

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

Специальность: 09.05.01 «Применение и эксплуатация
автоматизированных систем специального
назначения»

Специализация: «Эксплуатация вычислительных машин, комплексов,
систем и сетей специального назначения»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ
РАБОТА СПЕЦИАЛИСТА

на тему

Программное обеспечение технической поддержки
коммутационного оборудования

Студент

Кош
(подпись, дата)

Кошлаков А.С.

Руководитель

Фед
(подпись, дата)

Федюнин Р.Н.

Нормоконтролёр

Ку
(подпись, дата)

Кучин А.В.

Рецензент

(должность, место работы)

Васильков
(подпись, дата)

Васильков А.В.

Работа допущена к защите (протокол зас. кафедры от 10.06.2020г. № 13)

Заведующий кафедрой

Митрохин
(подпись)

Митрохин М.А.

Работа защищена с отметкой

18.06.2020 № 6

Отлично

(протокол заседания ГЭК от

Секретарь ГЭК

Гурин
(подпись)

Гурин Е.И.

Пенза 2020

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
Вычислительная техника

«Утверждаю»
Заведующий кафедрой
 М.А. Митрохин
«30» апреля 2020 г.

ЗАДАНИЕ
ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ
РАБОТА СПЕЦИАЛИСТА

1. Студент Кошляков Александр Сергеевич гр. 15BO2 факультета
вычислительной техники.

2. Тема работы: «Программное обеспечение технической поддержки
коммутационного оборудования».

Тема утверждена приказом ПГУ № 0335а/о от 30.04.2020 г.

3. Руководитель работы к.т.н., доцент каф. ВТ ПГУ Федюнин Р.Н.

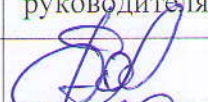
4. Задание на работу (назначение разработки, исходные данные и т.п.).

Назначение разработки – Система предназначена для разработки клиентского приложения работающего в качестве утилиты индивидуальной и/или массовой настройки сетевых устройств. Сетевые устройства работают под управлением операционной системы RouterOS В системе необходимо предусмотреть работу с графическим интерфейсом, СУБД, Mikrotik API и скриптами RouterOS.

5. Перечень подлежащих разработке вопросов.

Требуется произвести анализ предметной области, выбрать функциональный состав системы, осуществить выбор стека технологий для реализации системы, разработать алгоритм работы системы, разработать структурную схему системы, разработать диаграммы: прецедентов, активностей, последовательностей, объектов, классов, с последующим написанием программы.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	85	05.03.20	
2.	Выбор компонентов для реализации принципиальной схемы	95	01.04.20	
3.	Составление принципиальной схемы устройства	95	10.04.20	
4.	Написание программы-прошивки для устройства	180	27.04.20	
5.	Отладка системы	100	28.04.20	
6.	Моделирование и поиск уязвимостей, исправление ошибок в работе системы	104	04.05.20	
7.	Оформление выпускной работы	205	01.06.20	

Дата выдачи задания 30 «апреля» 2020 г.

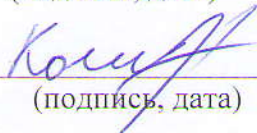
Руководитель бакалаврской работы



(подпись, дата)

Федюнин Р.Н

Задание к исполнению принял студент



(подпись, дата)

Кошляков А.С.

Работу к защите допустить

Декан факультета



(подпись, дата)

Фионова Л.Р.

Реферат

Работа содержит 134 листов, 50 рисунков, 6 источников, 3 приложения.

УТИЛИТА, СЕТЕВОЕ ОБОРУДОВАНИЕ, RouterOS, Mikrotik API, скрипты сетевого оборудования.

Цель работы – разработать мобильное приложение для рутинной настройки сетевого оборудования на базе ОС RouterOS.

Объектом разработки является программное приложение, обеспечивающее клиентам доступ к сетевым приложениям компании Mikrotik и скриптам настройки данного оборудования.

В результате проведенной работы должен быть разработан проект, удовлетворяющий поставленной задаче, а именно, создан программный продукт – клиент, бизнес логика и хранилище скриптов настройки RouterOS. Необходимо было разработать алгоритмы, реализующие возможность хранить, обрабатывать и доставлять до сетевых устройств информацию в виде сообщений (скриптов), с последующим сохранением нового состояния устройства и отработки им заданных режимов.

Данное приложение рекомендуется использовать для индивидуальной и/или массовой настройки сетевого оборудования Mikrotik. Система может быть внедрена в системе администрирования сетевого оборудования на технологических или тестовых площадках.

					ПГУ 1.090501.07.001 ПЗ			
Изм.	ЛистЛ	№ докум.№	Подпись	Дата	Программное обеспечение технической поддержки коммутационного оборудования Пояснительная записка	Лит.Лит.	ЛистЛис	ЛистовЛисто
Разраб.		Кошлаков А.С.						
Провер.		Федюнин Р.Н.					2	125
Н. Контр_Н.		Кучин А.В.				ФВТ, каф. ВТ гр. 15В02		
Утверд.		Митрохин М.А.						

Содержание

ВВЕДЕНИЕ.....	7
1 Анализ предметной области.....	9
1.1 Сбор материала по теме проекта.....	9
1.2 Исследование стека технологий проекта.....	11
1.2.1 Операционная система RouterOS.....	11
1.2.2 Средства автоматизации.....	18
1.2.3 Средства программирования.....	20
2 Разработка системы.....	37
2.1 Разработка скриптов RouteOS.....	39
2.2 Разработка СУБД.....	54
2.2.1 Концептуальное проектирование.....	55
2.2.2 Дatalogическое проектирование.....	56
2.3 Разработка клиентского приложения.....	62
2.4 Развертывание приложения.....	70
2.5 Описание работы приложения.....	71
3 Разработка клиентского приложения.....	78
4 Экономическое обоснование разработки мобильного приложения.....	86
4.1 Обоснование выбора объекта.....	86
4.2 Расчет затрат на модернизацию мобильного приложения.....	88
4.2.1 Расчет затрат на оплату труда.....	88
4.2.2 Расчет отчислений в страховые фонды.....	89
4.2.3 Расчет накладных расходов.....	90
4.2.4 Расчет амортизационных отчислений.....	90
4.2.5 Расчет затрат на обслуживание оборудования.....	92
4.2.6 Расчет затрат на электроэнергию.....	92
4.2.7 Расчет стоимости модернизации системы.....	93
4.2.8 Расчет цены программы.....	93
4.3 Расчет эксплуатационных расходов.....	94
4.4 Расчет денежного потока.....	92
5 Гигиена труда в системе мероприятий по сохранению высокой	

трудоспособности пользователя ПЭВМ.....	100
5.1.1 Организация рабочего места пользователя ПЭВМ.....	101
5.1.2 Освещенность рабочего места.....	105
5.1.3 Параметры микроклимата на рабочем месте.....	106
5.1.4 Методы защиты от шума.....	108
5.2 Вывод.....	109
Заключение.....	110
Список используемых источников.....	111
Приложение А.....	112
Приложение В.....	122
Приложение С.....	128

Введение

Современная сфера связи демонстрирует сильную интеграцию инструментов, ассоциируемых со сферой разработки ПО, в область телекоммуникаций. В частности, при эксплуатации систем связи всё чаще используются программные средства автоматизации.

Использование подобных инструментов и методов имеет ряд особенностей и применимо не во всех реализациях:

внедрение средств автоматизации снижает затраты на фонд оплаты труда сотрудников за счёт снижения их числа, однако повышенные требования к профессиональным навыкам сотрудников увеличивает уровень заработной платы этих сотрудников;

усложнение инструментов и увеличение их числа требует постоянного профессионального развития сотрудников. Зачастую им приходится решать нетривиальные задачи, что снижает уровень рутины и повышает заинтересованность;

замена ручных средств эксплуатации на автоматизированные снижает уровень ошибок и, как следствие, повышает надёжность и доступность сервиса. Влияние данного аспекта увеличивается с ростом числа устройств в сети.

Таким образом, использование средств автоматизации в небольших компаниях, как правило, обеспечивается инициативностью технических специалистов, тогда как в крупных компаниях является необходимой мерой.

Под сетевыми задачами, решаемыми средствами автоматизации, в рамках этой статьи понимаются три типа задач:

контроль и обновление программного обеспечения устройств;

анализ и применение унифицированной конфигурации на устройствах с возможностью их изменения/дополнения;

					ПГУ 1.090501.07.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		5

контроль за состоянием устройств и каналов связи и оперативная реакция на возникающие события.

Данная работа предназначена для разработки клиентского приложения работы со скриптами настройки сетевого оборудования Mikrotik.

					ПГУ 1.090501.07.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		6

1 Анализ предметной области

1.1 Сбор материала по теме проекта

Современное сетевое оборудование представляет собой симбиоз аппаратной части и прошивки (скриптов и/или операционной системы). В данном проекте будет эксплуатироваться операционная система RouterOS. Выбор данной операционной системы вызван анализом существующих предложений на рынке операционных систем для работы сетевого оборудования (таблица 1.1).

В настоящий момент рынок операционных систем для сетевых устройств достаточно разнообразен и представлен как минимум следующими ОС: RouterOS, DD-Wrt, OpenWrt, LEDE, VyOS. Краткий сравнительный анализ ОС приведен в таблице 1.1, ниже сделаем краткий вводный экскурс в каждую из них.

MikroTik RouterOS является основным продуктом MikroTik (система на основе Linux).

Установленный на персональном компьютере или сервере, MikroTik превращает его в сетевой маршрутизатор, используя такие функции, как:

- правила брандмауэра,
- сервер и клиент виртуальной частной сети (VPN),
- формирование пропускной способности и качество обслуживания,
- функции беспроводной точки доступа,
- функции для маршрутизации и соединения сетей.

Программное приложение MikroTik Winbox предоставляет графический интерфейс пользователя для конфигурации и мониторинга RouterOS. ПО позволяет подключаться через FTP, telnet и защищенную оболочку (SSH).

					ПГУ 1.090501.07.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		7

Прикладное программирование доступно для прямого доступа из пользовательских приложений для управления и мониторинга.

DD-WRT - прошивка OpenSource на основе Linux, подходящая для большого количества маршрутизаторов и встроенных систем. Основной упор делается на обеспечение максимально простой обработки при одновременной поддержке большого числа функций в рамках используемой аппаратной платформы.

OpenWRT - нишевый дистрибутив Linux, в основном устанавливаемый на встроенные устройства. ОС реализована на основе ядра Linux и состоит из набора разнообразных пакетов программного обеспечения. Для легкой установки и удаления пакетов используется система управления пакетами `opkg`. OpenWrt в основном использует интерфейс командной строки `sh` или (необязательно) `bash`, но также имеет дополнительные веб-интерфейсы GUI, такие как LuCI или X-Wrt или встроенное программное обеспечение Gargoyle Router.

LEDE (Linux Embedded Development Environment) - встроенный дистрибутив Linux необходим для создания и настройки программного обеспечения встроенных устройств, особенно беспроводных маршрутизаторов.

Проект LEDE был настроен иначе, чем OpenWrt:

каналы связи являются общедоступными, некоторые из них предназначены только для чтения,

процесс принятия решений открыт, примерно 50/50 разработчиками и опытными пользователями,

инфраструктура упрощена, чтобы уменьшить затраты на техническое обслуживание,

проект общедоступен на github,

					ПГУ 1.090501.07.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		8

наличие автоматического тестирования в сочетании с упрощенным процессом выпуска своих прошивок.

Исходя из потребностей проекта выбор брендовой операционной системы Mikrotik позволит создать полноценное клиентское приложение для настройки аппаратного обеспечения.

1.2 Исследование стека технологий проекта

Стек технологий это набор инструментов, применяющийся при работе в проектах и включающий языки программирования, фреймворки, системы управления базами данных, компиляторы и т. д.

От выбранного разработчиком стека технологий зависят производительность работы, требования к аппаратным ресурсам, надежность работы программного обеспечения (ПО).

1.2.1 Операционная система RouterOS

Mikrotik RouterOS (рисунок 1.1) является операционной системой, которая ставится непосредственно на роутеры бренда RouterBoard. RouterBoard - это бренд, под которым MikroTik производит свои собственные роутеры.

В то же время Mikrotik RouterOS может быть установлена и на компьютеры, что позволит превратить компьютер в многофункциональный роутер. Маршрутизация, Firewall, VPN сервер, беспроводная точка доступа - все эти, и многие другие функции могут быть реализованы на компьютере под управлением этой ОС.

За основу RouterOS взято ядро Linux. Система специально разрабатывалась так, чтобы программное обеспечение могло быстро и без проблем настроить все функции, необходимые для работы компьютера в качестве роутера.

Таблица 1.1. – Сравнение сетевых операционных систем.

Функционал	RouterOS	DD-Wrt	OpenWrt	LEDE	VyOS
Поддерживаемое аппаратное обеспечение	Многоядерные и мультипроцессорные ПК IDE, SATA и USB флеш-карты памяти, HDD, CF, SD, SDD	Многоядерные и мультипроцессорные ПК IDE, SATA и USB	Многоядерные и мультипроцессорные ПК IDE, SATA и USB	Многоядерные и мультипроцессорные ПК IDE, SATA и USB	Многоядерные и мультипроцессорные ПК IDE, SATA и USB
Настройка	ПК Терминал Telnet SSH утилита с GUI Microtik API	ПК Терминал Telnet SSH	ПК Терминал Telnet SSH	ПК Терминал Telnet SSH	ПК Терминал Telnet SSH
Firewall	SPI source-NAT и destination-NAT UPnP	source-NAT и destination-NAT UPnP	source-NAT и destination-NAT UPnP	source-NAT и destination-NAT UPnP	source-NAT и destination-NAT UPnP
Маршрутизация	IPv4 RIP версии 1 и версии 2, OSPF v2, BGP v4. IPv6 RIPng, OSPF v3, BGP	IPv4 RIP версии 1 и версии 2, OSPF v2. IPv6 RIPng, OSPF v3	IPv4 RIP версии 1 и версии 2, OSPF v2. IPv6 RIPng, OSPF v3	IPv4 RIP версии 1 и версии 2, OSPF v2. IPv6 RIPng, OSPF v3	IPv4 RIP версии 1 и версии 2, OSPF v2. IPv6 RIPng, OSPF v3
Переадресация	Сетевые мосты, мещ и WDS	Сетевые мосты, WDS	Сетевые мосты, WDS	Сетевые мосты, WDS	Сетевые мосты, WDS

Продолжение таблицы 1.1

Функционал	RouterOS	DD-Wrt	OpenWrt	LEDE	VyOS
MPLS	Поддерживается	Поддерживается	Поддерживается	Поддерживается	Поддерживается
VPN	Поддерживается	Поддерживается	Поддерживается	Поддерживается	Поддерживается
Беспроводное соединение	Точка доступа Hotspot mesh-сеть	Точка доступа Hotspot	Точка доступа Hotspot	Точка доступа Hotspot	Точка доступа Hotspot
Hotspot	Поддерживается	Поддерживается	Поддерживается	Поддерживается	Поддерживается
QoS (управление ресурсами сети)	Поддерживается	Поддерживается	Поддерживается	Поддерживается	Поддерживается
Веб-прокси	Поддерживается	Поддерживается	Поддерживается	Поддерживается	Поддерживается
Инструменты сетевого администрирования	Поддерживается	Поддерживается	Поддерживается	Поддерживается	Поддерживается
Система мониторинга сети DUDE	Поддерживается	Отсутствует	Отсутствует	Отсутствует	Отсутствует
Лицензирование	платное	бесплатное	бесплатное	бесплатное	бесплатное

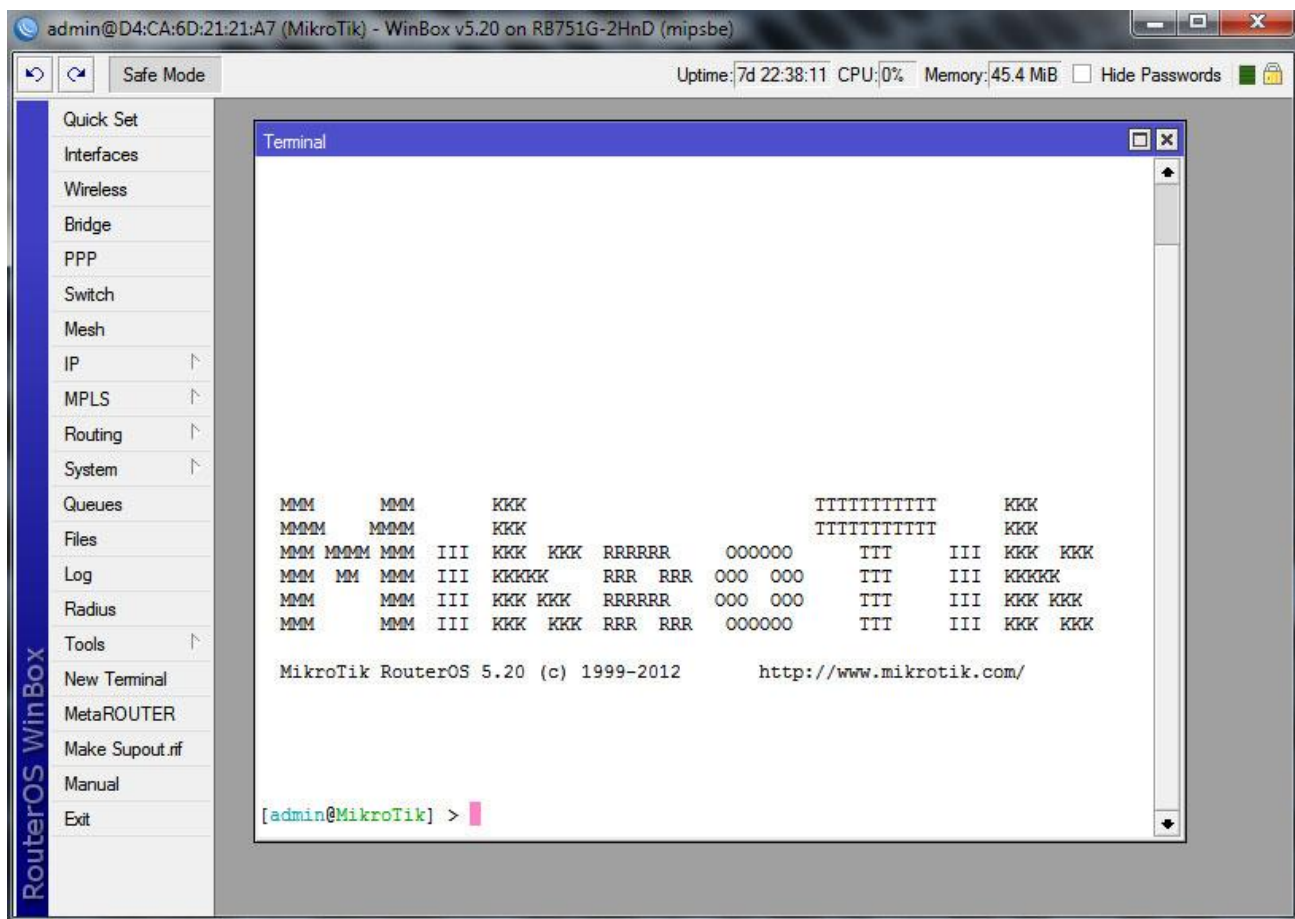


Рисунок 1.1 – Первая страница операционной системы RouterOS

RouterOS поддерживает многоядерные и мультипроцессорные компьютеры, так что можно установить эту систему на самые новые и мощные процессоры и платы компании Intel. RouterOS поддерживает установку на IDE, SATA и USB флеш-карты памяти, в том числе и на HDD, CF и SD карты, SSD диски и другие типы носителей. Для установки RouterOS требуется по крайней мере 64 MB свободного места. При установке раздел диска будет отформатирован, и RouterOS по умолчанию станет основной операционной системой, используемой компьютером. RouterOS поддерживает множество всевозможных проводных интерфейсов, в том числе 10-гигабитные сетевые карты, 802.11 a/b/g/n Wi-Fi модули и 3G модемы.



Рисунок 1.2 – объекты установки RouterOS

RouterOS поддерживает различные способы настройки: настройка непосредственно на компьютере, настройка через консольный порт при помощи терминала, Telnet и доступ по сети через защищенный протокол SSH, настройка с помощью специальной утилиты с графическим интерфейсом, которая называется Winbox, настройка через интерфейс, построенный на коде реализованных приложений. Также имеется API, который позволит построить систему управления компьютером/роутером.

RouterOS предоставляет простую для изучения командную строку с возможностью использования скриптов. Непосредственно в RouterOS v4 начал использоваться скриптовый язык Lua, который предоставляет обширные возможности для программирования и автоматизации роутера.

Firewall осуществляет фильтрацию пакетов, что позволяет ему обеспечивать безопасность сети, контролируя входящие и исходящие данные. Вместе с NAT это позволяет предотвратить несанкционированный доступ к сети извне и контролировать исходящий трафик.

RouterOS поддерживает технологию SPI, которая позволяет запомнить состояние текущих подключений и избежать таким образом подключения к

порту злоумышленников. Также система поддерживает source-NAT и destination-NAT, и набор сервисных протоколов UPnP.

Firewall может производить фильтрацию информации: по конкретному IP-адресу, по диапазону IP-адресов, порту, диапазону портов, IP протоколу, DSCP и другим параметрам, также поддерживая статические и динамические списки адресов. Firewall может сравнивать пакеты по общим закономерностям в их содержимом. (Layer 7 matching) IPv6 также поддерживается.

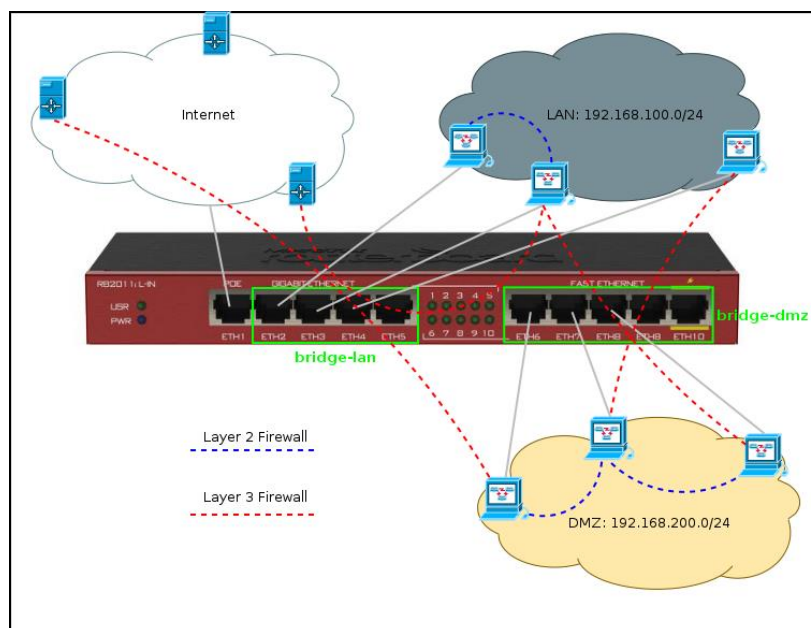


Рисунок 1.3 – Firewall в системе MikroTik

Поддерживаются следующие протоколы маршрутизации:

для IPv4 RIP версии 1 и версии 2, OSPF v2, BGP v4.

