Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ КОСМИЧЕСКИХ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
Кафедра «Информационных систем»

УТВЕРЖДАЮ
И. О. Заведующий кафедрой ИС
_____ Л.С. Троценко

«13» июня 2018 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.02 «Информационные системы и технологии»

Разработка программного модуля проверки доступности информационных узлов локальной сети

Руководитель	_____ подпись, дата	доцент, к.т.н	Е.А. Мальцев
Выпускник	_____ подпись, дата		М.П. Цисневич
Нормоконтролер	_____ подпись, дата		Ю.В. Шмагрис

Красноярск 2018

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ КОСМИЧЕСКИХ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
Кафедра «Информационных систем»

УТВЕРЖДАЮ
Заведующий кафедрой ИС

_____ С.А Виденин
подпись инициалы, фамилия

«2» марта 2018 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

в форме бакалаврской работы

Студенту _____Цисневич Максиму Павловичу _____
фамилия, имя, отчество

Группа ЗКИ13-13б Направление (специальность) _____09.03.02_____
номер код

_____Информационные системы и технологии_____
наименование

Тема выпускной квалификационной работы Разработка программного модуля проверки доступности информационных узлов локальной сети _____

Утверждена приказом по университету № 3758/с от __14.03.2018_____

Руководитель ВКР Мальцев Е.А. консультант к.т.н., доцент кафедры систем искусственного интеллекта

Исходные данные для ВКР Рекомендации руководителя, учебная литература, рекомендации
заказчика._____

Перечень разделов ВКР 1. Исследование и анализ системм аналогов
___

2. Проектирование. Выбор технического решения_____

3. Описание проектных решений и реализация системы_____

Перечень графического материала Презентация, выполненная в Microsoft: PowerPoint 2007_____
_____
_____
_____

Руководитель ВКР _____ Е. А. Мальцев
подпись инициалы и фамилия

Задание принял к исполнению _____ М. П. Цисневич
подпись, инициалы и фамилия студента

«2» марта 2018 г.

РЕФЕРАТ

Выпускная квалификационная работа по теме «разработка программного модуля проверки доступности информационных узлов локальной сети для компании ООО «ИК «А-ИНЖ»» содержит 49 страниц текстового документа, 23 иллюстрации и 14 использованных источников.

АВТОМАТИЗИРОВАННАЯ ИНФОРМАЦИОННАЯ СИСТЕМА, КОНТРОЛЬ, СИСТЕМА ОПОВЕЩЕНИЯ, ПРОГРАММА, ОБЪЕКТ, ИНФОРМАЦИЯ, ОТЧЕТ.

Объектом исследования работы является программный модуль проверки доступности информационных узлов локальной сети».

Целью работы является разработка программного модуля проверки доступности информационных узлов локальной сети.

В ходе выполнения данной работы была создана система по контролю и оповещению доступности узлов локальной сети на предприятии, сформирована база данных и разработан алгоритм работы программной части системы, также создан пользовательский интерфейс программы.

В результате была реализована система по контролю и оповещению доступности узлов локальной сети на предприятии. Разобранная по шагам работа реализованной системы дает понимание функциональности реализованной системы в рамках данного предприятия. Исходя из этого, главная цель выпускной квалификационной работы была выполнена.

СОДЕРЖАНИЕ

Техническое задание

На кафедре Систем искусственного интеллекта было получено задание по разработке программного модуля проверки доступности информационных узлов локальной сети для компании ООО «ИК «А-ИНЖ» ул. Вавилова 1, стр. 1

Цель выпускной квалификационной работы - спроектировать модуль проверки доступности информационных узлов локальной сети

Требования, согласованные с руководителем «Выпускной Квалификационной Работы» после выявления потребности от фирмы заказчика, можно определить следующей последовательностью: при открытии программы, должна быть возможность ввести несколько разных ip и запустить «пинг» отдельно, для каждого ip, размер пакета стандартный, чтобы не забивать трафиком сеть.

Должна быть возможность настроить программу через прокси сервер. Если «пинг» пропал, к примеру, потеряно 10 пакетов в подряд, нужно слать уведомление на электронную почту и по смс. В сообщении должно быть указано ip, до какого время потери и когда, начались. Сама программа должна или работать как служба или запускаться автоматом, сворачиваться в «трей» и висеть в «трее». Настройки после перезагрузки компа не должны слетать. Работать должна под «Windows 7,8.1, 10, server 2008».

Так же, необходимо присылать сообщение, когда связь восстановилась. Должна быть возможность настроить время рассылки сообщений. К примеру, если сервер не доступен, то отправлять сообщения каждые 10 минут. Так же, должна быть возможность настроить почту, с которой идет рассылка, номер телефона и отправку сообщений в мессенджер «Telegram» - это пожелание системного администратора, который и будет заниматься мониторингом локальной сети организации.

Еще одно не маловажное требование, непосредственно определенное заказчиком программного обеспечения, оперативность получения

информации для администратора сети фирмы, для того чтобы, потери связи и их причины были устранены незамедлительно во избежание финансовых потерь организации т.к. работа фирмы полностью построена на работоспособности локальной сети и сети «Enternet» при их сбоях фирма несет серьезные финансовые потери, кроме этого системный администратор обслуживает еще три фирмы в здании и не в состоянии присутствовать постоянно на рабочем месте. Имея при себе мобильный телефон с подключением к «Enternet» может удобно вести мониторинг сети удаленно т.к. получает рассылку оповещений на аккаунт «Telegram».

Для выполнения поставленной цели необходимо решить несколько задач:

- изучить теоретическую часть;
- изучить готовые аналогичные модули проверки;
- ознакомиться с преимуществами аналогов, представить свои;
- выбрать программную среду и оболочку для проектирования системы.

Система проверки доступности информационных узлов локальной сети "pinger" предназначена для мониторинга доступности узлов локальной сети. С помощью нее легко можно отследить активность сетевого оборудования, при этом, в случае потери связи с удаленным узлом отправляется письмо - оповещение на e-mail и в Telegram об отсутствии связи, времени потери связи, доступности и количестве потерянных пакетов. Если же, связь возобновлена, то отправляется повторное оповещение о восстановлении связи. Кроме того, утилита имеет много вариаций настройки проверок и проверку каждого узла локальной сети вручную, при этом будет конечно отображена скорость перемещения пакетов по всей сети и непосредственно на заданных к проверке узлах.

Оборудование системы доступности узлов «pinger». Ключевыми компонентами системы являются:

- Интерфейс пользователя;
- База знаний;
- Интерпретатор;
- Модуль создания системы.

ВВЕДЕНИЕ

Очень часто, в среде системного администрирования локальных и не только сетей, возникают ситуации, когда необходимо определить доступность и работоспособность узла, сайта или сервера в интернете. В большинстве этих случаев может прояснить ситуацию команда «Ping», применять данный вид команды можно практически в любой операционной системе.

Использование команды «Ping» позволяет проверить соединение с узлом на уровне IP. Она же позволяет определить наличие той или иной неполадки. С ее помощью вы посылаете на хост, указанный в команде, пакет определенного размера, через определенное время вы получаете ответ - пакет возвращается. На основе полученных данных пакета, можно судить о совместимости настроек, определять проблемы с аппаратным обеспечением, а также оценивать стабильность подключения компьютера к сетевым ресурсам и сети TCP/IP в целом.

Универсальное средство, доступное для каждого пользователя и в любой момент времени это команда «Ping». Синтаксис ее использования очень прост, он придется по вкусу даже человеку, которому вовсе не приходилось иметь дело с компьютерными сетями. Команда включает в себя широкий арсенал параметров, с помощью которых можно поставить перед системой более точные параметры работы и эксплуатации.

В операционных системах таких как Windows, Unix команда «Ping» выполняется достаточно просто. Для этого нужно просто в соответствующем приложении её запустить. К примеру, в Windows нужно в командной строке ввести следующий текст: ping <имя хоста или его ip-адрес>, в треугольных скобках узел, для которого вы будете проверять «Ping». Пользователям Unix – систем, все описанные действия можно исполнить в терминале.

С параметрами -t, -s, -i часто используется команда «Ping».

– Первый параметр - позволяет отслеживать действие команды неограниченное число раз («Ping» не прекращает работу по завершению отведенного стандартными настройками времени);

– Второй параметр – позволяет изменять размер ICMP-пакета, который посылается определённому компьютеру или серверу;

– Третий параметр – позволяет изменять интервалы между посылкой этих пакетов.

По существующей сети, Команда «Ping» дает общее представление о скорости перемещения пакета. Если он проходит между узлами без частых скачков, с определенным интервалом, это является свидетельство корректности работы, а если пакеты вообще не доходят или приходят с сильной задержкой, тогда это означает, что в сети возникли определенные проблемы, которые, необходимо решать системному администратору компьютерной системы.

Если затронуть вопрос системы автоматизированной проверки доступности информационных узлов сети, то найдется достаточно решений по своему удобству в использовании. Их основная цель вести «ping» информационных узлов сети в автономном или ручном режиме, и обрабатывать данные, при этом в зависимости от возвращаемого значения выхода утилиты способность сигнализировать о доступности или недоступности узла.

Если локальная сеть предприятия состоит из множества сегментов, а именно:

– Пользователи (клиенты);

– Устройства вывода информации на бумажные носители (принтеры);

– Несколько маршрутизаторов;

– Сервера;

– Множество установленных IP камер.

При возникновении неполадки в соединении (отсутствия передачи пакетов) программа фиксирует задержку или же потерю определенного количества данных, определенного узла в сети со своим IP, вся информация в кротчайшие сроки, поступает в виде сообщения в мессенджер администратора, который даже находясь не в этом здании может принять решение об оперативности реагирования и устранения поломки. В сообщении прописан узел, количество потерянных пакетов и время восстановления – если связь возобновлена. Так вот, при «падении» сервера, парализуется работа всего офиса, так же немаловажна работа принтеров и соединения клиентов. Большая часть IP камер на постоянном контроле и их отслеживает служба безопасности, а другая часть менее востребованные пишут в архив и смотрят по ним данные только тогда, когда что-то случится. Так вот, такие камеры могут просто «тихо» выйти из строя, никого об этом «не оповестив», следовательно, и информацию, которую они должны были писать мы не увидим. Подводя итог, очень важна оперативность реагирования на недоступные узлы локальной сети, для таких целей нужны утилиты, основанные на постоянной проверке узлов локальной сети.

В наше время существует большое количество версий таких утилит типа «pinger», мы рассмотрим несколько распространённых, сравним их функционал, удобство в использовании и предложим свою версию этой программы со своими преимуществами.

Исследование и анализ систем аналогов

На сегодняшний день существует много приложений для мониторинга компьютерных сетей. Мы предложим к рассмотрению аналогов самые популярные из них

Популярное приложение для администрирования, мониторинга и инвентаризации компьютерных сетей «Friendly Pinger».

