Оглавление

[Введение 3](#_Toc357961318)

[1. Назначение и область применения 6](#_Toc357961319)

[2. Постановка задачи 7](#_Toc357961320)

[3. Техническое задание на разработку программного модуля. 8](#_Toc357961321)

[3.1 Введение 8](#_Toc357961322)

[3.2 Основания для разработки 8](#_Toc357961323)

[3.3 Назначение разработки 9](#_Toc357961324)

[3.4 Требования к программному продукту 9](#_Toc357961325)

[3.4.1 Требования к функциональным характеристикам 9](#_Toc357961326)

[3.4.2 Требования к надёжности 10](#_Toc357961327)

[3.4.3 Требования к эксплуатации 10](#_Toc357961328)

[3.4.4 Требование к составу и параметрам технических средств 10](#_Toc357961329)

[3.4.5 Требование к информационной и программной совместимости 11](#_Toc357961330)

[3.4.6 Требования к маркировке и упаковке. 11](#_Toc357961331)

[3.4.7 Требования к транспортированию и хранению. 11](#_Toc357961332)

[3.5 Программная документация 11](#_Toc357961333)

[3.6 Стадии разработки 12](#_Toc357961334)

[3.7 Порядок контроля и приёмки 13](#_Toc357961335)

[4. Специальная часть 14](#_Toc357961336)

[4.1 Обзор предметной области (АСОТ) 14](#_Toc357961337)

[4.2 Теория качества информации 18](#_Toc357961338)

[4.3 Обзор существующих программных средств 22](#_Toc357961339)

[4.3.1 Система отслеживания ошибок 22](#_Toc357961340)

[4.3.2 Система электронного документооборота 23](#_Toc357961341)

[4.4 Математическое и информационное обеспечение 25](#_Toc357961342)

[4.4.1 Теория конечных автоматов 25](#_Toc357961343)

[4.4.2 Метаязык 27](#_Toc357961344)

[4.4.3 Система управления базами данных 28](#_Toc357961345)

[4.5 Структура модуля 31](#_Toc357961346)

[4.5.1 Основные понятия 31](#_Toc357961347)

[4.5.2. Концепция модуля. 33](#_Toc357961348)

[4.5.3 Описание правил на метаязыке 35](#_Toc357961349)

[4.5.4. Структура служебной базы данных. 39](#_Toc357961350)

[5. Конструкторско-технологическая часть 43](#_Toc357961351)

[5.1 Технология программирования 43](#_Toc357961352)

[5.1.1 Экстремальное программирование 44](#_Toc357961353)

[5.1.2 Методология SCRUM 47](#_Toc357961354)

[5.1.3 Методология Kanban 50](#_Toc357961355)

[5.2 Выбор используемого программного обеспечения. 53](#_Toc357961356)

[5.2.1 MySQL-сервер 53](#_Toc357961357)

[5.2.2 MySQL Workbench 55](#_Toc357961358)

[5.2.3 SQLyog Community 56](#_Toc357961359)

[5.2.4 Borland Delphi 7 58](#_Toc357961360)

[5.2.5 WinCVS 60](#_Toc357961361)

[6. Охрана труда 62](#_Toc357961362)

[6.1 Основные положения 62](#_Toc357961363)

[6.2 Безопасность труда при работе с персональным компьютером 63](#_Toc357961364)

[6.3 Микроклимат в рабочей зоне 64](#_Toc357961365)

[6.4 Расчет защитного зануления 65](#_Toc357961366)

[7. Экологическая часть 69](#_Toc357961367)

[7.1 Введение 69](#_Toc357961368)

[7.2 Разработка эргономичного интерфейса 69](#_Toc357961369)

[7.3 Вывод: 72](#_Toc357961370)

[8. Решение задачи на ЭВМ 74](#_Toc357961371)

[8.1 Структура классов модуля 74](#_Toc357961372)

[8.2 Графический интерфейс пользователя 78](#_Toc357961373)

[8.2.1 Редактирование основных сущностей схем маршрутизации запросов 78](#_Toc357961374)

[8.2.2 Настройка схемы маршрутизации запросов 80](#_Toc357961375)

[8.2.3 Настройка бизнес-процессов 82](#_Toc357961376)

[8.3 Апробация модуля 82](#_Toc357961377)

[Заключение 85](#_Toc357961378)

[Список используемой литературы 86](#_Toc357961379)

# Введение

Автоматизация технологических процессов в любой сфере деятельности невозможна без работы с информацией об объектах ее инфраструктуры. Поэтому качество автоматизации напрямую зависит от качества и структурированности информации. В автоматизированных системах с распределенным вводом информации конечный пользователь системы должен иметь возможность оперативно редактировать информацию, однако бесконтрольное внесение изменений, как правило, приводит к снижению качества данных, дублированию и потере их актуальности.

Особенно актуально решение проблемы контроля качества данных в корпоративных системах ведения нормативно-справочной информации, данные которых используются всеми смежными автоматизированными системами.

Нормативно-справочная информация (НСИ) – это условно-постоянная составляющая общей корпоративной информации. НСИ – это ядро единого информационного пространства организации. В состав НСИ входит набор справочников, словарей, классификаторов, стандартов, регламентов, используемых в деятельности предприятия. Управление нормативно-справочной информацией происходит с помощью системы управления НСИ или MDM-системы (Master Data Management).

Master Data Management – это набор процессов и инструментов для определения и управления нормативно-справочной информацией, её сбора из всех информационных систем, обеспечения её качества, поиска потенциальных дублей, консолидации, распространения по всей организации в рамках единой политики. Главной задачей MDM-системы является обеспечение максимально полной, достоверной, уникальной информации, используемой всеми автоматизированными системами и подразделениями компании. MDM-системы активно используются в распределённых информационных системах [32].

Распределённые информационные системы – это информационные системы, компоненты которой распределены по нескольким автоматизированным системам. По архитектуре распределённые информационные системы делятся на файл-серверные и клиент-серверные [21].

В файл-серверной архитектуре клиентское приложение и система управления базой данных находится на рабочей станции, в то время как базы данных располагаются на сервере. Клиент-серверная архитектура заключается в том, что на рабочих станциях (компьютерах) расположены клиентские приложения, а система управления базами данных и сами базы данных находятся на сервере. Клиент-серверные информационные системы можно подразделить на двухзвенные и многозвенные [19].

В двухзвенных информационных системах существует только два типа звеньев: рабочая станция, на которой располагается клиентское приложение и сервер, на котором хранятся СУБД и базы данных. В такой архитектуре приложения напрямую обращаются к СУБД. В многозвенных ИС существуют ещё промежуточные звенья – серверы приложений. Клиентские приложения в этом случае не обращаются напрямую к СУБД, а взаимодействуют только с промежуточными звеньями. При реализации такой архитектуры данные вводятся и редактируются децентрализовано, из-за чего возникает проблема репликации данных [16] – синхронизации нескольких копий содержимого баз данных. Из-за этого появляется главная проблема MDM-систем – контроль качества информации.

Под контролем качества информации подразумеваются меры, направленные на снижение вероятности ошибок при модификации информации, гарантируя тем самым работу с достоверной информацией [14].

Для решения проблемы контроля качества данных зачастую применяется механизм подачи запросов на изменение информации. Суть данного механизма заключается в том, что конечный пользователь вносит необходимые изменения не напрямую в хранилище данных, а создаёт некоторый пакет изменённой информации, который передаётся для контроля сделанных им изменений группе пользователей, ответственных за проверку данных, их подтверждение или отклонение. Разумеется, в зависимости от существующих технологических / регламентных процессов конкретного предприятия сама цепочка групп пользователей, работающих с пакетом измененной информации (далее – запросом) может быть длиннее, а сами пользователи помимо функции контроля могут выполнять и функцию дополнения информации.

Данная дипломная работа посвящена исследованию проблемы контроля качества данных в системах с распределенным вводом информации, а также созданию программного обеспечения, обеспечивающего настройку правил передачи запросов на изменение информации от одной группы пользователей другой и реализующего полный цикл прохождения этих запросов, начиная от их создания, и заканчивая их обработкой или отклонением.

Практическим примером, подчеркивающим актуальность создания и внедрение такого программного обеспечения, является автоматизированная система создания и ведения адресных справочников и справочников объектов теплоснабжения (далее – АСОТ), разработанная компанией «Маппл Групп», на которой мною была пройдена производственная практика, в интересах ОАО «Московская объединенная энергетическая компания» (ОАО «МОЭК»). АСОТ - корпоративная географическая информационная система ведения НСИ ОАО «МОЭК».

Географическая информационная система (ГИС) – это система, обеспечивающая сбор, хранение, обработку, отображение и распространение данных, а также получение на их основе новой информации о пространственно-ориентированных явлениях [15]. В более узком смысле географической информационной системой является программный продукт, позволяющий пользователям работать не только с графической информацией, но и редактировать другие атрибуты объектов, представленных визуально на цифровой карте, такие как адрес, тип здания, наименование улицы и так далее [4]. ГИС преимущественно хранят информацию в системах управления базами данных и обладают мощными инструментами работы с данными, например, редакторами растровой и векторной графики [29].

Исследование задачи обеспечения качества информации, разработка программного модуля маршрутизации запросов на изменение данных будет произведено в рамках внедрения в автоматизированную систему АСОТ.

# 1. Назначение и область применения

Главной задачей разрабатываемого модуля маршрутизации запросов на изменение данных является контроль качества вносимой в систему информации за счет:

* Маршрутизации запросов пользователям в зависимости от роли пользователя в системе и параметров запроса;
* Динамического изменения маршрута запроса при выполнении определённых условий;
* Проверки корректности изменений и запрет обработки и внесений изменений в базу данных в случае ошибки;
* Автоматического отклонения запросов по времени.

Областью применения разрабатываемого модуля являются распределённые информационные системы и информационные системы с повышенным контролем качества информации. Разработанный программный модуль позволит снизить риски ошибок модификации информации при децентрализованном вводе и редактировании данных. Автоматизация отклонения и обработки запросов на изменение данных, а также динамическое изменение маршрута запроса в зависимости от его параметров позволят сократить время от создания запроса до фактической записи в базу данных, не влияя на качество информации, то есть, не повышая риска внесения ошибочных данных, а также гибко настроить схему маршрутизации для каждого автоматизируемого технологического процесса предприятия.

# 2. Постановка задачи

В рамках выполнения дипломной работы требуется выполнить следующие задачи:

* Рассмотреть существующие аналоги разрабатываемого программного обеспечения, их плюсы и минусы;
* Разработать программный модуль, реализующий маршрутизацию запросов на изменение данных;
* Спроектировать и запрограммировать пользовательский интерфейс, позволяющий настроить маршруты запросоы и задать условия, при которых запрос будет пропускать или останавливаться в том или ином состоянии;
* Разработать метаязык для описания условий;
* Внедрить разработанный модуль в программный комплекс АСОТ:
* Проверить работоспособность разработанного модуля на тестовых примерах:
* Сформировать отчёт о проделанной работе.

# 3. Техническое задание на разработку программного модуля.

## 3.1 Введение

**Наименование**

Программный модуль маршрутизации запросов на изменение данных в автоматизированной системе АСОТ.

**Область применения**

В настоящее время широко распространены предприятия с распределёнными информационными системами. Работа с данными на таких предприятиях осуществляется децентрализовано, что ведёт за собой необходимость использования механизмов повышающий контроль качества информации. Одним из таких решений является внедрение системы автоматизированной маршрутизации запросов на изменение или внесение данных. Такая система позволит проверять информацию на нескольких уровнях, что обеспечит снижение риска внесения ошибочных данных. Этот программный модуль получит широкое применение на предприятиях с распределёнными информационными системами, а также на предприятиях, где особо важен повышенный контроль качества информации. Автоматическое распределение запросов по системе, а также динамическое изменение маршрута, в зависимости от атрибутов запроса не повлияют на время фактического внесения изменений в базу данных, но сохранят надёжность и достоверность информации.

## 3.2 Основания для разработки

Документ, на основании которого ведется разработка:

Форма Запроса Изменений к договору № 1/11 на оказание услуг и выполнение работ по технической поддержке автоматизированной системы ведения адресных справочников и справочников объектов теплоснабжения ЕКС НСИ ОАО «МОЭК» и использованием технологий ГИС ОАО «МОЭК».

Документ утверждён ОАО «МОЭК» 01.09.2012.

Наименование и (или) условное обозначение темы разработки:

Разработка компонента распределения объектов мультизапроса по ролям в зависимости от типа объекта, значений его атрибутов и операции над ним в АС АСОТ ОАО «МОЭК».

## 3.3 Назначение разработки

Программный модуль, разрабатываемый для системы АСОТ должен выполнять следующие задачи:

* Правильная адресация запросов пользователям в зависимости от роли пользователя в системе, а также содержимого запроса;
* Динамическое изменение маршрута, при выполнении определённых условий;
* Проверка валидности изменений и запрет обработки и внесений изменений в базу данных в случае ошибки;
* Автоматическое отклонение запросов по времени;
* Возможность настройки маршрута и условий, при которых маршрут будет динамически изменяться, с помощью пользовательского интерфейса.

## 3.4 Требования к программному продукту

### 3.4.1 Требования к функциональным характеристикам

Для осуществления правильной адресации запросов пользователям, следует спроектировать и разработать программное ядро, где будут реализованы все сущности, с которыми работает система маршрутизации, связи этих сущностей друг с другом, а также необходимые для взаимодействия с этими сущностями функции.

Динамическое изменение настроенного маршрута подразумевает под собой автоматическое прохождение определённых состояний запросом при выполнении конкретного условия в зависимости от атрибута этого запроса. Для реализации этой функции необходимо разработать и внедрить метаязык, на котором будут описываться условия пропуска состояний, а также функции для преобразования описанного условия в логическое выражение и дальнейшее вычисление его значения. Входными данными этой функции будут являться значения атрибутов запроса, значения атрибутов объекта записанные в базе данных, а также условие, описанное на метаязыке. На выходе функции – булевское значение (true или false) однозначно определяющее необходимость пропуска или остановки в текущем состоянии маршрута.

Для предотвращения внесения в базу данных ошибочной информации следует реализовать функцию автоматической проверки значений атрибутов запроса на предмет логических ошибок.

Для автоматического отклонения запросов по времени, заданному администратором системы, необходимо реализовать сервис, который постоянно через определённый интервал времени будет проверять длительность простоя каждого запроса, а по истечении указанного времени будет автоматически отклонять такие запросы. Все данные о времени берутся из актуальной базы данных.

Пользовательский интерфейс должен позволять полностью настроить атрибуты всех сущностей системы маршрутизации, а также проверять и предотвращать неправомерные действия по изменению уже существующих схем маршрутизации. Интерфейс должен обладать визуальной интерпретацией настраиваемой схемы маршрутизации и отражать все изменения, вносимые пользователем, в режиме реального времени.

### 3.4.2 Требования к надёжности

В случае пропуска ошибочной информации через всю схему маршрутизации и записи таковой в базу данных, осуществлять откат на основе данных о последних изменениях объектов.

### 3.4.3 Требования к эксплуатации

Стандартные требования эксплуатации программного продукта. Администратор системы, выполняющий настройку схемы маршрутизации, должен уметь описывать условия остановки в состояниях на метаязыке, разработанном специально для этой цели.

### 3.4.4 Требование к составу и параметрам технических средств

Средства вычислительной техники должны иметь следующий минимальный состав:

1. ПК с Intel-совместимым процессором с тактовой частотой не ниже 1,7 ГГц;
2. объем ОЗУ – 256 МБ;
3. объём свободной памяти на жёстком диске – 500 МБ;

В состав программного обеспечения должны входить:

1. ОС семейства Windows (Windows 2000 и более поздних версий);
2. СУБД MySQL 5.0 и выше;
3. среда Borland Delphi 5.0 и выше.

### 3.4.5 Требование к информационной и программной совместимости

Разрабатываемый программный модуль является частью системы АСОТ, поэтому требования к информационной и программной совместимости будут аналогичными. Программа должна работать в операционной системе Windows XP или Windows 7, поэтому она должна иметь совместимость со стандартной библиотекой функций Win API. Все данные находятся на базах данных формата MySQL, поэтому в системе должен быть реализован интерфейс, позволяющий обращаться к СУБД. Программный модуль рекомендуется разрабатывать в среде Borland Delphi 7, на языке Delphi поскольку система, в которую модуль будет внедрён, была разработана именно на этом программном обеспечении. Это облегчит интеграцию разрабатываемого модуля.

### 3.4.6 Требования к маркировке и упаковке.

Не предъявляются.

### 3.4.7 Требования к транспортированию и хранению.

Не предъявляются.

## 3.5 Программная документация

Программная документация для пользователя должна содержать в себе описание функций работы программного модуля, объяснять назначение управляющих элементов интерфейса и содержать примеры выполнения стандартных операций при работе с программой. Также документация должна содержать раздел, обучающий пользователя метаязыку для настройки условий остановки запросов в состояниях. Документация программного интерфейса должна содержать описание классов и функций программного модуля для возможности расширить его функциональность или использования модуля в других проектах.

