Департамент образования Вологодской области

бюджетное профессиональное образовательное учреждение

Вологодской области «Череповецкий химико-технологический колледж»

Специальность 09.02.07

«Информационные системы и программирование»

**«Автоматизации процесса управления заявками»**

**Руководство системного программиста**

**КП.09.02.07.00.08**

|  |  |
| --- | --- |
| Руководитель: |  |
|  | (подпись, дата) |

|  |  |
| --- | --- |
| Проект разработал: |  |
|  | (подпись, дата) |

2024 г.

**Введение**

В условиях стремительного развития информационных технологий системный программист занимает центральное место в автоматизации процессов управления заявками. Это руководство предназначено для системных программистов, работающих в области автоматизации, и его цель — помочь разобраться в специфике работы, понять ключевые процессы и адаптироваться к быстро меняющейся среде.

Системные программисты играют важную роль в создании и поддержке эффективных решений, которые обеспечивают бесперебойное функционирование систем управления заявками. Успешная работа включает в себя все этапы — от регистрации и обработки заявок до формирования отчетов. Работа в этой области требует не только глубоких технических знаний, но и навыков коммуникации, способности работать в команде и готовности к изменениям.

Данное руководство охватывает основные аспекты, необходимые для успешного выполнения задач, взаимодействия с коллегами и личного развития. Оно состоит из нескольких разделов, каждый из которых подробно рассматривает ключевые аспекты работы системного программиста в области автоматизации управления заявками. В руководстве представлена информация о том, как организована работа в команде, какие инструменты и технологии используются, как проходит процесс разработки и тестирования, а также на какие аспекты личностного и профессионального роста стоит обратить внимание.

**1. Архитектура системы**

**1.1 Общая структура**

Автоматизация процесса управления заявками основана на модульной архитектуре, что является ключевым аспектом, обеспечивающим гибкость и масштабируемость системы. Каждый модуль представляет собой независимый компонент, отвечающий за определённую функциональность, что позволяет разрабатывать, обновлять и масштабировать его без влияния на другие части системы. Это способствует более эффективному управлению проектами и упрощает разработку, тестирование и внедрение новых функций.

Основные компоненты архитектуры автоматизации управления заявками:

1. Регистрация заявок: Модуль регистрации заявок отвечает за прием и обработку входящих заявок от пользователей. Он обеспечивает валидацию данных, управление статусами заявок и их маршрутизацию к соответствующим исполнителям. Разделение этой функциональности в отдельный модуль позволяет оптимизировать процесс обработки заявок и улучшить пользовательский опыт, а также упростить интеграцию с другими системами.

2. Управление статусами: Модуль управления статусами заявок отслеживает и обновляет текущие статусы каждой заявки на протяжении всего процесса её обработки. Он отвечает за автоматическое уведомление пользователей о изменениях статуса и обеспечивает прозрачность процесса.

3. Обработка данных: Модуль обработки данных отвечает за анализ и хранение информации о заявках, включая метаданные, историю изменений и комментарии. Он обеспечивает доступ к данным для формирования отчетов и аналитики. Разделение обработки данных на отдельный модуль позволяет использовать различные технологии хранения

4. Отчетность и аналитика: Модуль отчетности и аналитики занимается генерацией отчетов на основе данных о заявках и их обработке. Он предоставляет пользователям возможность получать информацию о производительности, времени обработки и других ключевых показателях. Отделение этой функциональности в отдельный модуль позволяет быстро разрабатывать новые отчеты и визуализации

**1.2 Технологии**

* Языки программирования: C#
* СУБД: SQL Server Management Studio

**2.1 Общая информация**

Автоматизация процесса управления заявками основывается на модульной архитектуре, что является ключевым аспектом, обеспечивающим гибкость и масштабируемость системы. Каждый модуль представляет собой независимый компонент, отвечающий за конкретную функциональность, что позволяет разрабатывать, обновлять и масштабировать его без влияния на другие части системы. Это способствует более эффективному управлению проектами и упрощает разработку, тестирование и внедрение новых функций.