для IPv6 RIPng, OSPF v3, BGP

RouterOS поддерживает VRF (виртуальная маршрутизация и переадресация), маршрутизацию на основании политики, маршрутизацию на основании интерфейса и ECMP маршрутизацию. Вы можете использовать Firewall чтобы отметить определённые соединения с маршрутизационными метками, после чего заставить отмеченный таким образом трафик

использовать другой ISP. VRF - это технология, которая позволяет использовать сразу несколько таблиц адресов, у которых имеется общие IP-адреса, не вызывая при этом конфликтов между этими таблицами. К тому же VRF повышает уровень безопасности сети. VRF часто используется вместе с так же поддерживаемым MPLS, но может быть использоваться и с другими протоколами.

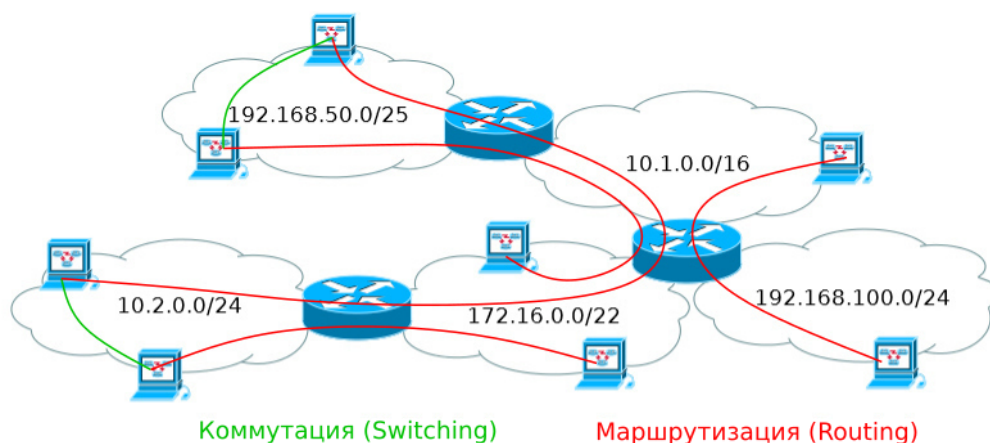


Рисунок 1.4 – Маршрутизация на основе Mikrotik

Инструменты сетевого администрирования RouterOS обладает набором инструментов и возможностей, позволяющих упростить и автоматизировать управление сетью:

команды ping и traceroute

тестирование сетевой способности, ping flood

анализатор трафика, torch

telnet, SSH протоколы

инструменты SMS, e-mail рассылки

автоматизированные скрипты выполнения программы

CALEA зеркалирования данных

file fetch

таблица активных соединений

NTP клиент и сервер

TFTP сервер

динамическое обновление DNS

поддержка VRRP избыточности

SNMP для предоставления графиков и статистики

программа Radius для менеджмента пользователей (клиент и сервер).

1.2.2 Средства автоматизации

В рамках данной статьи будут описаны следующие средства автоматизации: шаблоны конфигураций, NMS/EMS, скрипты, выполняемые на устройствах с RouterOS, скрипты, выполняемые на локальном устройстве.

Шаблоны конфигураций

Под шаблонами конфигураций понимается сценарий, в котором администратор заранее составляет несколько шаблонов для конфигураций устройств, каждый из шаблонов дополняет уникальными данными, например IP-адресом, и применяет их на устройствах вручную. Использование того или иного шаблона может определяться выполняемой функциональностью устройства в сети, географическим местоположением или другим критерием.

Недостатком данного метода является то, что объём ручного труда остаётся достаточно большим. Кроме того, трудно поддерживать такую инфраструктуру в случае необходимости внесения в конфигурацию каких-либо изменений, однако, в начале эксплуатации, конфигурация оборудования будет унифицирована.

Система управления NMS/EMS (DUDE)

Системы управления устройствами отлично справляются с задачей контроля за состоянием устройств и каналов связи, а также обновление программного обеспечения. Многие системы позволяют выполнять предварительную настройку в соответствии с заданными шаблонами.

Особенностью NMS является вендорнезависимость, т. е. для формирования рабочего сервера управления устройствами понадобятся большие трудозатраты по организации рабочей схемы управления конфигурацией с заданным производителем оборудования. EMS вендорозависимы, что, в свою очередь, не позволит использовать такие системы для инфраструктуры, построенной с использованием оборудования разных производителей.

Система управления TR-069

Операционная система RouterOS поддерживает спецификацию TR-069, описывающую протокол управления абонентским оборудованием через глобальную сеть. Изначально протокол разрабатывался для операторов связи, однако может быть использован и в корпоративных сетях.

Архитектура решения предусматривает выделение сервера конфигурации, который, взаимодействуя с абонентскими устройствами по протоколу CWMP, выполняет сбор данных и конфигурацию. Устройства, находящиеся под управлением RouterOS, могут выступать в роли клиентских устройств. Для этого потребуется дополнительная установка и конфигурация пакета tr069.

Достоинством такого решения является вендорнезависимость, т.к. спецификация предусматривает возможность унифицированной настройки оборудования разных производителей, а среди недостатков можно выделить высокий порог вхождения при освоении навыков написания конфигурационных скриптов.

Скрипты, выполняемые на устройствах с RouterOS

RouterOS позволяет запускать предварительно загруженные скрипты, написанные для данной системы. Скрипты MikroTik обладают гибкостью, т. к. в них может быть настроен сбор сведений об устройстве и выполнение каких-то команд в соответствии с анализом этих данных. Кроме того,

возможности RouterOS позволяют выполнить скрипт, как реакцию на возникшее событие или запускать его по расписанию.

Недостатком этого метода является плохая масштабируемость и возможность редактирования: скрипт должен быть унифицирован, чтобы однозначно был выполнен на устройствах разных моделей, а изменение скрипта потребует последовательную загрузку исполняемого файла на все устройства с получением отчёта о выполняемых операциях, т. к. часть устройств в момент загрузки могут быть недоступны.

Скрипты, выполняемые на локальном устройстве

Данный метод подразумевает запуск скрипта на локальном устройстве администратора, который выполняет подключение к заданному списку устройств с RouterOS (операции могут быть выполнены как последовательно, так и параллельно) с помощью одного из методов (telnet, SSH, API) и выполнение заданных команд. Выполнение команд может сопровождаться получением и анализом данных об устройстве.

Данный метод является достаточно гибким и может быть реализован с помощью одного из распространённых языков программирования. Это выгодно отличает его от запусков скриптов на устройствах, т.к. написания исполняемых файлов для RouterOS потребует знать специфический синтаксис.

Этот инструмент хорошо масштабируется и позволяет получить отчёт об успешности выполнения изменений в конфигурации.

В рамках данной статьи будет рассмотрен вариант подключения через API и использование одной из систем управления конфигурациями. В качестве языка для написания скриптов будет выбран C#, а в качестве системы управления конфигурациями Ansible.

1.2.3 Средства программирования

Сравнение инструментов программирования

					ПГУ 1.090501.07.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		18

Какой из общецелевых, объектно-ориентированных языков программирования лучше всего подойдет реализации клиентского приложения проекта в сравнении C# и Java.

Они оба имеют обширные библиотеки, которые можно использовать для создания приложений для ПК, веб, мобильных устройств и других платформ. У обоих есть большие сообщества восторженных поклонников и много онлайн-поддержки.

Имея так много общего, выбор правильного языка требует более тонкого, тщательного подхода. Ниже приводится сравнительный анализ C# и Java.

Общее описание языка C#

Язык программирования C# это язык программирования общего назначения, который появился в рамках инициативы Microsoft .NET. Он был разработан для общей языковой инфраструктуры (CLI) – открытой спецификации, разработанной Microsoft и стандартизированной ISO и ECMA. Приложения C# скомпилированы в байт-код, который может запускаться при реализации CLI.

Общее описание языка JAVA

Язык программирования Java является языком программирования общего назначения, который был разработан с конкретной целью, позволяющей разработчикам “write once, run anywhere”, то есть написать код единожды и запускать в любом месте. Java-приложения скомпилированы в байт-код, который может запускаться при реализации виртуальной машины Java (JVM). Подобно CLI, JVM помогает преодолеть разрыв между исходным кодом и 1 и 0, которые понимает компьютер.

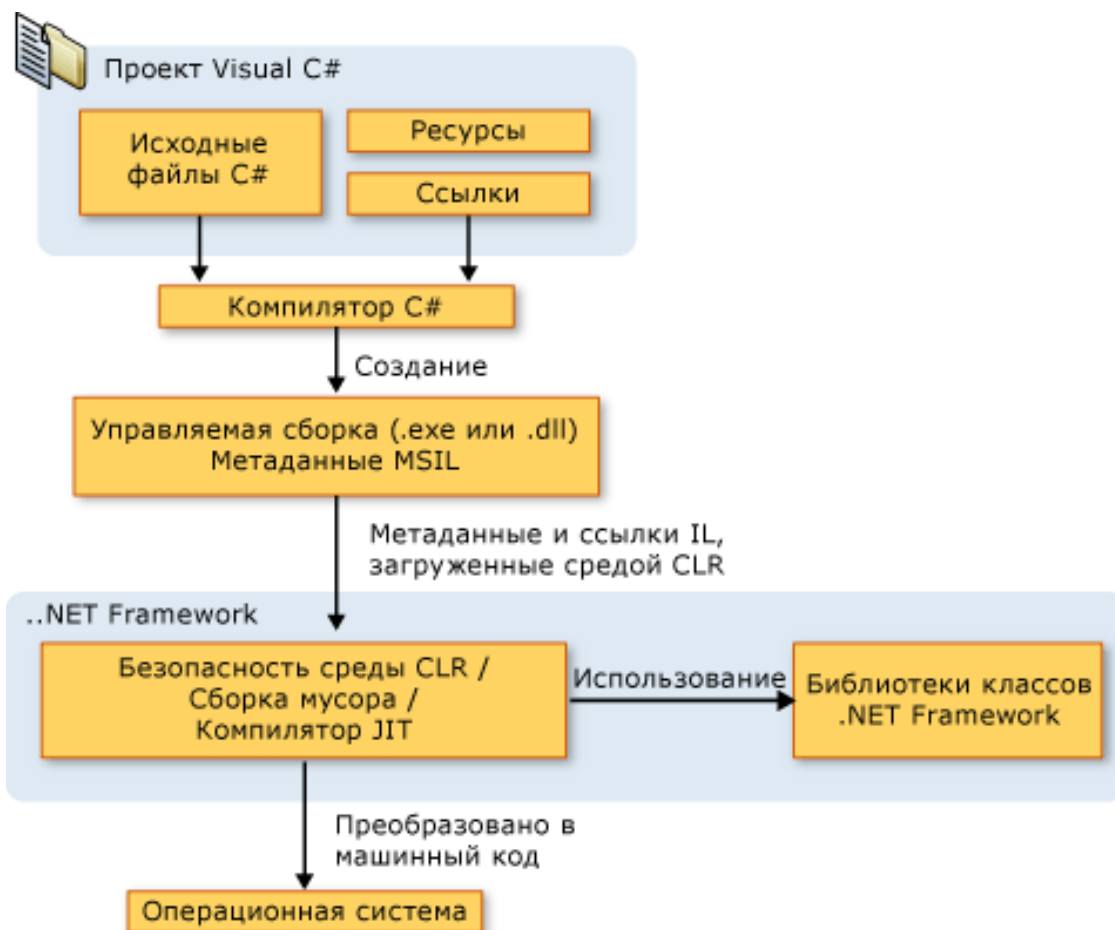


Рисунок 1.5 – Архитектура приложения на базе .NET

Основные сходства C# и JAVA

Появление как Java, так и C#, тесно связано с переходом от низкоуровневых языков программирования, таких как языки программирования C++, к языкам более высокого уровня, которые компилируются в байт-код. Байт-код можно запустить на виртуальной машине. С этим связан ряд преимуществ, в первую очередь, возможность написания кода, который будет понятен человеку и будет работать на любой аппаратной архитектуре, на которой установлена виртуальная машина. Если отбросить синтаксические причуды в сторону, то неудивительно, что эти два подобных между собой языка так популярны для разработчиков приложений.

Безопасность типов. Ошибка типа возникает, когда тип данных одного объекта ошибочно назначается другому объекту, создавая непреднамеренные побочные эффекты. И С#, и Java работают на то, чтобы гарантировать выявление таких типов незаконных приведений во время компиляции. Если приведение не может быть применено к новому типу, тогда во время выполнения такие исключения будут удалены.

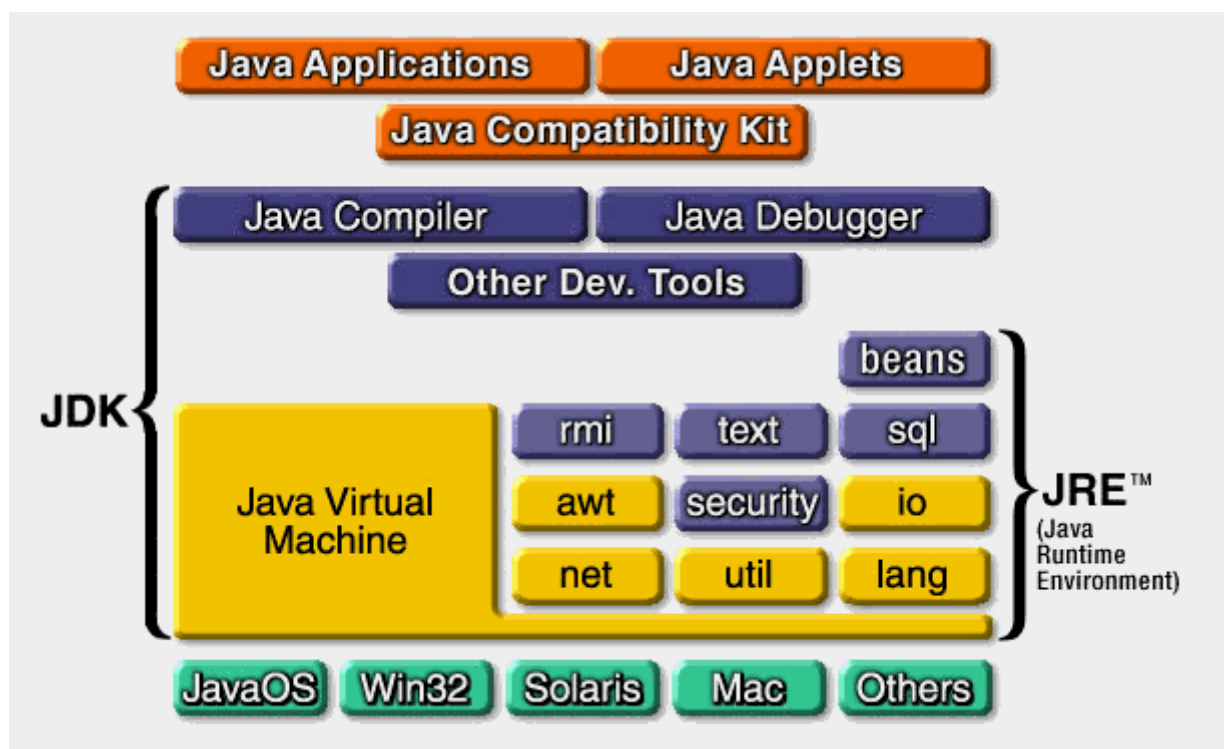


Рисунок 1.6 – Архитектура приложения на базе Java

Сборка мусора. На языках более низкого уровня управление памятью может быть утомительным, ведь нужно помнить о том, что необходимо правильно удалить новые объекты, чтобы освободить ресурсы. На С# и Java есть встроенная сборка мусора, которая помогает предотвратить утечку памяти путем удаления объектов, которые больше не используются приложением. Утечки памяти все еще могут возникать, но благодаря основам управления памятью, очистка памяти будет осуществляться автоматически.

Одиночное наследование. Оба языка поддерживают одиночное наследование – это означает, что существует только один путь из любого базового класса в любой из его производных классов. Это ограничивает непреднамеренные побочные эффекты, которые могут возникать при наличии нескольких путей между несколькими базовыми классами и производными классами.

Интерфейсы. Интерфейс представляет собой абстрактный класс, где все методы абстрактны. Абстрактным методом является тот метод, который объявлен, но не содержит подробностей его реализации. Код, определяющий любые методы или свойства, определенные интерфейсом, должен предоставляться классом, который его реализует. Это помогает избежать двусмысленности паттерна diamond, поскольку всегда ясно, какой базовый класс реализует данный производный класс во время выполнения. Результатом является чистая иерархия линейных классов одиночного наследования в сочетании с некоторой универсальностью множественного наследования. Фактически использование абстрактных классов является одним из способов множественного наследования языков.

Основные различия C# и JAVA

Важно помнить, что C# берет свое начало в желании Microsoft иметь собственный «Java-подобный» язык для платформы .NET. Язык C# настроен для решения проблем, с которыми сталкивались разработчики Microsoft, когда они изначально пытались создать свою платформу на Visual J++. В то же время сообщество Java с открытым исходным кодом продолжало расти и между этими двумя языками развивалась гонка технических вооружений. Вот некоторые из основных различий между C# и Java.

Windows vs open-source. Хотя существуют реализации с открытым исходным кодом, C# в основном используется в разработке для платформ Microsoft – .NET Framework CLR и является наиболее широко используемой

реализацией CLI. На другом конце спектра Java имеет огромную экосистему с открытым исходным кодом и у него открылось второе дыхание отчасти благодаря тому, что Google использует JVM для Android.

Поддержка обобщений (Generics): Generics улучшает проверку типов с помощью компилятора, в основном удаляя приведения из исходного кода. В Java средства обобщений реализуются с использованием стираний. Параметры общего типа «стираются», а при компиляции в байт-код добавляются приведения. C# также использует обобщения, интегрируя его в CLI и предоставляя информацию о типе во время выполнения, что дает небольшое увеличение производительности.

Поддержка делегатов (указателей). В C# есть делегаты, которые по существу служат в качестве методов, которые могут быть вызваны без знания целевого объекта. Для достижения такой же функциональности в Java вам необходимо использовать интерфейс с одним методом или другим способом обхода, который может потребовать нетривиального количества дополнительного кода, в зависимости от приложения.

Проверяемые исключения: Java различает два типа исключений – проверяемые и непроверяемые. C# имеет только один тип исключения. Хотя способность ловить исключения может быть полезна, она также может отрицательно влиять на масштабируемость и контроль версий.

Полиморфизм C# и Java используют очень разные подходы к полиморфизму. Java допускает полиморфизм по умолчанию, C# же должен вызывать ключевое слово «virtual» в базовом классе и ключевое слово «override» в производном классе.

Перечисления (Enums): в C# перечисления представляют собой простые списки именованных констант, где базовый тип должен быть целым. Java представляет перечисления более глубоко, рассматривая его как

именованный экземпляр типа, что упрощает добавление пользовательского поведения к отдельным перечислениям.

Инструмент для программирования, который в конечном итоге используется, будет во многом зависеть от платформы, которую выбирает разработчик для своего проекта. Сегодня C# используется в основном для реализации CLI на .NET Framework, Mono и Portable.NET. Если программное обеспечение или веб-приложение создаются для Windows, C# будет работать лучше всего с набором технологий .NET.

Тем не менее, если требуется разрабатывать для Unix, Linux или других платформ за пределами платформы Microsoft, экосистема с открытым исходным кодом Java лучший выбор. Сообщество постоянно создает новые библиотеки и инструменты. Появились новые мощные языки, такие как Scala, Clojure и Groovy, и они все тоже основаны на JVM. К тому же это неплохо, что большинство реализаций JVM являются общедоступными и бесплатными. Java основной язык разработки, который использует Google для Android – самой большой мобильной операционной системы в мире в настоящее время.

В связи с тем, что система будет со временем расширяться и рассчитана на кроссплатформенное использование выбор остановлен на языке Java.

Сравнение СУБД для реализации проекта

Базы данных это специально разработанное хранилище для различных типов данных. Каждая база данных, имеет определённую модель (реляционная, документно-ориентированная), которая обеспечивает удобный доступ к данным. Системы управления базами данных (СУБД) - специальные приложения (или библиотеки) для управления базами данных различных размеров и форм.

СУБД должна обеспечивать реляционную модель работы с данными. Сама модель подразумевает определенный тип связи между сущностями из разных таблиц. Чтобы хранить и работать с данными, такой тип СУБД должен иметь определенную структуру (таблицы). В таблицах каждый столбец может содержать данные разного типа. Каждая запись состоит из множества атрибутов (столбцов) и имеет уникальный ключ, хранящейся в той же таблице - все эти данные взаимосвязаны между собой, как описано в реляционной модели.

Типы данных СУБД и отношений между ними

Отношения в базах данных можно рассматривать как математическое множество, содержащее в себе число атрибутов, которые суммарно представляют собой базу данных и информацию, хранящуюся в ней (фраза для тех, кто понимает, что такое математическое множество).

При создании структуры таблицы каждое поле записи должно иметь заранее описанный тип (например: строка, целочисленное значение и т.д.). Все СУБД имеют в своем составе различные типы данных, которые не всегда взаимозаменяемы. При работе с СУБД всегда приходится сталкиваться с подобными ограничениями.

Если требуется использовать неструктурированные данные (т.е. записи в одной и той же таблице могут иметь разную структуру), то следует рассмотреть использование NoSQL (безсхемная СУБД).

Популярные и основные реляционные базы данных:

- SQLite - очень мощная встраиваемая система управления;
- MySQL - самая популярная и распространённая СУБД;
- PostgreSQL - наиболее продвинутая СУБД.

Большим плюсом свободно распространяемого ПО является возможность изменять его под свои нужды. Почти всегда можно свободно получить копию репозитория из системы контроля версий.

СУБД SQLite

Легко встраиваемая в приложения база данных. Так как это система базируется на файлах, то она предоставляет довольно широкий набор инструментов для работы с ней, по сравнению с сетевыми СУБД. При работе с этой СУБД обращения происходят напрямую к файлам (в эти файлах хранятся данные), вместо портов и сокетов в сетевых СУБД. Именно поэтому SQLite очень быстрая, а также мощная благодаря технологиям обслуживающих библиотек.