## 3.6 Стадии разработки

|  |  |  |  |
| --- | --- | --- | --- |
| **Этап** | **Название** | **Сроки (неделя)** | **Исполнитель** |
| 1 | Проектирование структуры классов и функций ядра программного модуля | 1-2 | Тырин А.А. |
| 2 | Проектирование структуры классов и функций механизма создания и обработки условий остановки запросов в состоянии | 2-3 | Тырин А.А. |
| 3 | Программирование ядра программного модуля | 4-6 | Тырин А.А. |
| 4 | Программирование механизма создания и обработки условий остановки запросов в состоянии | 7-9 | Тырин А.А. |
| 5 | Настройка связей между ядром и механизмом создания и обработки условий остановки запросов в состоянии | 10 | Тырин А.А. |
| 6 | Проектирование и разработка программного интерфейса, привязка к ядру модуля | 11-13 | Тырин А.А. |
| 7 | Проверка программного модуля на работоспособность на тестовых данных и исправление недочётов | 14 | Тырин А.А. |
| 8 | Создание программной документации и формирование отчёта о проделанной работе | 15 | Тырин А.А. |

## 3.7 Порядок контроля и приёмки

Внедрение и апробация разработанного программного модуля в системе АСОТ. Проверка работы системы со специально подготовленными запросами для тестирования всех функций модуля:

* Заведомо ошибочные запросы для тестирования механизма отклонения неверных запросов;
* Запросы с атрибутами, удовлетворяющие условиям остановки в состоянии;
* Проверка автоматического отклонения запроса, не подвергавшегося редактированию в течение количества дней, указанного администратором.

# 4. Специальная часть

## 4.1 Обзор предметной области (АСОТ)

АСОТ – это автоматизированная система создания и ведения адресных справочников и справочников объектов теплоснабжения, разработанная на предприятии «Маппл Групп» в интересах ОАО «Московская объединенная энергетическая компания» (ОАО «МОЭК»). АСОТ является корпоративной географической информационной системой ведения НСИ ОАО «МОЭК».

В состав АСОТ входят следующие компоненты:

* импортирования адресных данных внешних источников (отмечено цифрой 1 на рисунке 4.1);
* сопоставления адресной информации единой государственной картографической основы (ЕГКО), общемосковских классификаторов (ОМК), адресного реестра бюро технической инвентаризации (АР БТИ) (отмечено цифрой 2 на рисунке 4.1);
* периодического обновления адресных справочников по новым версиям данных из внешних источников (отмечено цифрой 3 на рисунке 4.1);
* ведения справочников (отмечено цифрой 4 на рисунке 4.1);
* сопоставления списков объектов теплоснабжения, используемых в различных адресных справочниках (АС) ОАО “МОЭК” (отмечено цифрой 5 на рисунке 4.1);
* администрирования и разграничения прав пользователей (отмечено цифрой 6 на рисунке 4.1);
* настройки схем маршрутизации запросов на редактирование справочников (отмечено цифрой 7 на рисунке 4.1);
* взаимодействия с интеграционной шиной (отмечено цифрой 8 на рисунке 4.1).

Графическое представление структуры и взаимодействия компонентов АСОТ изображено на рисунке 4.1.

Описание: архитектураАСОТv2

*Рис. 4.1 Структурная схема модуля АСОТ*

АСОТ обеспечивает выполнение следующих функций:

* импортирование адресных данных внешних источников (ЕГКО, АР БТИ, ОМК);
* сопоставление адресной информации ЕГКО, ОМК, АР БТИ в автоматизированном режиме;
* формирование адресных справочников;
* периодическое обновление адресных справочников по новым версиям данных из внешних источников;
* сопоставление списков объектов теплоснабжения, используемых в различных АС ОАО “МОЭК”;
* ведение и поддержка справочников в актуальном состоянии;
* настройка схем маршрутизации запросов на редактирование справочников для поддержки принятых в ОАО «МОЭК» технологических процессов по ведению информации;
* организация информационного обмена с другими АС ОАО «МОЭК» посредством интеграционной шины ОАО «МОЭК»

Главным образом нас интересует функция настройки схем маршрутизации запросов на редактирование справочников, которая реализована в компоненте настройки схем маршрутизации запросов на редактирование справочников, так как именно эта часть АСОТ будет переработана в рамках данной дипломной работы.

На текущий момент компонент настройки схем маршрутизации запросов на редактирование справочников решает следующие задачи:

* создание и редактирование схем маршрутизации запросов на редактирование справочников АСОТ, посредством задания детерминированного конечного автомата состояний и переходов запросов;
* задание и редактирование ролей (например, заявитель, технолог и т.д.) пользователей в схемах маршрутизации запросов;
* присвоение пользователям тех или иных ролей в схемах маршрутизации запросов.

Источником входных данных для настройки схем маршрутизации служит информация об обязанностях сотрудников подразделений ОАО «МОЭК» по работе с АСОТ в соответствии с установленным регламентом ведения нормативно-справочной информации (НСИ) ОАО «МОЭК» в части справочников АСОТ.

В процессе эксплуатации АСОТ были выявлены следующие проблемы и недостатки компонента настройки схем маршрутизации запросов на редактирование справочников:

* отсутствие возможности пропускать те или иные состояния запроса в схеме в зависимости от типов операций с редактируемым объектом (создание, редактирование, удаление) и значений его атрибутов;
* необходимость многократной настройки однотипных схем маршрутизации из-за того, что для каждого справочника должна быть настроена собственная схема;
* Невозможно поддерживать мультизапрос – редактирование нескольких объектов разных справочников в рамках единого запроса;
* отсутствие визуальной интерпретации настраиваемой схемы.

Функция пропуска состояния позволяет направить запрос в то или иное состояние схемы в зависимости от его атрибутов. Таким образом, мы получаем динамически изменяющийся маршрут в рамках одной схемы. Такая гибкость схемы маршрутизации позволит настраивать одну схему для нескольких разнотипных объектов, что также сократит время работы пользователей системы. Также актуальность функции пропуска состояния возникает в том случае, когда нет необходимости в участии пользователя определённой роли при обработке запроса на изменение данных. В этом случае скорость обработки данных существенно увеличивается.

Различие схем маршрутизации у разных справочников не позволяет реализовать механизм обработки мультизапросов. Мультизапрос – это запрос, состоящий из нескольких объектов, принадлежащих разным справочникам, логически связанных между собой. Мультизапросы позволяют вести справочники объектов теплоснабжения, редактируя несколько разнотипных объектов, объединённых в одну логическую группу. Например, при редактировании участков тепловой сети (УТС) необходимо также редактировать и прилегающую инфраструктуру, такую как тепловые камеры (ТК). Объекты типа УТС и ТК принадлежат разным справочникам, а значит, они не смогут пройти обработку ввиду отсутствия единой схемы маршрутизации.

При отсутствии визуальной интерпретации схемы маршрутизации легко допустить ошибку при настройке, особенно если настраиваемая схема обладает большим количеством состояний и переходов. Интерактивная визуализация схемы позволит избежать ошибок при настройке и наглядно покажет маршрут запроса в конкретной схеме, так как восприятие графической информации намного проще восприятия текстовой.

В связи с перечисленными проблемами и недостатками было принято решение о переработке действующего компонента настройки схем маршрутизации запросов на редактирование справочников, который будет сочетать в себе прежнюю функциональность с новыми возможностями.

Функциональные требования к разрабатываемому модулю:

* обеспечение маршрутизации запросов (в том числе мультизапросов) в соответствии с настроенной схемой;
* настройка одной схемы маршрутизации для нескольких справочников;
* описание условий остановки запроса в определенном состоянии;
* визуальное отображение состояний и переходов схемы маршрутизации.

## 4.2 Теория качества информации

Под качеством информации понимают степень её соответствия потребностям пользователей. Свойства информации являются относительными, так как зависят от потребностей пользователей информации [7].

«Качество» часто воспринимают как субъективную величину, так как качество информации может сильно варьироваться в зависимости от пользователей информации, а также методах и способах её использования [8]. Тем не менее, чем выше уровень качества информации, тем выше её объективность. Точность можно рассматривать как один из элементов качества информации, так и как несколько обобщённых, в зависимости от того, что подразумевать под точностью [10].

Ванг и Стронг предлагают разделять качество информации на следующие четыре группы [2]:

1. Внутреннее качество информации:
   * Точность
   * Объективность
   * Достоверность
   * Адекватность
2. Контекстуальное качество информации
   * Релевантность
   * Актуальность
   * Полнота
   * Количество
3. Репрезентативное качество информации
   * Интерпретируемость
   * Формат
   * Согласованность
   * Совместимость
4. Доступность качества информации
   * Доступность
   * Безопасность доступа

**Объективность**

Объективность информации характеризует её независимость от чьего-либо мнения или сознания, а также от методов получения. Более объективна та информация, в которую методы получения и обработки вносят меньший элемент субъективности [9].

**Достоверность**

Достоверность – верность информации, не вызывающая сомнений. Объективная информация всегда достоверна, но достоверная информация может быть как объективной, так и субъективной. Причинами недостоверности могут быть: преднамеренное искажение (дезинформация); непреднамеренное искажение субъективного свойства; искажение в результате воздействия помех; ошибки фиксации информации. В общем случае достоверность информации достигается: указанием времени свершения событий, сведения о которых передаются; сопоставлением данных, полученных из различных источников; своевременным вскрытием дезинформации; исключением искажённой информации и др [10].

**Адекватность**

Адекватность – степень соответствия смысла реально полученной информации и её ожидаемого содержимого. Например задан вопрос – "Сколько у человека пальцев на руке?" "На руке у человека пять пальцев" – ответ достоверный и адекватный, "У человека две руки" - ответ достоверный, но неадекватный [9].

**Актуальность**

Актуальность информации – это степень соответствия информации текущему моменту времени [9].

**Полнота**

Информацию можно считать полной, когда она содержит минимальный, но достаточный для принятия правильного решения набор показателей. Как неполная, так и избыточная информация снижает эффективность принимаемых на основании информации решений [10].

**Доступность**

Доступность информации – мера возможности получить ту или иную информацию.

**Безопасность доступа**

Безопасность доступа к информации – это меры и уровень защищенности информации.

Обеспечение и контроль качества информации очень важный аспект жизненного цикла любой информационной системы. Чем сложнее информационная система, тем строже должны быть требования к системе контроля качества информации, так как ошибка при обработке данных сможет вызвать цепную реакцию, подставив под угрозу всю информационную систему [26]. Восстановление достоверности данных в таком случае может занять большое количество времени, что негативно отразится на рабочем процессе информационной системы в целом.

На сегодняшний день существует множество механизмов контроля качества информации. Вот некоторые из них:

**Многоуровневая обработка информации**

Принцип этого механизма заключается в том, что при любом редактировании информации перед непосредственным внесением изменений в базу данных, новые данные подвергаются нескольким итерациям проверки на их качество (достоверность, актуальность, качество и т.д.) [22]. При реализации этого механизма следует разбить данные на несколько, желательно не связанных между собой, логических групп, если это возможно. Тогда каждая итерация сведётся к проверке на качество отдельной логической группы. Проверка основывается на имеющейся, эталонной информации. Все итерации могут быть выполнены разными людьми, что увеличит надёжность процесса обработки информации.

**Автоматическая проверка изменений**

Механизм автоматической проверки изменений реализуется с помощью программы. Программа выполняет алгоритм проверки введённой информации на достоверность, основываясь на существующих данных. Главным преимуществом данного метода является скорость проверки информации. За короткий промежуток времени программа может проверить большое количество условий, при которых информация считается недостоверной и выдать соответствующее сообщение о запрете на изменение редактируемой информации. Минусом этого метода является проблематичность программирования всех возможных условий, особенно при работе с графической информацией.

**Логирование**

Под логированием подразумевается запись всех изменений, производимых с информацией, в базу данных [23]. С помощью записей в базе данных об изменении той или иной информации можно осуществить откат этой информации и вернуть предыдущие значения атрибутов, если это необходимо. Логирование уместно применять в совокупности с другими методами контроля качества информации (например, с описанными выше), но использовать его как основной метод нецелесообразно. В информационных системах, где возможно частое изменение одних и тех же данных, откат, возможно, будет выполнить проблематично.

Из рассмотренных методов обеспечения и контроля качества информации наиболее оптимальным для решения поставленной задачи является метод многоуровневой обработки информации. Поскольку разработка ведётся для крупной распределённой информационной системы, проверку информации на качество целесообразно выполнять поэтапно и, возможно, распределять этапы между разными пользователями системы с разными полномочиями. Выбранный метод позволяет это и обеспечивает высокий контроль качества информации. Это особенно важно для систем обладающих репликацией данных.

## 4.3 Обзор существующих программных средств

Рассмотрим несколько аналогов, решающих поставленную задачу.

### 4.3.1 Система отслеживания ошибок

Система отслеживания ошибок - прикладная программа, разработанная с целью помочь разработчикам программного обеспечения (программистам, тестировщикам и др.) учитывать и контролировать ошибки, найденные в программах, пожелания пользователей, а также следить за процессом устранения этих ошибок и выполнения или невыполнения пожеланий. Принцип работы системы отслеживания ошибок рассмотрим на примере программного продукта австралийской компании Atlassian. Atlassian JIRA – это система, предназначенная для отслеживания ошибок в процессе разработки и сопровождения программного обеспечения [1]. Также данная программа иногда используется для ведения проектов. Задачи, которые решаются в процессе ведения проекта, реализованы в качестве запросов. Запрос представляет собой конкретную задачу, назначенную определённому пользователю. В зависимости от настроек системы пользователю может быть выделено определённое количество времени для решения задачи, если это необходимо. У запроса есть такой атрибут как статус, который определяет текущее состояние запроса. По этому статусу можно определить, на какой стадии находится решение поставленной задачи. Администратор системы может произвольно настроить состояния, которые может принимать запрос, а также маршрут, по которому запрос попадает из одного состояния в другое. Например: пусть решение задачи предполагает выполнение трёх этапов: подготовительный, основной и заключительный. Для обозначения начального и завершённого состояний запроса введём ещё 2 статуса. Тогда мы можем построить цепочку, по которой будет двигаться запрос в ходе решения поставленной задачи: начальное – подготовительный этап – основной этап – заключительный этап – завершён. В этом случае запрос не сможет пройти в заключительный этап, не остановившись в основном. В случае если необходимо добавить возможность возврата запроса по цепочке назад, этот переход добавляется администратором системы. Таким образом, реализована маршрутизация данных в программном продукте Atlassian JIRA.

Преимущества:

* Удобный инструментарий для настройки маршрута;
* Наглядное визуальное представление маршрута;
* Возможность создать маршрут высокой сложности.

Недостатки:

* Нет возможности динамически изменять маршрут при определённых условиях;
* Отсутствие открытого кода лишает возможности использовать этот модуль в своих целях.

### 4.3.2 Система электронного документооборота

Система электронного документооборота (СЭД) – это автоматизированная система управления корпоративным документооборотом. Автоматизация позволяет сократить время на обработку документов, а также снижает риски случайной потери данных, кроме того, СЭД позволяет руководству контролировать выполнение управленческих решений и отслеживать все действия, производимые с зарегистрированными в системе документами [28]. Рассмотрим одну из наиболее известных систем электронного документооборота, разработанного российской компанией «ЛАНИТ» LanDocs.

Как специализированная для решения задач СЭД программная платформа, LanDocs позволяет строить полный спектр систем электронного документооборота – от самых простых до систем высокой сложности. Сочетание функциональности с открытостью, позволяет при внедрении настроить создаваемую систему в соответствии с нормативной базой и практикой работы с документами конкретного предприятия, ведомства или учреждения. LanDocs имеет высокий уровень масштабируемости решений: в контур документооборота могут быть включены как пользователи вычислительной сети главного офиса, так и сотрудники удаленных подразделений. Платформа включает в себя ПО высокой степени готовности – автоматизация стандартных широко распространенных бизнес-процессов сводится к настройкам платформы и, как правило, не требует программирования. Наряду с этим, LanDocs предоставляет развитые инструментальные средства, с помощью которых могут быть автоматизированы бизнес-процессы специфичные для данного предприятия или предметной области [12].

LanDocs обеспечивает автоматизированное создание списка рассылки документа и регистрацию факта отправки его или его копии адресату (адресатам). Рассылка документов может выполняться по личному или общему списку рассылки, по принадлежности к подразделению или роли (группе работников, объединенных по какому-либо признаку). Стандартные алгоритмы рассылки – «последовательно по цепочке» и «веером».

LanDocs ведет автоматизированный учет движения документа, т.е. фиксирует в своей БД все действия пользователей, совершенные с документом, и изменения его статуса на протяжении всего жизненного цикла документа от регистрации до сдачи в архив.

Преимущества:

* Широкий спектр возможностей, позволяющий реализовать маршрутизацию запросов любой сложности;
* Возможность динамически изменять маршрут, в зависимости от атрибутов конкретного запроса;
* Наличие открытого документированного программного интерфейса, позволяющего внедрять систему в своё приложение с помощью OLE-интерфейса.

Недостатки:

* Высокая стоимость лицензионной версии программы не оправдана, в виду не использования большого количества функций системы.

