связанных с программным обеспечением. Первичный ключ: ID.
* Application\_device\_type – хранит информацию о типах заявок, связанных с техникой.
* Application\_software\_type – хранит информацию о типах заявок, связанных с программным обеспечением.
* Comment – хранит информацию о комментариях сотрудников, оставленных к заявке. Первичный ключ: ID.
* Attachment – хранит информацию о приложенных сотрудниками файлах, оставленных к заявке. Первичный ключ: ID.

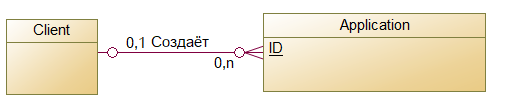
На основании, выделенных сущностей и требований, в системе можно выделить следующие связи:

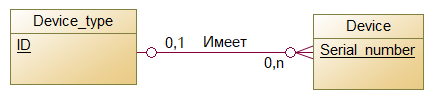
1. Рабочее место находится в офисе.
2. Сотрудник занимает рабочее место.
3. Сотрудник работает в отделе.
4. Сотрудник занимает пост.
5. Клиент, администратор, специалист ТХП являются ролями сотрудников в системе.
6. Сотрудник прикладывает вложения
7. Сотрудник оставляет комментарии.
8. Клиент создает заявки.
9. Специалист ТХП обрабатывает заявки.
10. Заявки бывают двух видов: связанные с техникой и связанные с программной обеспечением.
11. Заявка содержит комментарии.
12. Заявка содержит вложения.
13. Заявка имеет приоритет.
14. Заявка имеет статус.
15. Заявка, связанная с техникой, имеет тип.
16. Заявка, связанная с техникой, относится к технике.
17. Техника имеет тип.
18. Техника имеет состояние.
19. Заявка, связанная с программным обеспечением, имеет тип.
20. Заявка, связанная с программным обеспечением, относится к программному обеспечению
21. Программное обеспечение имеет тип.