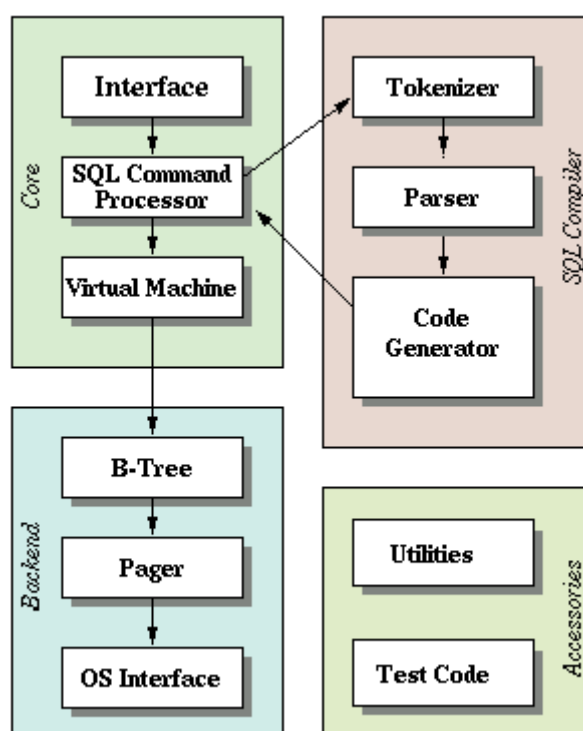


Рисунок 1.7 – Архитектура СУБД SQLite

Типы данных SQLite:

- **NULL** - значение NULL
- **INTEGER** - знаковое целочисленное значение, использует 1, 2, 3, 4, 6, или 8 байт в зависимости от порядка числа
- **REAL** - число с плавающей точкой, занимает 8 байт для хранения числа в формате IEEE

- TEXT - текстовая строка, при хранении используются кодировки UTF-8, UTF-16BE или UTF-16LE
- BLOB - тип данных BLOB, массив двоичных данных (предназначенный, в первую очередь, для хранения изображений, аудио и видео).

Преимущества SQLite:

- Файловая структура - вся база данных состоит из одного файла, поэтому её очень легко переносить на разные машины;
- Используемые стандарты - хотя может показаться, что эта СУБД примитивная, но она использует SQL;
- Отличная при разработке и тестировании - в процессе разработки приложений часто появляется необходимость масштабирования; SQLite предлагает всё что необходимо для этих целей, так как состоит всего из одного файла и библиотеки написанной на языке C.

Недостатки SQLite

- Отсутствие системы пользователей - более крупные СУБД включают в свой состав системы управления правами доступа пользователей. Обычно применения этой функции не так критично, так как эта СУБД используется в небольших приложениях;
- Отсутствие возможности увеличения производительности - опять, исходя из проектирования, довольно сложно выжать что-то более производительное из этой СУБД.

Когда использовать SQLite

- Встроенные приложения - если вам важна возможность легкого переноса приложения и не важна масштабируемость. Например однопользовательские приложения, мобильные приложения или игры.

- Прямой доступ к диску - при необходимости напрямую обращаться к диску можно выиграть при переходе на эту СУБД в функционале и простоте использования SQL языка
- Тестирование - использование дополнительных процессов при тестировании функционала, очень замедляет приложение.

Когда отказаться от SQLite:

- Многопользовательские приложения - если вам необходимо обеспечить доступ к данным для нескольких пользователей, да и к тому же различать их по правам доступа, то, наверное, полноценная СУБД (например: MySQL) будет более логичным выбором;
- Запись больших объемов данных - одно из ограничений SQLite это операции записи. Разрешен только один процесс записи в промежуток времени, что сильно ограничивает производительность.

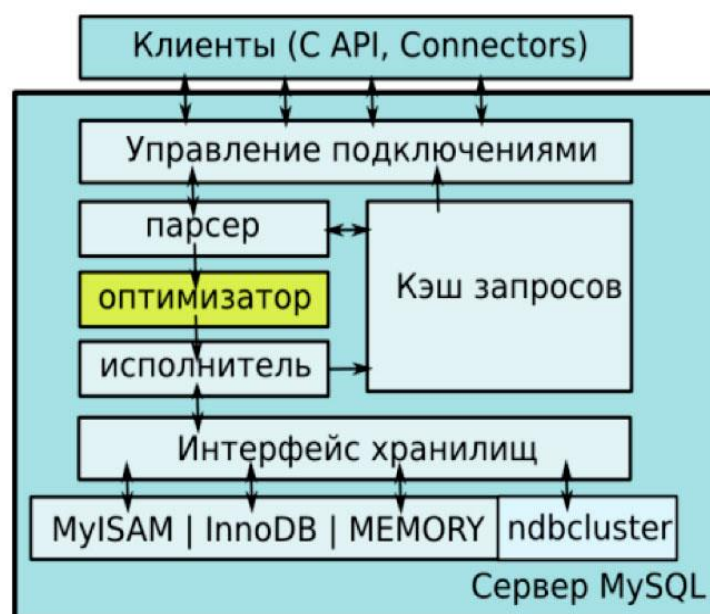
СУБД MySQL

СУБД MySQL это самая распространенная полноценная серверная СУБД. MySQL очень функциональная, свободно распространяемая СУБД, которая успешно работает с различными сайтами и веб приложениями. Обучиться использованию этой СУБД довольно просто, так как на просторах интернета можно легко найти большее количество информации.

Плюсом СУБД MySQL является и наличие плагинов и расширений для решения задач средствами MySQL.

Несмотря на то, что в ней не реализован весь SQL функционал, MySQL предлагает довольно много инструментов для разработки приложений. Так как это серверная СУБД, приложения для доступа к данным, в отличии от SQLite работают со службами MySQL.

Архитектура MySQL



HighLoad++ Junior

Рисунок 1.8 – Архитектура СУБД MySQL

Преимущества MySQL:

- Простота в работе - установить MySQL просто. Дополнительные приложения, например GUI, позволяет довольно легко работать с БД;
- Богатый функционал - MySQL поддерживает большинство функционала SQL;
- Безопасность - большое количество функций обеспечивающих безопасность, которые поддерживаются по умолчанию;
- Масштабируемость - MySQL легко работает с большими объемами данных и легко масштабируется;
- Скорость - упрощение некоторых стандартов позволяет MySQL значительно увеличить производительность.

Недостатки MySQL:

- Известные ограничения - по задумке в MySQL заложены некоторые ограничения функционала, которые иногда необходимы в особо требовательных приложениях;
- Проблемы с надежностью - из-за некоторых способов обработки данных MySQL (связи, транзакции, аудиты) иногда уступает другим СУБД по надежности;
- Медленная разработка - Хотя MySQL технически открытое ПО, существуют жалобы на процесс разработки. Стоит заметить, что существуют другие довольно успешные СУБД созданные на базе MySQL, например MariaDB.

Когда следует использовать MySQL:

- Распределённые операции - если функционала SQLite не хватает, то стоит рассмотреть MySQL. Так как эта СУБД сочетает в себе продвинутый функционал и свободный доступ к исходному коду;
- высокий уровень безопасности - система безопасности MySQL включает в себе простые и в то же время достойные способы защиты доступа к данным;
- Веб сайты и веб приложения - большинство сайтов и онлайн приложений спокойно работают с MySQL несмотря на некоторые ограничения. Будучи легкой в настройке и масштабируемой системой - MySQL проверена временем;
- Индивидуальные решения - если вы работаете с каким либо специфическим проектом, MySQL легко сможет вам помочь благодаря широким возможностям в настройке и функционалом.

Когда лучше отказаться от MySQL:

- Соответствие стандартам - MySQL не ставит для себя целью полностью соответствовать стандартам SQL, то эта СУБД не

полностью поддерживает SQL. Если в будущем вы планируете перейти на подобную систему, то MySQL - не лучший выбор;

- Многопоточность - хотя некоторые движки БД довольно легко выполняют параллельное чтение, параллельные операции чтения-записи могут создать проблемы;
- Недостаток функционала - некоторые движки MySQL, например, не поддерживают полнотекстовый поиск.

СУБД PostgreSQL

PostgreSQL является самым профессиональным из всех трех рассмотренных нами СУБД. Она свободно распространяемая и максимально соответствует стандартам SQL. PostgreSQL или Postgres стараются полностью применять ANSI/ISO SQL стандарты своевременно с выходом новых версий.

От других СУБД PostgreSQL отличается поддержкой востребованного объектно-ориентированного и/или реляционного подхода к базам данных. Например, полная поддержка надежных транзакций, т.е. атомарность, последовательность, изоляционность, прочность (Atomicity, Consistency, Isolation, Durability (ACID).) Благодаря мощным технологиям Postgre очень производительна. Параллельность достигнута не за счет блокировки операций чтения, а благодаря реализации управления многовариантным параллелизмом (MVCC), что также обеспечивает соответствие ACID. PostgreSQL очень легко расширять своими процедурами, которые называются хранимые процедуры. Эти функции упрощают использование постоянно повторяемых операций.

PostgreSQL и не имеет большой популярности в отличии от MySQL, существует довольно большое число приложений облегчающих работу с PostgreSQL, несмотря на всю мощность функционала. Сейчас довольно легко

установить эту СУБД используя, стандартные менеджеры пакетов операционных систем.

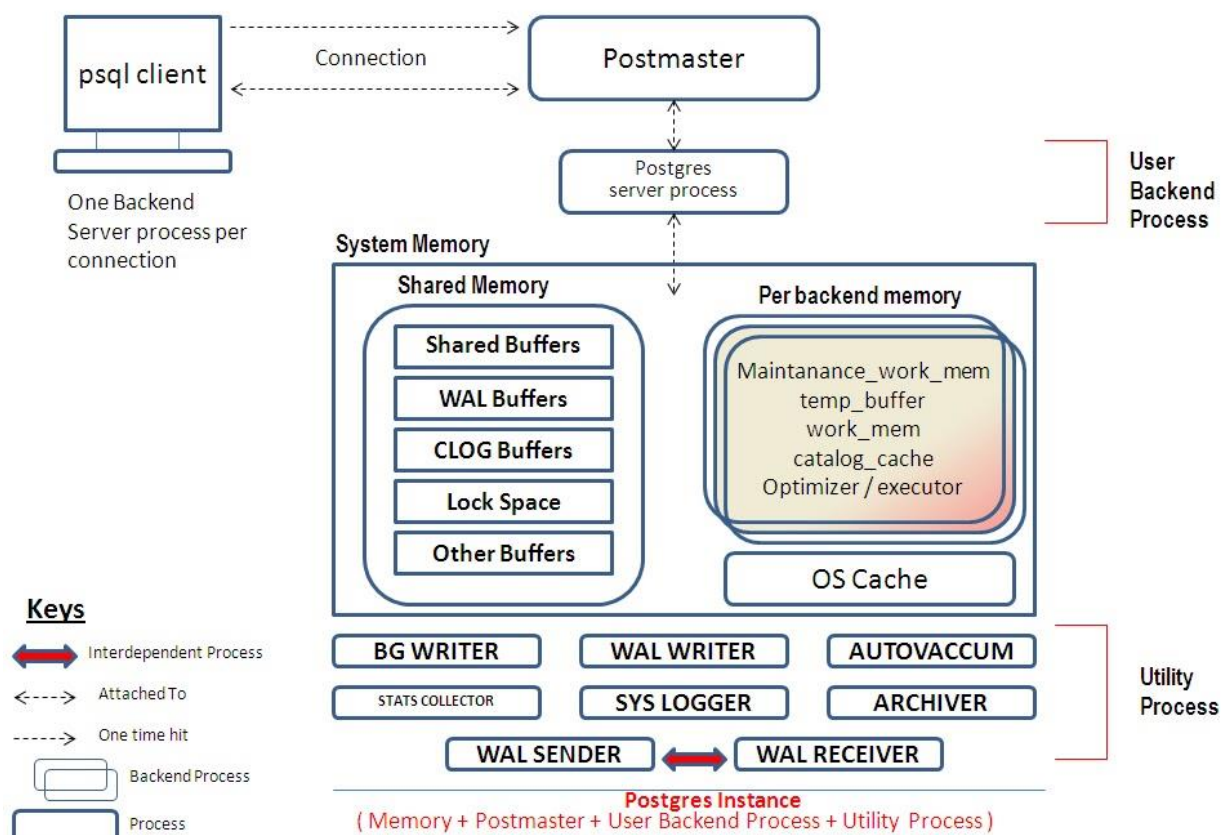


Рисунок 1.9 – Архитектура СУБД Postges

Достоинства PostgreSQL:

- Открытое ПО соответствующее стандарту SQL - PostgreSQL - бесплатное ПО с открытым исходным кодом. Эта СУБД является очень мощной системой;
- Большое сообщество - существует довольно большое сообщество в котором вы запросто найдёте ответы на свои вопросы;
- Большое количество дополнений - несмотря на огромное количество встроенных функций, существует очень много дополнений,

позволяющих разрабатывать данные для этой СУБД и управлять ими;

- Расширения - существует возможность расширения функционала за счет сохранения своих процедур;
- Объектность - PostgreSQL это не только реляционная СУБД, но также и объектно-ориентированная с поддержкой наследования и много другого.

Недостатки PostgreSQL:

- Производительность - при простых операциях чтения PostgreSQL может значительно замедлить сервер и быть медленнее своих конкурентов, таких как MySQL;
- Популярность - по своей природе, популярностью эта СУБД похвастаться не может, хотя и присутствует довольно большое сообщество;
- Хостинг - в силу выше перечисленных факторов иногда довольно сложно найти хостинг с поддержкой этой СУБД.

Когда использовать PostgreSQL:

- Целостность данных - когда надежность и целостность данных - ваши требования, PostgreSQL будет, пожалуй, лучшим выбором;
- Сложные пользовательские процедуры - если вам необходимо использовать пользовательские процедуры, то PostgreSQL имеет встроенную поддержку для них;
- Интеграция - если в будущем вы планируете переход на платные СУБД, например Oracle, то сделать это с PostgreSQL будет довольно просто по сравнению с другими бесплатными СУБД;
- Сложная структура данных - по сравнению с другими открытыми СУБД PostgreSQL предоставляет больше возможностей для создания сложных структур данных.

Когда не следует использовать PostgreSQL:

- Скорость - если быстрое чтение для вас единственный фактор, то стоит присмотреться к другим СУБД;
- Простая настройка - если вам не нужна целостность данных, соответствие ACID или сложные структуры данных, то настройка PostgreSQL будет реализована трудозатратой по сравнению с MySQL;
- Репликация - если разработчик не готов тратить время на то, что мог бы с легкостью сделать MySQL, то требуется выбрать более легкое в настройке СУБД.

Выводы по главе

В результате исследования в первой главе были сделаны следующие шаги:

Исследование предметной области и доказательство ее актуальности, как тематика дипломного проектирования.

Был исследован и выбран стек технологий.

Исследованы способы хранения информации и выбрана СУБД для хранения служебной информации и данных.

2 Разработка системы

В распределенной системе одновременно работает множество коммутационного оборудования, которое требует от технического персонала технической поддержки, настройки и обслуживания.

Самостоятельное клиентское приложение необходимо для автоматизации работы с RouterOS от Mikrotik в трех случаях: реализовать задачу встроенными средствами RouterOS трудозатратно, скриптовый язык RouterOS в рамках своих ограничений не позволяет реализовать задачу, рутинные задачи требуется автоматизировать на множестве однотипных устройств в рамках распределенной системы.

Если операция, которую требуется произвести на множестве роутеров однотипна, то целесообразно реализовать программную систему, показанную на рисунке 2.1.

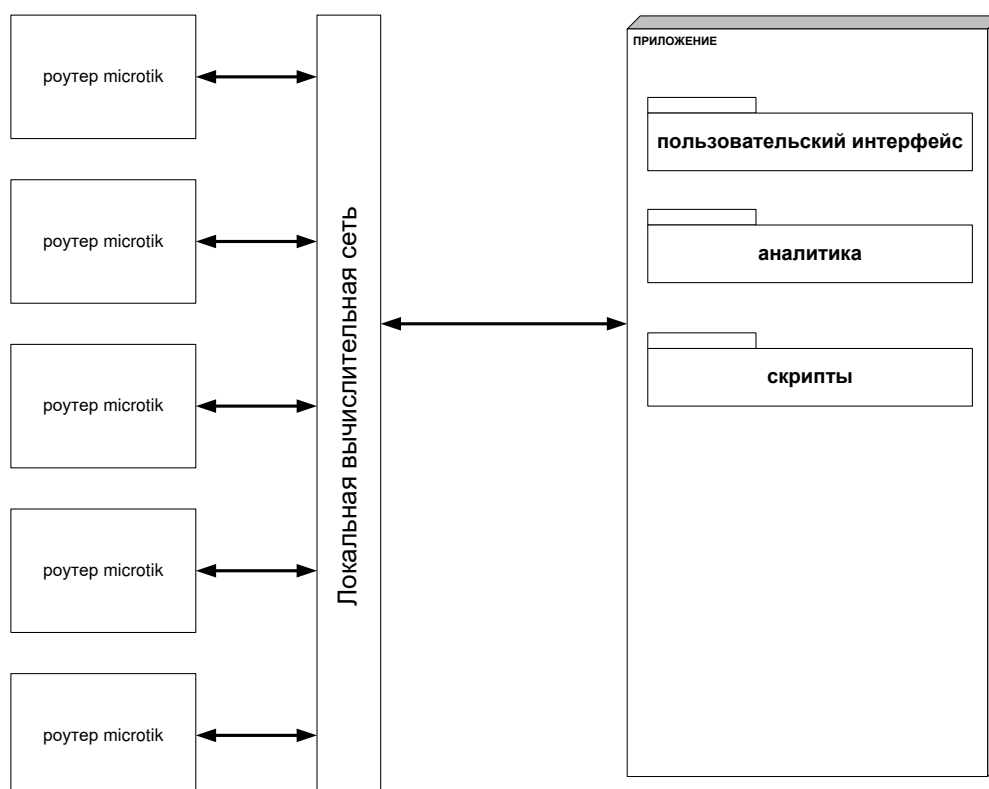


Рисунок 2.1 - Общая организация структуры

На рисунке 2.1 представлена система, в которой присутствует в рамках локальной сети некоторое количество роутеров. Роутеры изначально настроены и выполняют свои функции в штатном режиме. Также в системе имеется GUI приложение, которое осуществляет элементарные функции мониторинга, аналитики и настройки роутеров, подключенных к модулю мониторинга приложения. Отметим также два момента, данное приложение не заменяет стандартных средств управления устройствами от производителя или специализирующихся на данной тематике вендоров, оно лишь позволяет оперативно помочь техническому персоналу произвести рутинные настройки на множестве однотипных устройств, заменив ручной труд (рисунок 2.2).

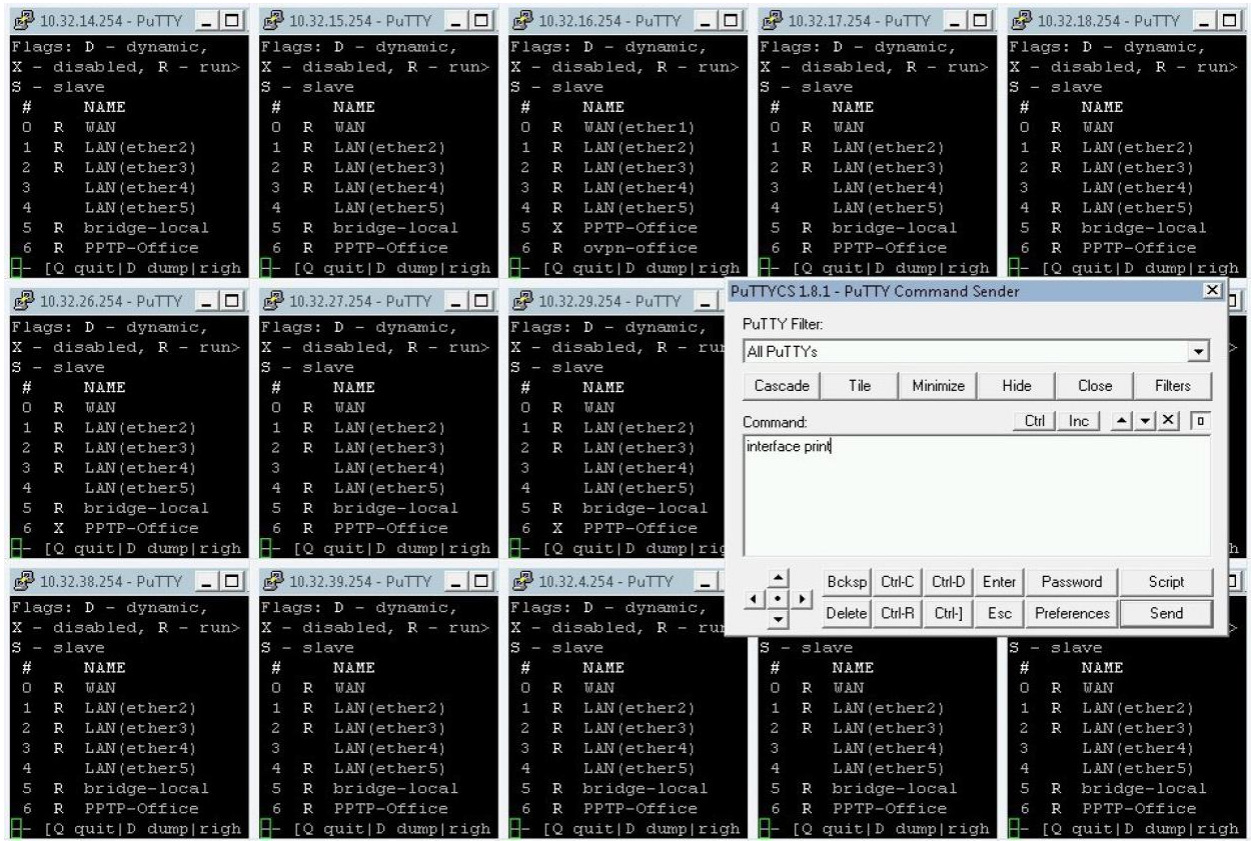


Рисунок 2.2 - Организация ручного рутинного обслуживания коммутаторов MikroTik.

Для реализации приложения необходимо определить функционал приложения, описать и произвести тест скриптов которые требуется

автоматизировать, реализовать базу данных для хранения информации, написать приложение для автоматизации процесса работы с роутерами Mikrotik.

2.1 Разработка скриптов RouteOS

В данном разделе будет реализована разработка скриптов, которые будут использованы в приложении. Будет проведен эксперимент по настройке оборудования, который подтвердит их работоспособность.