## 4.4 Математическое и информационное обеспечение

### 4.4.1 Теория конечных автоматов

Конечным автоматом называется формальная система M = (Q, Σ, δ, q0, F), где Q – конечное непустое множество состояний; Σ – конечный входной алфавит; δ – отображение типа Q × Σ → Q; q0 ∈ Q – начальное состояние; F ⊆ Q – множество конечных состояний [17].

Конечный автомат состоит из конечного управления и входной ленты, разбитой на ячейки. В каждой ячейке записан один символ из входного алфавита Σ, и все они образуют конечную входную цепочку. Конечное управление первоначально находится в состоянии q0 и сканирует крайнюю левую ячейку ленты. По мере чтения входной цепочки слева направо автомат переходит в другие состояния из множества Q. Если, прочитав входную цепочку, автомат оказывается в некотором конечном состоянии из множества F, то говорят, что он принял ее. Множество цепочек, принимаемых конечным автоматом, называется языком, распознаваемым данным конечным автоматом.

Конечные автоматы подразделяются на детерминированные и недетерминированные:

Детерминированным конечным автоматом называется такой автомат, в котором нет дуг с меткой ε (неопределённым символом) и из любого состояния по любому символу возможен переход в точности в одно состояние. Все остальные конечные автоматы являются недетерминированными.

Конечные автоматы описывают с помощью двух способов [24]:

1. Таблица переходов

Таблица переходов – табличное представление функции δ. Обычно в такой таблице каждой строке соответствует одно состояние, а столбцу – один допустимый входной символ. В ячейке на пересечении строки и столбца записывается состояние, в которое должен перейти автомат, если в данном состоянии он считал данный входной символ.

На рисунке 4.2 приведен пример описания конечного автомата с помощью таблицы переходов.

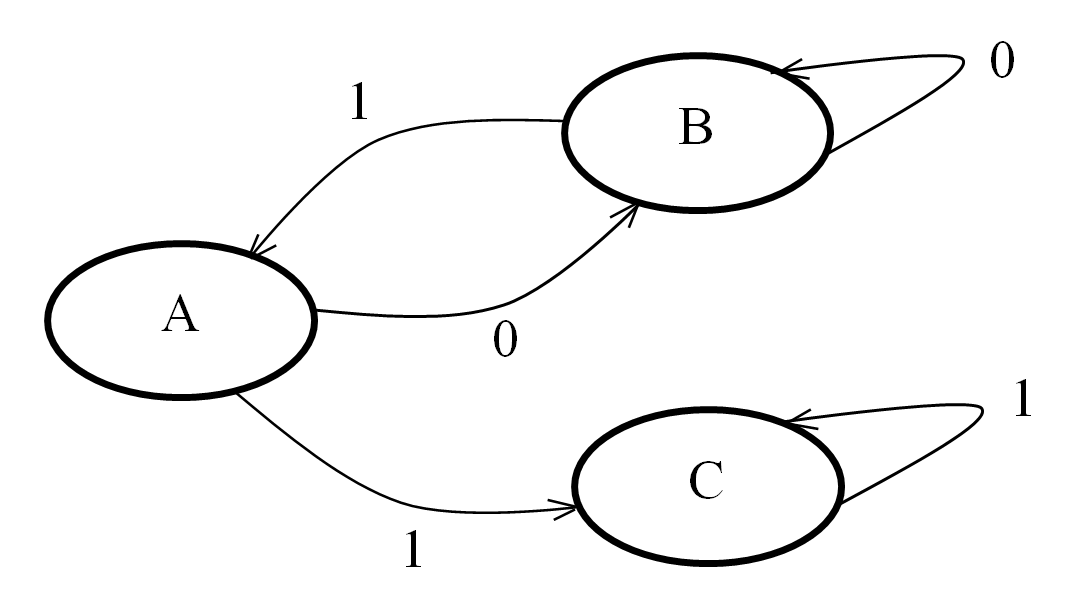
|  |  |  |
| --- | --- | --- |
| Символы  Состояния | **0** | **1** |
| **A** | B | C |
| **B** | B | A |
| **C** | - | C |

*Рис. 4.2 Таблица переходов*

1. Диаграмма состояний

Диаграмма состояний (или иногда граф переходов) – графическое представление множества состояний и функции переходов. Представляет собой размеченный ориентированный граф, вершины которого – состояния конечного автомата, дуги – переходы из одного состояния в другое, а метки дуг – символы, по которым осуществляется переход из одного состояния в другое. Если переход из одного состояния в другое может быть осуществлен по одному из нескольких символов, то все они должны быть надписаны над дугой диаграммы.

На рисунке 4.3 изображена диаграмма состояний конечного автомата, который был описаны выше с помощью таблицы переходов.



*Рис. 4.3 Диаграмма состояний*

Для решения поставленной задачи систему маршрутизации запросов можно интерпретировать с помощью конечного автомата следующим образом [31]:

Конечное непустое множество состояний Q – состояния, которые может принимать запрос, проходя по схеме маршрутизации;

Конечный входной алфавит Σ – набор возможных пользователям действий для перемещения запроса из одного состояния в другое;

Функция переходов δ – набор переходов, связывающих состояния межу собой;

Начальное состояние q0 – состояние, в котором пользователь создаёт запрос;

Множество конечных состояний F – состояние «обработан», обозначающее правильность вносимых изменений и позволяющее фактическую запись в базу данных. В данном случае множество конечных состояний будет состоять из одного элемента.

Главным отличием работы схемы маршрутизации от традиционного конечного автомата, будет являться отсутствие заранее составленной входной цепочки. Группа пользователей не может заранее знать, какие действия они будут совершать с запросом и в какой последовательности. В данном случае конечный автомат будет совершать анализ действия и последующий переход в требуемое состояние по факту получения этого действия.

### 4.4.2 Метаязык

Метаязык – это язык, используемый для выражения суждений о другом языке, языке-объекте. С помощью метаязыка изучают структуру выражений языка-объекта, его свойства и отношения к другим языкам. Изучаемый язык называется также предметным языком по отношению к данному метаязыку. Как предметный язык, так и метаязык могут быть естественными языками. Метаязык может отличаться от языка-объекта, но может и совпадать с ним или отличаться лишь частично, например специальной терминологией. Другими словами метаязык – это «сверхязык», предназначенный для описания языка. В информатике метаязыком принято считать дополнительные данные, служащие для описания уже имеющихся. В математической логике метаязык используется для определения истинности или ложности выражения предметного языка.

Примером метаязыка может служить русский язык в учебнике английского языка для русских, в то время как английский язык будет являться его предметным языком.

Одной из задач данной дипломной работы является создание механизма описания правил для остановки запроса в конкретном состоянии схемы маршрутизации. Этот механизм будет реализован с помощью метаязыка, где предметным языком будут являться сведения о параметрах и свойствах запроса. Для вычисления истинности или ложности выражения метаязык должен обладать операторами, которые однозначно смогут определить значение описанного условия.

Необходимо также создать интерпретатор для разбора описанного условия на метаязыке [25]. В основе работы интерпретатора лежит теория конечных автоматов, о которой написано выше. Интерпретатор разбивает выражение на условия, соединенные друг с другом булевскими операторами «и», «или», «не», а также учитывает скобки для определения порядка вычисления логических действий.

### 4.4.3 Система управления базами данных

Вся информация об объектах системы АСОТ, в том числе и графическая, содержится в базах данных. Манипуляции с этой информацией происходят с помощью системы управления базами данных.

База данных – представленная в объективной форме совокупность самостоятельных материалов (статей, расчётов, нормативных актов, судебных решений и иных подобных материалов), систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины [6].

Система управления базами данных (СУБД) – совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных [5].

СУБД выполняет следующие функции:

* управление данными во внешней памяти;
* управление данными в оперативной памяти;
* журнализация изменений, резервное копирование и восстановление базы данных после сбоев;
* поддержка языков БД.
* По модели данных СУБД подразделяются на следующие виды:
* Иерархические
* Сетевые
* Реляционные
* Объектно-ориентированные
* Объектно-реляционные

Транзакция – это последовательность операций над БД, рассматриваемых СУБД как единое целое. Либо транзакция успешно выполняется, и СУБД фиксирует изменения БД, произведенные этой транзакцией, во внешней памяти, либо ни одно из этих изменений никак не отражается на состоянии БД [3].

Одним из основных требований к СУБД является надежность хранения данных во внешней памяти. Под надежностью хранения понимается то, что СУБД должна быть в состоянии восстановить последнее согласованное состояние БД после любого аппаратного или программного сбоя. Обычно рассматриваются два возможных вида аппаратных сбоев: так называемые мягкие сбои, которые можно трактовать как внезапную остановку работы компьютера (например, аварийное выключение питания), и жесткие сбои, характеризуемые потерей информации на носителях внешней памяти. Примерами программных сбоев могут быть: аварийное завершение работы СУБД (по причине ошибки в программе или в результате некоторого аппаратного сбоя) или аварийное завершение пользовательской программы, в результате чего некоторая транзакция остается незавершенной [11]. Первую ситуацию можно рассматривать как особый вид мягкого аппаратного сбоя; при возникновении последней требуется ликвидировать последствия только одной транзакции. Понятно, что в любом случае для восстановления БД нужно располагать некоторой дополнительной информацией. Другими словами, поддержание надежности хранения данных в БД требует избыточности хранения данных, причем та часть данных, которая используется для восстановления, должна храниться особо надежно. Наиболее распространенным методом поддержания такой избыточной информации является ведение журнала изменений БД. Журнал – это особая часть БД, недоступная пользователям СУБД и поддерживаемая с особой тщательностью, в которую поступают записи обо всех изменениях основной части БД. Иногда поддерживаются две копии журнала, располагаемые на разных физических дисках для увеличения надёжности. В разных СУБД изменения БД журнализуются на разных уровнях: иногда запись в журнале соответствует некоторой логической операции изменения БД (например, операции удаления строки из таблицы реляционной БД), иногда – минимальной внутренней операции модификации страницы внешней памяти; в некоторых системах одновременно используются оба подхода [27]. Во всех случаях придерживаются стратегии "упреждающей" записи в журнал, так называемого протокола Write Ahead Log – WAL. Эта стратегия заключается в том, что запись об изменении любого объекта БД должна попасть во внешнюю память журнала раньше, чем измененный объект попадет во внешнюю память основной части БД. Известно, что если в СУБД корректно соблюдается протокол WAL, то с помощью журнала можно решить все проблемы восстановления БД после любого сбоя.

Для работы с базами данных используются специальные языки, называемые языками баз данных. В современных СУБД обычно поддерживается единый интегрированный язык, содержащий все необходимые средства для работы с БД, начиная от ее создания, и обеспечивающий базовый пользовательский интерфейс с базами данных. Стандартным языком наиболее распространенных в настоящее время реляционных СУБД является язык SQL (Structured Query Language). Прежде всего, язык SQL позволяет определять схему реляционной БД и манипулировать данными. При этом именование объектов БД (для реляционной БД – именование таблиц и их столбцов) поддерживается на языковом уровне в том смысле, что компилятор языка SQL производит преобразование имен объектов в их внутренние идентификаторы на основании специально поддерживаемых служебных таблиц-каталогов. Внутренняя часть СУБД (ядро) вообще не работает с именами таблиц и их столбцов. Наконец, авторизация доступа к объектам БД производится также на основе специального набора операторов SQL. Идея состоит в том, что для выполнения операторов SQL разного вида пользователь должен обладать различными полномочиями. Пользователь, создавший таблицу БД, обладает полным набором возможных действий для работы с этой таблицей. В число этих действий входит возможность передачи всех или части полномочий другим пользователям. Ограничение возможностей пользователей описываются в специальных таблицах-каталогах, а их контроль поддерживается на языковом уровне.

## 4.5 Структура модуля

### 4.5.1 Основные понятия

*Пользователь* – зарегистрированный в системе человек.

*Физическая таблица* – зарегистрированная в базе данных таблица для хранения информации об однотипных объектах. Таблица может содержать только часть информации об объекте, а также может являться таблицей связи для реализации N:N связей [20] между другими таблицами.

*Справочник* – виртуальное хранилище данных для работы с однотипными объектами. Справочник обязательно имеет главную физическую таблицу, а также может иметь несколько связанных таблиц, в случае если информация об объектах такого типа разбита на несколько частей, которые содержатся в разных физических таблицах.

*Объект* – конкретная запись в справочнике.

*Запрос* – пакет информации, содержащий данные об изменении существующей в базе данных информации о конкретном объекте. Запрос может содержать информацию о создании нового объекта или редактировании/удалении существующего.

*Мультизапрос* – это запрос, объединяющий в одну логическую группу несколько объектов для удобной обработки информации. Объекты мультизапроса могут являться как однотипными, так и разнотипными.

*Схема маршрутизации запросов* – детерминированный конечный автомат, вершинами которого являются состояния, а дугами – переходы между состояниями, реализующий логику взаимодействия пользователей при редактировании данных для обеспечения контроля их качества.

*Роль* – некоторая логическая группа пользователей, выполняющая определенные действия с запросами при их прохождении по схеме маршрутизации. Каждому состоянию схемы маршрутизации, за исключением конечных, должна соответствовать ровно одна роль.

*Тип роли* – логическая группа ролей, объединенная определённым признаком для удобного разграничения полномочий между разнотипными ролями.

*Состояние* – атрибут запроса, обозначающий его текущее положение в схеме маршрутизации запросов. Состояние привязано к конкретной роли, поскольку работа с запросом в определённом состоянии может выполняться пользователями, обладающими одной и той же ролью. Состояния бывают четырёх типов: «Начальное», «В процессе», «Обработан», «Отклонён». Каждое из этих типов обозначает прогресс запроса в схеме от его формирования до внесения изменений в базу данных или его отклонения пользователем или системой.

*Переход* – процесс изменения состояния запроса. Существует 3 типа переходов: *Forward, Back, Abort*. Тип *Forward* обозначает продвижение запроса по схеме вперед к состоянию, где он будет обработан. *Back* возвращает запрос по цепи назад. Переход *Abort* отклоняет запрос, который далее не рассматривается пользователями. Из любого промежуточного состояния схемы маршрутизации должен быть задан ровно один переход типа *Forward*, может быть задан максимум один переход типа *Abort* и несколько переходов типа *Back*. Из начального состояния схемы маршрутизации должен быть задан ровно один переход. Этот переход должен иметь тип *Forward*. Из конечных состояний не может быть задано ни одного перехода.

*Смежные состояния* – состояния схемы маршрутизации, соединённые переходом типа Forward, при этом исходящее состояние называется предыдущим, а входящее – следующим. Двум смежным состояниям не может соответствовать одна и та же роль.

*Набор полей* – перечень атрибутов объекта доступных на редактирование для роли.

*Бизнес-процесс (БП)* – перечень соответствий между ролями схемы маршрутизации запросов и набором полей, доступных для редактирования, а также набор допустимых операций со справочниками. В случае если в рамках одной схемы маршрутизации при создании запроса пользователю доступны несколько бизнес-процессов, то он должен выбрать один из них.

### 4.5.2. Концепция модуля.

Необходимость разработки модуля настройки маршрутизации запросов образовалась в процессе внедрения в систему АСОТ технологии мультизапросов. В первоначальной версии системы у каждого справочника была своя схема маршрутизации запросов. Такая структура программы не позволяла разнотипным объектам проходить по одной и той же схеме маршрутизации в рамках одного запроса, что исключало возможность использования технологии мультизапросов. Также, стоит отметить, что на 35 справочников существовало всего лишь 4 уникальные схемы маршрутизации, что говорит об избыточности данных. Также изменение одинаковых схем являлось довольно трудоёмким и неудобным процессом. Эти факторы тоже повлияли на решение о смене структуры программной части по маршрутизации запросов.

Настройка схем маршрутизации

Маршрутизация запросов

Запись в БД

Интерфейс создания и редактирования запросов

Ядро.

Основной компонент АСОТ

*Рис 4.4 Модуль настройки схем маршрутизации*

Часть, выделенная голубым фоном (рис. 4.4), показывает структуру разрабатываемого модуля настройки схем маршрутизации. Модуль разбит на три логических элемента: «Настройка схем маршрутизации», «Маршрутизация запросов» и «Запись в БД». Также на схеме изображены компоненты, с которыми модуль настройки схем маршрутизации обменивается данными. С помощью интерфейса создания и редактирования запросов пользователи формируют информацию об изменении данных и отправляют её по настроенной схеме. В ядре АСОТ реализованы все базовые функции работы программы, в том числе и обмен данными с БД.

«Настройка схем маршрутизации» представляет собой интерфейс, с помощью которого администратор системы создаёт и редактирует все сущности системы маршрутизации: регистрирует физические таблицы, создаёт справочники, типы ролей и схемы маршрутизации. В схемах маршрутизации необходима настройка состояний, переходов, ролей, наборов полей и бизнес-процессов.

«Маршрутизация запросов» выполняет функцию распределения запросов по ролям по предварительно настроенной схеме маршрутизации.

«Запись в БД» осуществляет запись изменённой информации в базу данных с помощью функций описанных в ядре АСОТ. Перед непосредственной записью данных программа выполняет несколько операций, для проверки полноправности вносимых изменений.