Среди ключевых аспектов системы автоматизации управления заявками можно выделить следующие элементы:

* Модули для обработки заявок: Система включает в себя набор модулей, каждый из которых отвечает за определённые этапы обработки заявок, такие как регистрация, управление статусами, обработка данных и отчетность. Это позволяет командам сосредоточиться на улучшении конкретных функций, не затрагивая другие компоненты системы.

**2.2 Примеры использования API**

Система автоматизации управления заявками предоставляет множество возможностей для разработки приложений и интеграции с другими сервисами. Ниже приведены несколько примеров использования системы, которые подчеркивают её функциональность и гибкость:

* Регистрация новой заявки: для регистрации новой заявки можно использовать модуль регистрации, который принимает данные от пользователя и создает новую запись в системе. Этот процесс включает в себя валидацию данных и автоматическую маршрутизацию заявки к соответствующему исполнителю.
* Обновление статуса заявки: Используя модуль управления статусами, разработчики могут обновлять статус заявки в зависимости от её текущего состояния.
* Получение информации о заявках: Система позволяет получать информацию о заявках с помощью модуля обработки данных. Разработчики могут использовать методы для фильтрации и сортировки заявок по различным параметрам, таким как статус.
* Генерация отчетов: Модуль отчетности и аналитики предоставляет возможность генерировать отчеты на основе данных о заявках. Это может быть полезно для анализа производительности команды и выявления узких мест в процессе обработки.

**3. Интеграция и разработка**

**3.1 Интеграция с другими сервисами**

На данный момент интеграция с другими сервисами отсутствует. Это означает, что система автоматизации управления заявками функционирует как самостоятельное решение и не поддерживает взаимодействие с внешними платформами или API. В будущем планируется разработка механизмов интеграции, которые позволят расширить функциональность системы и улучшить взаимодействие с другими инструментами и сервисами.

**3.2 Разработка приложений**

Система автоматизации управления заявками предоставляет возможности для разработки приложений, однако в настоящее время поддержка интеграций и расширений ограничена. Это означает, что разработка приложений в рамках данной системы в основном сосредоточена на внутренней функциональности, без возможности взаимодействия с внешними сервисами.

* Простые приложения: Разработчики могут создавать приложения, которые обеспечивают базовые функции управления заявками, такие как создание, редактирование и просмотр заявок. Эти приложения могут быть полезны для небольших команд, которым необходимо организовать процесс обработки заявок без сложных интеграций.
* Сложные решения: в рамках системы можно разрабатывать более сложные решения, которые автоматизируют внутренние процессы, такие как маршрутизация заявок или уведомления о статусах. Однако, без интеграции с внешними сервисами, возможности для расширения функциональности остаются ограниченными.

**3.3 Использование SDK**

В настоящее время система автоматизации управления заявками не предоставляет набор инструментов разработки (SDK) для различных языков программирования. Это ограничивает возможности разработчиков в создании приложений и интеграции с другими системами.

**4. Разработка и тестирование**

Процесс разработки и тестирования является ключевым этапом в создании надежного программного обеспечения. Он включает в себя написание кода, его проверку на работоспособность, соответствие требованиям и выявление возможных ошибок. В следующих разделах подробно рассматриваются основные аспекты разработки и тестирования.

**4.1 Среда разработки**

Создание удобной и эффективной среды для разработки модуля автоматизации управления заявками является основой успешного проекта. Рекомендуется использовать локальные окружения для разработки и тестирования, что позволяет избежать конфликтов между различными проектами и зависимостями.

* Локальные окружения: Использование локальных окружений позволяет разработчикам тестировать изменения в коде в безопасной среде. Это способствует быстрому выявлению и устранению ошибок, а также позволяет контролировать версию приложения в процессе разработки. Популярные инструменты для упрощения работы с локальными окружениями включают Vagrant и Docker.
* Системы контроля версий: Использование систем контроля версий, таких как Git, является обязательным для управления изменениями в коде. Это позволяет отслеживать историю изменений, работать в команде и легко возвращаться к предыдущим версиям кода в случае необходимости.

