**Введение**

В условиях быстрого прогресса информационных технологий системный программист занимает ключевую позицию в автоматизации процессов управления заявками. Это руководство создано для системных программистов, работающих в сфере автоматизации, и направлено на то, чтобы помочь им разобраться в особенностях своей профессии, понять важнейшие процессы и адаптироваться к динамично меняющейся среде.

Данное руководство охватывает ключевые аспекты, необходимые для успешного выполнения задач, взаимодействия с коллегами и личностного развития. Оно состоит из нескольких разделов, каждый из которых подробно рассматривает важные моменты работы системного программиста в области автоматизации управления заявками. В руководстве представлена информация о том, как организована командная работа, какие инструменты и технологии применяются, 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 важным аспектом, обеспечивающим гибкость и масштабируемость системы. Каждый модуль представляет собой самостоятельный 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 процесс разработки, тестирования и внедрения новых функций.

Основные компоненты архитектуры автоматизации управления заявками:

1. Регистрация заявок: Этот модуль отвечает за прием и обработку входящих заявок от пользователей. Он обеспечивает валидацию данных, управление статусами заявок и их маршрутизацию к соответствующим исполнителям. Выделение этой функциональности в отдельный модуль позволяет оптимизировать процесс обработки заявок, улучшить пользовательский опыт и упростить интеграцию с другими системами.
2. Управление статусами: Модуль управления статусами отслеживает и обновляет текущие статусы каждой заявки на протяжении всего процесса её обработки. Он также отвечает за автоматическое уведомление пользователей об изменениях статуса, обеспечивая прозрачность процесса.
3. Обработка данных: Этот модуль отвечает за анализ и хранение информации о заявках, включая метаданные, историю изменений и комментарии. Он обеспечивает доступ к данным для формирования отчетов и аналитических материалов. Выделение обработки данных в 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 строится на модульной архитектуре, что является важным аспектом, обеспечивающим гибкость и масштабируемость системы. Каждый модуль функционирует как самостоятельный компонент, отвечающий за определённые задачи, что позволяет разрабатывать, обновлять и масштабировать его без воздействия на другие элементы системы. Это способствует более эффективному управлению проектами и упрощает процессы разработки, тестирования и внедрения новых функций.

К числу ключевых элементов системы автоматизации управления заявками относятся:

* Модули для обработки заявок: Система включает 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 предлагает множество возможностей для разработки приложений и интеграции с другими сервисами. Вот несколько примеров использования системы, которые подчеркивают её функциональность и гибкость:

* Регистрация новой заявки: Модуль регистрации позволяет принимать данные от пользователя и создавать новую запись в системе. Этот процесс включает в себя валидацию данных и автоматическую маршрутизацию заявки к соответствующему исполнителю, что обеспечивает оперативное начало обработки.
* Обновление статуса заявки: С использованием модуля управления статусами разработчики могут обновлять статус заявки в зависимости от её текущего состояния. Это позволяет поддерживать актуальность информации и информировать пользователей о ходе обработки их заявок.
* Получение информации о заявках: Система предоставляет возможность получать информацию о заявках через модуль обработки данных. Разработчики могут использовать методы для фильтрации и сортировки заявок по различным параметрам, таким как статус, что упрощает анализ и управление заявками.
* Генерация отчетов: Модуль отчетности и аналитики позволяет генерировать отчеты на основе данных о заявках. Это полезно для анализа производительности команды, выявления узких мест в процессе обработки и принятия обоснованных решений для улучшения работы системы.

**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, но 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, обеспечивающие 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 возможно 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. Это может затруднить адаптацию системы под специфические нужды бизнеса и интеграцию с другими инструментами и платформами.

**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, которые обеспечивают изоляцию и воспроизводимость окружений.
* 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. Системы контроля версий также способствуют лучшему сотрудничеству между разработчиками и обеспечивают прозрачность в процессе разработки

**4.2 Тестирование кода**

Тестирование кода — ключевой процесс для оценки качества и функциональности модуля автоматизации управления заявками. Оно включает несколько методов, обеспечивающих корректность работы функций.

* Юнит-тестирование: Проверяет отдельные функции и компоненты модуля, такие как создание и обработка заявок. Это позволяет быстро выявлять ошибки на ранних этапах разработки, снижая затраты на исправление и повышая качество продукта.
* Функциональное тестирование: Оценивает, работают ли все функции модуля согласно ожиданиям. Например, проверяется, что при нажатии кнопки "Обработать заявку" появляется соответствующее сообщение, а статус заявки обновляется корректно. Это тестирование гарантирует соответствие пользовательского интерфейса и логики приложения требованиям пользователей.
* Нагрузочное тестирование: Необходимо для проверки способности модуля обрабатывать ожидаемые уровни трафика и большое количество заявок одновременно. Инструменты, такие как JMeter или LoadRunner, помогают проводить нагрузочное тестирование и анализировать производительность.