Новая структура модуля решила ряд проблем первоначальной версии программы, но она не может обеспечить должную гибкость схем маршрутизации при решении каких-либо конкретных задач, например: динамическое изменение пути запроса в зависимости от заполненных атрибутов и их значений. Для достижения этой цели в систему были внедрены такие средства как: бизнес-процесс и условия остановки в состояниях.

Бизнес-процесс – это режим обработки запроса, со своими ограничениями и допусками. С помощью этого режима можно задавать допустимые операции со справочниками и наборы полей объектов. Также бизнес-процесс используется как флаг для условий остановки в состоянии. При определении бизнес-процесса программа маршрутизации может остановить запрос в конкретном состоянии или пропустить его, в зависимости от описанного администратором системы условия.

По умолчанию переход в следующее смежное состояние выполняется автоматически. С помощью метаязыка можно устанавливать правила, при которых в состоянии будет произведена остановка. Правила остановки можно устанавливать только на промежуточные состояния и проверяться они будут только при выполнении переходов типа Forward. В правиле описывается набор условий, связанных логическими операторами.

В случае если при выполнении перехода обнаруживается, что пользователь, осуществляющий переход, может работать в следующем смежном состоянии, то переход в него выполняется автоматически, после чего вновь проверяется доступность работы пользователя в следующем смежном состоянии, и т.д.

Для того чтобы правила остановки выполнялись, они должны быть правильно описаны на специально разработанном метаязыке.

### 4.5.3 Описание правил на метаязыке

В схеме маршрутизации запрос проходит путь от начального состояния («создан») до конечного («обработан» или «отклонён») через множество промежуточных состояний, определённых администратором системы при настройке схемы. Состояния запроса определяют роль, которой этот запрос доступен для обработки. По умолчанию промежуточные состояния пропускаются системой, но при необходимости администратор системы может описать правила остановки для конкретных состояний, при выполнении условий которых запрос будет останавливаться в них для обработки пользователем.

Документация по использованию метаязыка приведена ниже.

**Общие положения.**

Правило – набор условий соединённых между собой логическими операторами с определённым порядком выполнения действий с помощью скобок.

Условие – выражение, сравнивающее значение некоторого поля объекта запроса с заданным пользователем или проверяющее определённые параметры запроса.

**Описание условий.**

Условие может проверять следующее:

* Операция над объектом
* Изменение значения поля
* Наличие изменённых полей кроме заданного
* Пустое поле
* Вложение
* Значение поля

Интерпретатор распознаёт следующие служебные слова: «операция», «изменено», «пустое», «приложен файл». Нераспознанные слова интерпретатор будет воспринимать как название поля.

**Проверка операции над объектом.**

Для того чтобы задать проверку операции над объектом необходимо написать служебное слово «операция» (без кавычек), далее через пробел задать оператор сравнения: либо «=» либо «<>» и после этого обозначить тип операции – один из: «добавление», «редактирование» или «удаление».

Пример: операция = удаление

операция <> редактирование

операция=добавление – ошибка

Примечание: очень важно соблюдать пробелы между словами и операторами, так как слитно написанную строку интерпретатор воспримет как одно слово и выдаст ошибку. Регистр в свою очередь не имеет значения, поэтому пользователь может написать «Операция» или «УДАЛЕНИЕ». Эти правила распространяются на все остальные служебные слова и конструкции, в том числе и на логические операторы.

**Проверка изменённого или пустого значения поля.**

Служебными словами для проверки изменённого или пустого значения поля являются «изменено» и «пустое» соответственно. Следующее слово после служебного будет определено как названия поля объекта. Важно, чтобы название поля являлось одним словом (т.е. без пробелов) и соответствовало имени атрибута объекта в базе данных.

Пример: изменено number

пустое street\_name

изменено full address – ошибка

**Проверка наличия изменённых полей кроме заданного.**

Конструкция проверки наличия изменённых полей кроме заданного состоит из служебного словосочетания «изменено кроме» и названия атрибута объекта. Разделение оператора и имени поля также осуществляется с помощью пробела.

Пример: изменено кроме graphics

изменено\_кроме street\_name – ошибка

**Проверка наличия приложенного файла.**

Проверка наличия приложенного файла описывается служебной конструкцией «приложен файл». Эта конструкция является условием, дополнительные параметры и операторы сравнения не требуются.

Пример: Приложен файл

приложенФайл – ошибка

**Сравнение значений полей.**

Конструкция сравнения значений полей состоит из трёх частей: наименование поля, оператор сравнения, сравниваемое значение. Все три части, как и в предыдущих случаях, должны быть отделены друг от друга пробелами, а значение должно быть заключено в кавычки. Доступные операторы сравнения:

* = - равно
* <> - не равно
* > - больше
* < - меньше
* >= больше или равно
* <= - меньше или равно

Пример: number <> “20”

street\_name = “ул. Арбат”

full\_number = 10/17 – ошибка

Примечание: При описании правил запрещено использовать символы «~» и «|», так как они являются служебными и требуются для разбора строки правил.

**Описание правил.**

При описании правил используются логические операторы «и», «или», отрицание «не», а также скобки для установки порядка вычисления выражения. Операторы могут стоять соединять как простые условия, так и целые логические выражения, заключённые в скобки. Отрицание также может стоять перед условием или выражением в скобках.

При вычислении логического выражения сначала вычисляется отрицание («не»), затем конъюнкция («и»), и последней – дизъюнкция («или»). Порядок действий вычисляется слева направо.

В выражении *«изменено graphics или version = “2” и операция <> удаление»* будет следующий порядок действий:

1. version = “2” и операция <> удаление = результат
2. изменено graphics или результат = конечный\_результат

Если в это выражение добавить скобки: *«(изменено graphics или version = “2”) и операция <> удаление»*, порядок действий изменится:

1. изменено graphics или version = “2” = результат
2. результат и операция <> удаление = конечный\_результат

Далее следуют несколько примеров описания правил:

не пустое structure или full\_number = “20а”

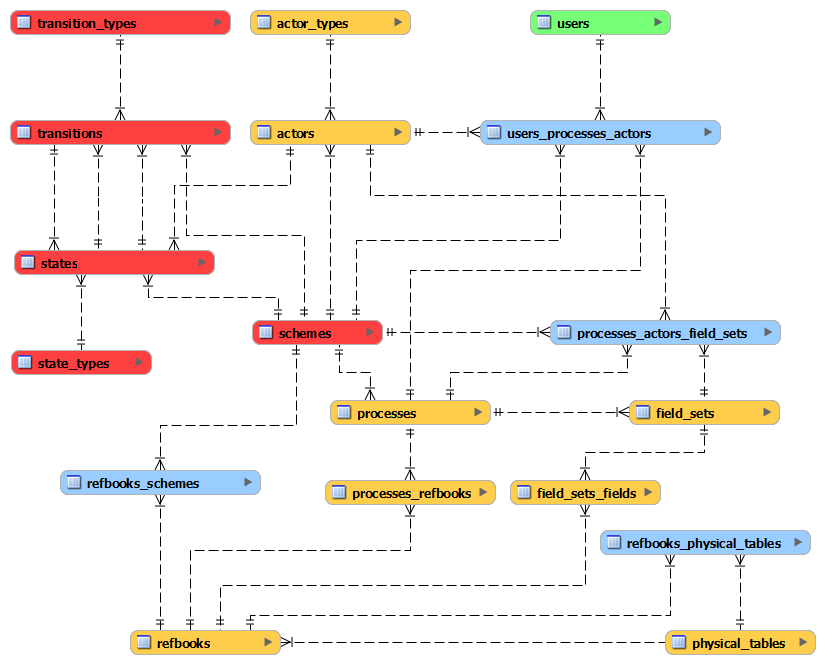
(изменено graphics или version = “2”) и операция <> удаление

(Приложен файл и version > 1) не пустое number – ошибка (между закрывающей скобкой и отрицанием должен стоять оператор)

И операция = удаление и не приложен файл – ошибка (логический оператор «и» не может стоять в начале выражения)

### 4.5.4. Структура служебной базы данных.

Изменение структуры модуля настройки и выполнения маршрутизации запросов, а также новая функциональность привела к необходимости изменения структуры служебной базы данных. Эта база данных отвечает за хранение всей информации о схемах маршрутизации, пользователях, их ролях, полномочиях этих ролей и так далее. Новая структура базы данных изображена на рисунке 4.5:



*Рис. 4.5 Структура служебной БД*

Схема разбита на несколько цветовых категорий:

Красная – описание схем маршрутизации;

Зелёная – администрирование пользователей;

Синяя – таблицы связей;

Жёлтая – остальные сущности, модуля настройки и выполнения маршрутизации запросов.

**Описание схем маршрутизации**

Данные о схемах маршрутизации распределены между несколькими таблицами базы данных. **Schemes** – таблица, в которой хранятся идентификаторы и названия схем маршрутизации. Также здесь содержится файл графического изображения схемы: состояния и переходы, их положение относительно друг друга. **States** – таблица состояний. В этой таблице хранится название состояния, схема, которой принадлежит состояние, роль, которая работает в данном состоянии, а также правило остановки в этом состоянии. Правило остановки является строкой, перечисляющей все условия, разделённые служебными символами. В **State\_types** перечислены типы состояний. На данный момент таких типа 4: «начальное», «в процессе», «обработан», «отклонён». **Transitions –** таблица переходов. Помимо информации о входном и выходном состоянии здесь хранятся значения двух флагов: обязательное указание пользователя при переходе, комментарий обязателен. **Transition\_types** содержиттипы переходов. Таких перехода всего 3: «Направить» (Forward), «уточнить» (Back), «отклонить» (Abort). Более подробно они описаны в пункте 4.5.1.

**Администрирование пользователей**

Для предоставления пользователям доступа к системе в АСОТ реализована система авторизации пользователей. Администрирование пользователей выполняет функцию обеспечения безопасности данных, с которыми работают пользователи следующим образом:

* Выдача уникального пароля пользователю для предотвращения входа в систему от чужого лица;
* Ограничение прав доступа к ресурсам и выделение полномочий индивидуально каждому пользователю;
* Привязка идентификатора пользователя к любой операции совершённой в системе для определения совершившего данную операцию пользователя.

Используемая таблица **Users**. В этой таблице хранятся идентификатор пользователя, его авторизационные данные, контактные данные (ФИО, e-mail), а также время последнего входа в систему и количество активных запросов, в которых он участвует. В таблице **Users\_processes\_actors** хранятся связи пользователей и ролей для конкретного бизнес-процесса в определённой схеме.

**Сущности схем маршрутизации**

Для хранения информации о ролях используется таблица **Actors**. У роли есть имя, принадлежность схеме маршрутизации, а также тип. Типы ролей в свою очередь хранятся в отдельной таблице **Actor\_types**.

Справочники, используемые для схем маршрутизации, добавляются в таблицу **Refbooks.** У справочника должна быть обязательно главная физическая таблица. Все необходимые физические таблицы регистрируются администратором и хранятся в **Physical\_tables**. У справочника есть также связанные таблицы. Для реализации связей между **Refbooks** и **Physical\_tables** типа N:N (много ко многим) используется таблица связей **Refbooks\_physical\_tables**. В схемемаршрутизацииможет использоваться определённый набор зарегистрированных в системе справочников**.** Данные озарегистрированных в схеме справочникаххранятся в таблице связей **Refbooks\_schemes**.

Бизнес-процессы перечислены в таблице **Processes**. В ней хранится название бизнес-процесса и его принадлежность схеме. Для определения допустимых операций со справочниками в конкретном бизнес-процессе существует таблица **Processes\_refbooks**. Можно создать бизнес-процесс, который допускает только удаление объектов конкретного справочника. Набор допустимых ролей определяет те атрибуты объекта, которые может редактировать пользователь. Информация об этих наборах хранится в **Field\_sets**. Сами же названия полей вынесены в отдельную таблицу **Field\_sets\_fields**. У этой таблицы есть также связь с **Refboooks**,так как имя поля является атрибутом справочника.

Ещё в базе данных есть две таблицы связей **Processes\_actors\_field\_sets** и **Users\_processes\_actors**. Первая описывает связь ролей и наборов полей в бизнес-процессе, а вторая связь пользователей и ролей в бизнес процессе. Это означает, что у одной роли может быть несколько наборов полей, а у одного пользователя может быть несколько ролей и наоборот.

# 5. Конструкторско-технологическая часть

## 5.1 Технология программирования

Разработка программного обеспечения – это процесс, направленный на создание и поддержание работоспособности качества и надежности программного обеспечения, используя технологии, методологию и практики из информатики, управления проектами, математики, инженерии и других областей знания.

На сегодняшний день существует большое количество подходов к разработке программного обеспечения, описанных различными методологиями. Задача этих методологий заключается в улучшении производительности, качества и надёжности разработки программного обеспечения. Не существует единого стандарта, которого стоит придерживаться при разработке, так как в зависимости от требований, которые предъявляются к продукту, а также условий, в рамках которых будет вестись работа, выгодно выбрать тот или иной метод, который лучшим образом отвечает поставленной задаче.

На сегодняшний день существует множество типов методологий разработки программного обеспечения со своими преимуществами и недостатками. Но при решении поставленной задачи выбор лучшей методологии стоит производить из группы гибких методик разработки. Гибкая методология, обычно, предполагает наличие небольшой команды разработчиков, а также очень проста для реализации и легко адаптируется к изменениям требований к продукту на любом этапе жизненного цикла.

Гибкая методология разработки – серия подходов к разработке программного обеспечения, ориентированных на использование итеративной разработки и динамическое формирование требований, и обеспечение их реализации в результате постоянного взаимодействия внутри самоорганизующихся рабочих групп, состоящих из специалистов различного профиля.

Основные идеи гибких методик разработки:

* Личности и их взаимодействия важнее, чем процессы и инструменты;
* Работающее программное обеспечение важнее, чем полная документация;
* Сотрудничество с заказчиком важнее, чем контрактные обязательства;
* Реакция на изменения важнее, чем следование плану.

Гибкая методология разработки придерживается следующих принципов:

* удовлетворение клиента за счёт ранней и бесперебойной поставки ценного программного обеспечения;
* приветствие изменений требований даже в конце разработки (это может повысить конкурентоспособность полученного продукта);
* частая поставка рабочего программного обеспечения (каждый месяц или неделю или ещё чаще);
* тесное, ежедневное общение заказчика с разработчиками на протяжении всего проекта;
* проектом занимаются мотивированные личности, которые обеспечены нужными условиями работы, поддержкой и доверием;
* рекомендуемый метод передачи информации – личный разговор (лицом к лицу);
* работающее программное обеспечение – лучший измеритель прогресса;
* спонсоры, разработчики и пользователи должны иметь возможность поддерживать постоянный темп на неопределённый срок;
* постоянное внимание улучшению технического мастерства и удобному дизайну;
* простота – искусство не делать лишней работы;
* лучшие технические требования, дизайн и архитектура получаются у самоорганизованной команды;
* постоянная адаптация к изменяющимся обстоятельствам.

Рассмотрим несколько методов более конкретно.

### 5.1.1 Экстремальное программирование

Экстремальное программирование (англ. Extreme Programming, XP) – это упрощенная методология организации разработки программ для небольших и средних по размеру команд разработчиков, занимающихся созданием программного продукта в условиях неясных или быстро меняющихся требований. Авторами методологии являются Кент Бек, Уорд Каннингем, Мартин Фаулер и другие.

**Цели экстремального программирования.**

Основными целями XP являются повышение доверия заказчика к программному продукту путем предоставления реальных доказательств успешности развития процесса разработки и резкое сокращение сроков разработки продукта. При этом XP сосредоточено на минимизации ошибок на ранних стадиях разработки. Это позволяет добиться максимальной скорости выпуска готового продукта и даёт возможность говорить о прогнозируемости работы. Практически все приемы XP направлены на повышение качества программного продукта.

**Принципы экстремального программирования.**

**Итеративность.** Разработка ведется короткими итерациями при наличии активной взаимосвязи с заказчиком. Итерации как таковые предлагается делать короткими, рекомендуемая длительность – 2-3 недели и не более 1 месяца. За одну итерацию группа программистов обязана реализовать несколько свойств системы. Отсутствие формализации описания входных данных проекта в XP стремятся компенсировать за счет активного включения в процесс разработки заказчика как полноправного члена команды.

**Простота решений.** Принимается первое простейшее рабочее решение. Экстремальность метода связана с высокой степенью риска решения, обусловленного поверхностностью анализа и жестким временным графиком. Реализуется минимальный набор главных функций системы на первой и каждой последующей итерации; функциональность расширяется на каждой итерации.

**Интенсивная разработка малыми группами** (не больше 10 человек) и **парное программирование** (когда два программиста вместе создают код на одном общем рабочем месте), активное общение в группе и между группами. Все это нацелено на как можно более раннее обнаружение проблем (как ошибок, так и срыва сроков). Парное программирование направлено на решение задачи стабилизации проекта. При применении экстремального программирования высок риск потери кода по причине ухода программиста, не выдержавшего интенсивного графика работы. В этом случае второй программист из пары играет роль «наследника» кода. Немаловажно и то, как именно распределены группы в рабочем пространстве – в XP используется открытое рабочее пространство, которое предполагает быстрый и свободный доступ всех ко всем.