Блокировка сайтов

Классической задачей системного администратора, является блокировка нежелательных сайтов.

Классическое решение данной задачи является следующий скрипт:

```
/ip firewall layer7-protocol
```

```
add name=youtube regexp="^(youtube).*$"
```

```
add name=facebook regexp="^(facebook).*$"
```

```
/ip firewall filter
```

```
add action=drop chain=forward layer7-protocol=facebook
```

```
add action=drop chain=forward layer7-protocol=youtube
```

У данного решения следующие минусы: высокая нагрузка на сри, увеличенная latency, потеря пакетов, youtube и facebook не блокируются.

Данные нагрузки вызваны тем, что, каждое соединение проверяется снова и снова, Layer7 проверяется не в том месте, что приводит к проверке всего трафика.

Второе решение с помощью регулярных выражений Layer7, лишено данных недостатков.

```

/ip firewall layer7-protocol
add name=youtube regexp="^(.+youtube).*$"

```

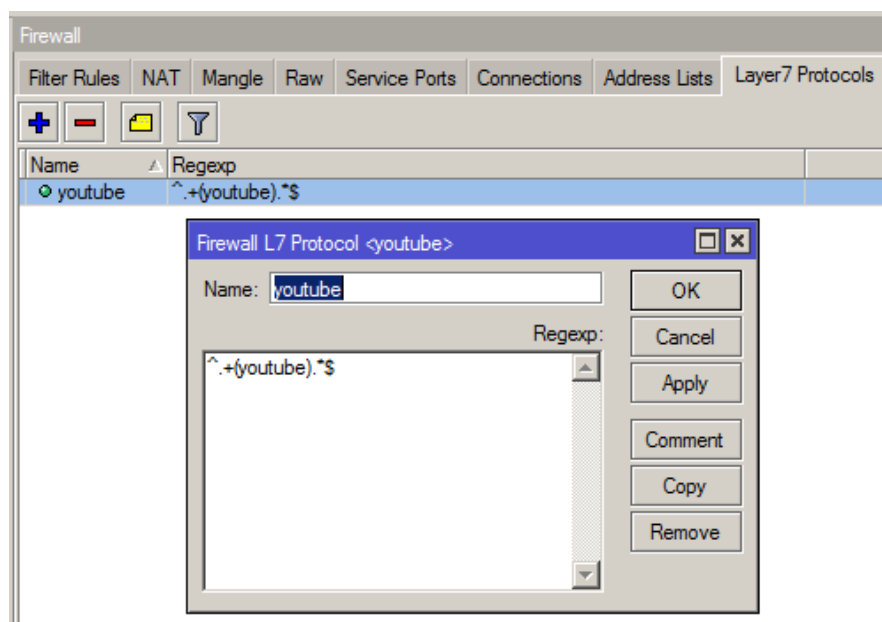


Рисунок 2.3 - Настройка скрипта на роутере Mikrotik.

Скрипт выше заблокировал только доступ к сервису youtube, если нужен facebook или что-то иное, создается отдельное правило:

```
add name=facebook regexp="^(.+facebook).*$"
```

Можно создавать правила и для других сервисов стримминга видео, вот один из вариантов:

```

regexp="^(.*youtube.com|youtu.be|netflix.com|vimeo.com|screen.yah
oo.com|dailyMotion.com|hulu.com|twitch.tv|liveleak.com|vine.co|b
reak.com|tv.com|metacafe.com|viewster.com).*$"

```

Далее создаем правила для маркировки соединений и пакетов:

```

/ip firewall mangle
add action=mark-connection chain=prerouting protocol=udp
dst-port=53 connection-mark=no-mark layer7-protocol=youtube new-
connection-mark=youtube_conn passthrough=yes

```

```
add action=mark-packet chain=prerouting connection-  
mark=youtube_conn new-packet-mark=youtube_packet
```

и правила для фильтра файрвола:

```
/ip firewall filter
```

```
add action=drop chain=forward packet-mark=youtube_packet
```

```
add action=drop chain=input packet-mark=youtube_packet
```

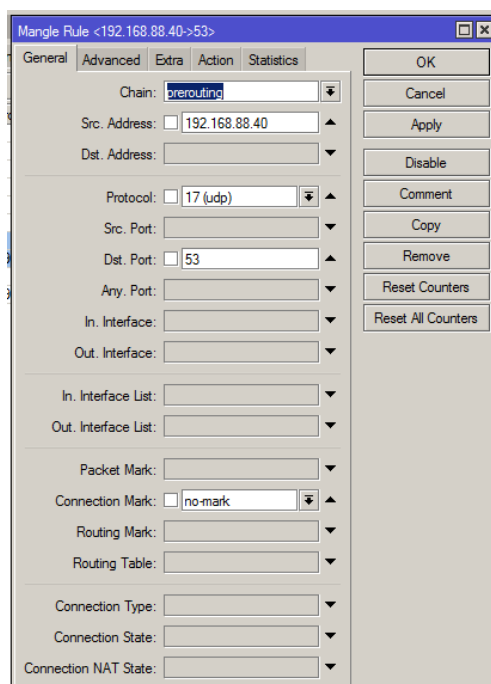


Рисунок 2.4 - Создание правила для маркировки соединений и пакетов.

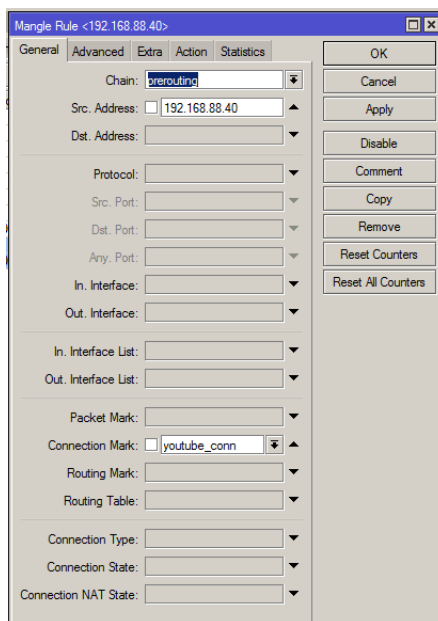


Рисунок 2.5 - Создание правила для маркировки соединений и пакетов.

Изм.	Лист	№ докум.	Подпись	Дата

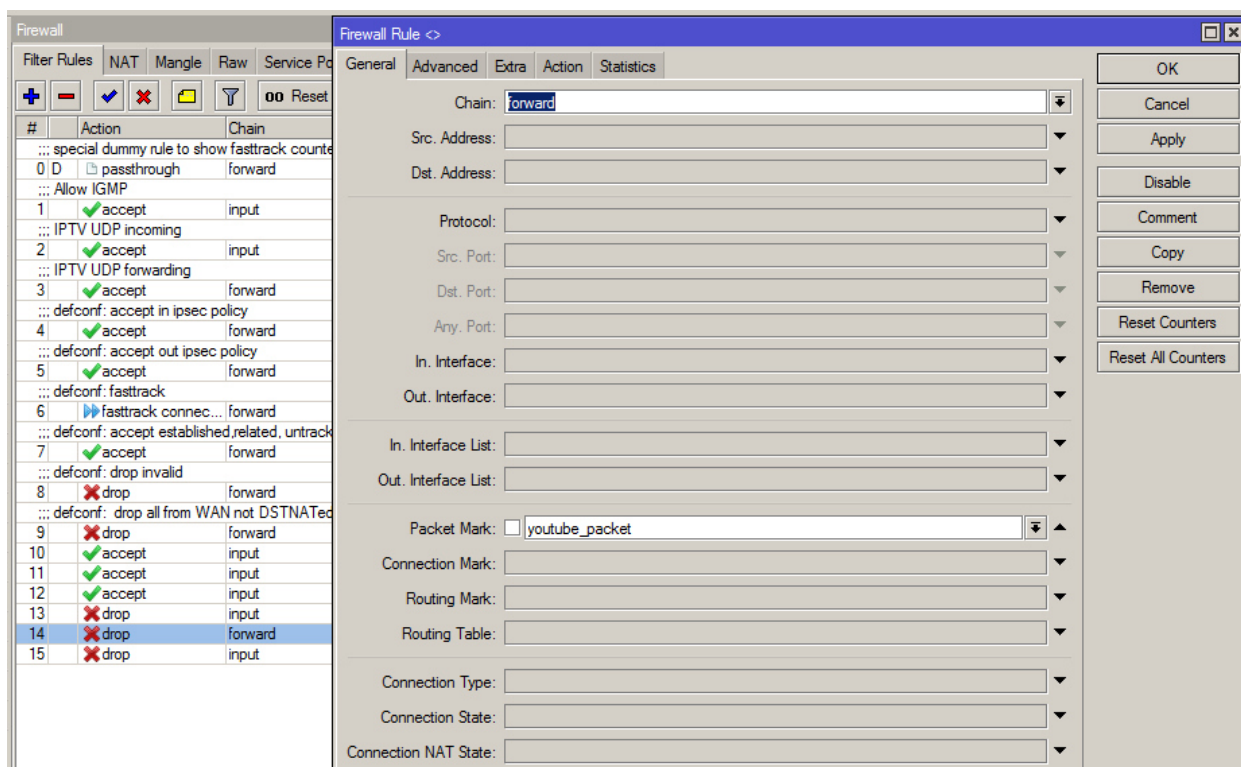


Рисунок 2.6 - Создание правила для маркировки соединений и пакетов.

В данном сегменте сети адреса раздаются ip-адреса по dhcp протоколу, поэтому фильтр применяется к ip-адресу смартфона пользователя, можно создать группу адресов и применить к ней. Для этого идем в меню **IP>Firewall>AddressList** нажимаем кнопку **Add**, вводим название группы и не забываем заполнить список адресов для блокировки.

Далее идем меню **IP>Firewall>Mangle** выбираем необходимые mark_connection и mark_packet и в поле Src. Address заносим информацию о блокируемом ip либо группе.

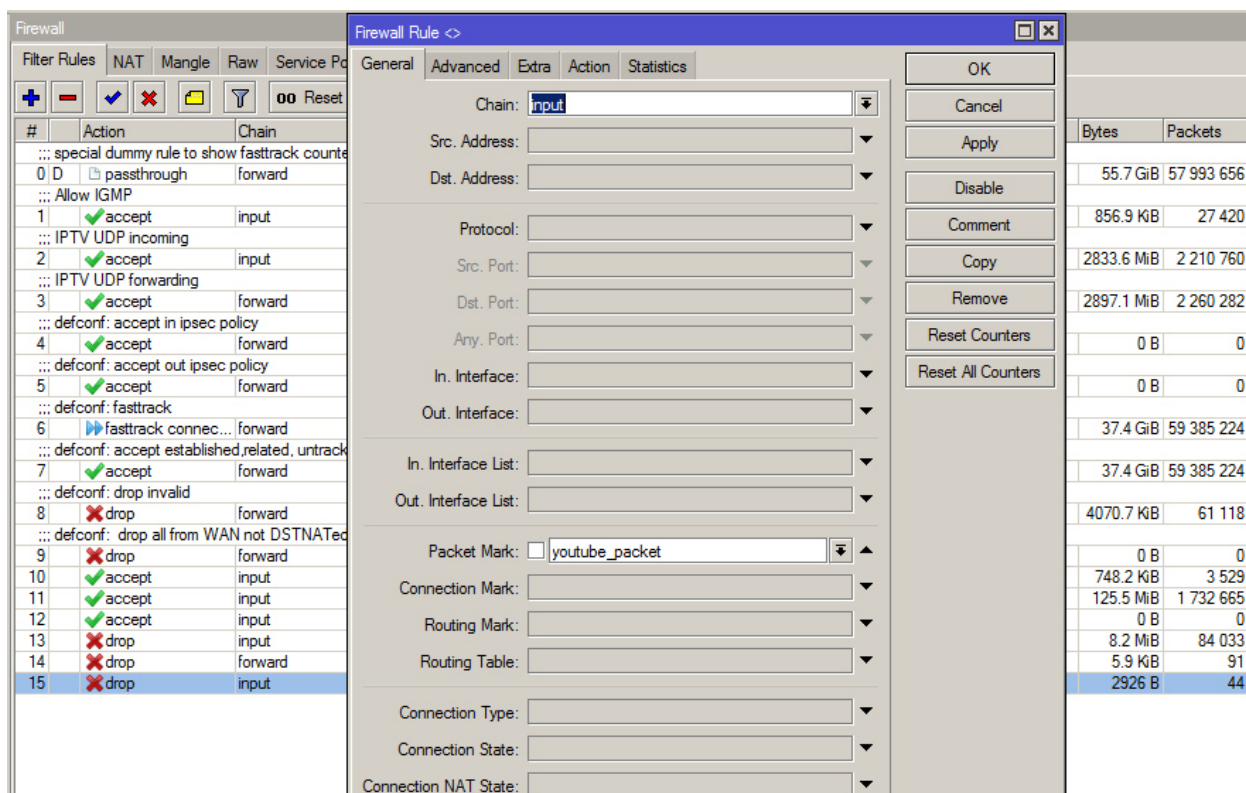


Рисунок 2.7 - Создание правила для маркировки соединений и пакетов.

Все, смартфон и/или компьютер остался без доступа к сервисам youtube. Так же можно применять эти правила по расписанию. Для блокировки устройств не только по IP адресам а и по mac необходимо добавить следующий участок кода:

```
/ip firewall filter
add chain=input src-mac-address=aa:bb:cc:dd:ee:ff action=drop
add chain=forward src-mac-address=aa:bb:cc:dd:ee:ff action=drop
```

Firewall											
Filter Rules											
#	Action	Chain	Src. Address	Dst. Address	Protocol	Src. Port	Dst. Port	In. Inter...	Out. Int...	Bytes	Packets
0	D passthrough	prerouting								55.6 GiB	57 923 011
1	D passthrough	forward								55.6 GiB	57 923 011
2	D passthrough	postrouting								55.6 GiB	57 923 011
3	mark connection	prerouting	192.168.88.40		17 (udp)		53			9.0 KiB	138
4	mark packet	prerouting	192.168.88.40							8.9 KiB	137

Рисунок 2.8 Результат работы блокировки

Настройка централизованного создания резервных копий

Операционная система RouterOS позволяет сохранить настройки MikroTik, и быстро восстановить резервную копию, чтобы возобновить работу устройства. Это очень удобно, если сбились настройки, или нужно выполнить одинаковую настройку нескольких устройств MikroTik. Ниже представлены классические шаги чтобы создать резервную копию настроек MikroTik, восстановить ее или перенести на другое устройство.

Сохранение и восстановление настроек MikroTik. Чтобы сохранить настройки MikroTik, выполните следующее:

- Откройте в программе Winbox меню Files.
- Нажмите кнопку Backup.
- Во всплывающем окне нажмите кнопку Backup.
- При желании можете указать в поле Name имя файла и пароль в поле Password.

По умолчанию файлы бэкапов шифруются, для защиты от извлечения паролей, например, к Wi-Fi. Галочка Don't encrypt позволяет отключить шифрование файла с резервными настройками.

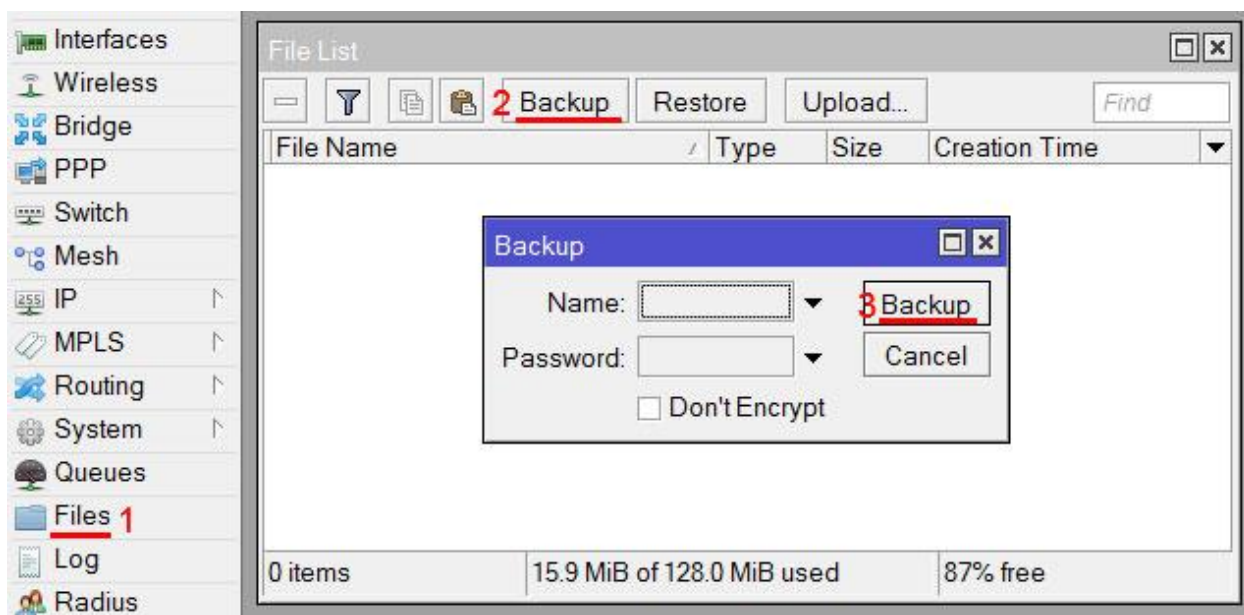


Рисунок 2.9 - Окно создания резервной копии

После этого в списке появится файл с именем MikroTik-xxxxxxxx-xxxx.Backup , где xxxxxxxx-xxxx - это дата и время создания файла.

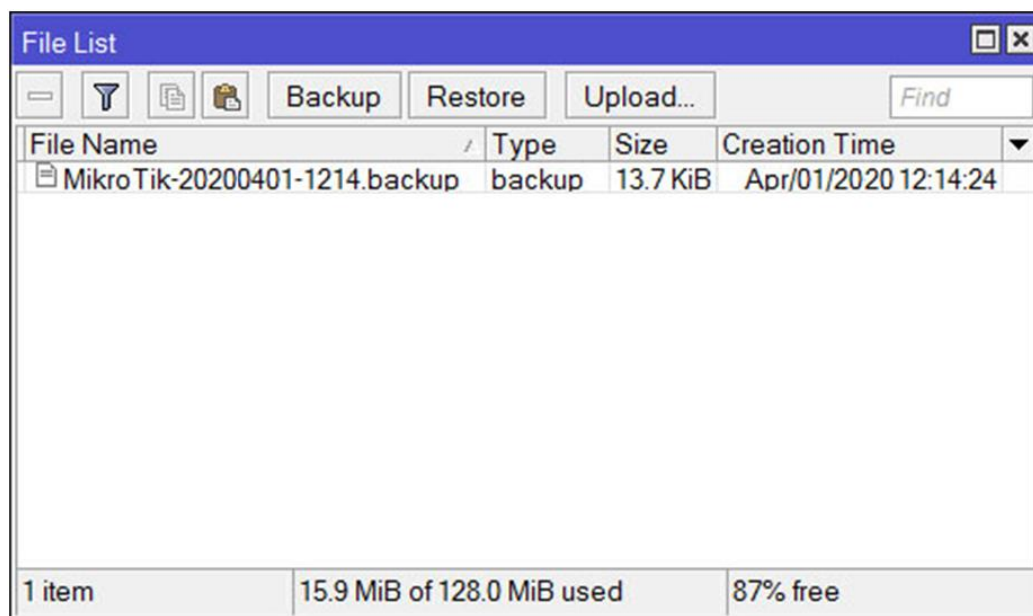


Рисунок 2.10 - Окно создания резервной копии

Файл MikroTik-xxxxxxxx-xxxx.Backup предназначен для восстановления настроек только на устройстве, на котором он создан. Его нельзя использовать для переноса настроек на другое устройство. Иначе перенесутся MAC-адреса интерфейсов, в сети будут два одинаковых устройства и возникнет конфликт. Для переноса настроек используются специальные файлы в текстовом формате.

Далее резервную копию конфигурации MikroTik можно сохранить на компьютер, перетащив файл на рабочий стол или в проводник Windows.

Перед восстановлением настроек, нужно сбросить все настройки устройства, чтобы в нем не остались какие-то настройки из текущей конфигурации.

Откройте меню System - Reset Configuration.

Поставьте галочку No Default Configuration.

Нажмите кнопку Reset Configuration.

В всплывающем окне нажмите Yes для сброса настроек и перезагрузки устройства.

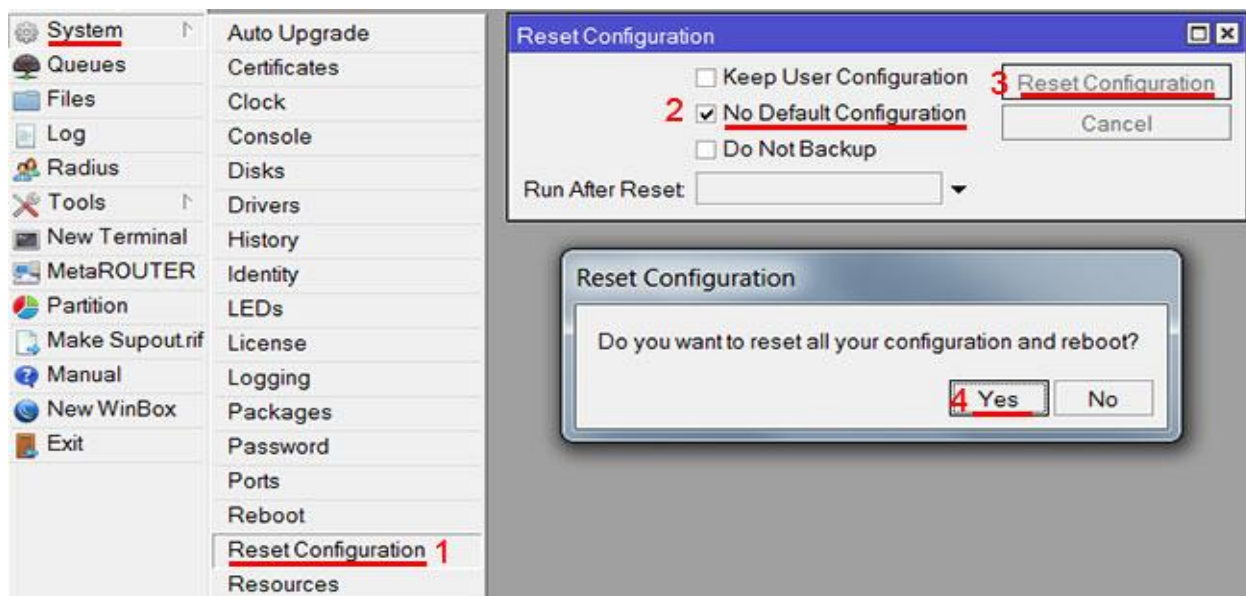


Рисунок 2.12 - Сброс настроек до заводских

После перезагрузки подключайтесь к устройству по MAC адресу, поскольку IP адрес будет отсутствовать. В программе Winbox нужно нажать мышкой на MAC-адрес устройства и нажать кнопку Connect.