В описании к скачиванию представлен основной функционал, выполняющийся этой программой:

- Визуализация компьютерной сети в анимационной форме;
- Отображение, какие компьютеры включены, а какие нет;
- Опрос всех устройств сети за раз;
- Оповещение в случае остановки/запуска серверов;
- Инвентаризация программного и аппаратного обеспечения всех компьютеров в сети;
- Отслеживание мониторинга и скачивания файлов вашего компьютера другими пользователями;
- Открытие компьютеров в проводнике, в Total Commander'e или в FAR'e;
- Функция "Создать дистрибутив" позволяет создать облегченную версию с Вашими картами и настройками;
- Поиск сетевых служб HTTP, FTP, e-mail и других;
- Назначение устройствам внешних команд: telnet, tracert, net.exe. Графический TraceRoute.

Преимущества программы состоит в том, что легко отслеживается активность сетевого оборудования, при этом сопровождается поддержка графической картой, где в наглядном виде изображены работающие

устройства и нет. Введя адреса локальной сети, которые требуется мониторить, запускаем единовременный «Ping» хостов всей сети и контролируем их работоспособность. В программе еще есть возможность настроить рассылку уведомлений на почту о работоспособности узлов локальной сети. На рисунках 1-3 показана визуализация сети и реализация отправки отчета на e-mail.

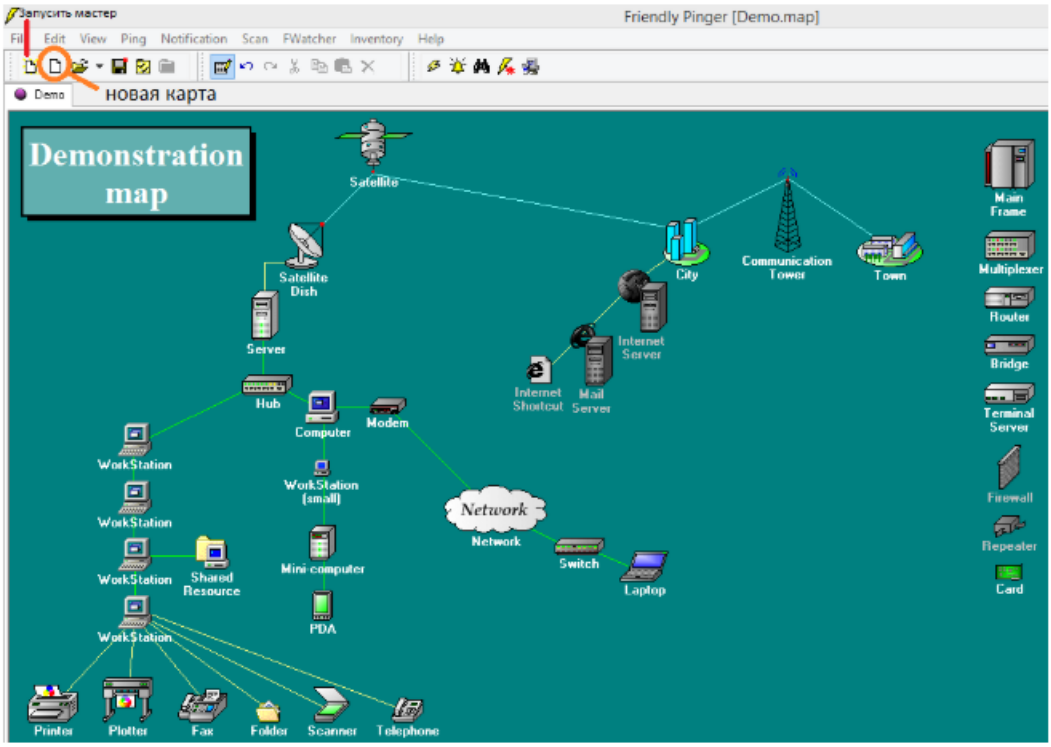

Рисунок 1- Графическое изображение локальной сети Friendly Pinger

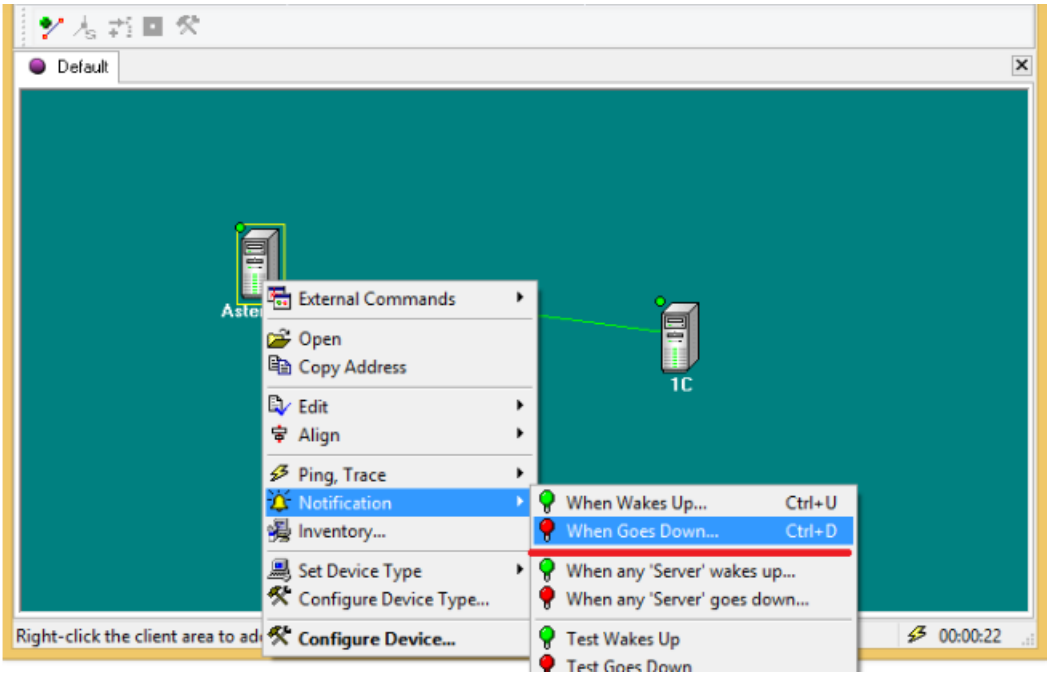

Рисунок 2 – Уведомление о состоянии соединения

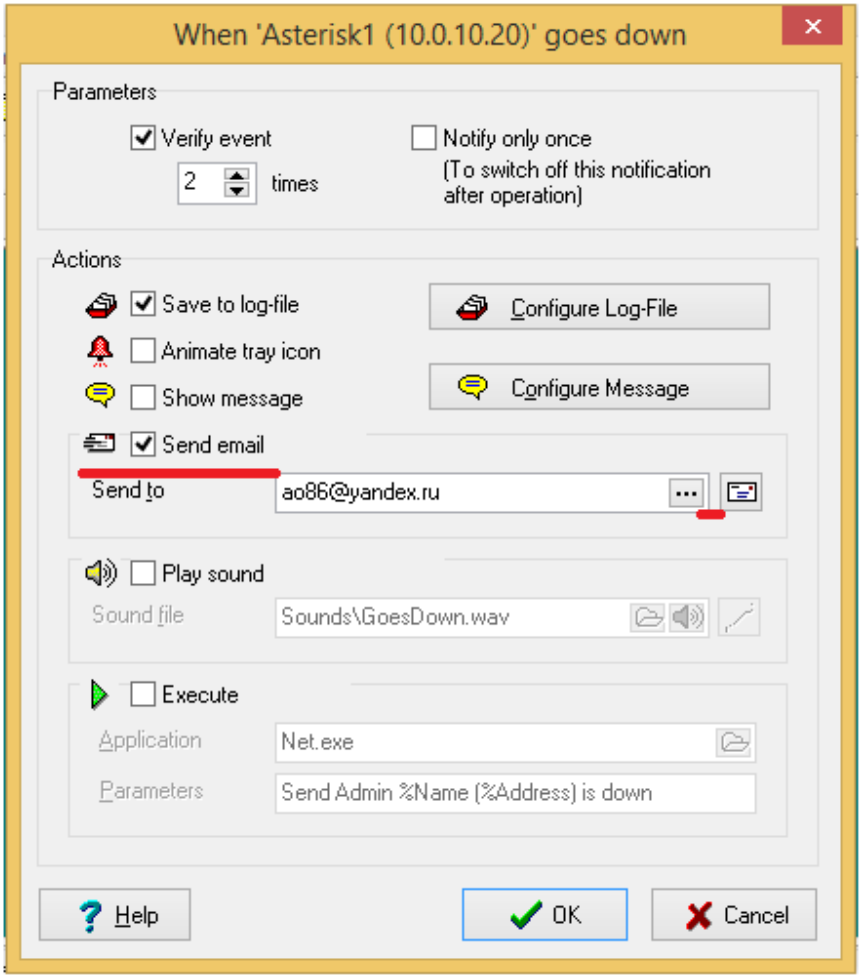

Рисунок 3 – Параметры уведомления

Приложение «NetView 2.94» - для проверки состояния хостов при помощи своего функционала: «Ping» /сканированием/ «ARP» запросами (позволяет видеть все компьютеры сети: которые отсутствуют в сети и те, которые имеют устаревший browse master). На рисунке 4 показан функционал программного решения.

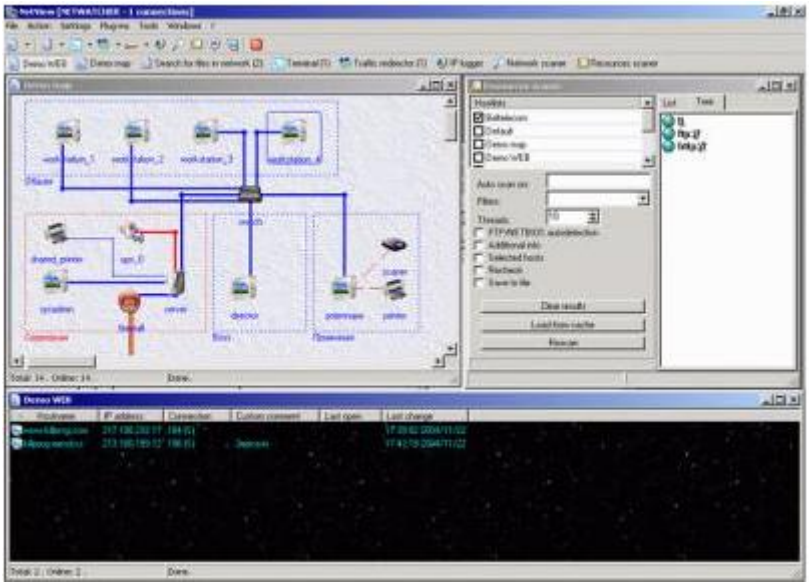

Рисунок 4 – Интерфейс и функционал NetView 2.94

Функционал «NetMessenger» - позволяет слать сообщения в Windows Messenger. Возможностью программы - это посылка сообщений от произвольного имени, аналог NET SEND. Имеет возможность так же и принимать сообщения. Работает как сервис в среде Windows 2000/XP/2003.