**Обратная связь с заказчиком**, представитель которого фактически вовлечен в процесс разработки.

**Достаточная степень смелости** и желание идти на риск.

**Основные приемы экстремального программирования.**

Двенадцать основных приёмов экстремального программирования [2] могут быть объединены в четыре группы:

* Короткий цикл обратной связи (Fine scale feedback)
  + Разработка через тестирование (Test driven development)
  + Игра в планирование (Planning game)
  + Заказчик всегда рядом (Whole team, Onsite customer)
  + Парное программирование (Pair programming)
* Непрерывный, а не пакетный процесс
  + Непрерывная интеграция (Continuous Integration)
  + Рефакторинг (Design Improvement, Refactor)
  + Частые небольшие релизы (Small Releases)
* Понимание, разделяемое всеми
  + Простота (Simple design)
  + Метафора системы (System metaphor)
  + Коллективное владение кодом (Collective code ownership) или выбранными шаблонами проектирования (Collective patterns ownership)
  + Стандарт кодирования (Coding standard or Coding conventions)
* Социальная защищенность программиста (Programmer welfare):
  + 40-часовая рабочая неделя (Sustainable pace, Forty hour week)

### 5.1.2 Методология SCRUM

Scrum – одна из самых популярных методологий гибкой разработки. Главным преимуществом этой методологии является её простота [18].

В Scrum существуют всего три роли:

* Scrum Master
* Product Owner
* Team

**Скрам Мастер (Scrum Master)**

Скрам Мастер (Scrum Master) – самая важная роль в методологии. Эта роль отвечает за успех Scrum в проекте. По сути, Скрам Мастер является интерфейсом между менеджментом и командой. Как правило, эту роль в проекте играет менеджер проекта. Важно подчеркнуть, что Скрам Мастер не раздает задачи членам команды. В гибких методах разработки команда является самоорганизующейся и самоуправляемой.

Основные обязанности Скрам Мастера таковы:

* Создает атмосферу доверия,
* Участвует в митингах в качестве человека, обеспечивающего легкую коммуникабельность между участниками митинга
* Устраняет препятствия
* Делает проблемы и открытые вопросы видимыми
* Отвечает за соблюдение практик и процесса в команде

**Ведущий проекта (Product Owner)**

Ведущий проекта (Product Owner) – это человек, отвечающий за разработку продукта. Как правило, это менеджер для продуктовой разработки, менеджер проекта для внутренней разработки и представитель заказчика для заказной разработки. Ведущий проекта – это единая точка принятия окончательных решений для команды в проекте, именно поэтому это всегда один человек, а не группа или комитет.

Обязанности ведущего проекта таковы:

* Отвечает за формирование концепции продукта
* Управляет ожиданиями заказчиков и всех заинтересованных лиц
* Координирует и приоритизирует бэклог продукта
* Предоставляет понятные и тестируемые требования команде
* Взаимодействует с командой и заказчиком
* Отвечает за приемку кода в конце каждой итерации

Ведущий проекта ставит задачи команде, но он не вправе ставить задачи конкретному члену проектной команды в течение спринта.

**Команда (Team)**

В методологии Scrum команда является самоорганизующейся и самоуправляемой. Команда берет на себя обязательства по выполнению объема работ на спринт перед ведущим проекта. Работа команды оценивается как работа единой группы. В Scrum вклад отдельных членов проектной команды не оценивается, так как это разваливает самоорганизацию команды.

Обязанности команды таковы:

* Отвечает за оценку элементов бэклога
* Принимает решение по дизайну и имплементации
* Разрабатывает софт и предоставляет его заказчику
* Отслеживает собственный прогресс (вместе со Скрам Мастером).
* Отвечает за результат перед ведущим проекта

Размер команды ограничивается размером группы людей, способных эффективно взаимодействовать лицом к лицу. Типичные размер команды – от пяти до девяти человек.

**Спринт (Sprint)**

В Scrum итерация называется спринт (sprint). Ее длительность обычно составляет 1 месяц (30 дней). Результатом спринта является готовый продукт, который можно передавать заказчику (по крайней мере, система должна быть готова к показу заказчику). Короткие спринты обеспечивают быстрый отзыв проектной команде от заказчика. Заказчик получает возможность гибко управлять возможностями системы, оценивая результат спринта и предлагая улучшения к созданной функциональности. Такие улучшения попадают в бэклог продукта, получают приоритет наравне с прочими требованиями и могут быть запланированы на один из следующих спринтов. В течение спринта делаются все работы по сбору требований, дизайну, кодированию и тестированию продукта. Требования к спринту должен быть фиксированным. Это позволяет команде давать обязательства на тот объем работ, который должен быть сделан в спринте. Это означает, что бэклок спринта никем не может быть изменен, кроме команды.

В процессе разработки ведутся специальные документы, описывающие план работ команды. Это бэклог продукта и бэклог спринта.

Бэклог продукта – это приоритезированный список имеющихся на данный момент бизнес-требований и технических требований к системе. Бэклог продукта включает в себя дефекты, технологии, проблемы, особенности, усовершенствования и т.д. Бэклог продукта также включает задачи, важные для команды, например «провести тренинг», «добить всем памяти».

Бэклог спринта содержит функциональность, выбранную ведущим проекта из бэклога продукта. Все функции разбиты по задачам, каждая из которых оценивается командой. Каждый день команда оценивает объем работы, который нужно проделать для завершения задач. Сумма оценок оставшейся работы может быть построена как график зависимости от времени. Такой график демонстрирует прогресс команды по ходу спринта.

Жизненный цикл спринта состоит из планирования спринта, непосредственной работы над задачами и демонстрации продукта заказчику.

Во время планирования участники проекта определяют цель спринта и формируют бэклог спринта, то есть функциональность, которая будет разработана в течение следующего спринта для достижения цели. Затем команда вместе со скрам мастером обсуждает средства достижения поставленных задач и оценивают примерное количество времени, требуемое для выполнения каждого элемента бэклога спринта.

При работе над продуктом ежедневно устраиваются непродолжительные митинги в начале рабочего дня. Их целью является поделиться информацией. Они предназначены для того, чтобы все члены команды знали, кто и чем занимается в проекте. Длительность таких митингов строго ограничена и не должна превышать 15 минут.

В конце спринта команда демонстрирует новую версию продукта, созданную за последний спринт. Ведущий проекта, менеджмент, заказчики, пользователи, в свою очередь, его оценивают. Команда рассказывает о поставленных задачах, о том как они были решены, какие препятствия были у них на пути, какие были приняты решения, какие проблемы остались нерешенными. На основании демонстрации принимающая сторона может сделать выводы о том, как должна дальше развиваться система. Участники митинга делают выводы о том, как шел процесс в команде и предлагает решения по его улучшению. Скрам Мастер отвечает за организацию и проведение этого митинга. Команда помогает ему распланировать, кто и в какой последовательности что представляет.

### 5.1.3 Методология Kanban

Методология Канбан (Kanban) является самой гибкой из методологий, представленных на обзор в данной дипломной работе. Основным принципом Канбана является минимизация количества выполняемой в данный момент времени работы. Достигается эта цель с помощью визуализации задач проекта посредством карточек, каждая из которых интерпретирует ту или иную задачу. Такие карточки располагаются на Канбан-доске, которая наглядно демонстрирует количество задач в проекте и их текущее состояние.

Канбан-доска разбита на столбцы, определяющие состояния задач. В каждый столбец одновременно может быть помещено ограниченное количество задач-карточек. Это количество вычисляется экспериментально, в зависимости от количества людей в команде определенного профиля или их опыта, типа задачи и так далее. Задачей ведущего проекта является оптимизировать максимальное количество задач, находящихся в конкретном состоянии, для минимизации количества выполняемой работы на данный момент времени – устранение простоев и задержек в рабочем процессе.

Для разработки конкретного программного обеспечения канбан-доска может иметь уникальный набор столбцов. Приведём один из вариантов структуры такой доски со значением каждого отдельного столбца.

**Цели проекта:**

В этом столбце содержатся карточки с целями проекта. Этот столбец является статичным, то есть карточки из него не перемещаются в процессе работы над проектом. Назначение этого столбца несёт мотивационный характер.

**Очередь задач:**

Этот столбец заполняется карточками-задачами, которые необходимо выполнить. Приоритет задач градируется сверху вниз, то есть вверху столбца располагается задача с наивысшим приоритетом. Приоритет задач также определяется ведущим проекта. Для того, чтобы приступить к выполнению задачи, участник команды должен переместить верхнюю карточку в следующий столбец-состояние.

**Проработка дизайна:**

В этом состоянии производится разработка дизайна интерфейса или архитектура программного кода.

**Разработка:**

В этом столбце задача находится во время разработки описанной в ней программной части. Как только разработка завершена, задача переходит в следующий столбец. Если в процессе разработки выясняется, что архитектура не верна или не точна, то задача перемещается в предыдущий столбец.

**Тестирование:**

В этом состоянии производится тестирование разработки. В случае если при тестировании были обнаружены ошибки, задача перемещается обратно в состояние «разработка».

**Развертывание:**

Под развертыванием понимается набор действий, приводящих разработанный и оттестированный код в состояние, позволяющее дальнейшее использование этой части программы. Например, фиксирование изменений в репозитории проекта или установка новой версии программы на сервере.

**Закончено:**

Сюда задача попадает, как только завершены все предыдущие работы по этой задаче.

**Неотложно:**

Для каждого состояния можно выделить специальный, наивысший приоритет для задач, которые необходимо выполнить срочно. В таком приоритете может находиться не больше одной задачи для каждого состояния проекта.

На карточке-задании помечается дата, когда она попала в очередь задач, потом дата, когда ее взяли в работу и дата, когда ее завершили. По этим трем точкам для нескольких задач можно посчитать среднее время ожидания в очередь задач и среднее время выполнения задачи. На основе этих данных можно выполнить оптимизацию рабочего процесса – реструктурировать команду, изменить количество максимальных задач в состоянии, пересмотреть приоритет задач и так далее.

Всю методологию Канбан можно описать всего тремя основными правилами:

* 1. Визуализировать производство
  2. Ограничивать количество работы, выполняемой одновременно на каждом этапе производства.
  3. Измерять время цикла (среднее время на выполнение одной задачи) и оптимизировать процесс, чтобы уменьшить это время.

Главными отличиями Канбан от Скрам являются:

* Отсутствие лимитов времени ни на задачи, ни на спринты
* Задачи больше и их меньше
* Оценки сроков на задачу опциональные или вообще их нет
* «Скорость работы команды» отсутствует и считается только среднее время на полную реализацию задачи

Также в методологии Канбан не существует собраний, на которых обсуждаются цели и средства достижения задач, и ежедневных митингов. Всё это время для поддержания рабочего процесса в Скрам отведено на непосредственную работу в Канбан.

**Выбор методологии разработки программного обеспечения:**

Экстремальное программирование предполагает минимизацию ошибок на раннем этапе разработки программного продукта и, как следствие, быстрый выпуск продукта. При решении поставленной задачи скорость не является доминирующим фактором, а недостаток опыта разработчика не позволяет использовать все преимущества метода экстремального программирования.

Технология Канбан очень удобна для выполнения самостоятельного проекта, а также хорошо адаптируется при изменениях требований к проекту в течение рабочего процесса. Канбан не является итеративным методом, и успех продукта сильно зависит от качества формулировки задач и выбора их приоритета.

Методология Скрам является самым сбалансированным вариантом из предложенных на обзор. Итеративность метода поможет разработчику сконцентрироваться на конкретной части программы, а не на всей системе в целом. Митинги помогут осознать задачи текущего этапа и выбрать средства разработки. Ежедневные собрания помогут скрам мастеру заблаговременно узнать о возникших трудностях у разработчика и помочь быстро их устранить. Это очень важно, когда речь идёт о недостатке опыта разработки программного обеспечения у программиста.

## 5.2 Выбор используемого программного обеспечения.

### 5.2.1 MySQL-сервер

MySQL – это свободная система управления базами данных разработанная, распространяемая и поддерживаемая корпорацией Oracle. MySQL работает с реляционными базами данных. Информация в реляционных базах данных хранится в отдельных таблицах, а не в одном большом хранилище данных. Структуры таких баз данных организованы внутри физических файлов, что является оптимизацией работы системы. Логическая модель с такими сущностями как базы данных, таблицы, виды, кортежи, атрибуты, предлагает пользователю гибкую среду программирования. Пользователь сам настраивает условия, с помощью которых управляет отношениями между разными атрибутами данных, такими как «одно к одному», «одно ко многим», «уникальное», «обязательное», «необязательное», а также указатели между разными таблицами. База данных выполняет эти условия, таким образом, при правильно сформированной базе данных, использующее её приложение никогда не столкнётся с противоречивой, дублирующейся, неактуальной, потерянной или пропавшей информацией.

Система управления база MySQL использует язык SQL (Structured Query Language – Структурированный язык запросов). SQL является наиболее распространённым стандартизированным языком, используемым для работы с базами данных. Для работы с MySQL можно вводить запросы на SQL напрямую, вставлять сформулированные тексты запросов в текст кода, написанного на любом другом языке или использовать различные библиотеки функций для работы с MySQL, чтобы не обращаться к синтаксису SQL.

MySQL является программным обеспечением, распространяемым с открытым программным кодом. Это означает, что такое программное обеспечение можно изменять и модифицировать для своих целей. Также MySQL распространяется бесплатно и доступна для скачивания через Интернет.

MySQL сервер отлично работает как на настольных компьютерах, так и на ноутбуках, не затрудняя работу других приложений или веб-серверов, а также не требует особого обслуживания. Если под работу СУБД выделяется отдельный компьютер, то программное обеспечение можно настроить таким образом, чтобы вся память и ресурсы процессора выделялись для нужд MySQL. Это актуально в том случае, когда с базой данных работает большое количество пользователей одновременно. MySQL сервер был специально разработан для быстрой работы с большими базами данных в сравнении с предоставленными на тот момент решениями. На сегодняшний день MySQL обладает широким набором функций. Простота подключения, скорость работы и высокий уровень безопасности делают эту СУБД максимально пригодной для решения любых задач, связанных с доступом и работой с базами данных.

Система управления базами данных MySQL имеет клиент-серверную архитектуру и состоит из многопотокового SQL сервера, который поддерживает различные модификации, разнообразных клиентов и библиотек, инструментов администрирования и широкого спектра интерфейсов программирования приложений (APIs).

### 5.2.2 MySQL Workbench

MySQL Workbench – инструмент для визуального проектирования баз данных, интегрирующий проектирование, моделирование, создание и эксплуатацию БД в единое окружение для системы баз данных MySQL. MySQL Workbench предоставляет комплекс инструментов для настройки сервера, администрирования пользователей и многое другое. Программа доступна для работы с операционных системах: Windows, Linux и Mac OS.

MySQL Workbench позволяет администратору или проектировщику баз данных визуально моделировать, создавать и управлять базам данных. Программное обеспечение обладает всем необходимым для создания комплексной ER-моделей, прямой и обратной разработки, а также позволяет легко произвести сложные изменения в базе данных или исправления в документации, которые обычно занимают много времени и усилий.

Программа позволяет визуально создавать, выполнять и оптимизировать SQL-запросы. SQL редактор поддерживает синтаксическую подсветку и историю выполненных SQL-запросов. Панель подключений к базам данных даёт возможность легко переключаться между разными базами данных, что позволяет работать с несколькими БД одновременно. Просмотрщик объектов обеспечивает быстрый доступ к таблицам баз данных и объектам этих таблиц.

MySQL Workbench упрощает разработку и поддержку баз данных, автоматизирует выполнение наиболее долгих и сложных задач и улучшает взаимодействие между разработчиками и администраторами баз данных. Программа позволяет проектировщикам данных наглядно предоставить требования и, связавшись с коллегами, быстро решить проблему до того как будет потрачено большое количество рабочих ресурсов и времени. С помощью MySQL Workbench можно легко создавать надёжные, хорошо структурированные базы данных и в то же время достаточно гибкие, для того чтобы изменяться и улучшаться, отвечая новым требованиям бизнес задач. Утилиты для проверки моделей данных и структур таблиц обеспечивают высокую надёжность при разработке. Это избавляет разработчика от ошибок по время моделирования новой ER-диаграммы или создания физической MySQL базы данных.

**Главные преимущества программы:**

* Позволяет наглядно представить модель базы данных в графическом виде;
* Наглядный и функциональный механизм установки связей между таблицами, в том числе «многие ко многим» с созданием таблицы связей;
* Reverse Engineering – восстановление структуры таблиц из уже существующей на сервере БД;
* Удобный редактор SQL запросов, позволяющий сразу же отправлять их серверу и получать ответ в виде таблицы;
* Возможность редактирования данных в таблице в визуальном режиме.

### 5.2.3 SQLyog Community

SQLyog – это графический интерфейс пользователя для популярной системы реляционных баз данных MySQL. Программа создана компанией Webyog Softworks Pvt. Ltd. Основная задача, которую решает SQLyog – это удобная работа с MySQL базами данных посредством наглядных графических элементов.