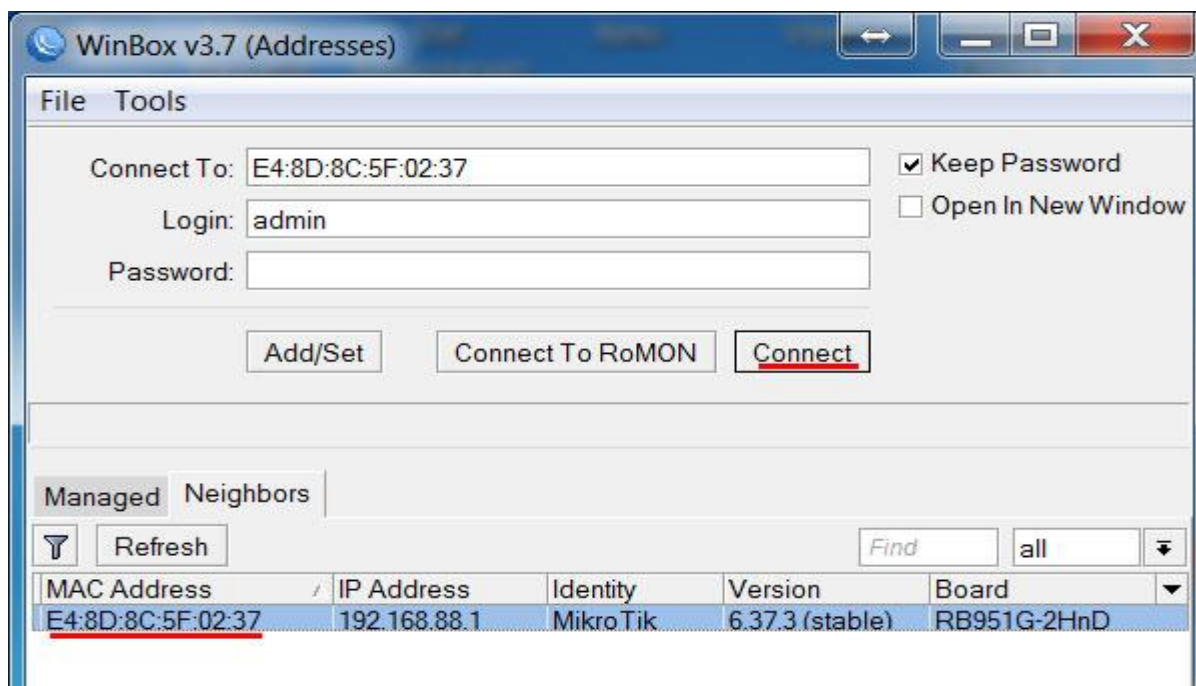


Рисунок 2.13 - Подключение к устройству по MAC адресу

Чтобы восстановить конфигурацию MikroTik, выполните следующее:

- Откройте меню Files.
- Выберите файл с резервной копией.
- Нажмите кнопку Restore.
- В появившемся окне нажмите кнопку Restore.
- Если вы делали бэкап с паролем, то введите пароль в поле Password.
- В всплывающем окне нажмите Yes для восстановления конфигурации и перезагрузки устройства.

Перенос настроек MikroTik на другое устройство

Для переноса настроек на другое устройство MikroTik, нужно сохранить настройки в файл в текстовом виде. Этот файл можно будет редактировать, если возникнут проблемы при переносе настроек.

Переносить настройки можно только на устройства MikroTik со схожей аппаратной конфигурацией. Вы не сможете перенести настройки с роутера на 10 LAN портов, на роутер с 5-ю LAN портами. При этом будут выдаваться ошибки. Можно конечно вручную отредактировать файл с настройками, но это довольно сложный и долгий процесс.

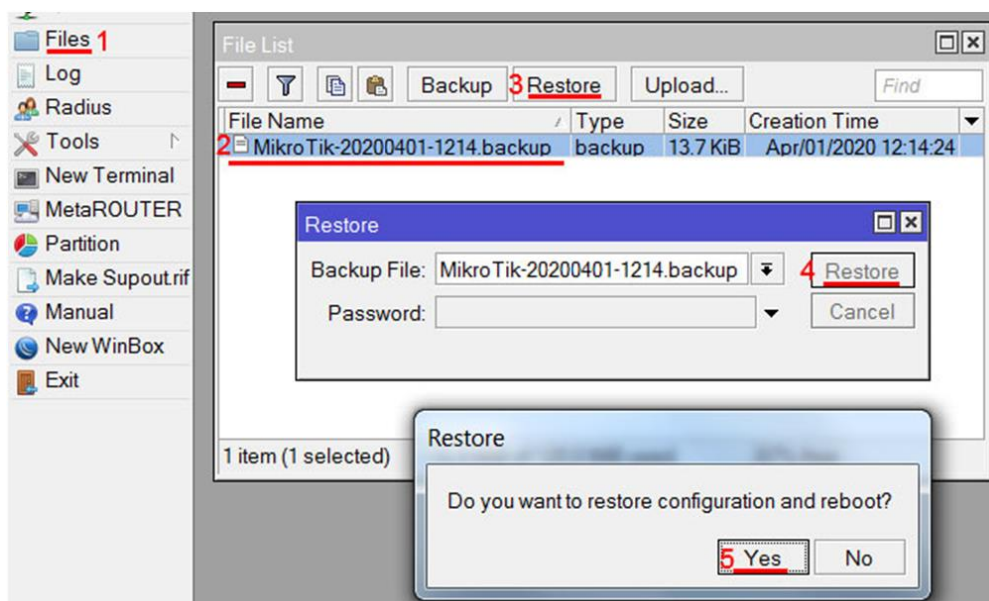


Рисунок 2.14 - Перенос настроек MikroTik на другое устройство

Чтобы сохранить настройки MikroTik в текстовом формате, откройте меню New Terminal и выполните следующую команду:

`export file=config_Backup_20200403.rsc`

Название файла может быть любым. В названии файла желательно указывать дату, чтобы потом не запутаться.

Далее нужно скопировать созданный файл в другое устройство. Для этого сначала перетащите файл на рабочий стол Windows. После этого перетащите его в новое устройство в меню Files.

Перед импортированием настроек, нужно сбросить конфигурацию устройства, иначе будет конфликт с текущими настройками и при импорте будут выдаваться ошибки.

- Откройте меню System - Reset Configuration.
- Поставьте галочку No Default Configuration.
- Нажмите кнопку Reset Configuration.
- В всплывающем окне нажмите Yes для сброса настроек и перезагрузки устройства.

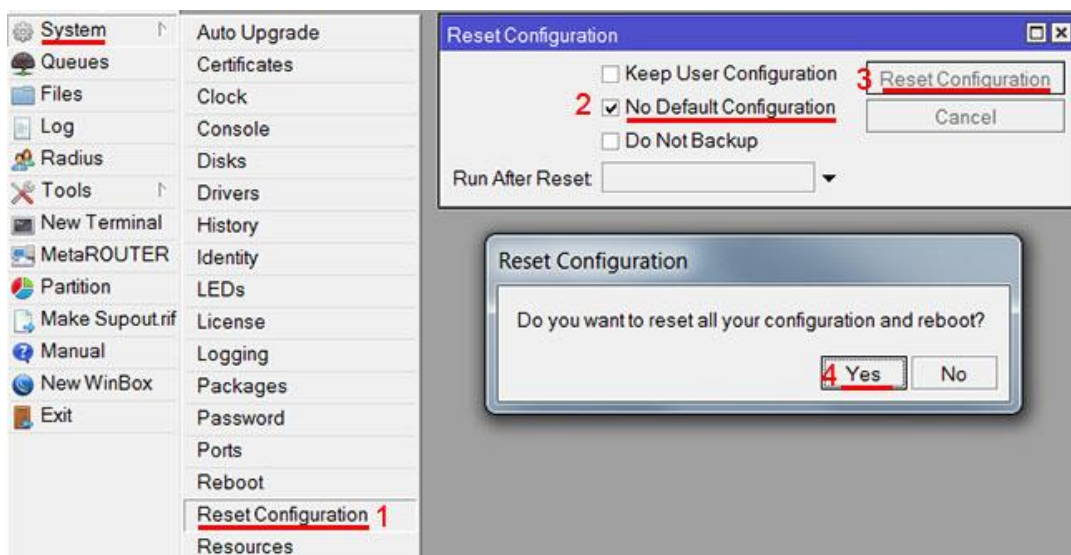


Рисунок 2.14 - Перезагрузка устройства

Частичный перенос настроек

Бывают ситуации, когда нужно сохранить часть настроек, например, правила фаервола, скрипты, NAT, правила маркировки пакетов и т.п. Для этого помогут следующие команды:

```
ip address export file=ip.rsc
ip firewall mangle export file=mangle.rsc
ip firewall nat export file=nat.rsc
ip firewall filter export file=filter.rsc
queue simple export file=simple.rsc
ip dns export file=dns.rsc
files Backup export file=Backup.rsc
system script export file=script.rsc
system scheduler export file=scheduler.rsc
tool e-mail export file=email.rsc
ip firewall address-list export file=address-list.rsc
ip route export file=route.rsc
ip dhcp-server network export file=network.rsc
queue type export file=type.rsc
queue tree export file=tree.rsc
queue simple export file=simple.rsc
```

```
interface ethernet export file=ethernet.rsc
ip pool export file=pool.rsc
ppp profile export file=profile.rsc
log export file=log.rsc
```

Автоматическое создание резервных настроек MikroTik и отправка на email

Чтобы не забывать делать резервные копии конфигурации MikroTik, можно автоматизировать этот процесс и отправлять бэкапы себе на email.

Создание скрипта состоит из этапов:

- Откройте меню System – Scripts и нажмите "красный плюсики", чтобы добавить новый скрипт.
- В поле Name укажите название скрипта, в поле Source вставьте скрипт и нажмите кнопку ОК.
- Не забудьте в скрипте изменить значения your_account@gmail.com и your_gmail_password на индивидуальные.

Чтобы протестировать работу скрипта, нажмите кнопку Run Script, и на почтовый ящик будут высланы резервные копии. Процесс выполнения скрипта можно отслеживать в меню Log. Скрипт для отправки резервной копии настроек MikroTik на email:

```
:log info "Starting Backup Script...";
:local sysname [/system identity get name];
:local sysver [/system package get system version];
:log info "Flushing DNS cache...";
/ip dns cache flush;
:delay 2;
:log info "Deleting last Backups...";
:foreach i in=[/file find] do={:if ([:typeof [:find [/file get $
i name] \
"$sysname-Backup-"]!="nil") do={/file remove $i}};
```

```

:delay 2;
:local smtpserv [:resolve "smtp.gmail.com"];
:local Eaccount "your_account@gmail.com";
:local pass "your_gmail_password";
:local Backupfile (" $sysname-Backup-" . \
[:pick [/system clock get date] 7 11] . [:pick [/system \
clock get date] 0 3] . [:pick [/system clock get date] 4 6] . ".
Backup");
:log info "Creating new Full Backup file...";
/system Backup save name=$Backupfile;
:delay 2;
subject=("$sysname Full Backup (" . [/system clock get date] . "
)") \

```

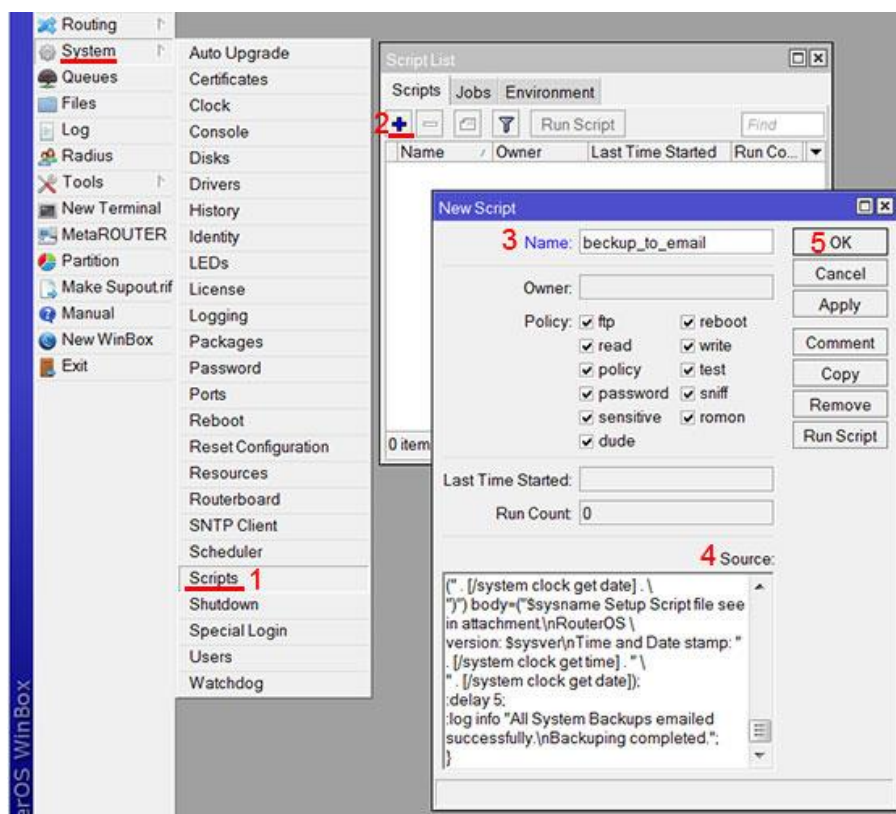


Рисунок 2.15 - Реализация скрипта отправки backup на e-mail.

В примере указан скрипт для сервиса Gmail. При отправке через Gmail нужно предварительно разрешить ненадежным приложениям доступ к

Изм.	Лист	№ докум.	Подпись	Дата

аккаунту. Для этого необходимо войти в свой google аккаунт, на странице "Мой аккаунт" перейдите в раздел "Ненадежные приложения" и выбрать настройку "Включить". Только после этого система отправки backup файлов на почту будет работать.

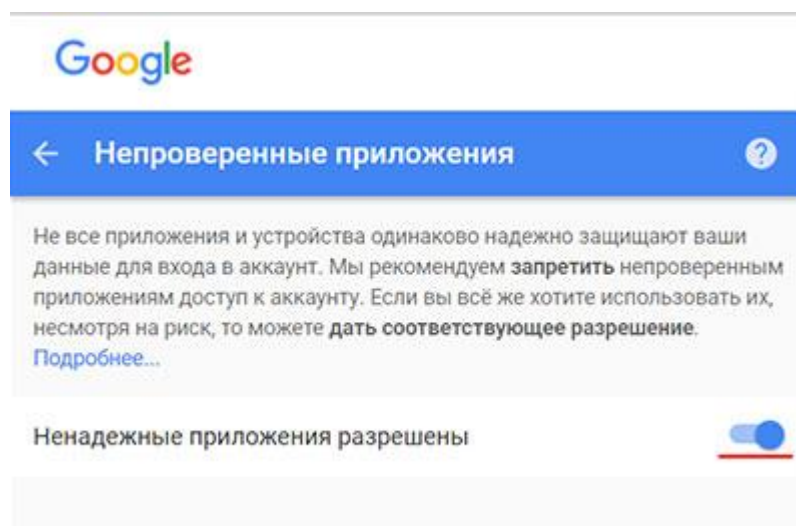


Рисунок 2.16 - Реализация скрипта отправки backup на e-mail.

Настройка планировщика

Теперь нужно настроить с какой периодичностью будут создаваться резервные копии, и отправляться на email. Для этого настроим планировщик (Sheduler). Откройте меню System – Sheduler и нажмите кнопку «плюсик», чтобы добавить новую задачу.

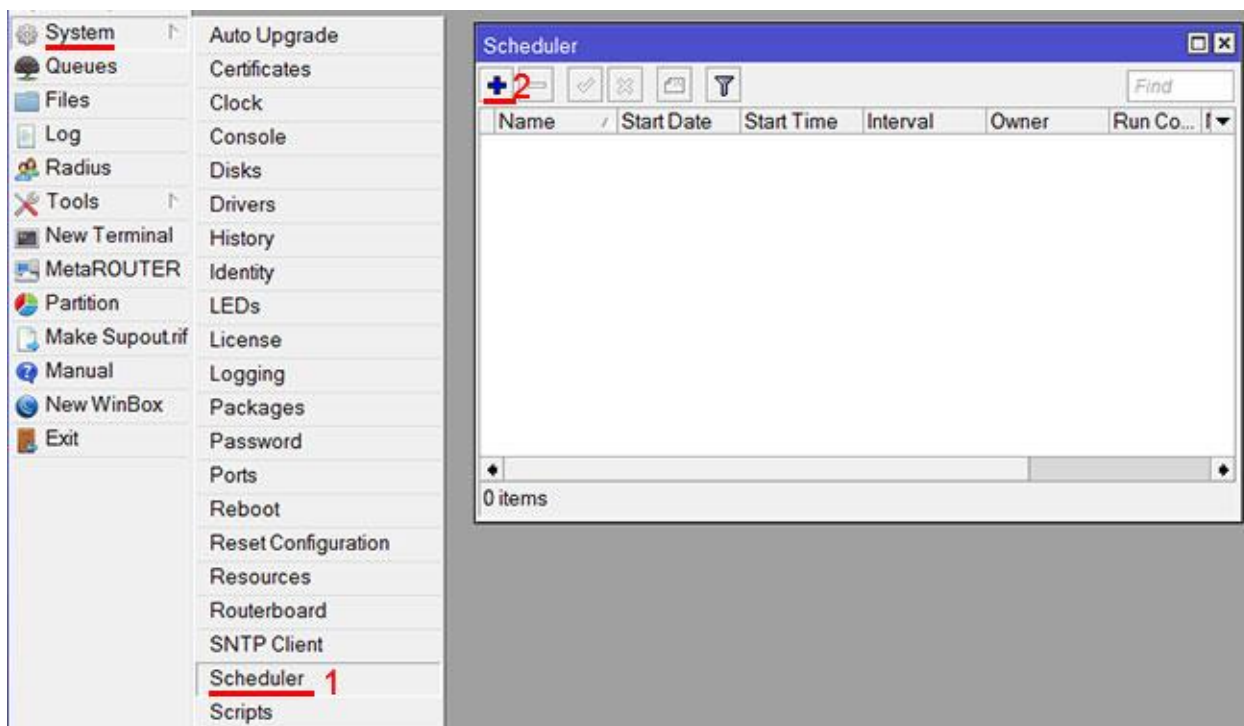


Рисунок 2.17 - Настройка планировщика.

В открывшемся окне настраиваем параметры задачи в поле Name указываем имя задачи, например `send_Backup_to_email`; в поле Interval требуется указать с какой периодичностью будет выполняться скрипт. По умолчанию указан интервал 24 часа. Для отладки работы планировщика целесообразно указать интервал 5 минут и текущее время Start Time. Также требуется настроить время на MikroTik в меню System - Clock; в поле On Event указываем, какую команду необходимо выполнить при запуске задачи. В данном случае команду `/system script run Backup_to_email`, которая запускает скрипт `Backup_to_email`; нажимаем кнопку ОК для сохранения задачи.

Рисунок 2.17 - Настройка времени отправки резервной копии.

Теперь каждый день в 00:00:00 будет создаваться резервная конфигурация MikroTik и отправляться на указанный email.

2.2 Разработка СУБД

Приложение работает со встраиваемой реляционной базой данных SQLite. СУБД SQLite не использует парадигму клиент-сервер, то есть SQLite не является отдельно работающим процессом, с которым взаимодействует программа, а предоставляет библиотеку, с которой программа работает на одной платформе и библиотека становится составной частью программы. Таким образом, в качестве протокола обмена используются вызовы функций

Изм.	Лист	№ докум.	Подпись	Дата

(API) библиотеки SQLite. Такой подход уменьшает накладные расходы, время отклика и упрощает программу. SQLite хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том устройстве, на котором выполняется программа.

2.2.1 Концептуальное проектирование

Процесс построения инфологической модели состоит из следующих шагов:

- определение сущностей;
- определение зависимостей между сущностями;
- задание первичных и альтернативных ключей;
- определение атрибутов сущностей.

Далее в таблице 2.1 приведен обзор сущностей, где представлены все спроектированные сущности БД, заголовки столбцов означают следующее:

- название сущности – название выделенной в предметной области сущности;
- идентификация таблицы – физическое имя сущности;
- количество записей – предлагаемое количество записей сущности;
- описание – описание сущности.

Таблица 2.1 – Таблица сущностей

Название	Идентификатор	Описание
шлюзы	gateway	Информация о шлюзах
пользователи	users	Информация о пользователе
Скрипты на исполнение	scripts	Информация о используемых в программе скриптах

Продолжение таблицы 2.1

Название	Идентификатор	Описание
Настройки DHCP	dhcр	Информация о настройках позволяющих автоматически получать IP адрес
VPN	vpn	Настройки vpn соединения
Firewall	firewall	Настройки межсетевого экрана
NAT	nat	Настройки преобразования сетевых адресов

2.2.2 Даталогическое проектирование

На данном этапе разработки БД была сформирована физическая модель БД, изображенная в таблицах 2.2 -2.9.