Еще одна программа для мониторинга сетевого оборудования и серверов – «Observium», имеет внушительный список поддерживаемых устройств, использующих «SNMP» протокол. Как и всё программное обеспечение, «LAMP», «Observium» относительно легко установить и настроить, требует обычных установок Apache, PHP и MySQL, создания базы данных, конфигурации Apache и тому подобного. Устанавливается программа как собственный сервер с выделенным URL-адресом. На рисунке 5 показан функционал программного решения.

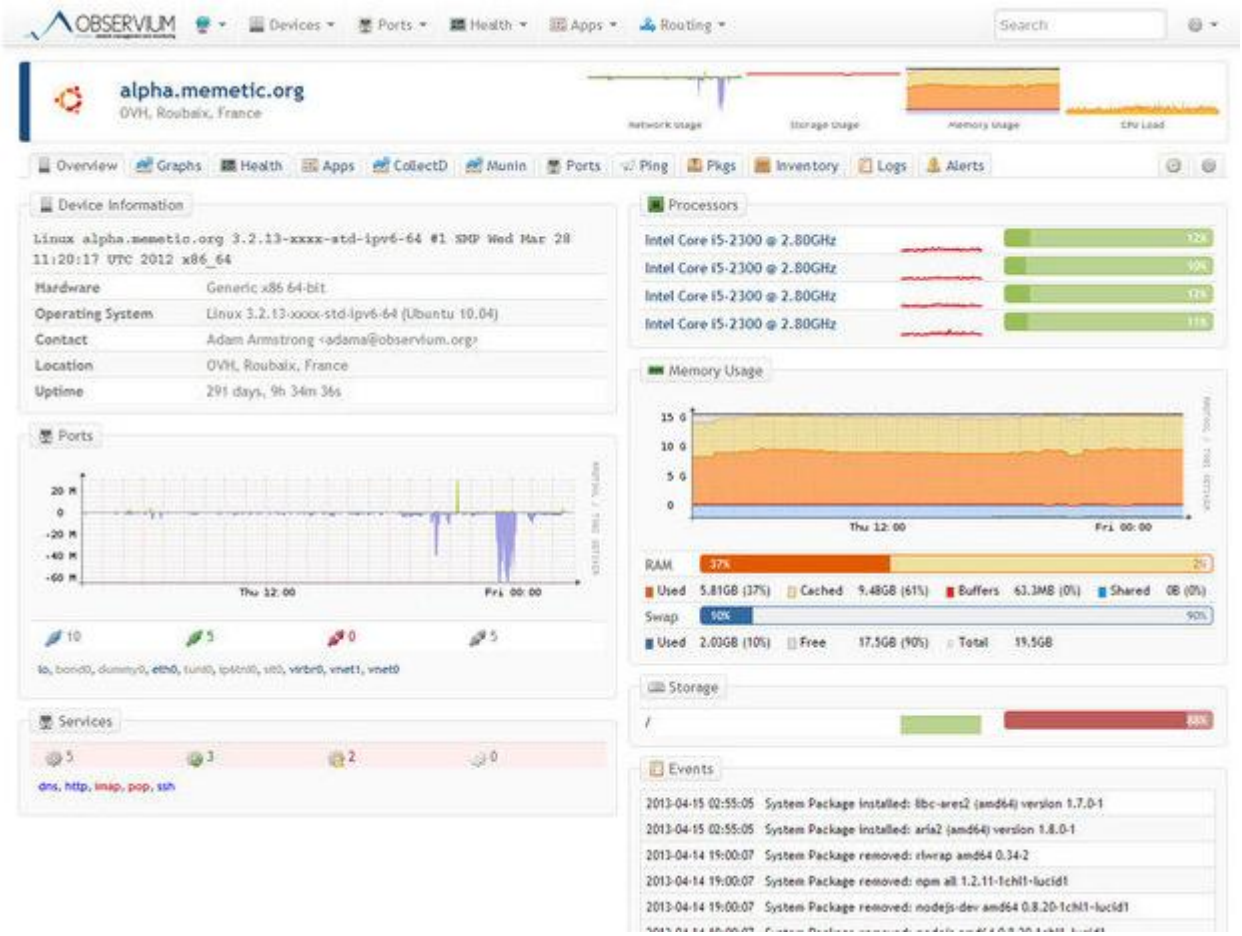

Рисунок 5 – Интерфейс и функционал Observium

«Observium» сочетает в себе систему мониторинга систем и сетей с анализом характера производительности.

В графическом интерфейсе добавляются хосты и сети, задаются данные «SNMP» и диапазоны для автоматического обнаружения, чтобы программа «Observium» задействовала свой функционал. Исследуя окружающие сети и сводя данные по каждой выводит их на экран в графическом представлении. «Observium» также обнаруживает сетевые устройства через протоколы «CDP», «LLDP» или «FDP», а удаленные агенты хоста при этом могут быть развернуты на «Linux-системах», чтобы помочь в сборе данных.

Через легкий и удобный в использовании интерфейс имеет возможности для статистического отображения данных, в виде диаграмм и графиков. Вы можете получать все данные:

- Время отклика «Ping» и «SNMP»;
- Графиков пропускной способности;
- Количества IP-пакетов;
- Фрагментации и т. д..

Вплоть для каждого обнаруженного порта (в зависимости от устройства), эти данные могут быть доступны.

Для серверов, «Observium» отображает информацию о состоянии центрального процессора, хранилища данных, оперативной памяти, температуры свопа, и т. д. из журнала событий. Также в функционал включен сбор данных и графическое отображение производительности для различных сервисов, в том числе:

- «Apache»;
- «MySQL»;
- «BIND»;
- «Memcached»;
- «Postfix».

«Observium» в дополнение работает как виртуальная машина, поэтому может быть основным инструментом для получения информации о состоянии серверов и сетей.

Рассмотрим программное решение «Ntop» - сейчас для «нового поколения» более известную как «Ntopng» - программа подобна вышеперечисленным утилитам, прошла долгий путь развития за последнее десятилетие. «Ntop» или «Ntopng», - надежный инструмент для мониторинга сетевого траффика с поддержкой быстрого и простого веб-интерфейса. Процесс, настроенный на определенный сетевой интерфейс - это все, что нужно, для получения отчетной информации. На рисунке 6 показан функционал программного решения.

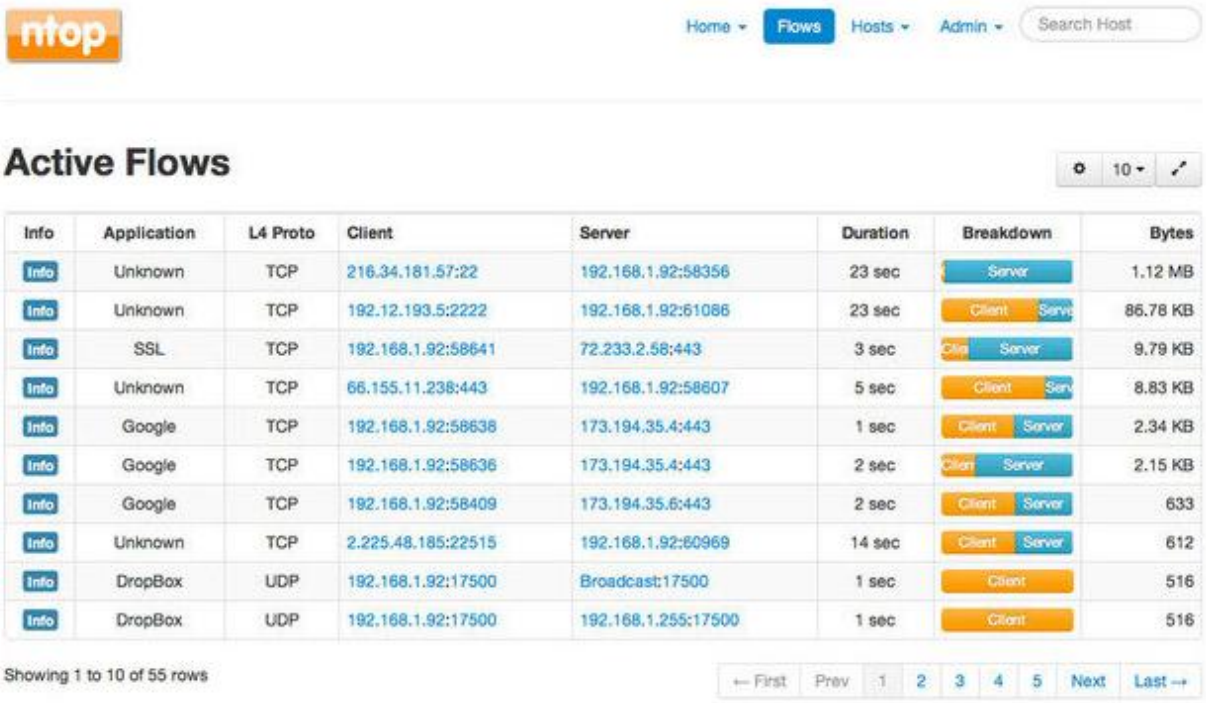

Рисунок 6 – Интерфейс и функционал Ntop

«Ntop» - это инструмент для анализа пакетов с легким веб-интерфейсом, который показывает в режиме реального времени данные о сетевом трафике. Также в режиме реального времени доступна информация о потоке данных через хост и о соединении с хостом. Кроме того, вы можете найти впечатляющий набор графиков, диаграмм и карт использования сети в реальном времени, и еще модульную архитектуру для огромного количества надстроек, таких как добавление мониторов «sFlow» и «NetFlow». Также в арсенале программы имеется «Nbox» - аппаратный монитор, который встраивает в «Ntop».

Контроль трафика в конкретном месте является одним из весомых аргументов применения «Ntopng». Например, на карте сети часть сетевых каналов подсвечены красным, информацией почему – вы не располагаете, так вот с помощью «Ntopng» вы можете получать поминутный отчет о сегменте сети с проблемой и сразу узнаёте, какие хосты ответственны за проблему.

Программы от таких «монстров» как «Zabbix», «Nagios» мы не рассматриваем так как они слишком громоздкие, с большим количеством

лишнего функционала, который в формате нашей фирмы будет просто не востребован, как и переплата за его стоимость.

Проектирование. Выбор технического решения

Сравнивая приведенные выше аналоги, видим, что все программы выполняют основные задачи модуля по определению доступности узлов сети.

Исходя из технического задания, мы сразу находим требования по которым, программы не совсем подходят к нашему предприятию, поскольку, системный администратор обслуживает еще несколько офисов, то он не постоянно на рабочем месте и продвинутая система оповещений играет важную роль в построении архитектуры программы.