Главные функции SQLyog:

* Конструктор запросов
* Умное автозавершение
* Интеллектуальное дополнение кода
* Туннелирование HTTP и HTTPS
* Туннелирование SSH
* Соединения
* Инструмент миграции в виде wizard.
* Синхронизация Структуры/Данных
* Полноценная поддержка Юникода.

Конструктор SQL-запросов позволяет избегать SQL синтаксиса при составлении запросов. Это значительно расширяет круг пользователей, которые могут работать с данным программным обеспечением.

Благодаря автозавершению количество ошибок при составлении запросов существенно снижается, а скорость набора текста запроса сильно вырастает. В сложных структурах баз данных обычно содержится большое количество таблиц. Поэтому для разграничения назначения таблиц их именуют таким образом, чтобы из названия можно было однозначно определить функцию таблицы. Из-за этого длина имён таблиц иногда достигает свыше 50 символов, например: «ext\_st\_lnk\_heat\_sources\_asot\_expert\_history\_changelog». При составлении запроса имя такой таблицы довольно долго набирать, а ошибиться очень просто. Автозавершеие решает эту проблему.

Функция туннелирования обеспечивает безопасную работу с данными. Благодаря механизмам туннелирования даже в случае похищения злоумышленником передаваемых данных, конфедициальность этих данных останется в силе, из-за шифрования передаваемых пакетов.

SQLyog поддерживает работу с несколькими соединениями одновременно. Эта функция позволяет работать с базами данных, которые физические располагаются в разных местах. Благодаря этому возможна реализация синхронизации баз данных и поддержка их в актуальном состоянии, вне зависимости от того, где эти базы данных хранятся.

### 5.2.4 Borland Delphi 7

Borland Delphi 7 – это интегрированная среда разработки программного обеспечения для операционной системы Microsoft Windows на языке Delphi.

Интерфейс этой среды представляет собой несколько отдельных окон. Каждое из окон выполняет свою функцию. Здесь есть: панель инструментов, редактор программного кода, просмотрщик объектов, редактор графического интерфейса приложения, окно стека вызовов и так далее. Каждое окно может иметь произвольный размер и располагаться в любом удобном для разработчика месте. Delphi 7 позволяет сохранить такое расположение окон для того, чтобы использовать такую рабочую поверхность в дальнейшем. Эту функцию среды очень удобно использовать для работы в двух наиболее часто используемых режимах: разработка и отладка. При разработке нужны такие окна как: редактор кода, редактор графического интерфейса, просмотрщик объектов. Во время отладки обычно редактируется код и просматривается стек вызовов, а остальные окна можно скрыть. Функция рабочих поверхностей позволяет сделать рабочий процесс максимально комфортным в зависимости от режима, в котором работает разработчик. Рассмотрим функциональность основных окон среды подробнее.

**Панель инструментов**

На панели инструментов находятся пиктограммы для доступа к основным функциям среды разработки: открыть/сохранить проект, запустить приложение, добавить или удалить файл из проекта и так далее. Особо важны кнопки «войти в функцию» и «пройти далее», которые позволяют перемещаться по коду программы по мере его выполнения в режиме отладки. В этом окне также производится выбор или сохранение текущей рабочей поверхности, о которых было сказано выше.

Но самым необходимым элементом на панели инструментов является набор компонентов. Набор компонентов содержит список установленных в среде компонентов, объекты которого можно использовать в разрабатываемом приложении. Delphi 7 предоставляет довольно большой спектр стандартных компонентов, но при желании можно установить дополнительные. Компонент Delphi представляет собой библиотеку классов. Классы, описанные в компоненте, могут являться графическими элементами, которые вы можете использовать в своём приложении, просто перетащив иконку элемента к себе на редактор графического интерфейса. Также там могут содержаться запрограммированные алгоритмы, которые можно использовать для реализации определённой части вашего приложения, например: поиск кратчайшего пути. В сети Интернет есть множество компонентов, распространяющихся как на платной, так и на бесплатной основе. Это предоставляет широкий инструментарий для разработчиков и позволяет сократить большое количество время на программировании уже готовых решений.

**Редактор кода**

Редактор кода является, по сути, текстовым редактором с множеством дополнительных функций. Редактор кода отображает текущее положение курсора в тексте кода – номер строки и столбца. Это очень удобно, особенно когда размер кода одного файла превышает тысячу строк. Индексация положения курсора сильно облегчает навигацию разработчика по коду программы. Редактор кода в Delphi 7 поддерживает контекстную подсветку. Это означает, что текст будет принимать определённое форматирование в зависимости от своего назначения. Например, служебные лексемы выделяются жирным, а комментарии окрашиваются в синий цвет и приобретают атрибут курсива. Форматирование текста можно произвольно менять. Можно окрашивать не только цвет шрифта, но и его фона. Также в данной среде поддерживается система шаблонов текста. Разработчик может предварительно установить шаблон для определённого заменяемого значения и по нажатию определённого сочетания клавиш описанный шаблон вставится в текст кода. Эта функция помогает поддерживать единый стиль кода приложения, а также сократить количество времени на наборе распространённых конструкций циклов, условий, заголовка функций и процедур и так далее. В Delphi 7 есть функция перехода от объявления функции в классе к её описанию в коде, а также история перемещения по тексту, которая позволит возвращаться в предыдущие места редактирования кода программы.

**Редактор графического интерфейса и просмотрщик объектов**

Редактор графического интерфейса пользователя позволяет разработчику визуально форматировать пользовательский интерфейс. Это даёт возможность просто и быстро корректировать элементы графического интерфейса, поскольку нет необходимости запускать приложение, чтобы оценить результат работы. Благодаря этой форме, разработчик сразу может увидеть, как поведёт себя интерфейс при расширении или сужении окна, и внести необходимые изменения.

В просмотрщике объектов отображаются все атрибуты выбранного объекта. Выбор можно произвести из выпадающего списка, который расположен в самом просмотрщике, или, кликнув на нужном объекте в окне редактора графического интерфейса. При изменении атрибута объекта результат тут же отобразится в редакторе интерфейса, в том случае если это возможно. Также с помощью просмотрщика объектов можно присваивать функции обработки событий для описанных в этом классе событий.

Borland Delphi 7 предоставляет комфортную и многофункциональную среду для разработки программного обеспечения. Также эта среда поддерживает внедрение различных дополнений, которые смогут сделать рабочий процесс ещё более эффективным.

### 5.2.5 WinCVS

WinCVS является системой контроля версий, разработанной специально для операционной системы Microsoft Windows. WinCVS, как и другие CVS системы, предназначена для ведения изменяющегося во времени проекта, хранения файлов предыдущих версий и обеспечения системы синхронизации файловой информации между некоторым количеством пользователей. CVS используют обычно в рамках разработки программного обеспечения. Две главные задачи, решаемые CVS:

* Хранение истории изменения файла для возможности отката к предыдущей версии;
* Облегчение кооперативной работы участников проекта в процессе изменения данных файлов.

Наиважнейшие задачи, которые выполняют CVS системы – это синхронизация данных проекта, то есть обновление файлов до последней версии, а также внесение изменений файлов в репозиторий. При совершении этих операций CVS сервер сравнивает отличия файла клиента и файла, хранящегося на сервере. Если изменения файлов не пересекаются, то операция выполняется, в противном случае сервер оповестит клиент о конфликте версий файлов и изменения придётся отслеживать вручную. Это и есть главное отличие WinCVS от многих других CVS систем. Обычно при работе пользователя с определённым файлом этот файл блокируется, и работа с ним невозможна другим пользователям до тех пор, пока его изменения не будут внесены, и работа с ним, тем самым, будет прекращена. Это сильно затрудняет совместную работу команды над одним файлом и тормозит процесс разработки. В WinCVS с одним файлом может работать неограниченное число пользователей при условии, что каждый из участников проекта будет редактировать разную часть файла.

Для компрессии хранимых данных CVS используют механизм дельта-компрессии. Дельта-кодирование (англ. Delta encoding) – способ представления данных в виде разницы (дельты) между последовательными данными вместо самих данных. Этот способ довольно эффективен для хранения текстовых данных, но недостаточно хорошо для хранения бинарных файлов.

CVS также поддерживает систему различных веток проекта. Обычно отлаженную и стабильную версию проекта хранят на одной ветке. Её обновляют только для исправления ошибок в текущей версии. Для активных разработок, внедрения новой функциональности, значительных улучшений создают параллельную ветвь и работают на ней. Это позволяет быстро развивать проект, не опасаясь за потерю стабильности в разрабатываемом программном продукте.

# 6. Охрана труда

## 6.1 Основные положения

Охрана труда – система сохранения жизни и здоровья работников в процессе трудовой деятельности, включающая в себя следующие мероприятия [30]:

* правовые
* организационно-технические
* социально-экономические
* лечебно-профилактические
* санитарно-гигиенические
* реабилитационные

Система управления охраной труда (СУОТ) – часть общей системы управления организации, обеспечивающая управление рисками в области охраны здоровья и безопасности труда, связанными с деятельностью организации (ГОСТ 12.0.230-2007).

Система включает:

* организационную структуру;
* распределение ответственности;
* деятельность по планированию;
* процессы и ресурсы для разработки, внедрения, достижения целей, анализа результативности политики и мероприятий по охране труда в организации.

Основу нормативно-правовой базы создания и функционирования СУОТ организации составляют федеральные законы (ФЗ), в том числе "Об обязательном социальном страховании от несчастных случаев на производстве и профессиональных заболеваний", "О промышленной безопасности опасных производственных объектов" и др., а также ТК РФ, постановления Правительства РФ по вопросам охраны труда, нормативные правовые акты и нормативно-технические документы федеральных органов исполнительной власти и субъектов РФ в соответствии с их компетенцией.

## 6.2 Безопасность труда при работе с персональным компьютером

В соответствии с Санитарными правилами СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам (ПЭВМ) и организации работы» осуществляется за производством и эксплуатацией ПЭВМ.

Используемый на предприятии монитор ПЭВМ обеспечивает поворот корпуса в горизонтальной и вертикальной плоскости с фиксацией в заданном положении, а так же регулировку яркости и контрастности. Дизайн монитора ПЭВМ, а также подключённых периферийных устройств предусматривают окраску корпуса в спокойные и мягкие тона с рассеиванием света. Корпус, клавиатура и другие блоки имеет матовую поверхность. Помещения для эксплуатации ПЭВМ имеют естественное и искусственное освещение, соответствующее требованиям нормативной документации.

Оконные проемы снабжены устройствами типа жалюзи, для регулировки интенсивности естественного освещения в помещении.

Помещения оборудованы защитным заземлением (занулением) в соответствии с техническими требованиями по эксплуатации.

Шумящее оборудование (печатающие устройства, серверы и т.п.), уровни шума которого превышают нормативные, размещены вне помещений с ПЭВМ. Расстояния между рабочими столами с видеомониторами (в направлении тыла поверхности одного видеомонитора и экрана другого видеомонитора), составляет 2,5 метра. Расстояние между боковыми поверхностями видеомониторов – 1,2 м. Использованы рабочие столы, отвечают современным требованиям эргономики. Рабочий стул (кресло) оборудовано подъемно-поворотным устройством, регулируемым по высоте и углам наклона сидения и спинки, а также расстоянию спинки от переднего края сидения. Экран видеомонитора находится от глаз пользователя на расстоянии 670 мм, клавиатура располагается на поверхности стола на расстоянии 200 мм от края, обращенного к пользователю. Клавиатура оборудована специальной накладкой для ладоней, для снижения изгиба запястья и снижения напряжения рук при работе за ПЭВМ.

В помещениях оборудованных ПЭВМ, проводятся:

* ежедневная влажная уборка;
* систематическое проветривание после каждого часа работы на ПЭВМ;
* чистка стекол оконных рам и светильников не реже двух раз в год.

## 6.3 Микроклимат в рабочей зоне

Независимо от состояния природных метеорологических условий в производственных помещениях и на рабочих местах должны быть созданы климатические условия, безопасные для человека и наиболее благоприятные для выполнения работы.

Под микроклиматом производственных помещений понимаются метеорологические условия внутренней среды помещений, которые определяются действующими на организм человека сочетаниями температуры, влажности, скорости движения воздуха, а также температурой окружающих поверхностей (ГОСТ 12.1.005-88 ССБТ). Показатели микроклимата должны обеспечивать сохранение теплового баланса человека с окружающей средой и поддержание оптимального или допустимого теплового состояния организма. Повышенная влажность затрудняет теплоотдачу организма путем испарений при высокой температуре воздуха и способствует перегреву, а при низкой температуре, наоборот, усиливает теплоотдачу, способствуя переохлаждению. Оптимальны такие параметры микроклимата, которые при длительном и систематическом воздействии на человека обеспечивают сохранение нормального функционального и теплового состояния организма без напряжения реакций терморегуляции, что создает ощущение теплового комфорта и служит предпосылкой для высокой работоспособности.

Оценка микроклимата проводится на основе измерений его параметров на всех местах пребывания работника в течение смены и сопоставления с нормативами СанПиН 2.2.4.548-96 по показателям:

* температура;
* влажность воздуха;
* скорость движения воздуха;
* тепловое излучение.

Все показатели микроклимата должны обеспечивать сохранение теплового баланса человека с окружающей средой и поддержание оптимального и допустимого теплового состояния организма.

Оптимальные микроклиматические условия – обеспечивают общее и локальное ощущение теплового комфорта в течение 8-часовой рабочей смены, не вызывают отклонений в состоянии здоровья, создают предпосылки для высокого уровня работоспособности и являются предпочтительными на рабочих местах.

Согласно СанПиН 2.2.4.548-96 оптимальные величины показателей микроклимата на рабочих местах производственных помещений для категорий работ по уровню энергозатрат более 290 Вт/час, определяются для холодного и теплого периода времени года соответствующе:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Период года | Категория работ по уровню энергозатрат, Вт | Темпе-ратура воздуха, °С | Темпе­ратура поверх-ностей,  °С | Относительная влажность воздуха, % | Скорость движения воздуха, м/с |
| Холодный | более 290 | 16-18 | 15-19 | 60-40 | 0,3 |
| Теплый | более 290 | 18-20 | 17-21 | 60-40 | 0,3 |

Данные показатели микроклимата поддерживаются, благодаря системе кондиционирования состоящей из 2х настенных сплит-систем, общей мощностью 5.2 КВт. Она имеет дезодорирующий фильтр и обладает возможностью регулировки направления воздушного потока. Уровень шума внутреннего блока – 38 Дб, что соответствует негромкому разговору и не протеворчит ГОСТ 12.1.036-81 (2001) («Шум. Допустимые уровни в жилых и общественных зданиях»).

## 6.4 Расчет защитного зануления

Зануление – преднамеренное электрическое соединение с нулевым защитным проводником металлических нетоковедущих частей, которые могут оказаться под напряжением. Нулевой защитный проводник – проводник, соединяющий зануляемые части с нейтральной точкой обмотки источника тока или ее эквивалентом. Зануление применяется в сетях напряжением до 1000 В с заземленной нейтралью. В случае пробоя фазы на металлический корпус электрооборудования возникает однофазное короткое замыкание, что приводит к быстрому срабатыванию защиты и тем самым автоматическому отключению поврежденной установки от питающей сети. Такой защитой являются:

* плавкие предохранители
* максимальные автоматы, установленные для защиты от токов коротких замыканий;
* автоматы с комбинированными расцепителями.

Данная дипломная работа производилась на ПЭВМ с предустановленным блоком питания, мощность которого составляет 550 Вт. Для защиты от короткого замыкания в блоке установлен тугоплавкий предохранитель номиналом 25 А. Для обеспечения безопасности на предприятии установлен распределительный щит с защитным занулением. Используется электрическая сеть с рабочим напряжением 230 В с глухозаземленной нейтралью.

Требуется проверить отключающую способность зануления в сети, при нулевом защитном проводнике, состоящем из стальной полосы сечением 30x4 мм. От трансформатора, мощностью 100 кВт, до рабочего места проложена электролиния с медными проводами 3х6 мм2.

Решение данной проблемы сводится к проверке следующего условия:

, (6.1)



где – ток однофазного короткого замыкания, проходящий по петле фаза-нуль;



– наименьший допустимый ток по условию срабатывания защиты (предохранителя);



- номинальный ток плавкой вставки предохранителя.



- коэффициент кратности номинального тока . Так как установка защищается тугоплавким предохранителем, время которого зависит от тока, то в целях ускорения отключения принимают .



Выполнение этого условия обеспечит надежное срабатывание защиты при коротком замыкании (КЗ) фазы на зануленный корпус электродвигателя, т.е. при соединении нулевого защитного проводника с глухозаземленной нейтральной точкой трансформатора.

Определяем наименьшие допустимые значения токов для двигателя:

А; (6.2)



Находим полное сопротивление трансформатора:

Ом, согласно таблице 6.1 из [13]



Определяем на участке м или 0,15 км, активное и индуктивное сопротивления фазного провода; активное и индуктивное сопротивления нулевого защитного провода и внешнее индуктивное сопротивление петли фаза-нуль:



Согласно паспортным данным кабеля марки АПВ 3х6:

Ом/км



Ом/км



Ом,



Ом



Принимаем Ом



Находим ожидаемую плотность тока в нулевом защитном проводе – стальной полосе сечением

мм2;



А/мм2



По [13, табл. 6.2] для А/мм2 и мм2 находим:



Ом/км – активное сопротивление 1 км стального провода,



Ом/км – внутреннее индуктивное сопротивление 1 км стального провода.