**4.2 Тестирование кода**

Тестирование кода — это важный процесс, направленный на проверку качества и функциональности модуля автоматизации управления заявками. Оно включает в себя различные методы тестирования, которые помогают убедиться в том, что все функции работают корректно и соответствуют заданным требованиям.

* Юнит-тестирование: этот тип тестирования фокусируется на проверке отдельных функций и компонентов модуля, таких как создание, регистрация и обработка заявок. Использование юнит-тестов позволяет быстро выявлять ошибки на ранних этапах разработки, что снижает затраты на исправление и улучшает качество конечного продукта.
* Функциональное тестирование: этот тип тестирования направлен на проверку того, что все функции модуля работают так, как задумано. Например, необходимо протестировать, что при нажатии кнопки "Обработать заявку" появляется сообщение "Заявка обработана", а также что установка галочки о выполнении ремонта корректно обновляет статус заявки. Функциональное тестирование помогает убедиться, что пользовательский интерфейс и логика приложения соответствуют ожиданиям пользователей.
* Нагрузочное тестирование: важно проводить нагрузочное тестирование, чтобы убедиться, что модуль может справиться с ожидаемыми уровнями трафика и большим количеством одновременно обрабатываемых заявок. Это особенно актуально для систем, которые могут использоваться многими пользователями одновременно. Инструменты, такие как JMeter или LoadRunner, могут помочь в проведении нагрузочного тестирования и анализе производительности модуля.

**5. Безопасность**

Безопасность является неотъемлемой частью разработки программного обеспечения и должна уделяться особое внимание на всех этапах жизненного цикла приложения. Защита конфиденциальности и целостности данных, а также обеспечение устойчивости к различным угрозам — ключевые задачи для обеспечения безопасности модулей автоматизации управления заявками. Данный раздел подробно охватывает основные принципы и лучшие практики в области безопасности.

**5.1 Аутентификация и авторизация**

Хотя в данной системе не предусмотрены механизмы аутентификации и авторизации, важно понимать, что эти процессы играют ключевую роль в обеспечении безопасности приложений. Аутентификация и авторизация помогают гарантировать, что доступ к ресурсам имеют только уполномоченные пользователи, что особенно важно для защиты конфиденциальной информации и предотвращения несанкционированного доступа.

**5.2 Защита данных**

Защита данных в процессе управления заявками на ремонт является критически важной задачей, поскольку она включает в себя обработку конфиденциальной информации пользователей и данных о состоянии оборудования. Для обеспечения безопасности данных необходимо реализовать несколько ключевых мер.

* Шифрование: все данные, связанные с заявками, включая личные данные пользователей (имена, контактные телефоны), должны быть защищены с помощью шифрования как в состоянии покоя, так и в процессе передачи. Рекомендуется использовать современные алгоритмы шифрования, такие как AES (Advanced Encryption Standard), для защиты данных в базе данных, а также TLS (Transport Layer Security) для защиты данных при передаче по сети.
* Системы управления доступом: необходимо внедрить строгие механизмы управления доступом, чтобы гарантировать, что только авторизованные пользователи могут создавать, изменять или просматривать заявки.
* Логи и мониторинг: ведение журналов действий пользователей и системных событий поможет отслеживать доступ к данным и выявлять подозрительную активность. Регулярный анализ логов может помочь в обнаружении попыток несанкционированного доступа и других инцидентов безопасности.
* Регулярные аудиты безопасности: периодические аудиты безопасности системы, включая проверку кода и конфигураций, помогут выявить уязвимости и недостатки в защите данных. Использование инструментов статического и динамического анализа кода позволит автоматизировать процесс поиска уязвимостей и повысить общую безопасность системы.

**5.3 Логи и мониторинг**

Эффективное логирование и мониторинг процессов управления заявками на ремонт являются ключевыми аспектами обеспечения безопасности и надежности системы. Они позволяют отслеживать действия пользователей, выявлять аномалии и быстро реагировать на инциденты.