Оповещения в нашей системме будут рассылаться не только на e-mail но и на мессенджер Telegram

Кроме того, программы работают не во всех операционных системмах, точнее не в тех котрые понадобятся нам. «Friendly Pinger 5.0.1» например работает на «Windows 98/ME/2000/XP». А модуль проверки узлов локальной сети «NetView 2.94» на «Windows 2000/XP/2003»

Программа должна работать под «Windows 7,8.1, 10, server 2008»

В обзоре аналогов, мы так же нашли настройки оповещений о восстановлении связи, настройки времени рассылки и частоты оповещений.

Наша программа будет присылать сообщение, когда связь восстановилась. Будет возможность настроить время рассылки сообщений, к примеру, если сервер не доступен, то отправлять сообщения каждые 10 минут.

Программа «Observium» – имеет графический интерфейс, а также может отлично работать как виртуальная машина и может стать отличным инструментом для получения информации о состоянии серверов и сетей. Всё это сформировано при поддержке процессов, отраженных в графиках и таблицах, но по части настрой оповещений возможность отсутствует.

«Ntop» - инструмент для анализа пакетов с легким веб-интерфейсом, который показывает данные в реальном времени о сетевом трафике. Информация о потоке данных через хост и о соединении с хостом также доступны в режиме реального времени, но в тоже время программа не предусматривает настройку оповещений.

Еще одна из настроек, которую мы применим в функционале программы «pinger» это запуск программы вместе с «Windows» в свернутом режиме.

Исходя из выше перечисленных требований и руководствуясь своим опытом и навыками, выбираем среду разработки «Delphi 10 Seattle» в которой мы и сделаем пользовательский интерфейс, программный код. Базу данных мы создадим в Access.

В построении архитектуры системы мы будем продукционную модель системы, которая будет состоять из следующих элементов:

- Пользователь;
- Интерфейс пользователя;
- Интерпретатор;
- Модуль создания;
- База знаний;
- Проблемная область.

В системе, будет задействован принцип модульного программирования, программа на каждом шаге анализа образцов определяет, какой модуль, будет более подходящий для обработки данной ситуации. Каждый такой модуль будет реализовывать определенное правило. Функцию управления при этом будет осуществлять интерпретатор.

В распространённых операционных системах (linux, windows) это протокол межсетевых управляющих сообщений, в виде команды «ping» данный протокол и будет реализован. Протокол сетевого уровня модели OSI/ISO – это и есть ICMP

Разработка программного обеспечения

Программная оболочка - интерпретатор команд, обеспечивающий интерфейс для взаимодействия пользователя с функциями программы.

Интерфейс нашей утилиты будет выглядеть вот так: Рисунки 7 и 8

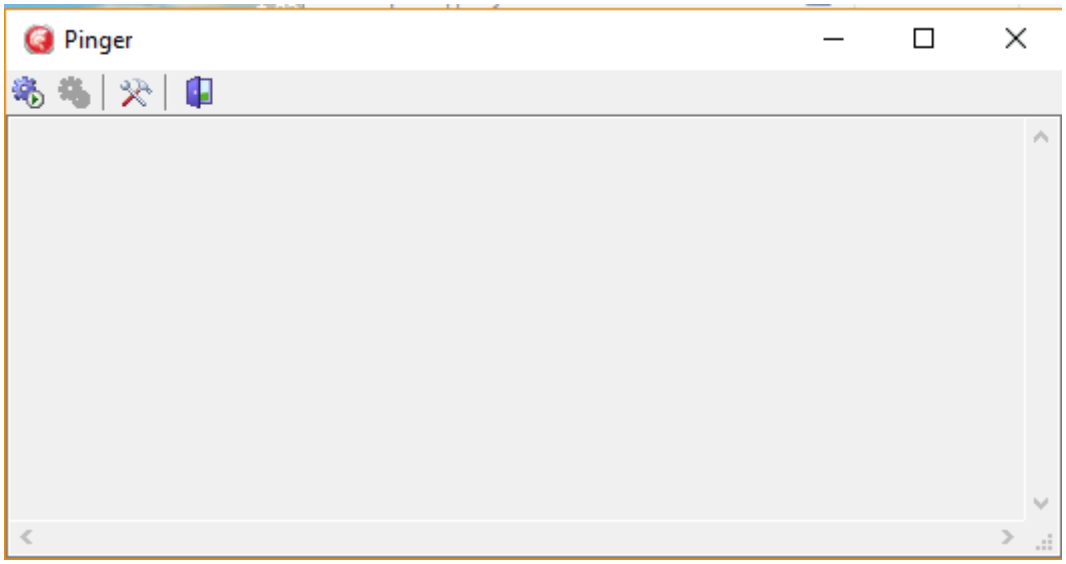

Рисунок 7 – Интерфейс «Pinger»

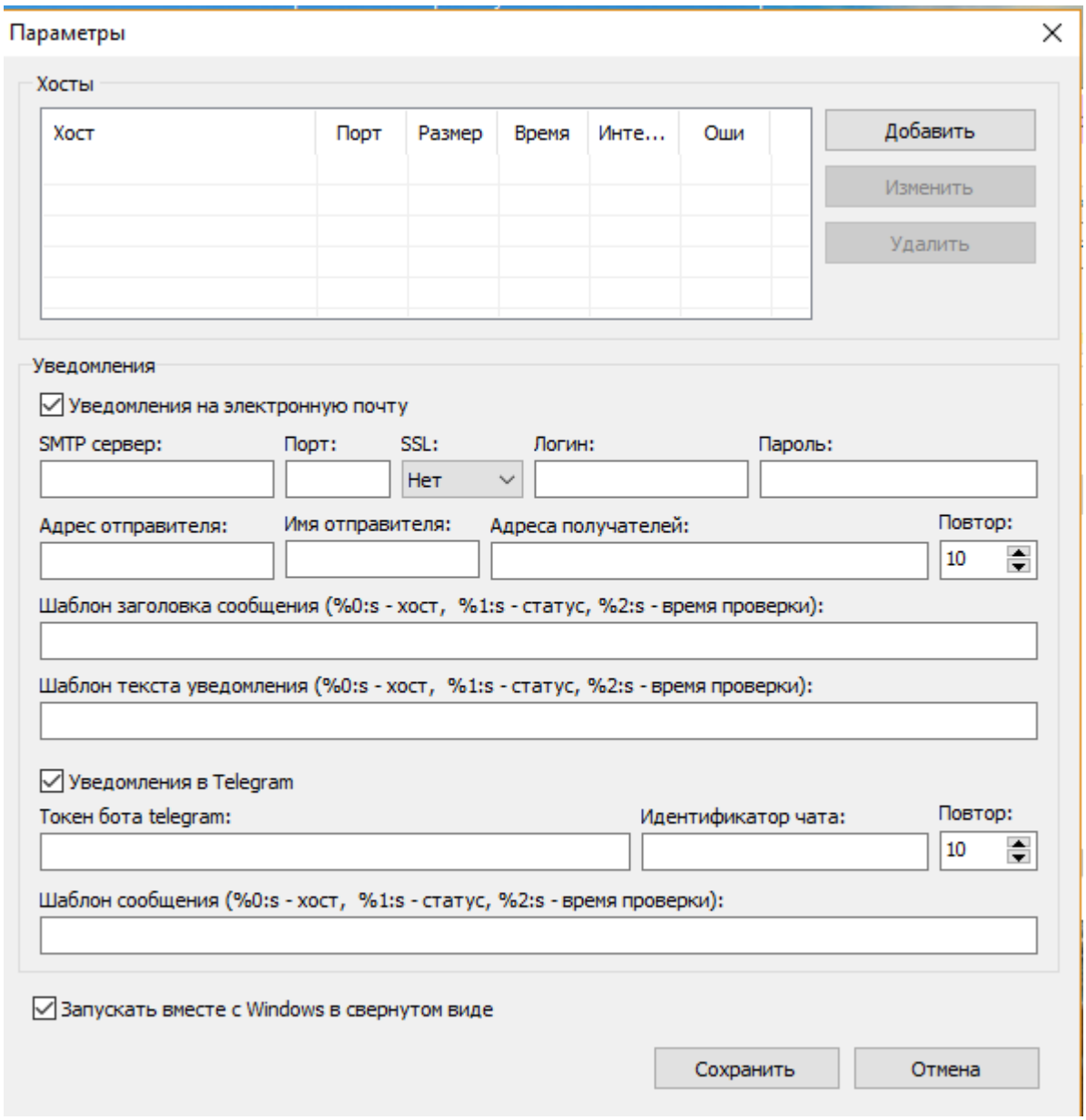

Рисунок 8 – Параметры

Продукционная модель. Если i-ая команда не является командой ветвления в общепринятом традиционном программировании, то i+1-ая команда будет за ней обязательно следующей. Удобство подобного способа программирования обоснованно в тех случаях, когда последовательность обработки мало зависит от обрабатываемых знаний.

В противном случае программу лучше рассматривать как совокупность независимых модулей, управляемых образцами:

– Такая программа на каждом шаге при анализе образцов определяет, какой модуль подходит для обработки данной ситуации;

– Управляемый образцами модуль подходит для обработки данной ситуации;

– Управляемый образцами модуль состоит из механизма исследования и модификации одной или нескольких структур;

– Каждый такой модуль реализует определенное продукционное правило;

– Функции управления при этом осуществляет интерпретатор.

С точки зрения представления знаний подход, при котором, используются управляемые образцами модули характеризуется, некоторыми особенностями:

– Разделение постоянных знаний, хранимых в БЗ;

– Распределение временных знаний из рабочей памяти;

– Структурная независимость модулей;

– Отделение схемы управления от модулей, несущих знания о проблемной области.