Таблица 2.3 – Отношение «Шлюзы» («gateway»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения Null	Роль
idgateway	Идентификатор шлюза	int	11	-	PrimaryKey
nameofic	Официальное наименование	VARCHAR	45	-	Attribute
nameuser	Пользовательское наименование	Int	11	-	Attribute
login	Логин				FK, Attribute

Продолжение таблицы 2.3

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения Null	Роль
password	Пароль				FK, Attribute
ipaddr	IP устройства				Attribute
mask	Маска				Attribute
gateway	Шлюз по умолчанию				Attribute
port	порт				Attribute

Таблица 2.4 – Отношение «Пользователи» («users»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения Null	Роль
idusers	Идентификатор пользователя	int	11	-	PrimaryKey
group	Официальное наименование	VARCHAR	45	-	Attribute
email		VARCHAR	45		

Продолжение таблицы 2.4

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения Null	Роль
enable	Пользователь действующий или заблокирован	int	11		

Таблица 2.5 – Отношение «Скрипты на исполнение» («scripts»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения Null	Роль
idscripts	Идентификатор скрипта	int	11	-	PrimaryKey
name	Имя скрипта	VARCHAR	45	-	Attribute
filepath	Путь к скрипту	VARCHAR	45		Attribute
description	Описание скрипта	TEXT			Attribute
enable	действующий или заблокирован	int	11		Attribute

Таблица 2.6 – Отношение «Настройки DHCP» («dhcp»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения Null	Роль
iddhcp	Идентификатор	int	11	-	PrimaryKey
interface	Имя скрипта	VARCHAR	45	-	Attribute
addressList	пул адресов	TEXT			Attribute
gatewaydefault	шлюз	VARCHAR	45		Attribute
dns	DNS сервер по умолчанию	VARCHAR	45		Attribute

Таблица 2.7 – Отношение «Настройки VPN» («vpn»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения Null	Роль
idvpn	Идентификатор	int	11	-	PrimaryKey
user	Имя скрипта	VARCHAR	45	-	Attribute
password	пул адресов	VARCHAR	45		Attribute
serviceport	порт	INT			

Продолжение таблицы 2.7

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения Null	Роль
protocol	протокол	VARCHAR	45		
chain	цепочка	VARCHAR	45		

Таблица 2.8 – Отношение «Настройки Firewall » («firewall»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения Null	Роль
idfirewall	Идентификатор	int	11	-	PrimaryKey
interface	Интерфейс к которому применяется правило	VARCHAR	45	-	Attribute
localnetwork	Адрес интерфейса	VARCHAR	45		Attribute
port	порт	int	11		Attribute

Таблица 2.9 – Отношение «Настройки NAT » («nat»)

Имя столбца	Содержательное описание	Тип данных	Размерность	Возможность значения Null	Роль
idnat	Идентификатор	int	11	-	PrimaryKey
rules	Правила	VARCHAR	45	-	Attribute
adresarList	Список адресов	TEXT			Attribute

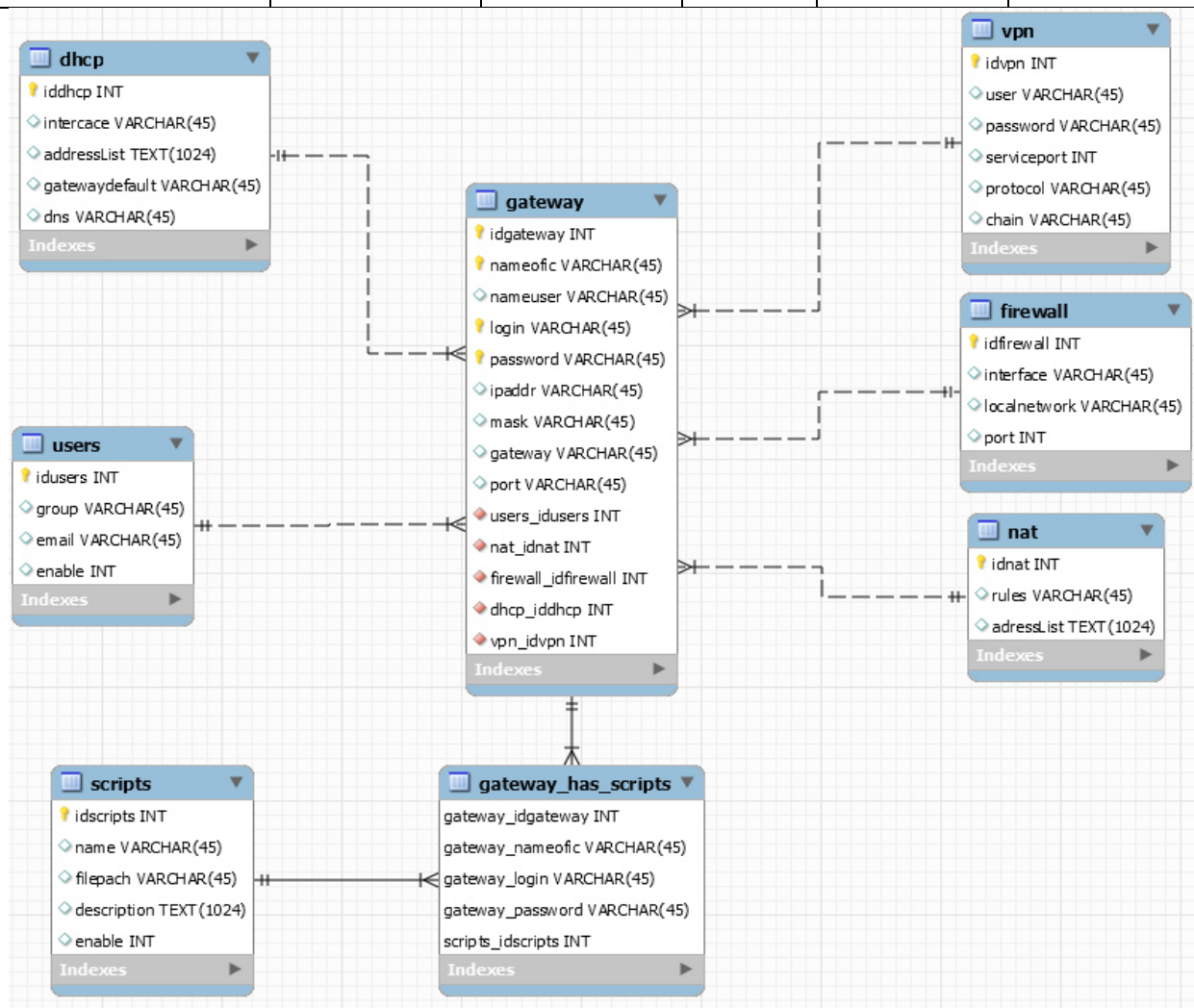


Рисунок 2.18 - Структура базы данных.

2.3 Разработка клиентского приложения

На рисунке 2.19 представлена упрощенная диаграмма классов приложения. Как видно из диаграммы основой функционала является класс MainPage, который использует объекты и функции других классов, таких как класс Statistica, Backup, System, Scripts и т.д.

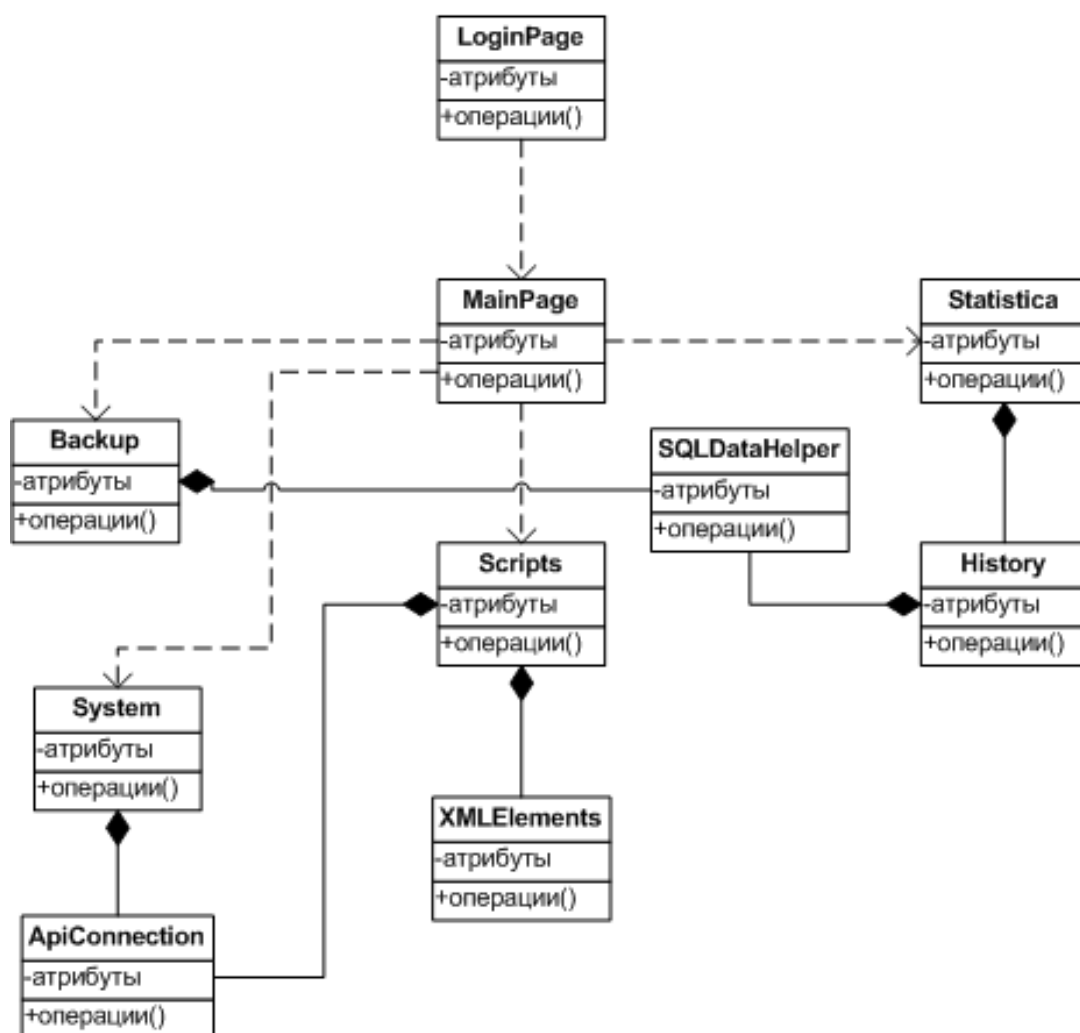


Рисунок 2.19 - Диаграмма классов

Класс MainPage является основным окном приложения, из которого вызываются все операции приложения. Класс Backup отвечает за создание и вывод информации о резервных копиях роутера, класс Statistica отвечает за отображение статистики входа в систему пользователя, для чего он

использует модель History. Класс Scripts является классом, который отображает результат работы скриптов. Для чего он использует экземпляр класса XMLElements, который загружает файл скрипта и затем с помощью объекта класса ApiConnection отправляет его на исполнение в устройство. Устройство выполняет скрипт, а в результате опроса устройства классом System пользователь видит результат работы скрипта.

Для наглядности представления вариантов использования классов и их объектов покажем процессы в виде диаграмм деятельности.

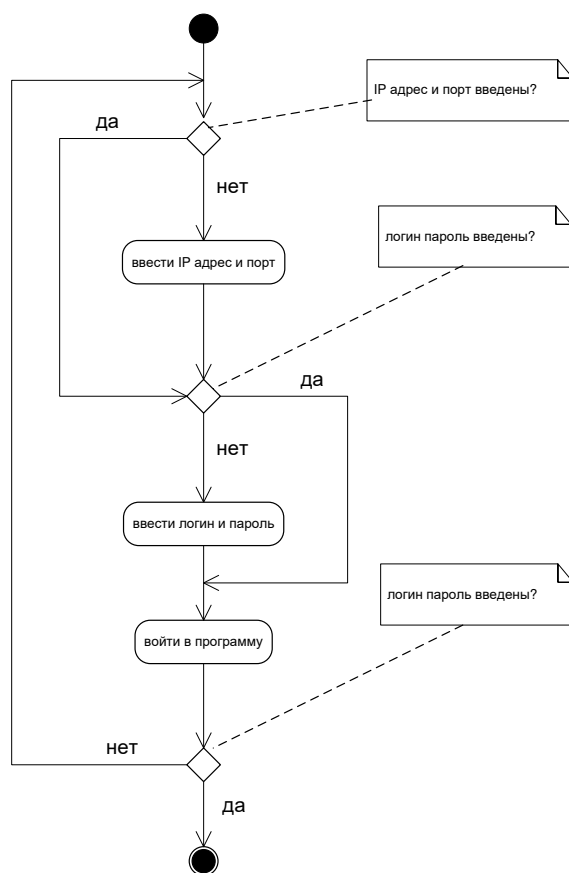


Рисунок 2.20 - Диаграмма деятельности для вариантов использования «Вход в систему».

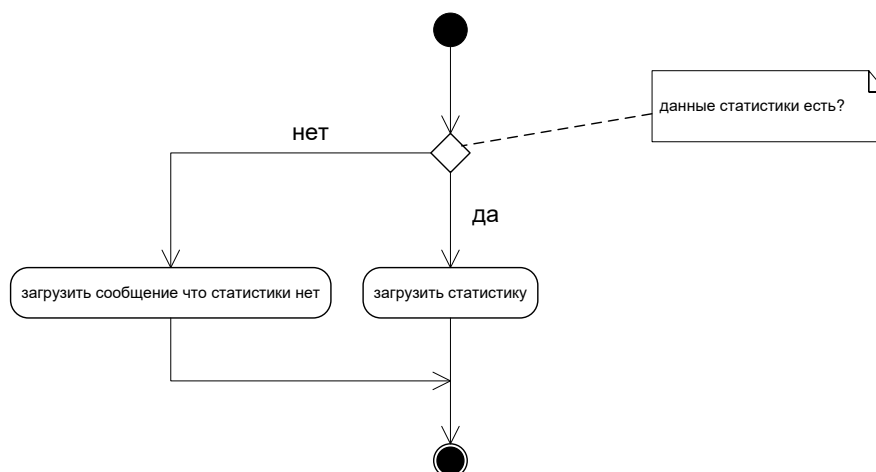


Рисунок 2.21 - Диаграмма деятельности для вариантов использования «Статистика».

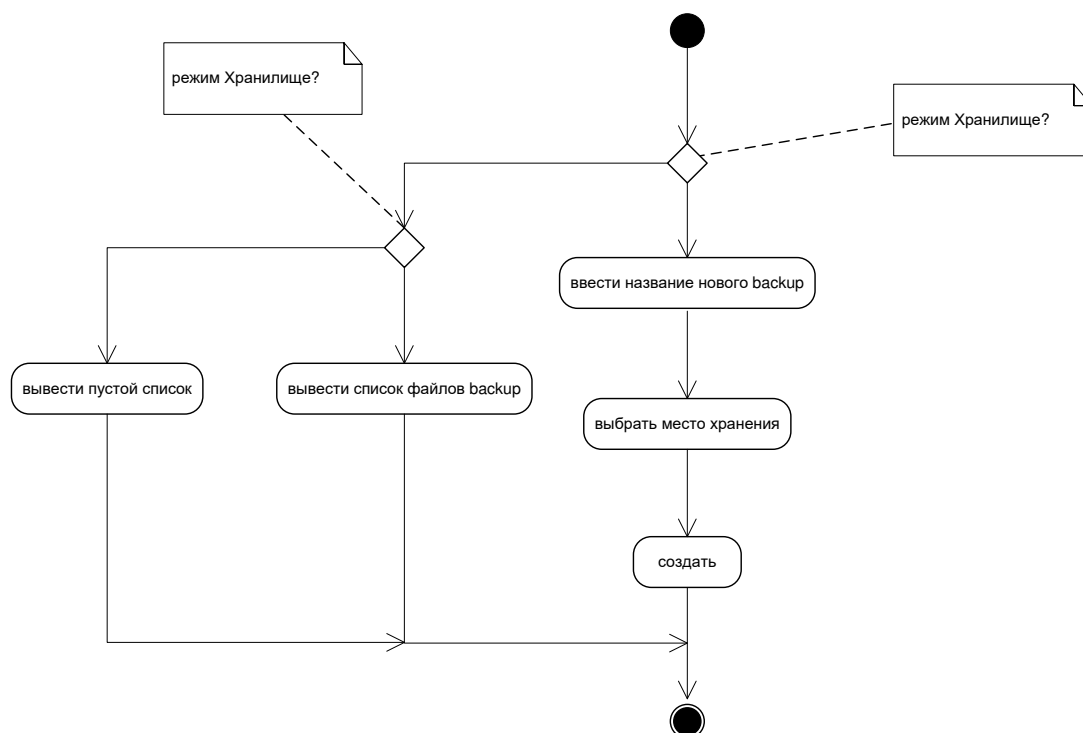


Рисунок 2.22 - Диаграмма деятельности для вариантов использования «Резервные копии».

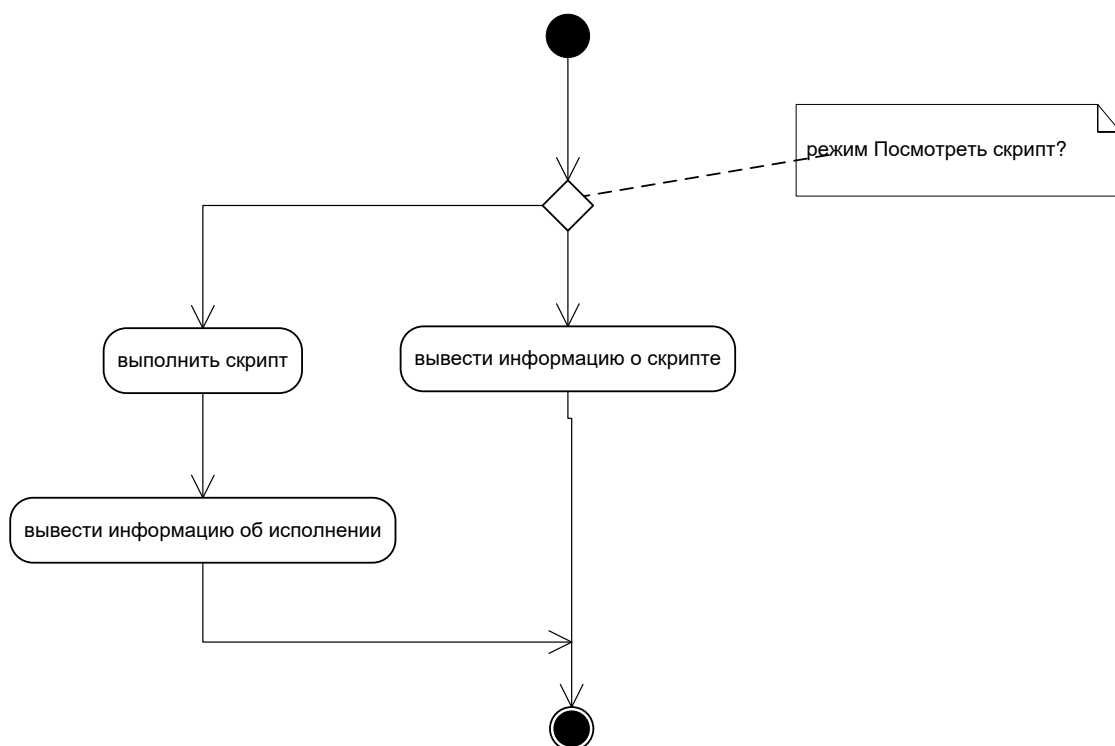


Рисунок 2.23 - Диаграмма деятельности для вариантов использования «Скрипты».

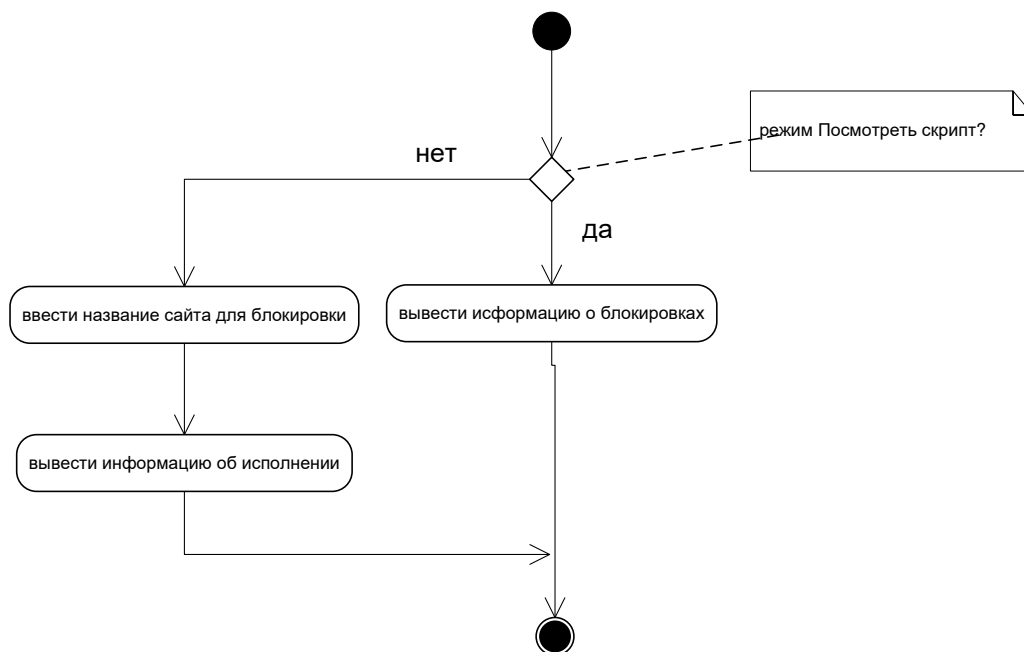


Рисунок 2.24 - Диаграмма деятельности для вариантов использования «Фильтрация».

Как показано на диаграмме последовательности пользователь из главного меню приложения последовательно вызывает объекты классов для реализации той или иной функции. Так для просмотра существующих и создания новых резервных копий, класс MainPage с помощью функции callBackup вызывает модуль Backup, который с помощью функции getBackupList показывает список всех созданных резервных копий. Далее метод callback возвращает пользователя на главную страницу. Далее пользователь с помощью функции callScript попадает в модуль Script, где метод getScriptsList отображает список доступных скриптов. Далее метод doScript загружает скрипт в файл, который транслируется в устройство методом runScript. Далее скрипт выполняется на устройстве. Результат исполнения можно посмотреть и вернуться в основное меню.