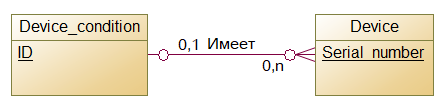


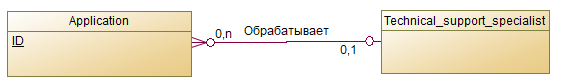


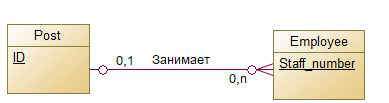


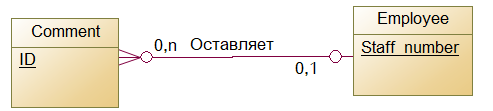


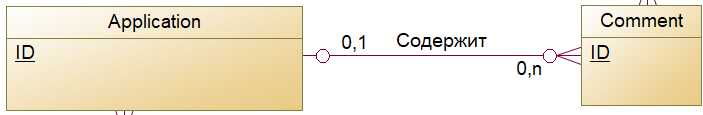


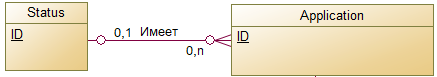


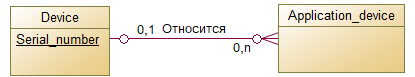


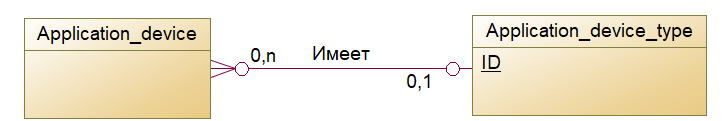


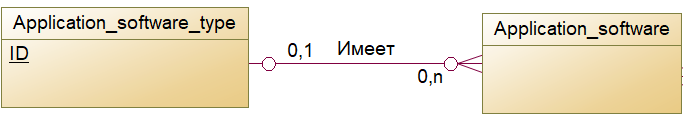


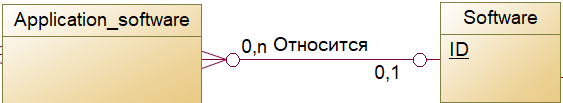


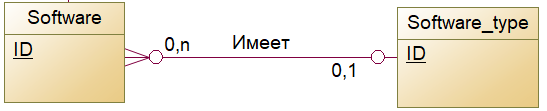


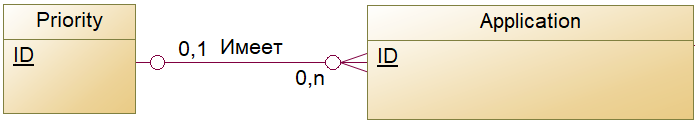


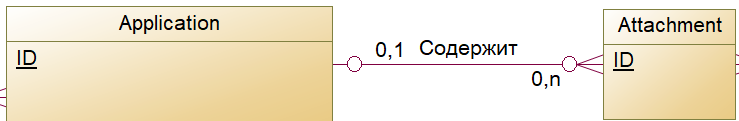


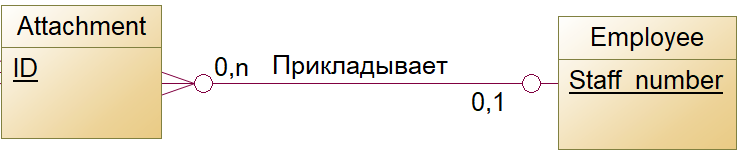


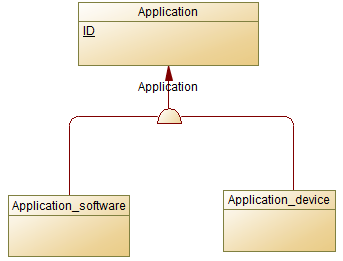


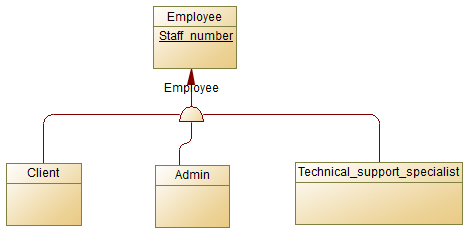


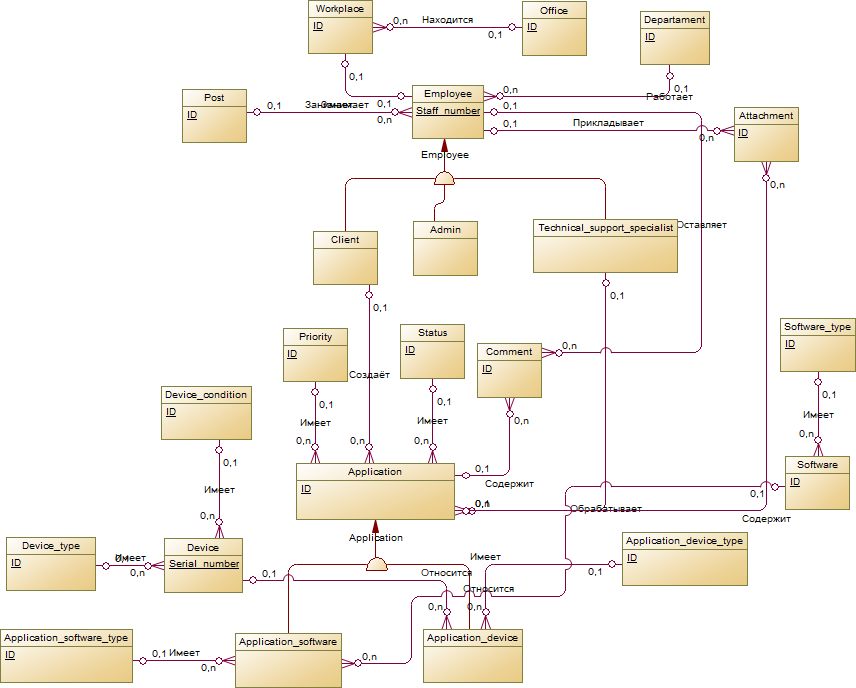












## Разработка серверной части.

## Разработка клиентской части.

## Интеграция клиентской и серверной части.

# Тестирование программного обеспечения

## Планирование тестирования

## Описание результатов тестирования

## Анализ результатов тестирования

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Черткова, Е. А. Программная инженерия. Визуальное моделирование программных систем. Учебник для среднего профессионального образования. Издательство Юрайт, 2023. – 147 с.
2. Роберт С. Мартин Чистая архитектура. Искусство разработки программного обеспечения. Издательство Питер, 2018. – 352 с.
3. Шитов, В. Н. Основы проектирования баз данных. Учебное пособие. Издательство ИНФРА-М, 2023. — 236 с.
4. Евдокимова Л.М., Корябкин В.В., Пылькин А.Н., Швечкова О.Г. Электронный документооборот и обеспечение безопасности стандартными средствами windows, Издательство КУРС, 2017 – 296c.
5. [Валентин Соломенчук](https://www.litres.ru/valentin-solomenchuk/), [Игорь Шишигин](https://www.litres.ru/igor-shishigin/), [Олег Колесниченко](https://www.litres.ru/oleg-kolesnichenko/) Аппаратные средства PC, БХВ-Петербург, 2009 – 782с
6. Таненбаум, Остин: Архитектура компьютера, Издательство [Питер](https://www.labirint.ru/pubhouse/104/), 2019-816с
7. Цай, Файзрахманова: Аппаратное обеспечение вычислительных систем. Учебное пособие, Издательство Фолиант, 2019 -320с
8. Евдокимова Л.М., Корябкин В.В., Пылькин А.Н., Швечкова О.Г. Электронный документооборот и обеспечение безопасности стандартными средствами windows, Издательство КУРС, 2017 – 296c.
9. Этан Браун Изучаем JavaScript. Руководство по созданию современных веб-сайтов. Издательство Альфа-книга, 2017 – 368 с.
10. Чиннатамби Кирупа Изучаем React. 2-е издание, изд-во Бомбора, 2019- 368с.
11. Девид Хэррон Node.js Разработка серверных веб-приложений на JavaScript. Издательство ДМК Пресс, 2016 -144 с.
12. Грофф Джеймс Р., Вайнберг Пол Н. SQL. Полное руководство. Издательство Вильямс, 2018 – 960с.
13. [Электронный ресурс]: metanit. URL: https://metanit.com/web/react/. (дата обращения 15.03.2020).

ПРИЛОЖЕНИЕ