Это позволяет рассматривать и реализовывать различные схемы управления, облегчает модификацию системы и знаний

Архитектура программы представлена на рисунке 9

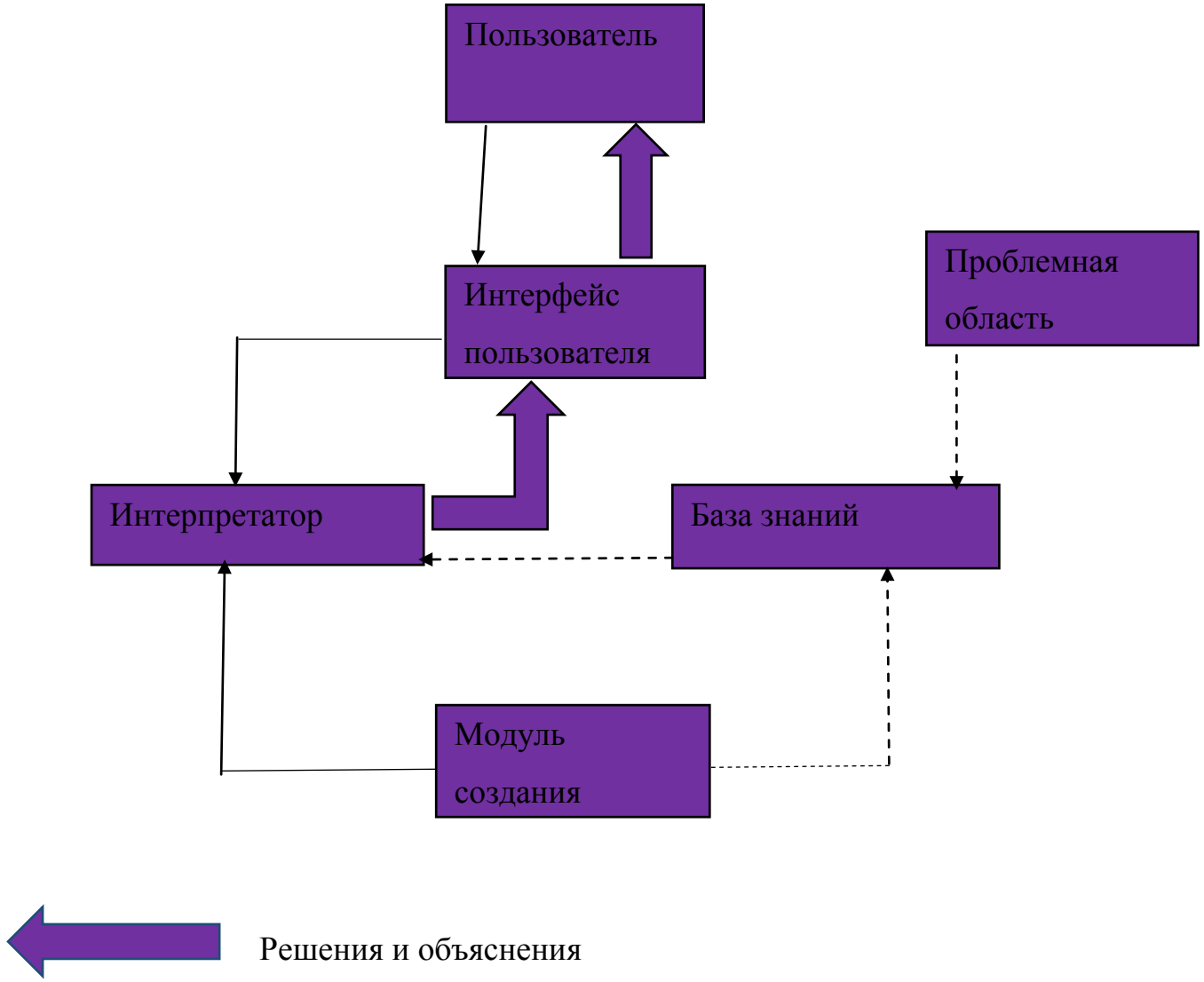

Рисунок 9 – Архитектура программы

Рассмотрим отдельно компоненты Архитектуры программы.

Пользовательский интерфейс – это компоненты и элементы программы, оказывающие влияние на взаимодействие пользователя с программным обеспечением.

Работая с устройствами вычислительной техники, пользователи взаимодействуют с ними (ведут диалог). Реакция ЭВМ на запросы и команды пользователей носит формальный характер. Поэтому программисты, создавая механизм взаимодействия пользователей с программой, формируют наборы:

- Форм;
- Окон;
- Меню;
- Пиктограмм;
- Активных кнопок;
- Справочных систем и т.п.

Эти инструменты в совокупности образуют интерфейс программы – внешний вид отдельных её элементов и видов на экране компьютера.

Важным показателем эффективности используемого интерфейса является выбранная форма диалога между системой и пользователем. Наиболее распространённые формы диалога это:

- Командный режим;
- Запросно-ответный режим;
- Режим заполнения пропусков в выражениях, предлагаемых компьютером;
- Режим меню.

Интерфейс должен обладать такими возможностями как:

- Манипуляция различными формами диалога, изменяя их в процессе принятия решения по выбору пользователя;

– Получение данных от различных устройств системы в различном формате;

– Передача данных системе различными способами;

– Гибко поддерживать знания пользователя (оказывать помощь по запросу, подсказывать).

Часть продукционной модели системы, производящая в определенном порядке обработку знаний, находящихся в базе знаний называется - интерпретатор. Последовательное рассмотрение совокупности правил это и есть функционал интерпретатора – «правило за правилом». В случае соблюдения условия, содержащегося в правиле, выполняется определенное действие, и пользователю предоставляется вариант решения его проблемы.

Кроме этого, во многих системах вводятся дополнительные блоки:

– База данных;

– Блок ввода;

– Блок расчета и корректировки данных.

Блок ввода и корректировки данных используется для оперативного и своевременного отражения текущих изменений в базе данных.

Модуль создания - служит для создания «иерархии» набора правил. Два подхода положены в основу модуля создания системы: использование оболочек экспертных систем, для представления базы знаний специально разработаны языки «Лисп» и «Пролог».

База знаний содержит факты, описывающие проблемную область, а также логическую взаимосвязь этих фактов. Главное место в базе знаний занимают правила. Правило определяет, что следует делать в данной конкретной ситуации, и состоит из двух частей:

– Действия, которое следует произвести, если условие выполняется;

– Условия, которые могут выполняться или нет.

В систему правил входят используемые в продукционной модели системы правила они её и образуют.

Функционал «Проблемной среды» включает в себя две части: решаемые в предметной области задачи и саму предметную область (множество сущностей, описывающих область экспертизы, т.е. множество объектов, значений их характеристик и связывающих их отношений). По-другому выражаясь, проблемная среда включает сущности (структуры данных) и решаемые над ними задачи, представляемые в виде исполняемых утверждений в виде правил, процедур, формул и т.п. Характеристики соответствующей предметной области с характеристиками типов решаемых в ней задач и определяется проблемная среда.

Характеристики предметной области определяются следующим набором параметров.

Существует несколько типов предметной области:

– Первый тип статический, т. е. входные данные не изменяются за время сеанса работы приложения, значения не входных данных изменяются только системой;

– Второй тип динамический, т. е. входные данные, поступающие от внешних источников, изменяются во времени, а значения других данных изменяются системой или подсистемой моделирования внешнего окружения.

Способы описания сущностей предметной области:

– Первый это совокупность атрибутов и их значений (фиксированный состав сущностей);

– Второй это совокупность классов (объектов) и их экземпляров (изменяемый состав сущностей).

Есть несколько способов организации сущностей в «БЗ»:

– Неструктурированная «БЗ»;

– Структурирование сущностей «БЗ» по различным иерархиям, наиболее распространенные иерархии это - «общее/частное» и «часть/целое», что обеспечивает наследование свойств сущностей, представляемых в БЗ.

Почему мы решили остановиться на «Delphi»? «Delphi» – это взаимодействие важнейших технологий, а именно это - высокопроизводительный компилятор в машинный код, объектно-ориентированная модель компонент, визуальное (а, следовательно, и скоростное) построение приложений из программных прототипов и масштабируемые средства для построения баз данных.

Компилятор встроенный в «Delphi» создающий условия преобразования в машинный код, в архитектуре “клиент-сервер” выдает высокую производительность, необходимую для построения приложений. Его ключевыми преимуществами являются: быстрое время проверки готового программного блока и легкость его разработки, что характерного для языков четвертого поколения «4GL» при этом уровень компилятора «3GL» обеспечивает качество кода. Компиляция в «Delphi» производится непосредственно в родной машинный код это конечно напрямую влияет на быстродействие готового приложения.

В объектно-ориентированной моделе программных компонент «Delphi» основной упор делается на максимальном «повторном» использовании кода. Это позволяет весьма быстро строить приложения из уже подготовленных объектов, и также предоставляет возможность создавать свои собственные объекты для среды «Delphi». Нет никаких ограничений по типам объектов, что могут создавать разработчики. В конечном итоге, всё в «Delphi» написано на нём же.

В стандартную комплектацию «Delphi» входят основные объекты, которые и образуют иерархию из 270 базовых классов. Это предоставляет большие возможности. Ещё, можно ознакомиться со списком свободно

распространяемых или коммерческих компонент, разработанных третьими фирмами программного обеспечения.

Также существует быстрая разработка работающего приложения из прототипов. Многие проектирующие программисты, работавшие на других языках до «Delphi», утверждают, что скорость изготовления сложного проекта выше раз в 10 на «Delphi».

В «Delphi» разработка интерфейса является самой простой задачей для программиста. Полный набор визуальных инструментов для скоростной разработки приложений включает в себя среда «Delphi». Один из таких «RAD - rapid application development», поддерживающий подключение к корпоративным базам данных и разработку пользовательского интерфейса. Библиотека визуальных компонентов «VCL», включает в себя стандартные объекты такие как:

– Объекты управления данными построения пользовательского интерфейса;
– Графические объекты;
– Объекты мультимедиа;
– Объекты управления файлами, управление «DDE» и «OLE» и диалоги.

Есть лишь одно упущение, что можно воспринимать, как недостаток «Delphi», это то, что готовых компонент, от «Borland», могло бы быть и больше. Однако этот недостаток уже восполнили, разработки других фирм, а также свободно распространяемые программистами «freeware» - компоненты. Идентичная ситуация складывалась в «Visual Basic» но к слову, визуальные компоненты обладают большей гибкостью в «Delphi».

Каждый язык начинается с алфавита - его множества символов, используемых в языке. Язык алфавита «Delphi» представляет собой подмножество набора символов «ASCII» кода:

– Символы, включающие строчные и прописные буквы латинского алфавита;

– Десятичные цифры;

– Символ подчеркивания;

А также такие символы как:

– Управляющие;

– Специальные;

– Пробела.

Программы и утилиты, написанные на языке «Delphi», состоят из лексем и разделителей.