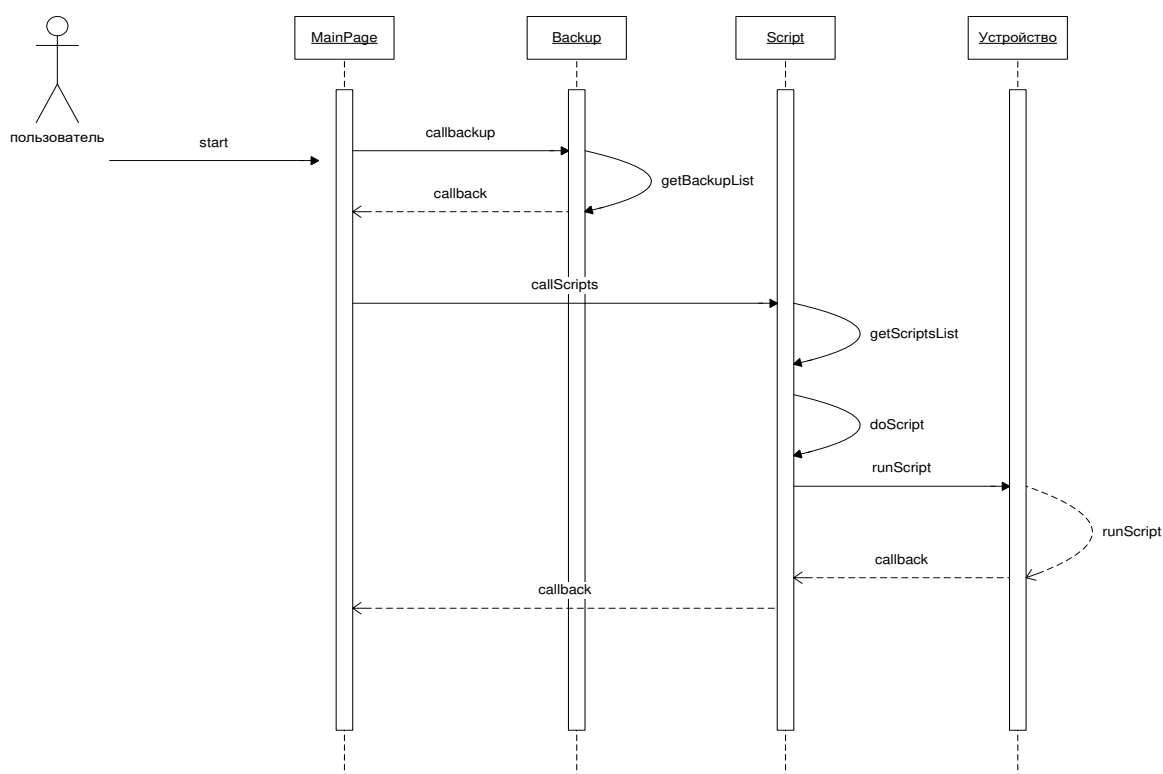


Рисунок 2.25 - Диаграмма последовательности работы части приложения.

Основным классом, обеспечивающим связь с устройством, является `ApiConnectionImpl`, который является потомком абстрактного класса `ApiConnection`. В классе `ApiConnectionImpl` содержится метод статического соединения с параметрами хост `String` (для адреса маршрутизатора) и целочисленное значение порта, на котором должно быть установлено соединение.

Этот метод возвращает новый экземпляр класса `ApiConnection`, в возвращенном объекте содержится информация о соединении или `null` если соединения с устройством не произошло.

В классе `ApiConnectionImpl` метод `open` обеспечивает работу с уже соединенным устройством, инициализирует потоки для отправки команд, читает полученные ответы и обеспечивает все рутинные команды во время сеанса:

```
private Socket sock = null;
private DataOutputStream out = null;
private DataInputStream in = null;
private boolean connected = false;
private BlockingQueue<Command> outQueue;
private Writer writer;
private Reader reader;
private Processor processor;
/**
 */
private void open(String host, int port, boolean secure) throws
    ApiConnectionException {
    try {
        InetAddress ia = InetAddress.getByName(host);
        if (secure) {
            sock = openSSLSocket(ia, port);
        } else {
            sock = new Socket();
            sock.connect(new InetSocketAddress(ia, port),
                SOCKET_CONNECT_TIMEOUT);
        }
    }
```

```

        in = new DataInputStream(sock.getInputStream());
        out = new DataOutputStream(sock.getOutputStream());
        connected = true;
        outQueue = new LinkedBlockingQueue<Command>();
        writer = new Writer();
        writer.setDaemon(true);
        writer.start();
        reader = new Reader();
        reader.setDaemon(true);
        reader.start();
        processor = new Processor();
        processor.setDaemon(true);
        processor.start();
    } catch (UnknownHostException ex) {
        connected = false;
        throw new ApiConnectionException(String.format("Unknown host
        '%s'", host), ex);
    } catch (IOException ex) {
        connected = false;
        throw new ApiConnectionException(String.format("Error connecting
        to %s:
        %d : %s", host, port, ex.getMessage()), ex);
    }
}

```

Внутренние классы Reader и Processor обеспечивают многопоточную обработку полученных от устройства ответов, соответственно. Данные классы получают ответы от класса Response. Класс Reader преобразует полученные ответы в символьные строки с использованием декодирования вспомогательного класса Util, а затем декодированные ответы отправляет в очередь LinkedBlockingQueue, откуда они извлекаются классом Processor путем вызова метода take и обрабатываются программой.

Класс ApiConnectionImpl использует тегирование отправленных команд, чтобы правильно определить правильный порядок отправки и получения сообщений (для этого используется класс ResultListener). В

данном классе имеется используется закрытый синхронизированный метод nextTag, который возвращает значение глобального целочисленного счетчика тегов в форме шестнадцатеричной строки.

Класс ConcurrentHashMap хранится в памяти, в которой находятся ключевые строковые значения тегов, а значение является ссылкой на интерфейс ResultListener. При отправке команды в класс ConcurrentHashMap вставляется запись с данным тегом и соответствующим экземпляром класса, реализующего данный интерфейс, поэтому после получения ответа известно, для какого объекта предназначен ответ.

Класс ConnectionManager сохраняет ссылку на экземпляр соединения, который иницируется при вызове метода createConnection, который открывает новое соединение. Ссылку на активное соединение можно получить, вызвав метод getConnection. Этот метод проверяет, что сохраненная ссылка не является нулевой, и если это так, метод генерирует исключение ConnectionNullException, в противном случае он возвращает экземпляр подключения.

Этот класс также поддерживает ссылку на маршрутизатор, с которым соединение было установлено в последний раз. Эта ссылка используется в предоставленном методе повторного подключения, который попытается восстановить соединение с этим маршрутизатором.

```
public class ConnectionManager {  
    private static ConnectionManager instance;  
    private ConnectionManager() {  
    }  
    public static synchronized ConnectionManager getInstance() {  
        if (instance == null) {  
            instance = new ConnectionManager();  
        }  
        return instance;  
    }  
}
```

Класс `ConnectionManager` сохраняет ссылку на экземпляр объекта соединения, который иницируется при вызове метода `createConnection`. Метод `createConnection` открывает новое соединение. Ссылку на активное соединение можно получить, вызвав метод `getConnection`. Этот метод проверяет, что сохраненная ссылка не является нулевой, и если это так, метод генерирует исключение `ConnectionNullException`, в противном случае он возвращает экземпляр подключения.

2.4 Развертывание приложения

Реализуемое приложение представляет собой исполняемый файл в формате APK.

APK формат архивных исполняемых файлов-приложений для Android. Каждое приложение Android скомпилировано и упаковано в один файл, который включает в себя весь код приложения (.DEX файлы), ресурсы, активы (assets), файл манифеста `AndroidManifest.xml` и нативные библиотеки (jniLibs). Файлы с данным расширением хранятся в магазинах приложений (например, Google Play), и/или загружаются в смартфон либо через компьютер для их использования, и устанавливаются пользователем вручную.

На рисунке 2.26 представлена диаграмма развертывания, которая отображает структуру приложения.

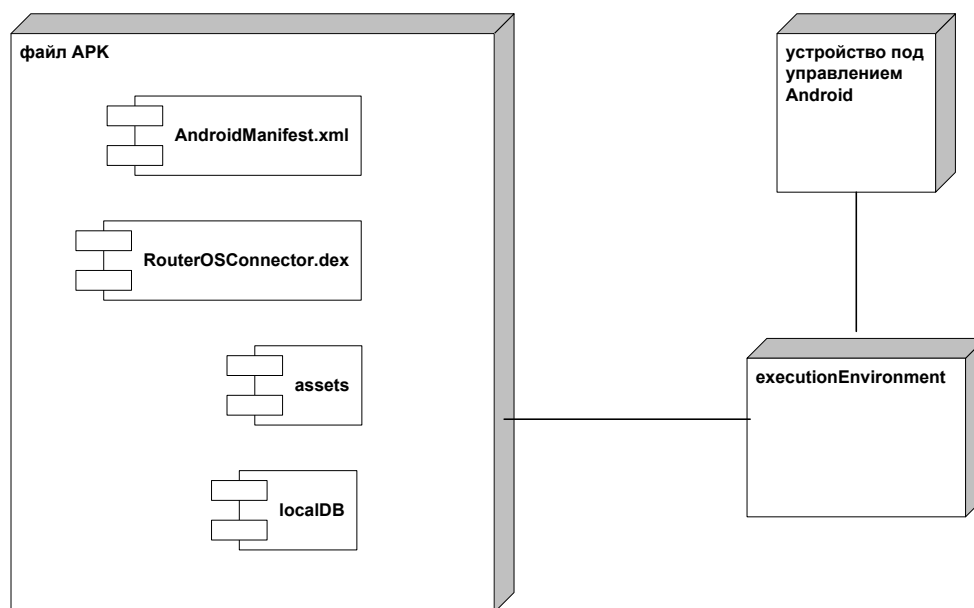


Рисунок 2.26 - Диаграмма развертывания.

2.5 Описание работы приложения

Работа в приложении начинается со входа в него. Для чего пользователю требуется ввести IP адрес устройства, порт по которому будет осуществляться взаимодействие с устройством, а также логин и пароль устройства.

Адрес устройства задается внутренним IP адресом и на этапе входа в систему не проверяется. В случае ввода ошибочного IP адреса и/или порта сообщение об ошибке появится при исполнении скриптов и соединении с устройством. В этом случае пользователю придется выйти из программы и снова ввести исходные данные.

В главном меню приложения имеются кнопки выбора функций, которые пользователь хочет реализовать.

Статистика – показывает когда пользователь входил в систему;

Бэкап – работа с резервными копиями;

Скрипты – исполнение скриптов на устройстве;

Фильтр – возможность фильтрации сайтов;

Скорость – ограничение скорости устройства;

Система – системная информация об устройстве.

RouterOSConnector

IP адрес 198.168.0.1 Порт 8728

Логин admin

Пароль

Войти

Рисунок 2.27 – Вход в систему

Главное меню

Статистика

Резервные копии

Скрипты

Фильтрация

Ограничение скорости

Система

Рисунок 2.28 – Главное меню

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.07.001 ПЗ

Лист

70

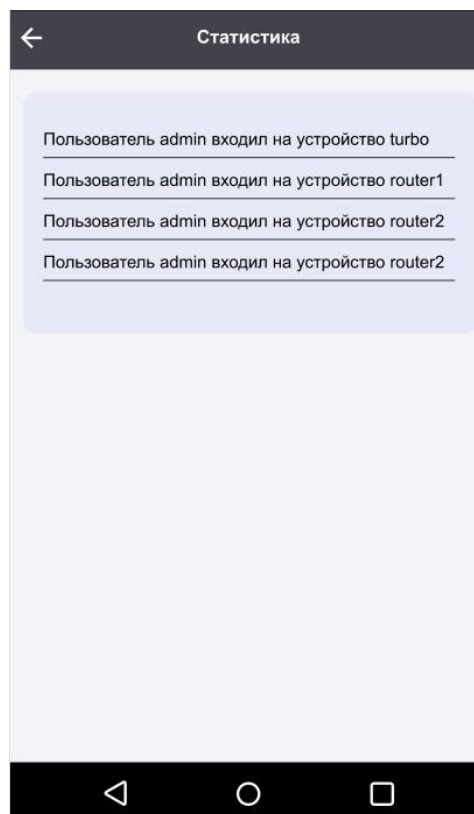


Рисунок 2.29 – Меню статистика

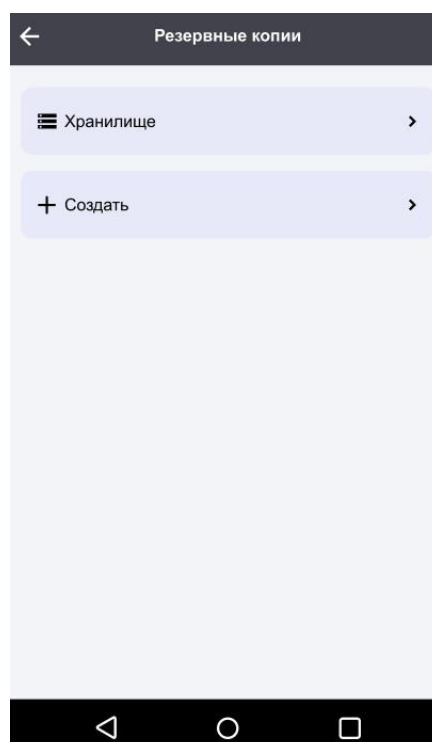


Рисунок 2.30 – Модуль резервных копий

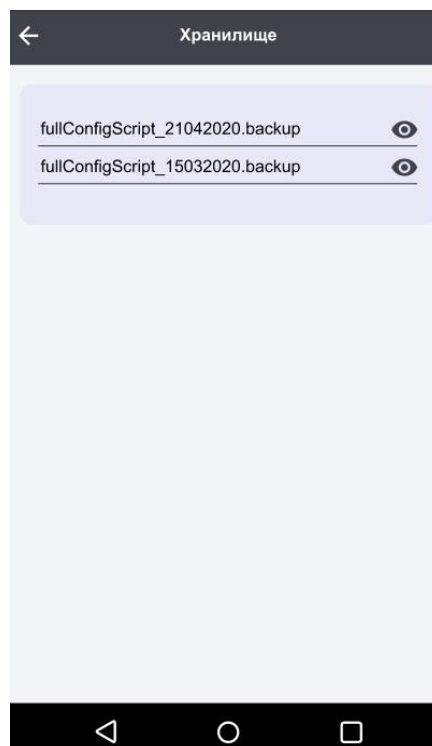


Рисунок 2.31 – Хранилище резервных копий

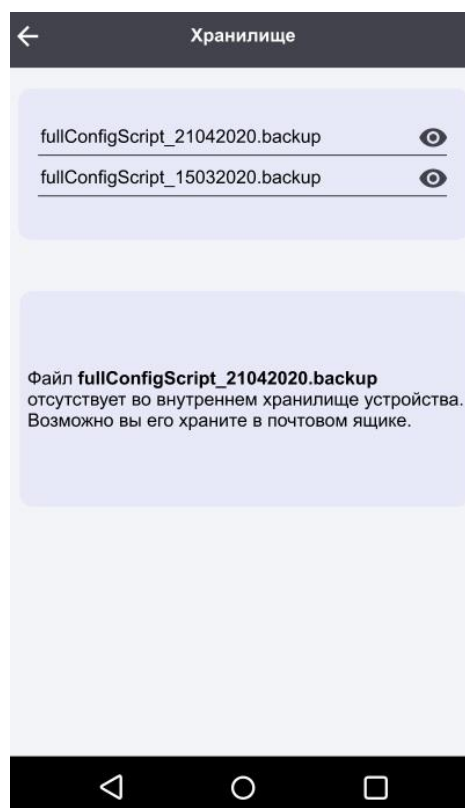


Рисунок 2.32 – Сообщение об ошибке

← Создание резервной копии

Устройство RB2019UIAS-2

Наименование turbo

Хранилище: Почта ☒ Роутер ☐

Название fullConfigScript

E - mail a.k.@gmail.com

Создать

Рисунок 2.33 – Создание резервных копий

← Скрипты

script_1.rsc	👁	✓
script_2.rsc	👁	✓

Рисунок 2.30 – Модуль скрипты

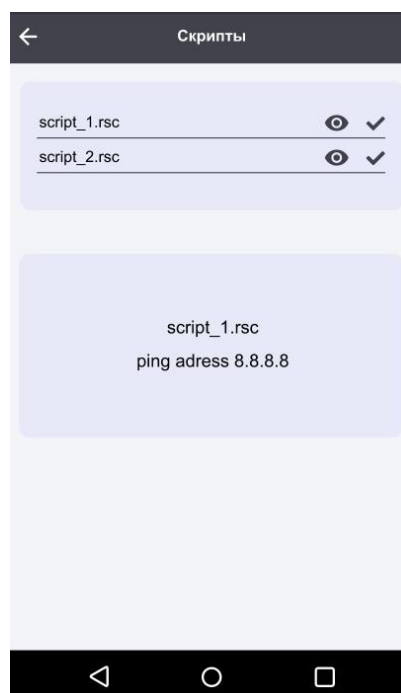


Рисунок 2.31 – Просмотр скрипта



Рисунок 2.32 – Исполнение скрипта

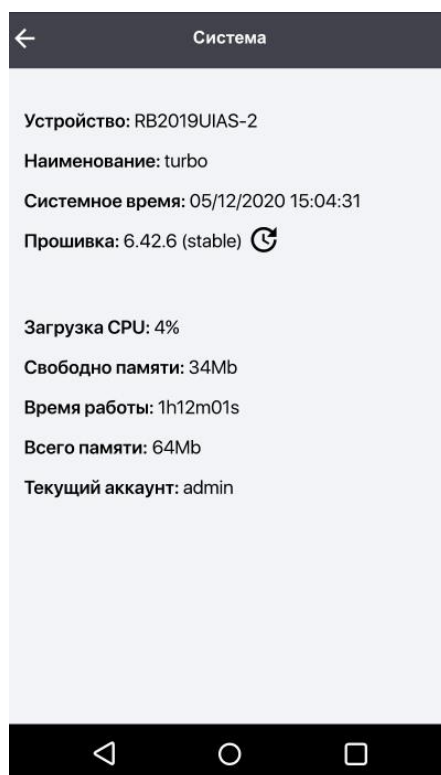


Рисунок 2.32 – Система

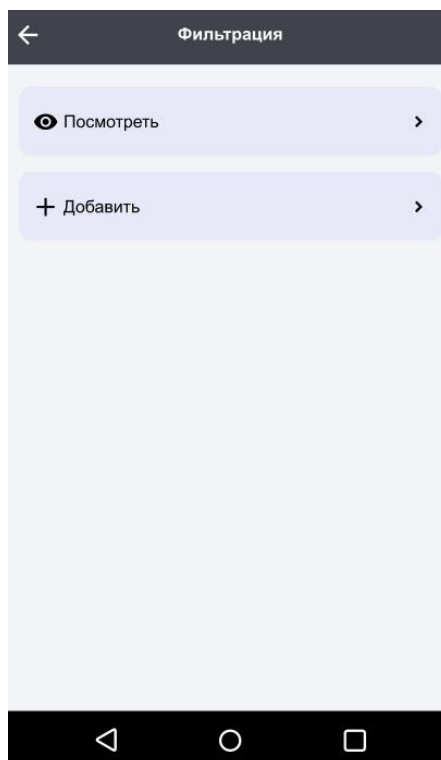


Рисунок 2.33 – Фильтрация

← Добавление ресурса

Устройство RB2019UIAS-2

Наименование turbo

Название ресурса для блокировки:
facebook.com

Добавить

Рисунок 2.34 – Добавление ресурса

← Фильтрация

Заблокированные ресурсы:

youtube.com x

facebook.com x

Рисунок 2.35 – Заблокированные ресурсы

← Ограничение скорости

Устройство RB2019UIAS-2

Наименование turbo

Ограничение на отдачу:

Название очереди psq-upload-2M

Ограничение, Мб 2

Ограничение на загрузку:

Название очереди psq-download-2M

Ограничение, Мб 2

Общее:

Общий лимит, Мб 18

Подсеть 192.168.0.0/24

Применить

Рисунок 2.36 – Ограничение скорости

← Ограничение скорости

Ограничения применены

Рисунок 2.37 – Применение ограничений

В результате данной главы разработано мобильное приложение в соответствии с требованиями задания.

3 Технология разработки скриптов routeros

Будет рассматриваться взаимодействие с устройством через специальный программный интерфейс управления (API - application programming interface). По функциональности этот интерфейс аналогичен конфигурации устройства через telnet или ssh. Отличие состоит в том, что telnet и ssh относятся к интерфейсам взаимодействия устройства и человека, а API-взаимодействия устройства и устройства, т.е. MikroTik API как нельзя лучше подходит для целей автоматизации.

В качестве языка программирования при демонстрации работы с API RouterOS будет использоваться Python. RouterOS API Python поддерживает взаимодействие с сетевыми устройствами через протоколы telnet и ssh, однако работа с этими модулями трудоёмка, например, результат выполнения команды необходимо выделять с помощью регулярного выражения из общего ответа устройства. Одним из модулей, который значительно упрощает взаимодействие с сетевыми устройствами, является модуль netmiko.

Для взаимодействия с устройством необходимо включить поддержку API непосредственно на устройстве, перейдя в раздел «/ip services». По умолчанию API работает через порт 8728. RouterOS поддерживает защищённый вариант программного интерфейса API over SSL (api-ssl по умолчанию использует порт 8729).

В рамках проекта реализован класс ApiRos, позволяющий выполнить подключение и взаимодействие с устройством через API. Акцент реализации сделан на три функции: открытие/закрытие сокета и вывод ответа устройства на экран. При демонстрации будут использованы следующие функции и классы:

Libapi - основные функции и классы для взаимодействия с устройством;

ApiRos -класс устройств под управлением RouterOS;

socketOpen - функция открытия сокета;

socketClose- функция закрытия сокета;

login - функция авторизации на устройстве;

writeSentence - функция отправки команды на устройство;

readResponse - функция получения результата выполнения команды с устройства.

Следует упомянуть о формате команд, т.к. он отличается от синтаксиса, используемого при конфигурации вручную. Команды вводятся в виде списка, где первый элемент - команда, в которой пробелы заменяются на символ «/», а последующие элементы уточняют запрос:

команды просмотра: фильтр запроса начинается с вопросительного знака, после чего следует текст фильтра, например «?type=ether».

Уточняющих команд может быть несколько;

команды изменения конфигурации: фильтр начинается с символа равенства, после чего следует имя параметра и его значение, например «=name=ISP1». Уточняющих команд может быть несколько.

Задача 1. Разработать программу для получения списка IP-адресов на устройствах R2 и R3.

```
import libapi
```

```
devices = [  
    { 'ip': '172.16.16.13',  
      'login': 'user',  
      'pass': 'user'},  
    { 'ip': '172.16.16.12',  
      'login': 'spw',
```

```

        'pass': 'spw'}}]

for device in devices:
    print("Connect to {}".format(device['ip']))

    #Создание сокета и объекта устройства
    s = libapi.socketOpen(device['ip'])
    dev_api = libapi.ApiRos(s)

    #Авторизация на устройстве
    if not dev_api.login(device['login'], device['pass']):
        break

    #Список команд
    command = ["/ip/address/print"]

    #Выполнение команды на устройстве
    dev_api.writeSentence(command)

    #Получение результата выполнения команды
    res = libapi.readResponse(dev_api)

    #Закрытие сокета
    libapi.socketClose(s)

    #Форматированный вывод результата команды
    print("    Command result:")
    print("        {:^18} {:10}".format('IP', 'Interface'))

    for element in res:
        print("            {:18}
{: ^10}".format(element[2].split('=')[2],
                    element[4].split('=')[2]))
    print('')

```

Принцип работы программы заключается в следующем:

- в словаре `devices` описаны реквизиты доступа к R2 и R3;
- в цикле обрабатывается словарь `devices` и выполняется подключение к каждому из устройств;
- после подключения на устройстве выполняется команда `"/ip address print"`;
- результат выполнения команды сохраняется в переменную `res`;
- закрывается подключение к устройству;
- выполняется форматированный вывод содержимого переменной `res`.