Далее находим и для м км:



Ом; Ом



Определяем для м км:



Ом



Ом/км – внешнее индуктивное сопротивление 1 км петли фаза-нуль, величина которого принята по рекомендации [13].



Находим действительные значения токов однофазного короткого замыкания, проходящих по петле фаза-нуль по формуле 6.8 [13]:

А



А



Вывод: Действительные значения токов однофазного короткого замыкания А превышают соответствующие наименьшие допустимые по условиям срабатывания защиты токи A нулевой защищенный провод выбран правильно, то есть отключающая способность системы зануления обеспечена. Электробезопасность при работе с ПЭВМ на предприятии обеспечена.



# 7. Экологическая часть

## 7.1 Введение

Важнейшим элементом экологии в программном обеспечении является эргономичность интерфейса.

Эргономика – наука о приспособлении должностных обязанностей, рабочих мест, предметов и объектов труда, а также компьютерных программ для наиболее безопасного и эффективного труда работника, исходя из физических и психических особенностей человеческого организма. Эргономичность интерфейса является залогом удобной, продуктивной и эффективной работы пользователя в программе. Хорошая эргономика интерфейса также не должна отрицательно влиять на физическое и психологическое состояние пользователя, что положительно отразится на результате и скорости работы человека.

К основным принципам проектирования качественных интерфейсов можно отнести:

* Естественность – взаимодействие с интерфейсом не должно вызывать у пользователя осложнения
* Непротиворечивость – взаимодействие должно соответствовать привычным для работы нормам
* Неизбыточность – пользователь должен вводить минимальную необходимую информацию
* Легкий доступ к системе помощи – справки и документация должна быть понятно интегрирована в интерфейс
* Гибкость – интерфейс должен быть рассчитан на пользователей с различными уровнями подготовки

## 7.2 Разработка эргономичного интерфейса

Одним из наиважнейших этапов проектирования интерфейса является выбор цветовой гаммы графических элементов интерфейса пользователя. Яркие и вызывающие цвета будут затруднять восприятие информации пользователем, и отвлекать его от рабочего процесса в целом. Такой выбор цветов будет способствовать снижению работоспособности человека и быстрее вызовет усталость и раздражение. Эти факторы напрямую повлияют на результат работы и увеличат частоту допущения ошибок пользователем. Выбор спокойных и нейтральных цветов поможет сконцентрироваться на работе и продлит высокую работоспособность пользователя за счёт снижения зрительного напряжения. Допускается цветовое группирование некоторых логических элементов интерфейса в одну подгруппу. Разграничение элементов с помощью цвета увеличит скорость ознакомления пользователя с интерфейсом. Визуально разбитая на группы информация поможет избежать необходимости запоминать, какая часть графического интерфейса за что отвечает. В этом случае также важно соблюдать выбор цветовой палитры.

Зрительное напряжение можно снизить с помощью правильной работы со шрифтами, которые используются в графическом интерфейсе пользователя. Мелкие и нечитабельные шрифты вызовут трудности в восприятии информации пользователем. Изучение всех надписей в процессе поиска необходимой пользователю функции отнимает большое количество рабочего времени, а также вынуждает сильно напрягать зрение. Ввиду этого, очень важно подбирать удобно читаемые шрифты и выбирать максимально допустимый интерфейсом размер шрифта. Не рекомендуется набирать надписи исключительно заглавными буквами, так как восприятие информации, набранной таким способом, значительно затрудняется («СОХРАНИТЬ В БАЗУ ДАННЫХ»). Для выделения надписи и привлечения внимания пользователя лучше воспользоваться присвоением шрифту атрибута «жирный» или изменить стандартный цвет шрифта, но таким образом, чтобы текст легко читался на собственном фоне («**Сохранить в базу данных**», «Сохранить в базу данных»).

При проектировании графического интерфейса пользователя важно заранее спланировать, какими функциями человек будет пользоваться чаще остальных. Те элементы интерфейса, с которыми будут взаимодействовать большую часть рабочего времени, стоит сделать крупнее, чем менее значительные, и расположить их настолько близко друг к другу, насколько это допустимо. Такой подход к проектированию интерфейса позволит сократить механическую работу пользователя по перемещению контроллера – мыши. Благодаря этому достигается снижение напряжения руки пользователя.

Функциональность элементов интерфейса должна быть максимальной. По возможности стоит объединить несколько операций программы в одну функцию, которая будет вызывать посредством нажатия одной кнопки. Сокращение кликов по управляющим элементам интерфейса также сокращает механическую работу, выполняемую пользователем, и увеличивает скорость работы в программе.

По возможности следует избавляться от большого количества форм (окон) при проектировке интерфейса. Чем больше форм, тем сложнее воспринимать информацию и определить принадлежность функций окна к функциональности программы в целом. Дополнительные возможности стоит встраивать в уже существующие формы, избегая создания новых. На сегодняшний день технологии программирования предоставляют богатый инструментарий для решения данной проблемы. Такие элементы как: выпадающие списки, всплывающие окна (формы внутри форм), «аккордеоны» позволяют добавить функциональность в рамках одной формы, но информируют пользователя в момент необходимости пользователем данных функций.

В интерфейсе программы обязательно стоит предусмотреть так называемую «защиту от дурака» (анг. «mistake-proofing»). Защита от дурака – защита предметов пользования (в особенности, техники), программного обеспечения от очевидно неверных действий человека, как при пользовании, так и при техническом обслуживании или изготовлении. Как известно, людям свойственно ошибаться и это стоит учитывать при проектировании интерфейса. Программа должна обладать различными системами проверки вводимых пользователем данных. В случае ввода некорректных данных об этом обязательно необходимо уведомлять пользователя, чтобы объяснить, в чём заключается ошибка. Эта проверка защитит вашу систему от неоперабельных данных, а пользователь будет работать внимательнее.

В случае если в интерфейсе предусмотрены управляющие элементы, которые выполняют функцию, приводящую к необратимым или тяжело обратимым последствиям, при работе с таким элементом стоит выводить диалоговое окно с вопросом о подтверждении действий пользователя. Диалоговое окно предотвратит выполнение функции в случае непреднамеренного нажатия на элемент графического интерфейса. Это избавит от необходимости совершать откат данных, что может являться довольно длительной и трудоёмкой процедурой.

При использовании диалоговых окон следует придерживаться всех правил, описанных выше. Также стоит уделить особое внимание формулировке текста диалогового окна. Нечёткая и непонятная формулировка текста диалога может привести к неправильному толкованию информации пользователем и ввести его в заблуждение. В таком случае пользователь может выполнить заведомо неверное действие, что также негативно отразится на результате работы. Пользователь может обратиться к справке или в техническую поддержку. И то, и другое отнимет определённое количество времени и заметно снизит скорость работы.

При проектировании интерфейса также стоит добавить функцию обработки «горячих клавиш». Сочетание клавиш (горячая клавиша) – разновидность интерфейса взаимодействия с вычислительной системой, представляющая собой нажатие клавиши (или сочетания клавиш) на клавиатуре, которому назначено некое действие – команды, исполняемые данной системой. Как правило, частично дублирует интерфейс меню или кнопок и служит для значительного ускорения работы, также – увеличения количества возможных действий, выполняемых с помощью клавиатуры.

Добавление обработки таких клавиш как: «escape», «enter», «delete» значительно увеличит производительность работы пользователя в системе. Как правило, значения этих клавиш в программном обеспечении унифицировано и воспринимается пользователем на интуитивном уровне. «Escape» – отклонить, «enter» – принять, «delete» – удалить. Если в интерфейсе предусмотрены функции, к которым пользователь будет часто обращаться, рекомендуется добавить горячую клавишу для её обработки. Опытные пользователи смогут сократить время работы с программой, а старая система выполнения функций с помощью нажатия на кнопку будет поддерживаться.

## 7.3 Вывод:

Эргономика графического интерфейса пользователя является одним из ключевых факторов результативной и качественной работы пользователей в системе. Хорошо разработанный интерфейс предоставляет пользователям комфортные условия для работы и расширяет круг людей, которые могут пользоваться таким программным обеспечением. Это особенно важно, поскольку даже на сегодняшний день есть много людей, являющихся неуверенными пользователями ПЭВМ. Качественная эргономика, простота и наглядность интерфейса позволяют работать с ним таким людям. Это отчасти решает кадровый вопрос. Соблюдение норм проектировки интерфейса программы позволяет снизить риск ошибок пользователя, увеличить его работоспособность и способствует меньшему расходу сил пользователя при работе с таким графическим интерфейсом. Это выгодно как для пользователя, так и для работодателя.

# 8. Решение задачи на ЭВМ

## 8.1 Структура классов модуля

Разработанный модуль настройки схем маршрутизации запросов состоит из трех отдельных элементов: ядро, интерпретатор метаязыка и графический интерфейс пользователя. Ядро делится на две части:

* Программный интерфейс настройки схемы маршрутизации
* Маршрутизация запросов пользователям, обработка и отклонение запроса

Интерпретатор метаязыка служит для реализации правил остановки в состояниях схемы маршрутизации. Эта часть отвечает за создание правил, их корректной записи в базу данных, а также проверку их выполнения для конкретного запроса.

Графический интерфейс пользователя является связующим звеном между администратором системы и программным интерфейсом настройки схемы маршрутизации. Именно с помощью пользовательского интерфейса производится настройка схемы и описания правил остановки в том числе.

Рассмотрим функции основных классов модуля подробнее.

TtrsAdmin – программный интерфейс настройки схемы маршрутизации. Одной из самых важных функций класса является загрузка всех сущностей схемы маршрутизации, а также вспомогательных сущностей в собственное хранилище данных. Вся информация загружается из базы данных и работа с ними происходит локально на компьютере пользователя. Такой подход был применён для оптимизации скорости работы с данными. Поиск конкретного объекта в локальном кэше занимает меньше времени, чем запрос к базе данных, так как с базой данных одновременно может работать большое количество пользователей. Загружать MySQL-сервер дополнительными запросами каждый раз при обновлении формы не имеет смысла. Функция загрузки сущностей выполняется всего единожды – при инициализации, в дальнейшем все операции по изменению сущностей дублируются: сначала происходит попытка записи в базу данных и, в случае успеха, в кэше также происходит обновление.

В этом классе содержатся все необходимые функции для получения и изменения данных всех основных и вспомогательных сущностей схемы маршрутизации запросов, а именно:

* Физических таблиц
* Справочников
* Типов ролей
* Схем маршрутизации
* Бизнес-процессов
* Наборов полей
* Ролей
* Связей роли-пользователи
* Связей роли-наборы полей
* Состояний
* Переходов

Для каждой из сущностей доступны 3 основные функции: создание, удаление, изменение имени, а также набор дополнительных функций для изменения собственных атрибутов сущности. Например, у роли кроме основных функций существует только операция смены её типа, а у состояния таких дополнительных функций намного больше: изменение роли состояния, изменение правила остановки, изменение перехода по истечению времени и установка лимита этого времени. Это зависит от сложности структуры конкретной сущности.

При совершении любой операции ядро проверяет возможность её выполнения. Модуль проверяет входные данные, и, в случае если выполнение операции приведёт к дублированию данных или потери связей между объектами, ядро выдаёт ошибку с поясняющим сообщением.

TtrsTicket – класс, описывающий структуру запроса. Этот класс содержит все атрибуты запроса и предоставляет функции маршрутизации и изменения запроса. Под маршрутизацией запроса подразумеваются следующие функции:

* Переход запроса из одного состояния в другое в соответствии с настроенной схемой маршрутизации запроса
* Проверка выполнения правила остановки в состоянии
* Откат последнего совершённого перехода
* Назначение запроса пользователю
* Обработка запроса и запись изменений в базу данных
* Отклонение запроса

Изменение запроса включает в себя следующие функции:

* Создание запроса
* Удаление запроса
* Сохранение изменений в данных
* Отмена последних изменений в данных
* Добавить/удалить объект в запрос
* Добавить/удалить вложение в запрос

При совершении любой операции ядро выполняет ряд проверок для того, чтобы в базе данных не образовалось утечки или противоречия данных.

В отличие от структуры TtrsAdmin в классе TtrsTicket нет локального кэша для хранения нескольких объектов и работа в конкретный момент времени ведётся исключительно с одним объектом – запросом, который загружается из базы данных. Все необходимые вспомогательные объекты, такие как: переходы, состояния, пользователи подгружаются по мере необходимости.

При совершении операции перехода запроса из одного состояния в другое ядро проверяет условие выполнения правила остановки запроса в этом состоянии. Для этого происходит создание объекта класса TtrsStateStopRule по экземпляру текста правила, который хранится в базе данных как атрибут состояния. Ядро заполняет структуру правила остановки значениями атрибутов объектов запроса. Затем интерпретатор, описанный в TtrsStateStopRule, выполняет логические операции правила и выдаёт однозначный результат необходимости остановки в данном состоянии – true или false. Далее речь пойдёт о структуре TtrsStateStopRule.

TtrsStateStopRule – класс, определяющий интерпретатор метаязыка, который подробно описан в главе 5.5 этой дипломной работы. TtrsStateStopRule позволяет описывать правила для состояния, при выполнении условий которых будет происходить остановка запроса в этом состоянии при маршрутизации. Правило остановки в состоянии состоит из одного или нескольких правил для конкретного справочника и бизнес-процесса. Такое правило состоит из условий, соединенных логическими операторами. Условие представляет собой операнд и наименование атрибута объекта со сравниваемым значением, если это предполагает операнд. Порядок расчёта действий при вычислении значения логического выражения определяется с помощью скобок и приоритета логических операторов.

Интерпретатор анализирует введённый текст правила и создаёт собственные структуры классов для каждого отдельного правила, в состав которого входит контейнер со всеми условиями, каждое из которых также является отдельным объектом. Значения параметров условия заполняются ядром, так как зависят от данных запроса. В том случае, если в запросе нет данных об атрибуте объекта, указанном в правиле, значение атрибута этого объекта загружается из базы данных и операция вычисляется с этим значением. Если правило выполняется, хотя бы для одного объекта запроса, в данном состоянии произойдёт остановка.

В классе содержится функция сохранения структуры правил в текстовое поле базы данных, а также воссоздание этой структуры по загруженному тексту. При создании нового правила остановки пользователь может некорректно составить текст правила. Каждой уникальной ошибке присвоен свой номер и поставлено в соответствие информирующее сообщение, для того чтобы пользователь знал, в чём именно он допустил ошибку.

Описание правил остановки в состояниях, как и вся остальная настройка схемы маршрутизации, выполняется посредством графического интерфейса пользователя. Детальное описание интерфейса приведено в следующем разделе.

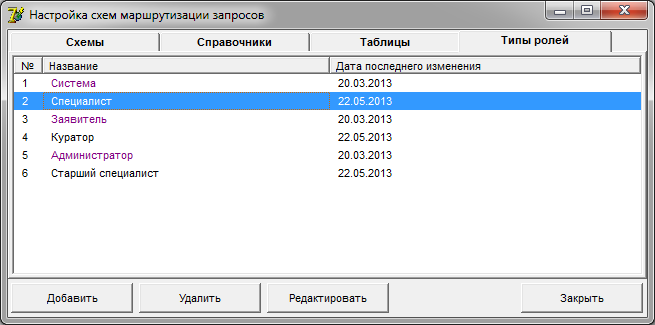
## 8.2 Графический интерфейс пользователя

Графический интерфейс пользователя состоит из трёх главных форм:

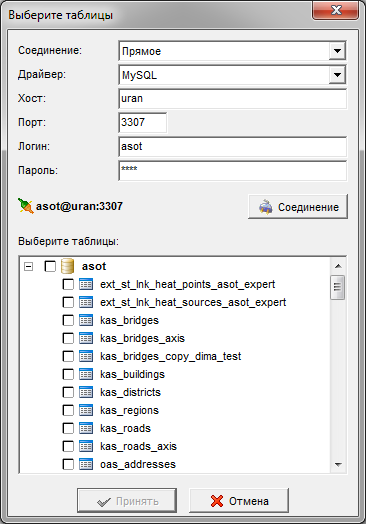
* редактирование основных сущностей схем маршрутизации запросов
* настройка схемы маршрутизации запросов
* настройка бизнес-процесса
* Также интерфейс содержит несколько вспомогательных форм:
* регистрирование физических таблиц в системе
* добавление/редактирование справочника
* настройка параметров роли
* настройка параметров состояния
* настройка параметров переходов

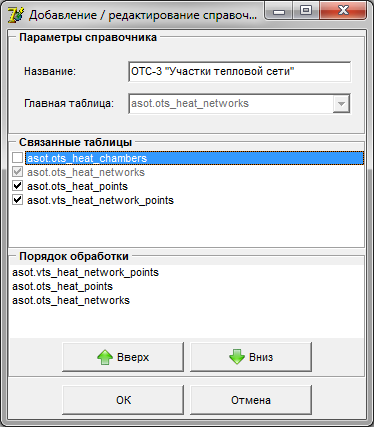
Рассмотрим функциональность и назначение каждой из форм подробнее.