Лексема - это минимально значимая единица текста программы. Лексемы неделимы, подобны словам естественного языка и сами по себе представляют некоторое содержание. Категории лексем представлены на рисунке 10

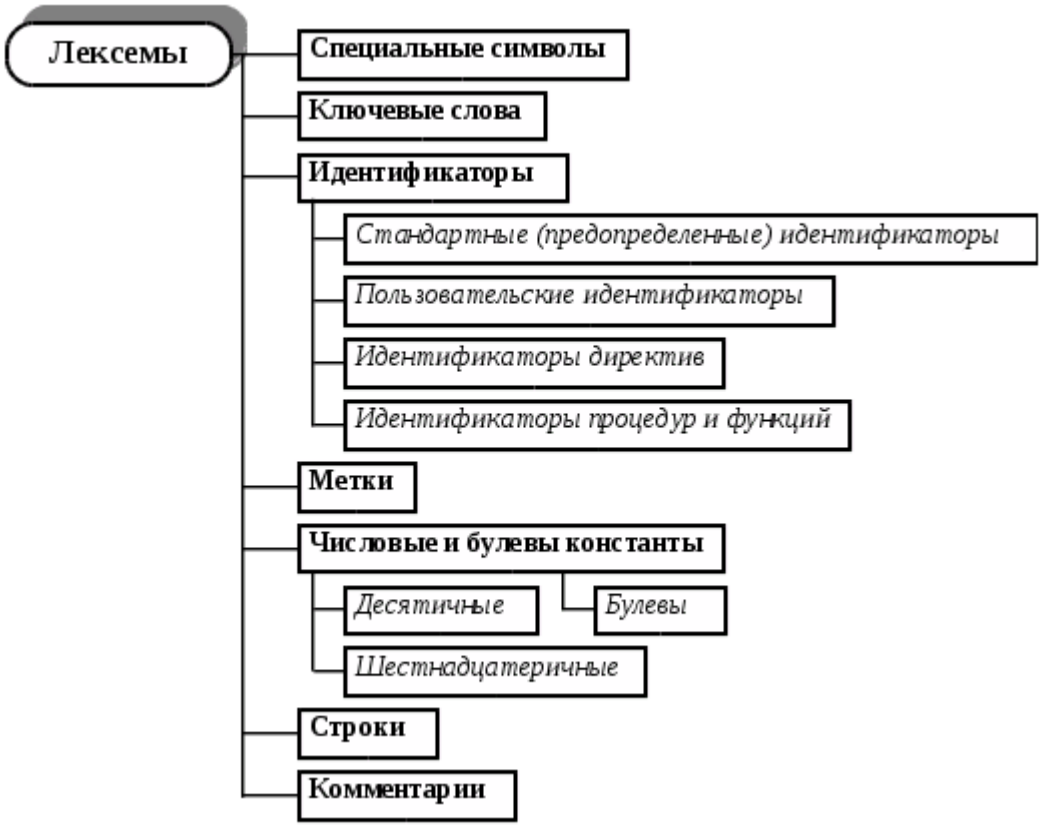

Рисунок 10 – Лексемы

К специальным символам относятся следующие символы: **: ; ^ @ # $+ - * / < = > [ ] ( ) { } . ,**.

Более 60 ключевых слов используется в языке «Delphi».

Для обозначения имен в записях, используются «идентификаторы»:

- Типов;
- Констант;
- Переменных;
- Процедур;
- Модулей;
- Функций;
- Программ;
- Полей.

Идентификатор - это последовательность обозначений: букв, цифр и знаков подчеркивания, которая начинается с символа подчеркивания или буквы и не содержит пробелов, он может иметь произвольную длину, однако только первые 255 символов являются значимыми.

В «Delphi» используются несколько разновидностей идентификаторов: пользовательские и стандартные (предопределенные).

Предопределенными стандартными идентификаторами являются имена всех встроенных в язык процедур и функций:

- Read;
- Write;
- Sin и др.

Директив:

– Absolute;

– Forward;

– Private;

– Public и др.

Типов:

– Integer;

– Real;

– Char и др.

Есть такие стандартные директивы, что, учитывая их специфику применения, называют также процедурными директивами.

Метки, используемые в программе для обозначения некоторых операторов, к которым можно делегировать управление. Они бывают двух разновидностей:

– Символьные;
– Числовые.

В программной среде «Delphi» используются целые: десятичные и шестнадцатеричные числа, и кроме того вещественные десятичные числа.

Целые десятичные числа записываются стандартным образом. Символ «$» используется для обозначения шестнадцатеричных целых чисел, он ставится перед числом, например, $1 $4E $FCB3437 $A20

Числа с десятичными точками или показателем степени являются константами вещественного типа, остальные десятичные и шестнадцатеричные числа являются константами целого типа.

Вещественные числа записываются показательной форме или десятичной дробью, с основанием 10. При первом способе записи вместо основания 10 ставится буква «E» прописная либо строчная, за ней указывается показатель степени, например, 6.3 -1476.9256 7.9E12 -34.83e7 0.2644e-10

Булевы константы «true» и «false» обозначают соответственно «истина» и «ложь» и используются в выражениях.

Строка символов является последовательностью символов из расширенного набора символов кода «ASCII», заключенную в одиночные кавычки.

Управляющие символы в строке представляются в виде целого десятичного числа, прямо перед которым, ставится символ «#». Указанное десятичное число должно быть кодом «ASCII» требуемого управляющего символа:

- #10 — символ " перевод строки";
- #7 — символ "звонок»;
- #13 — символ "возврат каретки"

Так могут быть представлены не только управляющие символы, но и любой другой символ кода «ASCII», коды от 0 до 255. К тому же, в строке допускается совмещать управляющие и печатные символы. Если в строку входят несколько управляющих символов, то между ними не должно быть разделителей:

- совмещение управляющих символов'#13#10;
- Совмещение печатных символов'#13#10'.

Комментарий представляет собой фрагмент текста программы:

- Ограниченный справа - символом **}** или составным символом ***)**;
- Слева символом **{** или составным символом **(***.

Информационную функцию выполняют в программе комментарии и служат для описания назначения отдельных подпрограмм, констант, типов, переменных и т.п. Комментарии пропускаются компилятором и не влияют на работу программы.

В целях повышения наглядности программы, используются разделители. Так, в качестве разделителей лексем друг от друга применяются символы:

- Табуляция (код ASCII 09);
- Пробел (код ASCII 32);
- Составной символ перехода в начало следующей строки (пара символов «возврат каретки» (код «ASCII 13»);
- «Перевод строки» (код «ASCII 10»);
- Любые управляющие символы набора кодов «ASCII» (из диапазона от символа с кодом «0» до символа с кодом «31»).

Лексемы группы «специальные символы» сами являются разделителями, поэтому, отделение их символами-разделителями от других лексем не обязательно.

Программа, либо программа и модули должны присутствовать обязательно при реализации любого приложения. Программа может быть одна, к ней возможно подключить сколько угодно новых модулей. Нужно, чтобы программа содержала декларативную часть и блок «begin...end», где и размещаются выполняемые операторы.

Программа не обязательно должна иметь заголовок, но если он присутствует в программе, то записан он должен быть синтаксически корректно:

- Program Print(Output);
- Program Simple;
- Program GetPut (Input, Output).

«Uses» это раздел указания используемых модулей. Большинство программ, кроме совсем простых создаются в виде нескольких программных единиц – набора модулей и собственно программы. Они все хранятся и транслируются по отдельности, а их исполняемые коды подключаются к программе при компоновке. Для того чтобы, этот процесс был успешным, в начале программы в предложении «uses», нужно указать имена используемых в ней модулей.

Раздел описаний представляет собой последовательность определений имен - указаний, что конкретно обозначается тем или иным идентификатором в данной программе. Идентификаторы могут обозначать константы или участки памяти, в которых хранятся определенные значения, а также другие, более сложные объекты.

Каждое определение заканчивается «**;**» разделителем. В рамках раздела описаний подразделы описания:

- Типов (type);
- Констант (const);
- Переменных (var);
- Функций (function);
- Процедур (procedure);
- Экспорта (exports).

Неоднократное использование одних и тех же подразделов описания применяется как для улучшения структурированности описаний, так и для повышения читабельности программы.

Иногда телом программы называется «раздел операторов» он содержит последовательность вызовов процедур и операторов, непосредственно обрабатывающих данные согласно с поведением алгоритма решаемой задачи. В структуре программы - это единственный обязательный раздел.

Основой модульного программирования в языке «Delphi» являются модули. Их используют для создания библиотек, включаемых в различные программы, а более обширные программы могут подразделяться на логически связанные модули.

Исходный код программы - это текст, понятный машине, выполненный на особом языке. Он может транслироваться в особый вид с помощью компилятора или выполняться непосредственно по тексту с помощью интерпретатора.

Исходный код программы представлен ниже UnitSettings.pas – рисунки с 11 по 17

```
unit UnitSettings;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes,
  System.UITypes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ComCtrls,
  Vcl.StdCtrls, Vcl.ExtCtrls, System.Actions, Vcl.ActnList, Vcl.Samples.Spin;

type
  TfrmSettings = class(TForm)
    gbHosts: TGroupBox;
    lvHosts: TListView;
    btnHostAdd: TButton;
    btnHostEdit: TButton;
    btnHostDel: TButton;
    gbNotify: TGroupBox;
    cbSMTP: TCheckBox;
    edSmtpServer: TLabeledEdit;
    edSmtpPort: TLabeledEdit;
    lblSmptSSL: TLabel;
    edSmtpSSL: TComboBox;
    edSmtpLogin: TLabeledEdit;
    edSmtpPassword: TLabeledEdit;
    edSmtpFromAddress: TLabeledEdit;
    edSmtpFromName: TLabeledEdit;
    cbTelegram: TCheckBox;
    edTgToken: TLabeledEdit;
    edTgChat: TLabeledEdit;
    btnSave: TButton;
    btnCancel: TButton;
    edSmtpToAddress: TLabeledEdit;
    ActionList: TActionList;
    actAdd: TAction;
    actEdit: TAction;
    actDel: TAction;
```

Рисунок 11 - UnitSettings.pas

```
    cbAutoRun: TCheckBox;
    lblSmtpRepeats: TLabel;
    edSmtpRepeats: TSpinEdit;
    edSmtpSubject: TLabeledEdit;
    edSmtpBody: TLabeledEdit;
    Label1: TLabel;
    edTgRepeats: TSpinEdit;
    edTgMsg: TLabeledEdit;
    procedure cbSMTPClick(Sender: TObject);
    procedure cbTelegramClick(Sender: TObject);
    procedure actAddUpdate(Sender: TObject);
    procedure actAddExecute(Sender: TObject);
    procedure actEditUpdate(Sender: TObject);
    procedure actEditExecute(Sender: TObject);
    procedure actDelUpdate(Sender: TObject);
    procedure actDelExecute(Sender: TObject);
    procedure lvHostsDblClick(Sender: TObject);
  private
    { Private declarations }
  public
    procedure LoadCfgFile;
    procedure SaveCfgFile;
  end;

implementation

{$R *.dfm}

uses
  IniFiles, Registry, UnitPing, UnitHost;

{ TfrmSettings }

procedure TfrmSettings.LoadCfgFile;
var
  cfgFile: TMemIniFile;
```