* Логирование событий: Система логирования настроена для фиксации всех критически важных событий, таких как создание, изменение и удаление заявок, а также действия по их обработке, включая нажатие кнопки "Заявка обработана". Используются структурированные логи, что значительно упрощает анализ и поиск информации, позволяя быстро находить необходимые данные в случае инцидента.

**Заключение**

Система управления заявками на ремонт представляет собой мощный инструмент, который значительно упрощает процесс обработки и отслеживания заявок. Она предлагает множество функций, позволяющих как пользователям, так и администраторам эффективно взаимодействовать с системой и оптимизировать рабочие процессы.

Знакомство с архитектурой системы, её функционалом и возможностями — это первый шаг к созданию эффективного решения для управления заявками. Понимание различных аспектов работы системы поможет вам максимально использовать её потенциал и сделать процесс управления более удобным и прозрачным. Основные аспекты, на которые стоит обратить внимание, включают:

* Изучение функционала: система управления заявками предоставляет доступ к множеству функций, таких как создание, редактирование и удаление заявок, а также отслеживание их статусов и обработка. Понимание структуры и принципов работы системы поможет вам эффективно использовать её возможности для улучшения взаимодействия с пользователями.
* Инструменты автоматизации: использование встроенных инструментов для автоматизации процессов, таких как уведомления о статусах заявок и отчеты о выполненных работах, может значительно ускорить обработку заявок. Эти инструменты позволяют сосредоточиться на более важных задачах и повышают общую продуктивность работы.
* Поддержка и обучение: работа с системой управления заявками не ограничивается только её функционалом. Существует множество ресурсов, таких как документация, обучающие материалы и вебинары, которые помогут вам лучше понять систему и её возможности. Участие в обучающих мероприятиях и форумах — отличные способы получить информацию и обменяться опытом с другими пользователями.

**Приложение**

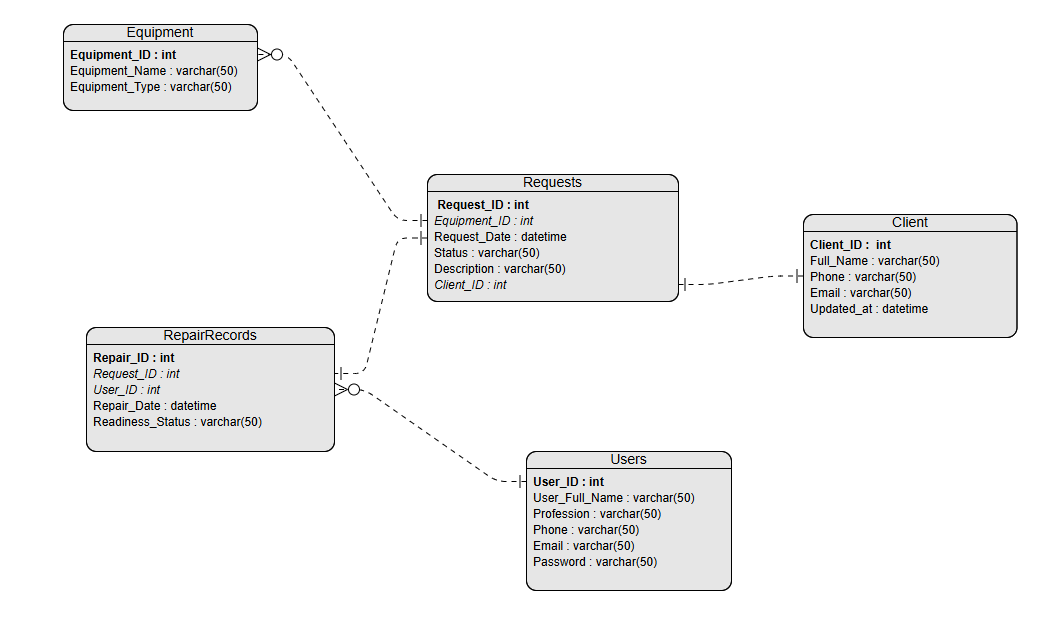
****

Рис.1 ER – модель



Рис 2. Блок – схема обработки заявки