**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**

Несмотря на отсутствие механизмов аутентификации и авторизации в данной системе, эти процессы критически важны для безопасности приложений. Они обеспечивают доступ к ресурсам только уполномоченным пользователям, что особенно необходимо для защиты конфиденциальной информации и предотвращения несанкционированного доступа.

**5.2 Защита данных**

Защита данных при управлении заявками на ремонт является критически важной, так как включает 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, должны быть защищены шифрованием как в состоянии покоя, так и при передаче. Рекомендуется использовать современные алгоритмы, такие как AES для хранения и TLS для передачи данных.
* Системы управления доступом: Необходимо внедрить строгие механизмы управления доступом, чтобы только авторизованные пользователи могли 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.
* Регулярные аудиты безопасности: Периодические аудиты безопасности, включая проверку кода и конфигураций, помогут выявить уязвимости. Использование инструментов анализа кода позволит автоматизировать поиск 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 должна быть 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". Использование структурированных логов значительно упрощает анализ и поиск информации, позволяя быстро находить необходимые данные в случае инцидента.

**Заключение**

Система управления заявками на ремонт является мощным инструментом, который значительно упрощает процесс обработки и отслеживания заявок. Она предлагает широкий спектр функций, позволяющих пользователям и администраторам эффективно взаимодействовать с системой и оптимизировать рабочие процессы.

Первый шаг к созданию эффективного решения для управления заявками — это знакомство с архитектурой системы, её функционалом и возможностями. Понимание различных аспектов работы системы поможет вам в полной мере использовать её потенциал и сделать процесс управления более удобным и прозрачным. Основные моменты, на которые следует 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. Осознание структуры и принципов работы системы поможет вам эффективно использовать её возможности для улучшения взаимодействия с пользователями.
* Инструменты автоматизации: Применение 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.
* Поддержка и обучение: Работа с системой управления заявками выходит за рамки её функционала. 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 — отличный способ получить информацию и обменяться опытом с другими пользователями.

**Приложение**

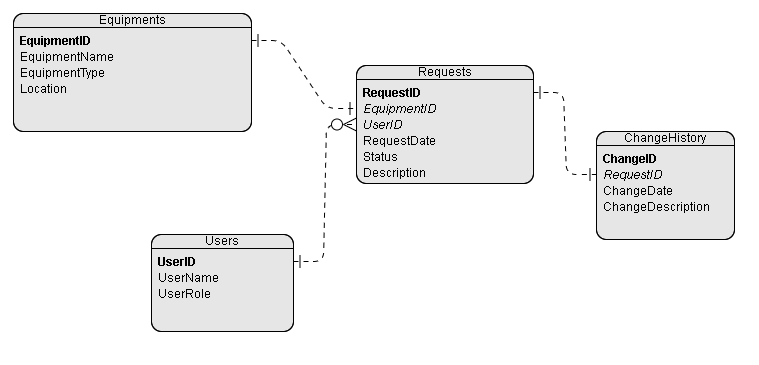
****

Рис.1 ER – модель

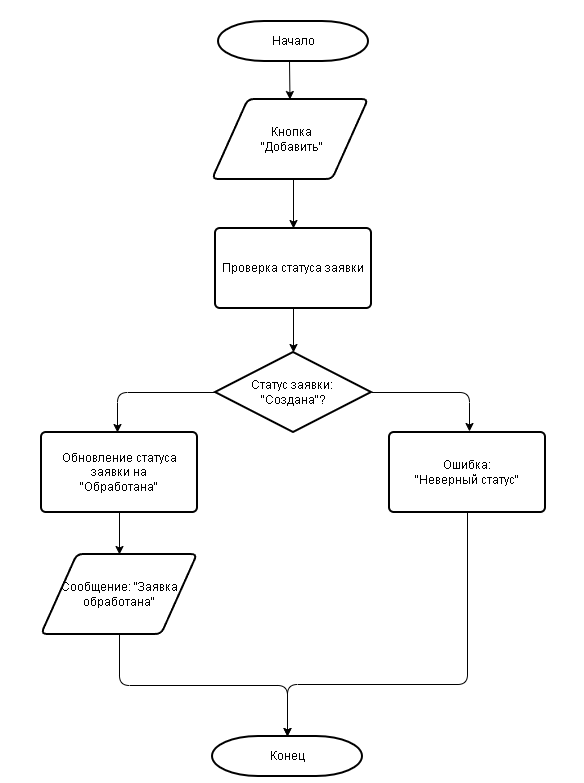


Рис 2. Блок – схема обработки заявки