Рисунок 12 - UnitSettings.pas

```
  regFile: TRegistryIniFile;
  i: Integer;
begin
  cfgFile := TMemIniFile.Create(ChangeFileExt(ParamStr(0), extCfgFile));
  try
    i := 1;
    lvHosts.Items.BeginUpdate;
    try
      lvHosts.Clear;
      while cfgFile.ValueExists(iniHosts, Format(iniHostName, [i])) do
      begin
        with lvHosts.Items.Add do
        begin
          Caption := cfgFile.ReadString(iniHosts, Format(iniHostName, [i]), EmptyStr);
          Subitems.Add(cfgFile.ReadString(iniHosts, Format(iniHostPort, [i]), '0'));
          Subitems.Add(cfgFile.ReadString(iniHosts, Format(iniHostPacket, [i]), '32'));
          Subitems.Add(cfgFile.ReadString(iniHosts, Format(iniHostTimeout, [i]), '5000'));
          Subitems.Add(cfgFile.ReadString(iniHosts, Format(iniHostInterval, [i]), '10'));
          Subitems.Add(cfgFile.ReadString(iniHosts, Format(iniHostErrors, [i]), '5'));

          Inc(i);
        end;
      end;
    finally
      lvHosts.Items.EndUpdate;
    end;

    cbSMTP.Checked := cfgFile.ReadBool(iniNotify, iniSmtpOn, cbSMTP.Checked);
    edSmtpServer.Text := cfgFile.ReadString(iniNotify, iniSmtpServer, edSmtpServer.Text);
    edSmtpPort.Text := cfgFile.ReadString(iniNotify, iniSmtpPort, edSmtpPort.Text);
    edSmtpSSL.ItemIndex := cfgFile.ReadInteger(iniNotify, iniSmtpSSL, edSmtpSSL.ItemIndex);
    edSmtpLogin.Text := cfgFile.ReadString(iniNotify, iniSmtpLogin, edSmtpLogin.Text);
    edSmtpPassword.Text := cfgFile.ReadString(iniNotify, iniSmtpPassword, edSmtpPassword.Text);
    edSmtpFromAddress.Text := cfgFile.ReadString(iniNotify, iniSmtpFromAddress, edSmtpFromAddress.Text);
    edSmtpFromName.Text := cfgFile.ReadString(iniNotify, iniSmtpFromName, edSmtpFromName.Text);
    edSmtpToAddress.Text := cfgFile.ReadString(iniNotify, iniSmtpToAddress, edSmtpToAddress.Text);
```

Рисунок 13 - UnitSettings.pas

```
    cfgFile.WriteInteger(iniNotify, iniSmtpRepeats, edSmtpRepeats.Value);
    cfgFile.WriteString(iniNotify, iniSmtpSubject, edSmtpSubject.Text);
    cfgFile.WriteString(iniNotify, iniSmtpBody, edSmtpBody.Text);

    cfgFile.WriteBool(iniNotify, iniTgOn, cbTelegram.Checked);
    cfgFile.WriteString(iniNotify, iniTgToken, edTgToken.Text);
    cfgFile.WriteString(iniNotify, iniTgChat, edTgChat.Text);
    cfgFile.WriteInteger(iniNotify, iniTgRepeats, edTgRepeats.Value);
    cfgFile.WriteString(iniNotify, iniTgMsg, edTgMsg.Text);

    cfgFile.UpdateFile;
  finally
    cfgFile.Free;
  end;

  regFile := TRegistryIniFile.Create(regAutoRunFile);
  try
    if cbAutoRun.Checked then
    begin
      if not SameText(regFile.ReadString(regAutoRunSection, regAutoRunKey, EmptyStr), ParamStr(0)) then
        regFile.WriteString(regAutoRunSection, regAutoRunKey, ParamStr(0));
    end
    else begin
      if regFile.ValueExists(regAutoRunSection, regAutoRunKey) then
        regFile.DeleteKey(regAutoRunSection, regAutoRunKey);
    end;
  finally
    regFile.Free;
  end;
end;

procedure TfrmSettings.cbSMTPClick(Sender: TObject);
begin
  edSmtpServer.Enabled := cbSMTP.Checked;
  edSmtpPort.Enabled := cbSMTP.Checked;
  edSmtpSSL.Enabled := cbSMTP.Checked;
```

Рисунок 14 - UnitSettings.pas

```
procedure TfrmSettings.cbSMTPClick(Sender: TObject);
begin
  edSmtpServer.Enabled := cbSMTP.Checked;
  edSmtpPort.Enabled := cbSMTP.Checked;
  edSmtpSSL.Enabled := cbSMTP.Checked;
  edSmtpLogin.Enabled := cbSMTP.Checked;
  edSmtpPassword.Enabled := cbSMTP.Checked;
  edSmtpFromAddress.Enabled := cbSMTP.Checked;
  edSmtpFromName.Enabled := cbSMTP.Checked;
  edSmtpToAddress.Enabled := cbSMTP.Checked;
  edSmtpRepeats.Enabled := cbSMTP.Checked;
  edSmtpSubject.Enabled := cbSMTP.Checked;
  edSmtpBody.Enabled := cbSMTP.Checked;
end;

procedure TfrmSettings.cbTelegramClick(Sender: TObject);
begin
  edTgToken.Enabled := cbTelegram.Checked;
  edTgChat.Enabled := cbTelegram.Checked;
  edTgRepeats.Enabled := cbTelegram.Checked;
  edTgMsg.Enabled := cbTelegram.Checked;
end;

procedure TfrmSettings.actAddUpdate(Sender: TObject);
begin
  actAdd.Enabled := True;
end;

procedure TfrmSettings.actAddExecute(Sender: TObject);
begin
  with TfrmHost.Create(Self) do
  begin
    try
      if ShowModal = mrOk then
      begin
        with lvHosts.Items.Add do
```

Рисунок 15 - UnitSettings.pas

```
        begin
          Caption := edHost.Text;
          Subitems.Add(IntToStr(edPort.Value));
          Subitems.Add(IntToStr(edPacket.Value));
          Subitems.Add(IntToStr(edTimeout.Value));
          Subitems.Add(IntToStr(edInterval.Value));
          Subitems.Add(IntToStr(edErrors.Value));
        end;
      end;
    finally
      Free;
    end;
  end;
end;

procedure TfrmSettings.actEditUpdate(Sender: TObject);
begin
  actEdit.Enabled := Assigned(lvHosts.Selected);
end;

procedure TfrmSettings.actEditExecute(Sender: TObject);
begin
  with TfrmHost.Create(Self) do
  begin
    try
      with lvHosts.Selected do
      begin
        edHost.Text := Caption;
        edPort.Value := StrToIntDef(Subitems[0], 0);
        edPacket.Value := StrToIntDef(Subitems[1], 32);
        edTimeout.Value := StrToIntDef(Subitems[2], 5000);
        edInterval.Value := StrToIntDef(Subitems[3], 10);
        edErrors.Value := StrToIntDef(Subitems[4], 5);
      end;

      if ShowModal = mrOk then
```

Рисунок 16 - UnitSettings.pas

```
      begin
        with lvHosts.Selected do
        begin
          Caption := edHost.Text;
          Subitems[0] := IntToStr(edPort.Value);
          Subitems[1] := IntToStr(edPacket.Value);
          Subitems[2] := IntToStr(edTimeout.Value);
          Subitems[3] := IntToStr(edInterval.Value);
          Subitems[4] := IntToStr(edErrors.Value);
        end;
      end;
    finally
      Free;
    end;
  end;
end;

procedure TfrmSettings.lvHostsDblClick(Sender: TObject);
begin
  if Assigned(lvHosts.Selected) then
    actEditExecute(Sender);
end;

procedure TfrmSettings.actDelUpdate(Sender: TObject);
begin
  actDel.Enabled := Assigned(lvHosts.Selected);
end;

procedure TfrmSettings.actDelExecute(Sender: TObject);
begin
  if MessageDlg('Удалить выделенную запись?', mtConfirmation, [mbYes, mbNo], 0) = ID_YES then
    lvHosts.DeleteSelected;
end;

end.
```

Рисунок 17 - UnitSettings.pas

UnitPing.pas – рисунки 18, 19

```
unit UnitPing;

interface

type
  THostData = record
    sHost: string;
    iPort: Integer;
    iPacket: Integer;
    iTimeout: Integer;
    iInterval: Integer;
    iIntCount: Integer;
    iErrors: Integer;
    bDisabled: Boolean;
    dtDisable: TDateTime;
    dtNotifySmtp: TDateTime;
    dtNotifyTg: TDateTime;
  end;

  THostsArray: array of THostData;

const
  // Константы для CFG-файла
  extCfgFile              = '.dat';

  iniHosts                = 'hosts';
  iniNotify               = 'notify';

  iniHostName             = 'Host_%d';
  iniHostPort             = 'Host_%d_Port';
  iniHostPacket           = 'Host_%d_Packet';
  iniHostTimeout          = 'Host_%d_Timeout';
  iniHostInterval         = 'Host_%d_Interval';
  iniHostErrors           = 'Host_%d_Errors';

  iniSmtpOn               = 'SmtpNotify';
```

Рисунок 18 - UnitPing.pas

```
  iniHostErrors           = 'Host_%d_Errors';

  iniSmtpOn               = 'SmtpNotify';
  iniSmtpServer           = 'SmtpServer';
  iniSmtpPort             = 'SmtpPort';
  iniSmtpSSL              = 'SmtpSSL';
  iniSmtpLogin            = 'SmtpLogin';
  iniSmtpPassword         = 'SmtpPassword';
  iniSmtpFromAddress      = 'SmtpFromAddress';
  iniSmtpFromName         = 'SmtpFromName';
  iniSmtpToAddress        = 'SmtpToAddress';
  iniSmtpRepeats          = 'SmtpRepeats';
  iniSmtpSubject          = 'SmtpSubject';
  iniSmtpBody             = 'SmtpBody';

  iniTgOn                 = 'TelegramNotify';
  iniTgToken              = 'TelegramToken';
  iniTgChat               = 'TelegramChat';
  iniTgRepeats            = 'TelegramRepeats';
  iniTgMsg                = 'TelegramMsg';

  // Автозапуск
  regAutoRunFile          = 'Software\Microsoft\Windows\CurrentVersion';
  regAutoRunSection       = 'Run';
  regAutoRunKey           = 'Pinger';


implementation

end.
```

Рисунок 19 - UnitPing.pas

UnitMain.pas – рисунок 20

```
  begin
    fIsWork := False;
    Timer.Enabled := False;
    StoreHosts;
    Log.Lines.Add('Контроль остановлен');
  end;

  procedure TfrmMain.TimerTimer(Sender: TObject);
  begin
    Timer.Enabled := False;
    try

    finally
      Timer.Enabled := fIsWork;
    end;
  end;

  procedure TfrmMain.actSettingsUpdate(Sender: TObject);
  begin
    actSettings.Enabled := not Timer.Enabled;
  end;

  procedure TfrmMain.actSettingsExecute(Sender: TObject);
  begin
    with TfrmSettings.Create(Self) do
    begin
      try
        LoadCfgFile;
        if ShowModal = mrOk then
          SaveCfgFile;
      finally
        Free;
      end;
    end;
  end;
```