# 8.2.1 Редактирование основных сущностей схем маршрутизации запросов



*Рис. 8.1 Редактирование основных сущностей схем маршрутизации запросов*

Форма редактирования основных сущностей схем маршрутизации запросов, изображенная на рисунке 8.1 является главной формой. В этом окне осуществляется просмотр и удаление объектов основных сущностей: схем, справочников, физических таблиц и типов ролей. Кнопки «добавить», «удалить» и «редактировать» выполняют с объектом соответствующие действия. Фиолетовым цветом выделены объекты являющиеся системными. Системные объекты создаются автоматически и не подлежат редактированию и удалению, поэтому кнопки «удалить» и «редактировать» становятся неактивными при выборе системного объекта. Кнопка «закрыть» заканчивает работу по настройке схем маршрутизации. Для регистрации физических таблиц в системе (рис. 8.2) и детальной настройки справочника (рис. 8.3) используются собственные формы.

Окно регистрации физических таблиц в системе имеет форму подключения к серверу, на котором располагаются необходимые базы данных. Для осуществления подключения к серверу администратору необходимо выбрать тип и драйвер соединения, прописать адрес сервера, порт подключения и указать логин и пароль для авторизации в СУБД. Если данные введены верно, то при нажатии на кнопку «соединение» программа предоставит пользователю список баз данных и их таблиц, располагающихся по указанным реквизитам. Для регистрации таблиц необходимо поставить галочки напротив наименований соответствующих таблиц, после чего нажать кнопку «принять». Результат будет отображен в главной форме на вкладке «таблицы».

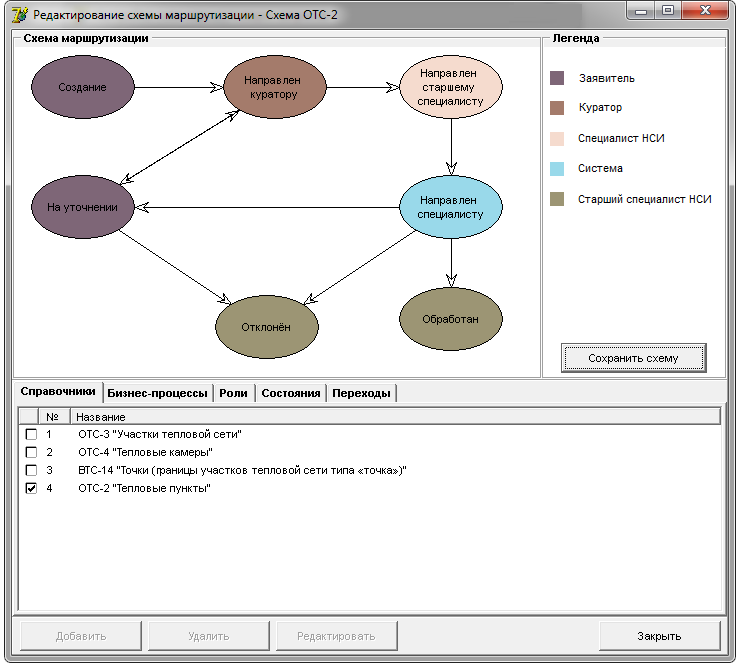
*Рис. 8.3*

*Редактирование справочников*

*Рис. 8.2 Форма регистрации таблиц*

Форма добавления и редактирования справочника представлена на рисунке 8.3. Для того чтобы создать новый справочник, необходимо указать название и выбрать главную физическую таблицу справочника из списка зарегистрированных таблиц. Также с помощью этой формы можно выбрать связанные таблицы и указать порядок их обработки управляющими кнопками «вверх» и «вниз». При редактировании справочника можно изменить название, набор связанных таблиц и их порядок, но нельзя изменить главную таблицу справочника. Для настройки схемы маршрутизации необходимо выбрать нужную в списке всех схем и нажать на кнопку «редактирование».

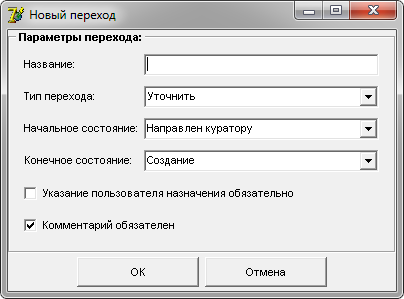
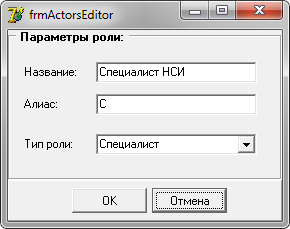
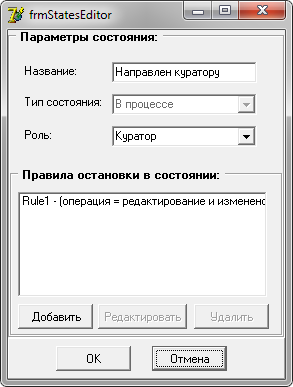
### 8.2.2 Настройка схемы маршрутизации запросов

****

*Рис. 8.4 Окно настройки схемы маршрутизации*

Окно настройки схемы маршрутизации (рис. 8.4) состоит из двух частей: графической и текстовой. Графическая (верхняя) часть состоит из изображения схемы маршрутизации и легенды этой схемы. Как видно из рисунка (рис. 8.4), каждое состояние окрашено в определённый цвет, этот цвет определяет принадлежность состояния одной из ролей схемы. Легенда поясняет соответствие цветов ролям. Администратор системы имеет возможность произвольно настраивать изображение схемы, то есть перетаскивать состояния схемы в любое положение на полотне. Для сохранения результатов изменений необходимо нажать кнопку «сохранить схему». Переходы привязаны к состояниям, поэтому их редактирование недоступно. При создании новой схемы все автоматически созданные состояния будут расположены в одном и том же месте поверх друг друга. Администратору требуется расположить их на полотне по своему усмотрению и сохранить результат в базу данных.

В текстовой (нижней) части формы расположена панель вкладок для выбора списка сущностей. Во вкладке «справочники» осуществляется выбор справочников, участвующих в данной схеме. Вкладка бизнес-процессов обеспечивает работу с бизнес-процессами. Списки ролей, состояний и переходов схемы доступны по соответствующим вкладкам, а редактирование атрибутов этих сущностей производится с помощью вспомогательных форм (рис 8.5).



*Рис. 8.5 Формы настройки атрибутов состояния(а), роли(б), перехода(в)*

*(а)*

*(б)*

*(в)*

Заполнение правил остановки (рис. 8.5а), алиаса роли (рис. 8.5б), а также флагов «указание пользователя при переходе» и «комментарий обязателен» (рис. 8.5в) является необязательной процедурой.

### 8.2.3 Настройка бизнес-процессов

Интерфейс окна настройки бизнес-процессов практически идентичен интерфейсу настройки схемы маршрутизации (рис. 9.4), за исключением отсутствия возможности редактирования изображения схемы маршрутизации, а также набор вкладок имеет следующие элементы:

* Справочники – определение доступных операций для справочников схемы в рамках данного бизнес-процесса (добавление/редактирование/ удаление);
* Набор полей – определение состава набора полей справочников схемы;
* Роли - наборы полей – определение наборов полей у ролей;
* Роли - пользователи – присвоение пользователям ролей.

## 8.3 Апробация модуля

Апробация модуля будет выполняться на схеме маршрутизации настроенной следующим образом:

Типы ролей:

* Система
* Специалист
* Заявитель
* Куратор
* Специалист
* Старший специалист
* Администратор

Физические таблицы:

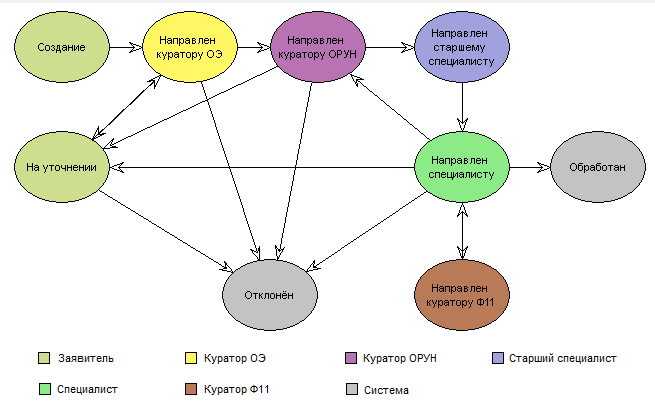
* asot.ots\_heat\_network

Справочники:

* ОТС-3 «Участки тепловой сети»
  + Главная таблица – asot.ots\_heat\_network
  + Связанные таблицы – нет

Схема:

* Зарегистрированные справочники: ОТС-3 «Участки тепловой сети»
* Состояния, переходы и роли изображены на рисунке 8.6.



*Рис. 8.6 Схема маршрутизации запросов на изменение участков тепловой сети*

Бизнес-процесс:

* Доступные операции со справочником: добавление, редактирование, удаление

|  |  |
| --- | --- |
| Роль | Набор полей (ОТС-3 «Участки тепловой сети») |
| Заявитель | Район, наличие предыдущего УТС, предыдущий УТС, тепловой источник, внешний тепловой источник, направление, тепловой пункт, тип УТС, наличие ЦО, наличие ГВС, наличие вентиляции, предприятие, рабочее место, длина УТС, длина по графике, графика, балансовая принадлежность по эксплуатации |
| Куратор ОЭ | Балансовая принадлежность по эксплуатации |
| Куратор ОРУН | Кадастровый номер, длина по кадастру, учетный номер, кадастровый адрес, балансовая принадлежность по недвижимости |

Правила остановки:

* «Направлен куратору ОЭ»: операция = добавление или операция = удаление или операция = редактирование и изменено кроме graphics
* «Направлен куратору ОРУН»: операция = добавление или операция = удаление или операция = редактирование и (изменено balance\_type\_muid или пустое balance\_type\_est\_muid)
* Направлен старшему специалисту: всегда
* Направлен специалисту: всегда

Любые запросы на добавление или удаление участка тепловой сети будут останавливаться во всех четырёх состояниях. Остановка в состоянии «Направлен куратору ОЭ» при редактировании УТС будет происходить только в том случае, если был изменён хотя бы один атрибут объекта кроме его графики. В состоянии «Направлен куратору ОРУН» запрос на редактирование останавливается в том случае, когда изменена балансовая принадлежность по эксплуатации или значение балансовой принадлежности по недвижимости не заполнено. Проверка работы модуля на тестовых данных показала, что система маршрутизации и механизмы пропуска состояний работают исправно.

# Заключение

В рамках данной дипломной работы были решены следующие задачи:

1. Изучена теория контроля качества информации
2. Изучена предметная область, в рамках которой поставлена проблема
3. Рассмотрены существующие решения проблемы
4. Разработана новая структура базы данных
5. Разработан модуль настройки схем маршрутизации запросов
6. Разработанный модуль внедрён в эксплуатацию автоматизированной системы АСОТ

Решение данных задач позволило:

1. Внедрить технологию мультизапросов
2. Добавить бизнес-процессы, определяющие режим работы пользователей
3. Увеличить гибкость схемы маршрутизации с помощью механизма пропуска и остановки в состояниях
4. Добавить визуальную интерпретацию схемы маршрутизации
5. Упростить систему настройки схемы маршрутизации

Благодаря новому модулю маршрутизации запросов были замечены количественные и качественные улучшения результатов работы пользователей в системе. Среднее время обработки запроса сократилось. Количество обработанных запросов в промежуток времени выросло, а отправленных на уточнение уменьшилось.

Выбранная технология программирования позволила правильно спроектировать модуль, который был разработан за установленный срок. Модуль поддерживает расширения и надстройки в виду своей структуры и наличия открытого программного интерфейса.

Внедрённая система маршрутизации обеспечивает высокий уровень контроля качества информации с высокой производительностью обработки данных благодаря технологии мультизапросов и механизмам пропуска состояний схемы маршрутизации.

# Список используемой литературы

1. Atlassian JIRA Documentation [https://confluence.atlassian.com/display/JIRA/ JIRA+Documentation](https://confluence.atlassian.com/display/JIRA/%20JIRA+Documentation)
2. Wang R. & Strong D. Beyond Accuracy: What Data Quality Means to Data Consumers // Journal of Management Information Systems – №12 (4). – 1996. – 34 с.
3. Андреев А.М. Березкин Д.В. Кантонистов Ю.А. Объектные СУБД на российских просторах. – "Компьютерная хроника", 1997.
4. Браун Л. А. История географических карт. – 2007. – М.: Центрполиграф, 2006. – 479 с.
5. Бураков П.В., Петров В.Ю. Введение в системы баз данных: Учебное пособие – СПб.: СПбГУ ИТМО, 2010. – 129 с.
6. Гайдамакин Н. А. Автоматизированные информационные системы, базы и банки данных. Вводный курс: Учебное пособие. – М.: Гелиос АРВ, 2002. – 368 с.
7. Герасименко В.А. Основы теории управления качеством информации. Деп. в ВИНИТИ, № 5392-В 89, 1989. – 410 с.
8. Герасименко В.А. Принципы и методы индустриализации управления на базе комплексного применения вычислительной техники. Деп. в ВИНИТИ, № 5391-В 89, 1989. – 111 с.
9. Герасименко В.А., Малышев Н.И., Таирян В.И. Постановка задачи и методологические подходы к управлению качеством информации в автоматизированных системах. – В сб. Проблемы повышения качества информации. МИИТ, 1988, № 808, 62-63 с.
10. Герасименко В.А., Таирян В.И. Проблемы компелексного управления качеством информации в современных вычислительных системах и сетях. – В сб. Классификаторы и документы, 1986, вып. 1-6.
11. Гришков В.И. Исследование возможностей объектного представления данных в прикладных системах // Труды СПИИРАН. Вып.1, т.3. – СПб: СПИИРАН, 2003.
12. Документация программы LanDocs <http://landocs.ru/system/>
13. Долин П.А. Основы техники безопасности в электроустановках: Учебное пособие для вузов 2-ое изд., перераб. и доп. – М.: Энергоатомиздат, 1984. – 448 с.
14. Дружинин Г. В. Надежность автоматизированных производственных систем. – М.: Энергоатомиздат, 1986.
15. Журкин И. Г., Шайтура С. В. Геоинформационные системы. – М.: Кудиц-пресс, 2009. – 272 с.
16. Избачков Ю. С. Информационные технологии. Учебное пособие. – Спб: Питер, 2007. – 656 с.
17. Касьянов В. Н. Лекции по теории формальных языков, автоматов и сложности вычислений. – Новосибирск: НГУ, 1995. – 112 с.
18. Кон М. Scrum: гибкая разработка ПО. – М.: «Вильямс», 2011. – 576 с.
19. Коржов В. Многоуровневые системы клиент-сервер. – М.: Открытые системы, 1997.
20. Коровкин С. Д., Левенец И. А., Ратманова И. Д., Старых В. А., Щавелёв Л. В. Решение проблемы комплексного оперативного анализа информации хранилищ данных // СУБД. – № 5-6. – 2007. – 247 с.
21. Курлаев С. А., Цильковский И. А. Распределённые информационные системы и базы данных: учебно-методическое пособие – Новосибирск: НГТУ, 2012. – 92 с.
22. Мамиконов А.Г., Кульбы В.В., Шелков А.Б. Достоверность, защита и резервирование информации в АСУ. – М.: Энергоатомиздат, 1986. – 304 с.
23. Мельников Ю.Н. Достоверность информации в автоматизированных системах. – М.: Сов. радио, 1973. – 136 с.
24. Нефедов В.Н., Осипова В.А. Курс дискретной математики. М.: Издательство МАИ, 1992.
25. Першиков В. И., Савинков В. М. Толковый словарь по информатике / Рецензенты: канд. физ.-мат. наук А. С. Марков и д-р физ.-мат. наук И. В. Поттосин. – М.: Финансы и статистика, 1991. – 543 с.
26. Пивоваров А.Н. Методы обеспечения достоверности информации в АСУ. – М.: Радио и связь, 1982. – 144 с.
27. Роб П., Коронел К. Системы баз данных: проектирование, реализация и управление. – 5-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2004. – 1040 с.
28. Романов Д. А., Ильина Т. Н., Логинова А. Ю. Правда об электронном документообороте. – М.: ДМК Пресс. – 224 с.
29. Рудько Г. И., Назаренко М. В., Хоменко С. А., Нецкий А. В., Федорова И. А. – Геоинформационные технологии в недропользовании (на примере ГИС K-MINE) – К.: «Академпрес», 2011. – 336 с.
30. Трудовой кодекс Российской Федерации. Под ред. Дегтяревой Т. – М.: Эксмо, 2013. – 208 с.
31. Хопкрофт Д., Мотвани Р., Ульман Д. Введение в теорию автоматов, языков и вычислений = Introduction to Automata Theory, Languages, and Computation. –М.: Вильямс, 2002. – 528 с.
32. Черняк Л. Задачи управления мастер-данными // Открытые системы. №5. – 2007.