Рисунок 20 - UnitMain.pas

UnitHost.pas – рисунок 21

```
unit UnitHost;

interface

uses
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Samples.Spin, Vcl.StdCtrls;

type
  TfrmHost = class(TForm)
    lblHost: TLabel;
    edHost: TEdit;
    lblPort: TLabel;
    edPort: TSpinEdit;
    lblPacket: TLabel;
    edPacket: TSpinEdit;
    lblTimeout: TLabel;
    edTimeout: TSpinEdit;
    lblErrors: TLabel;
    edErrors: TSpinEdit;
    btnSave: TButton;
    btnCancel: TButton;
    lblInterval: TLabel;
    edInterval: TSpinEdit;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

implementation

{$R *.dfm}

end.
```

Рисунок 21 - UnitHost.pas

Рисунок 22 – Показывает работоспособность программы

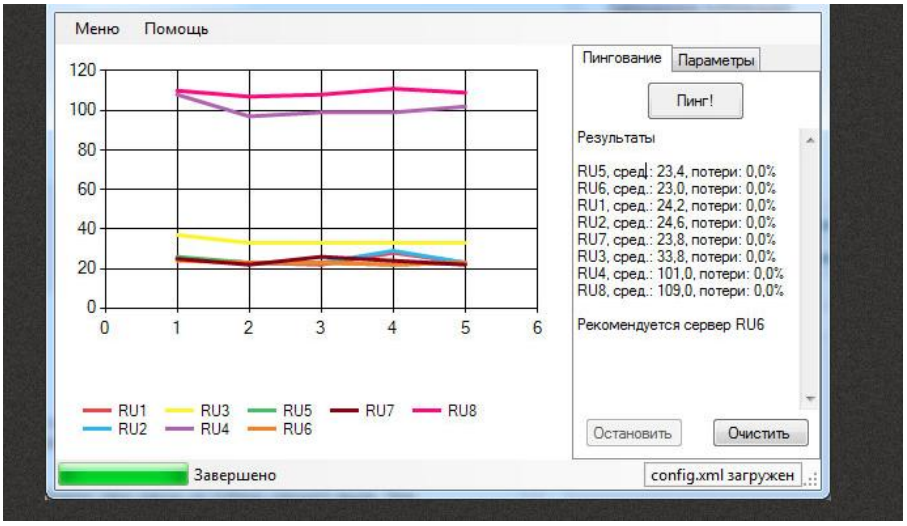

Рисунок 22 - Работоспособность

Чтобы обеспечить бесперебойную работу информационой корпоративной сети необходимо отладить работу интернет-соединения и подобрать количество провайдеров, рекомендуемое для надежного интернет соединения, предоставляющих доступ в через сеть (проводное подключение). Обычно выбирается примерно два провайдера, в большинстве технических решений. К примеру, через основной канал (первого провайдера) возникают проблемы с доступом в интернет, маршрутизаторы должны переключатся на резервный канал (второго провайдера) автоматически, и продолжать работу в штатном режиме с восстановленным соединением. По доступности ip-адреса будем судить о работоспособности канала. Для решения таковой задачи так же подходит наша разработка. Адрес крупного «web-сервиса» берётся для проверки, вероятность отказа такового (например, google dns) крайне мала.
К примеру "Netwatch" утилита в большинстве роутерах встроенная изначально, к примеру "Netwatch", позволяющая мониторить состояние хостов в сети, однако недостаточная гибкость отсутствие и наглядного визуализированного интерфейса выделяет нашу разработку.

(ICMP) Internet Control Message Protocol — В распространённых операционных системах (linux, windows) это протокол межсетевых управляющих сообщений, в виде команды «ping» данный протокол и реализован. Протокол сетевого уровня модели OSI/ISO – это и есть ICMP. Базовая Эталонная Модель Взаимодействия Открытых Систем) OSI - open systems interconnection basic reference model, таковой имеет 7 логических уровней работы сети:

Пример работы ICMP-протокола (команды Ping) имеет следующий вид: Синтаксис команды - ping google.com;

- Ответ хоста
- Pinging google.com [172.217.16.14] with 32 bytes of data;
- Reply from 172.217.16.14: bytes=32 time=99ms TTL=55;
- Reply from 172.217.16.14: bytes=32 time=92ms TTL=55;
- Reply from 172.217.16.14: bytes=32 time=86ms TTL=55;

– Reply from 172.217.16.14: bytes=32 time=91ms TTL=55;

– Статистика показателей доступности

– Ping statistics for 172.217.16.14:

– Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

– Approximate round trip times in milli-seconds:

– Minimum = 86ms, Maximum = 99ms, Average = 92ms

Разберем параметры ответы: reply from 172.217.16.14*:* - IP-адрес хоста; bytes=32 – размер эхо-ответа в байтах; time=99ms – время отклика

TTL=55 - максимальное количество маршрутизаторов, которое еще может быть пройдено пакетом

Packets: Sent = 4, Received = 4, Lost = 0 (0% loss), количество отправленных (Sent) пакетов, количество полученных (Received) пакетов, процент потерь (Lost).

Типы сообщений ICM-протокола:

– 0(Echo Reply) – Ответ на эхо;

– 3(Destination Unreachable) - Узел назначения недостижим;

– 4 (Source Quench)- Подавление источника;

– 5 (Redirect)- Перенаправление маршрута;

– 8(Echo Request) – Запрос эха;

– 9(Router Advertisement) - Информация о маршрутизаторах;

– 10 (Router Solicitation) - Регистрация маршрутизатора;

– И (Time Exceeded for a Data- gramm)- Лимит времени для дейтаграммы превышен;

– 12 (Parameter Problem on a Datagramm)- Проблема с параметром пакета;

– 13(Timestamp Request) - Запрос метки времени;

– 14(Timestamp Reply) - Ответ для метки времени;

– 15(не действует) - Запрос информации;

– 16(не действует) - Ответ на запрос информации;

– 17(Address Mask Request) - Запрос маски адреса;

– 18 (Address Mask Reply) - Ответ на запрос маски адреса.

Правила генерации ICMP – пакетов:

– Никогда не генерируется новый при потере ICMP-пакета;

– Чтобы не вызывать перегрузку в сети ICMP-пакеты никогда не генерируются в ответ на IP-пакеты с широковещательным или групповым адресом;

– В случае повреждения фрагментированного IP-пакета, только после получения первого повреждённого фрагмента ICMP-сообщение отправляется, потому как отправитель всё равно повторит передачу всего IP-пакета целиком.

Включение ICMP: Пуск -> Панель Управления -> Брандмауэр Windows -> Дополнительно -> Параметры ICMP

ЗАКЛЮЧЕНИЕ

Цель выпускной квалификационной работы, которая заключалась в проектировании системной утилиты для конкретного предприятия, с его требованиями, была достигнута. В процессе создания программного обеспечения были изучены аналогичные программы, проведен сравнительный анализ, в итоге которого, определены параметры, по которым не подойдут ныне имеющиеся на рынке на сегодняшний день программы, это помогло принятию решения по поставленным задачам, а именно основными параметрами не удовлетворяющих нашим требованиям было несколько : первое и самое важное требование было отправка уведомлений по смс или в мессенджер (предпочтительнее), из соображения мобильности системного администратора; второе это простота, надежность и быстродействие системы языка программирования, что тоже не маловажно исходя из выдвинутых требований, скорости реакции утилиты на отказ работы узлов локальной сети и обработки сигналов; и наконец третье требование (присутствующее в некоторых программа аналогах) настройка запуска программы вместе с «Windows» и работа её в «трее», реализована в нескольких программах аналогах но два других выше перечисленных там отсутствуют.

Исходя из требований программы была выбрана продукционная модель построения системы, в среде объектно - ориентированного программирования языка «Delphi» на платформе «Delphi 10 Seattle». Требования программы удалось реализовать в спроектированной утилите «Pinger»

Можем подвести итог успешного внедрения программного обеспечения на предприятии ООО «ИК «А-ИНЖ». На данном предприятии очень «остро» стоял вопрос по своевременному оповещению о недоступности локальных узлов сети и как следствие материальных потерь, которые несла компания, в связи с более долгим устранением их поломки.

После внедрения программной утилиты, системный администратор стал быстрее реагировать на отказы оборудования локальной сети компании и устранять поломку в короткие сроки, что способствовало: увеличению прибыли компании, уменьшению время простоя оборудования и отдельно каждого сотрудника компании, для администратора локальной сети компании было приятным дополнением (подспорьем) для удобства работы утилиты, ее настройки – по времени, интервалу команды «Ping» и оповещений. К слову он получал все нужные данные по пакетам и времени обрыва и восстановления связи, что тоже улучшило «КПД» его работы.

Исходя из всего изложенного, делаем вывод, что нам удалось реализовать поставленные задачи на выпускную квалификационную работу, а значит мы справились с заданием!

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Семкин, С. Н. Информационные автоматизированные системы. АИС Основы: курс лекций / С. Н. Семкин – Орел.: ВИПС, 2000. – 299 с.
2. Осипов, Д. Delphi. Профессиональное Программирование / Д.Осипов. – Санкт-Петербург: Символ-плюс, 2015. – 1006 с
3. Александр, Днепров Видео - самоучитель. «Microsoft Access 2007» (+ CD-ROM) / Днепров Александр. - Москва: Санкт - Питербург, 2008. - **902** с
4. Аверченков, В. И. Информационные системы на производстве: учебное пособие для вузов. / Ф. Ю. Лозбинев, А. А. Тищенко. – Москва: «Флинт» Издательский центр, 2011. – 247 с.
5. СТАНДАРТ ОРГАНИЗАЦИИ Система менеджмента качества Общие требования к построению, изложению и оформлению, документов учебной деятельности СТО 4.2–07–2014 – Красноярск: ИПК СФУ, 2014. – 60с
6. Пестриков, В. М. Delphi на примерах / В. М. Пестриков, А. Н. Маслобоев. — Санкт-Петербург: БХВ-Петербург, 2005. — 496 с.
7. Фленов М. Е. Библия Delphi / М. Е. Фленов. — 3-е издание, переработал и дополнил – Санкт-Петербург, 2008. — 800 с.
8. Хомоненко А., Гофман В. Москва «Delphi Самоучитель». 2005г. с.507
9. Куров А.В., Исаев А.Л., Машинная графика в среде программирования «Delphi»: Учебное пособие. – Москва,: Изд-во МГТУ им. Н. Э. Баумана, 2006. -65 с.: ил.
10. Колбин Р. В. Глобальные и локальные сети. Бином. Создание, настройка и использование (+ CD); Лаборатория знаний - Москва, **2012**. - 224 с
11. Фролов, А.В.; Фролов, Г.В. Работа с сервером «Novell»

«NetWare». Локальные сети персональных компьютеров; Диалог-Мифи - Москва, **2013**. - 168 с.

12. Новиков Ю.В., Кондратенко С.В. Основы локальных сетей. – Москва,: 2005

13. Санников, Е. В. Объектно-ориентированное программирование. Курс практического программирования в Delphi. / Е.В. Санников. – Москва,: Солон-Пресс, 2013. - 138 с.

14. Культин, Н. Delphi 6. Программирование на Object Pascal / Н. Культин. - Санкт-Петербург, 2002. - 528 с