Результат выполнения программы представлен на рисунке 3.1:

```

aleksandr@aleksandr-PC:~/spw$ python3.6 ip_get.py
Connect to 172.16.16.13:
  Socket open: done
  Authentication: done
  Command complete: done
  Socket closed: done
  Command result:
                IP          Interface
        172.16.16.13/28      ether1

Connect to 172.16.16.12:
  Socket open: done
  Authentication: done
  Command complete: done
  Socket closed: done
  Command result:
                IP          Interface
        172.16.16.12/28      ether1

aleksandr@aleksandr-PC:~/spw$ █

```

Рисунок 3.1 - Результат выполнения программы `ip_get`

Задача 2. Разработать программу, которая будет создавать интерфейс и ассоциировать созданный интерфейс с выделенным IP-адресом.

```

import libapi

devices = [
    { 'ip': '172.16.16.13',

```

```

        'login': 'user',
        'pass': 'user'},
        { 'ip': '172.16.16.12',
          'login': 'spw',
          'pass': 'spw'}}]
new_network = '10.52.52.{}/32'

new_interface = 'br_ospf'
for device in devices:

    print("Connect to {}".format(device['ip']))
    #Создание сокета и объекта устройства

    s = libapi.socketOpen(device['ip'])
    dev_api = libapi.ApiRos(s)
    #Авторизация на устройстве

    if not dev_api.login(device['login'], device['pass']):
        break

    #Команда для добавление bridge-интерфейса

    command = ["/interface/bridge/add",
"=name={}".format(new_interface)]
    #Выполнение команды на устройстве
    dev_api.writeSentence(command)
    #Получение результата выполнения команды
    res = libapi.readResponse(dev_api)
    #Команда для добавления IP-адреса на bridge-интерфейс
    command = ["/ip/address/add",

"=address={}".format(new_network.format(device['ip'].split('.')[
3]))),

        "=interface={}".format(new_interface)]
    #Выполнение команды на устройстве
    dev_api.writeSentence(command)
    #Получение результата выполнения команды

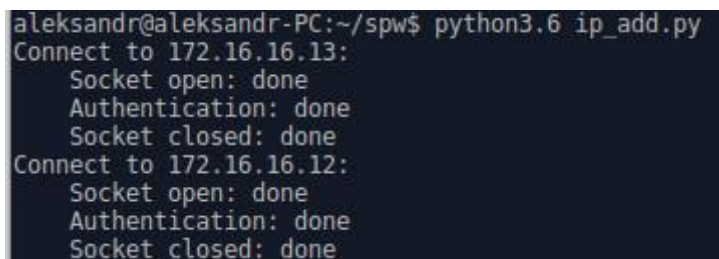
```

```

res = libapi.readResponse(dev_api)
#Закрытие сокета
libapi.socketClose(s)

```

Результат выполнения программы ip_add.py представлен на рисунке 3.2



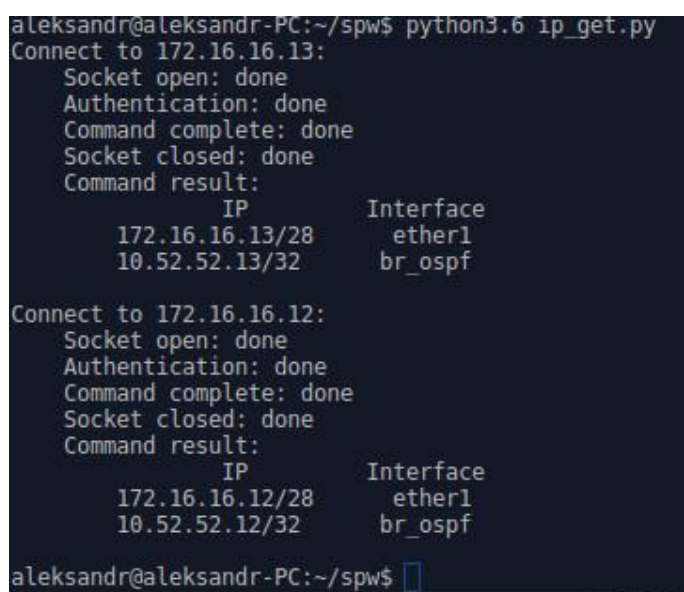
```

aleksandr@aleksandr-PC:~/spw$ python3.6 ip_add.py
Connect to 172.16.16.13:
Socket open: done
Authentication: done
Socket closed: done
Connect to 172.16.16.12:
Socket open: done
Authentication: done
Socket closed: done

```

Рисунок 3.2 - Результат выполнения программы int_dev_get

Несмотря на то, что в переменную res записывался результат применения команды, она не использовалась в текущей программе. Убедимся, что ip-адреса успешно добавлены, запустив программу ip_get.



```

aleksandr@aleksandr-PC:~/spw$ python3.6 ip_get.py
Connect to 172.16.16.13:
Socket open: done
Authentication: done
Command complete: done
Socket closed: done
Command result:
      IP      Interface
172.16.16.13/28 ether1
10.52.52.13/32  br_ospf

Connect to 172.16.16.12:
Socket open: done
Authentication: done
Command complete: done
Socket closed: done
Command result:
      IP      Interface
172.16.16.12/28 ether1
10.52.52.12/32  br_ospf

aleksandr@aleksandr-PC:~/spw$

```

Рисунок 3.3 - Проверка исполнения скрипта

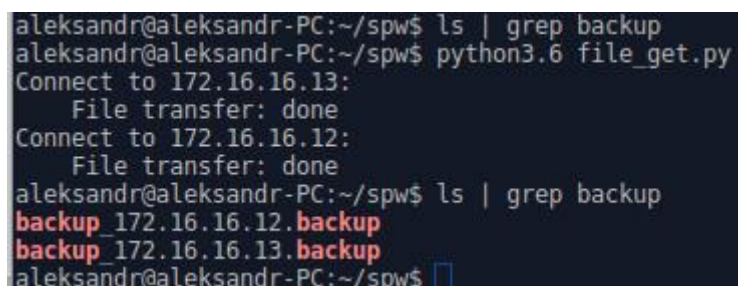
По результату работы программы видно, что программа ip_add выполнила действия согласно поставленной задаче: создан bridge-интерфейс и он ассоциирован с IP-адресом из выделенной сети.

Задача 3. Формирование backup прошивки роутера, с сборанием их на персональном компьютере пользователя.

```
import ftplib

devices = [
    { 'ip': '172.16.16.13',
      'login': 'user',
      'pass': 'user'},
    { 'ip': '172.16.16.12',
      'login': 'spw',
      'pass': 'spw'}}
filename_pattern = 'backup_{}.backup'
for device in devices:
    print("Connect to {}".format(device['ip']))

    filename = filename_pattern.format(device['ip'])
    with ftplib.FTP(device['ip'], device['login'],
device['pass']) as con:
        with open(filename, "wb") as f:
            con.retrbinary('RETR ' + filename, f.write)
            print("    File transfer: done")
```



```
aleksandr@aleksandr-PC:~/spw$ ls | grep backup
aleksandr@aleksandr-PC:~/spw$ python3.6 file_get.py
Connect to 172.16.16.13:
    File transfer: done
Connect to 172.16.16.12:
    File transfer: done
aleksandr@aleksandr-PC:~/spw$ ls | grep backup
backup_172.16.16.12.backup
backup_172.16.16.13.backup
aleksandr@aleksandr-PC:~/spw$
```

Рисунок 3.4 - Формирование backup

Рассмотрены основные методы автоматизации при эксплуатации устройств на базе RouterOS. Представлено описание и ряд базовых примеров

при работе с языком написания скриптов RouterOS, системой управления конфигураций на примере API RouterOS.

4 Экономическое обоснование разработки мобильного приложения

4.1 Обоснование выбора объекта

Расчет затрат на разработку мобильного приложения будет производиться в соответствии с методическими указаниями [Ошибка! Источник ссылки не найден., Ошибка! Источник ссылки не найден.]. В процессе дипломного проектирования была проведена модернизация системы автоматизации учета на рабочем месте сотрудника ОМТС филиала ОАО «МРСК волги» - «Пензаэнерго». Это позволило повысить эффективность труда, снизить количество ошибок при ведении учета, а также выполнять формирование отчетных форм с высокой скоростью и минимальными трудозатратами.

Капитальные вложения во внедрение системы следует рассматривать, как инвестиции, необходимые для получения прибыли. Поэтому возникает необходимость в проведении оценки эффективности инвестиционного проекта разработки. Среди модернизаций системы можно выделить создание отчетных форм, объектов базы данных и пользовательских процедур.

Над модернизацией модулей мобильного приложения в филиале «Пензаэнерго» работали следующие специалисты:

- ведущий специалист, знающий предметную область и формирующий задания на модернизацию;
- системный аналитик – технический консультант по технической части разработки ПО;
- программист, непосредственно занимающийся разработкой и доработкой ПО, а также выполняющий функции менеджера проекта;

– руководитель проекта, выполняющий функции по администрированию и ведению нормативно-справочной информации и общему курированию проекта.

Разработка будет вестись в помещениях и на оборудовании заказчика. В таблице 4.1 приведены данные для расчета затрат на модернизацию и внедрение мобильного приложения.

Таблица 4.1 – Данные для расчета затрат на внедрение мобильного приложения

Показатели, единица измерения	Значение показателя	Источник информации
Страховые взносы, %	30	В соответствие с НК РФ
Социальное страхование, %	2,9	В соответствие с НК РФ
Медицинское страхование, %	5,1	В соответствие с НК РФ
В пенсионный фонд, %	22	В соответствие с НК РФ
Эксплуатационные расходы до внедрения системы, руб.	50 000	По данным филиала ОАО «МРСК Волги» - «Пензаэнерго»
Накладные расходы, %	20	
Стоимость 1кВт/ч, руб.	2,32	
Стоимость одного часа работы системы, руб.	20	
Время разработки программы, месяц	2	По согласованию заказчика и исполнителя
Зарботная плата работающих, руб.	Время работы, раб. дни	
Показатели, единица измерения	Значение показателя	Источник информации

Продолжение таблицы 4.1

Показатели, единица измерения	Значение показателя	Источник информации	Показатели, единица измерения
Ведущий специалист	20 000	20	По данным филиала ОАО «МРСК Волги» - «Пензаэнерго»
Системный аналитик	25 000	15	
Программист	18 000	30	
Руководитель проекта	30 000	20	

4.2 Расчет затрат на модернизацию мобильного приложения

Затраты на модернизацию программы складываются из затрат на оплату труда специалистов с отчислениями на социальные нужды, в которые входят основная заработная плата разработчиков, страховые взносы, накладные расходы организации, где разрабатывается программа.

4.2.1 Расчет затрат на оплату труда

В таблице 4.2 приведены данные для расчета фонда оплаты труда в соответствии с таблицей 4.1.

Таблица 4.2 – Данные для расчета фонда оплаты труда

Специалист	Месячная заработная плата, руб.	Заработная плата в день, руб.
Ведущий специалист	20 000	909
Системный аналитик	25 000	1136
Программист	18 000	818
Руководитель проекта	30 000	1363

Заработная плата в день рассчитывается путем деления месячной заработной платы специалиста на 22 рабочих дня. В таблице 4.3 приведен расчет фонда оплаты труда.

Таблица 4.3 – Расчет фонда оплаты труда

Специалист	Время работы, раб. дни	Сумма оплаты, руб.
Ведущий специалист	20	18 180
Системный аналитик	15	17 040
Программист	30	24 540
Руководитель проекта	20	27 260
Всего		87 020

Сумма оплаты рассчитывается путем умножения заработной платы в день на отработанное время в соответствии с таблицей 4.1.

4.2.2 Расчет отчислений в страховые фонды

В таблице 4.4 приведен расчет страховых взносов, а именно на социальное страхование, обязательное медицинское и пенсионное страхование. Страховые взносы рассчитываются путем умножения заработной платы на соответствующую ставку.

Таблица 4.4 – Расчет страховых взносов

Специалист	Страховые взносы (30%), руб.	Социальное страхование (2,9%), руб.	Медицинское страхование (5,1%), руб.	Пенсионное страхование (22%), руб.
Ведущий специалист	5 454	527,22	927,18	3 999,6
Системный аналитик	5 112	494,16	869,04	3 748,8
Программист	7 362	711,66	1 251,54	5 398,8
Специалист	Страховые взносы (30%), руб.	Социальное страхование (2,9%), руб.	Медицинское страхование (5,1%), руб.	Пенсионное страхование (22%), руб.

Продолжение таблицы 4.4

Специалист	Страховые взносы (30%), руб.	Социальное страхование (2,9%), руб.	Медицинское страхование (5,1%), руб.	Пенсионное страхование (22%), руб.
Руководитель проекта	8 178	790,54	1 390,26	5 997,2
Всего	26 106	2 523,58	4 438,02	19 144,4

4.2.3 Расчет накладных расходов

По данным Пензаэнерго накладные расходы составляют 20% от затрат на оплату труда.

$$\text{Накладные расходы организации } P_{\text{НО}} = \frac{\alpha_{\text{НО}} * Z_0}{100},$$

где $\alpha_{\text{НО}}$ – процент накладных расходов организации;

Z_0 – основная заработная плата.

Таким образом, расходы на оплату труда составляют 87 020 рублей, Накладные расходы равны 17 404 рубля.

4.2.4 Расчет амортизационных отчислений

Для работы специалистов были использованы 4 компьютера стоимостью 20 000 рублей каждый. Компьютеры принадлежат заказчику. Расчет амортизационных отчислений зависит от выбранного способа расчета амортизации и от срока полезного использования. В этой работе расчет будет производиться линейным способом:

$$A_{\Gamma} = \frac{OC_{\text{п}} * H_{\text{а}}}{100},$$

где $OC_{\text{п}}$ – первоначальная стоимость компьютера,

$H_{\text{а}}$ – годовая норма амортизационных отчислений.

$$H_a = \frac{100}{n},$$

где n – количество лет срока полезного использования.

Срок полезного использования компьютера составляет 5 лет.
Следовательно, годовая норма амортизационных отчислений равна:

$$H_a = \frac{100}{5} = 20\%.$$

Таким образом, амортизация компьютера составит:

$$A_r = \frac{20000 \cdot 20\%}{100\%} = 4\,000 \text{ рублей.}$$

Так как работы велись по пятидневной рабочей неделе, то в году получается 258 рабочих дней. Дневная сумма амортизации рассчитывается путем деления суммы амортизационных отчислений за год на сумму рабочих дней в году:

$$\frac{4000}{258} = 15,5 \text{ рублей.}$$

Ежемесячная сумма амортизационных отчислений равна:

$$\frac{4000}{12} = 333,33 \text{ рубля.}$$

В таблице 4.5 приведены расходы на амортизацию.

Таблица 4.5 – Расходы на амортизацию

Компьютер	Время работы, раб. дни	Амортизационные отчисления, руб.
Компьютер ведущего специалиста	20	310
Компьютер системного аналитика	15	232,5
Компьютер программиста	30	465

Продолжение таблицы 4.5

Компьютер	Время работы, раб. дни	Амортизационные отчисления, руб.
Компьютер руководителя проекта	20	310
Всего		1 317,5

Таким образом, сумма затрат на амортизацию составит 1 317,5 рублей.

4.2.5 Расчет затрат на обслуживание оборудования

В соответствие с таблицей 4.1 затраты на обслуживание оборудования составляют 15% от амортизационных отчислений и равны:

$$1\,317,5 * 0,15 = 198 \text{ рублей.}$$

4.2.6 Расчет затрат на электроэнергию

В соответствие с паспортными данными компьютеров, их энергопотребление составляет 0,5 кВт/час. Стоимость одного кВт/час энергии составляет 2,32 рубля, компьютеры работают 8 часов в день. Тогда стоимость электроэнергии, потребляемой одним компьютером, составит: $0,5 * 2,32 * 8 = 9,28$ рублей в день. Затраты электроэнергии для всех компьютеров сведены в таблицу 4.6.

Таблица 4.6 – Расчет затрат на электроэнергию

Компьютер	Время работы, раб. дни	Расходы на электроэнергию, руб.
Компьютер ведущего специалиста	20	185,60
Компьютер системного аналитика	15	139,20
Компьютер программиста	30	278,40
Компьютер руководителя проекта	20	185,60
Всего		788,80

Общие затраты на электроэнергию за время внедрения проекта составят 788,80 рублей.

4.2.7 Расчет стоимости модернизации системы мобильного приложения

В таблице 4.7 приведена общая сумма затрат на модернизацию мобильного приложения.

Таблица 4.7 – Расчет стоимости модернизации мобильного приложения

Статья затрат	Сумма, руб.
Заработная плата	87 020
Страховые взносы	26 106
в том числе:	
Социальное страхование	2 523
Обязательное медицинское	4 438
В пенсионный фонд	19 144
Накладные расходы	17 404
Статья затрат	Сумма, руб.
Обслуживание оборудования	198
Амортизационные отчисления	1 317
Электроэнергия	788
Всего	132 833

4.2.8 Расчет цены программы

Норматив рентабельности, учитывающий прибыль организации, внедряющей систему автоматизации, равен 15%. Отсюда цена программы будет равна:

$$Ц_{\Pi} = 132\,833 * 1,15 = 152\,758 \text{ руб.}$$

4.3 Расчет эксплуатационных расходов

Расходы, связанные с эксплуатацией программы (P_{Σ}) в месяц, определяются по формуле:

$$P_{\Sigma} = T_{\text{МВ}} * P_{\text{rm}} + A_{\text{ПР}},$$

где $T_{\text{МВ}}$ – продолжительность машинного времени ЭВМ, используемой в течение месяца для решения задач с помощью мобильного приложения, машинно-час/месяц потребителя программы;

P_{rm} – эксплуатационные расходы, приходящиеся на 1 час машинного времени этой ЭВМ, руб/машинно-час;

$A_{\text{ПР}}$ – амортизационные отчисления с этой программы (руб/час).

Амортизационные отчисления находятся по формуле:

$$A_{\text{ПР}} = \text{Ц}_{\text{П}} * N_{\text{А}}/100$$

Срок полезного использования программы равен 5 лет. Таким образом, амортизация в год составит:

$$A_{\text{Г}} = \frac{152\,758 * 20\%}{100\%} = 30\,551 \text{ руб.}$$

Сумма ежемесячных амортизационных отчислений составит:

$$A_{\text{М}} = \frac{A_{\text{Г}}}{12} = \frac{30\,551}{12} = 2\,546 \text{ руб.}$$

Мобильное приложение будет эксплуатироваться 8 часов в сутки 258 дней в году. Тогда продолжительность машинного времени в год составит:

$$8\text{ч} * 258 \text{ дн.} = 2064 \text{ машинно-час/год.}$$

Продолжительность машинного времени в месяц составит:

$$\frac{2064}{12} = 172 \text{ машинно-час/месяц.}$$

Эксплуатационные расходы, приходящиеся на 1 час машинного времени ЭВМ в соответствие с таблицей 4.1 составят 20 руб/машинно-час. В итоге, эксплуатационные расходы в месяц после внедрения системы составят:

$$172 \cdot 20 + 2\,546 = 5\,986 \text{ руб.}$$

4.4 Расчет денежного потока

В целях экономического обоснования эффективности вложений необходимо произвести расчет показателей:

- чистого дисконтированного дохода (ЧДД);
- срока окупаемости вложений (Ток);
- индекса рентабельности (IP).

Денежный поток за первый месяц определяется как сумма затрат за первые 22 дня, в соответствие с данными «Пензаэнерго» эта сумма составляет 100 000,00 рублей. Денежный поток за 2 месяц определяется как сумма затрат за оставшиеся дни. По данным «Пензаэнерго» эта сумма составит 52 758 рублей.

Амортизационные отчисления представлены в виде суммы отчислений за оборудование и ПО:

$$A = A_{\text{ЭВМ}} + A_{\text{ПО}} = 1\,317 + 5\,986 = 7\,303 \text{ руб./месяц.}$$

Коэффициент дисконтирования K определяется так:

$$K = \frac{1}{(1+E)^t},$$

где t – номер месяца,

E – коэффициент (ставка) дисконтирования, в данной работе она принимает значение 11%.

$$K_1 = \frac{1}{(1 + 0.11)^1} = 0.901$$

$$K_2 = \frac{1}{(1 + 0.11)^2} = 0.812$$

$$K_3 = \frac{1}{(1 + 0.11)^3} = 0.731$$

$$K_4 = \frac{1}{(1 + 0.11)^4} = 0.659$$

$$K_5 = \frac{1}{(1 + 0.11)^5} = 0.594$$

$$K_6 = \frac{1}{(1 + 0.11)^6} = 0.535$$

$$K_7 = \frac{1}{(1 + 0.11)^7} = 0.482$$

$$K_8 = \frac{1}{(1 + 0.11)^8} = 0.434$$

$$K_9 = \frac{1}{(1 + 0.11)^9} = 0.391$$

$$K_{10} = \frac{1}{(1 + 0.11)^{10}} = 0.352$$

$$K_{11} = \frac{1}{(1 + 0.11)^{11}} = 0.317$$

$$K_{12} = \frac{1}{(1 + 0.11)^{12}} = 0.286$$

...

$$K_{24} = \frac{1}{(1 + 0.11)^{24}} = 0.082$$

Коэффициент K всегда меньше единицы, так как сумма денег, находящаяся на руках сегодня, больше той же суммы в будущем. ЧДД рассчитывается путем произведения денежного потока на коэффициент дисконтирования. Данные расчета приведены в таблице 4.8.

Исходя из расчета ЧДД видно, что денежный поток на первых 2-х месяцах разработки отрицательный, а начиная с 3-го месяца происходит экономия средств от использования системы.

Таблица 4.8 – Расчет ЧДД

№ ме- ся-ца	Инвести- ции, руб.	Текущие эксплуатационные расходы, руб.			Амор- тиза-ция, руб.	Денеж-ный поток, руб.	Ко- эффи- циент дис-кон- тиро- вания	ЧДД, руб.
		До внед- рения	Пос-ле внед-рения	Эконо-мия				
1	100 000,00	0	0	0	0	-100000	0,901	-90 090,09
2	52 758,00	0	0	0	0	-52758	0,812	-42 819,58
3	0	50 000	7 303,00	42 697,00	1 317,00	44 014,00	0,731	32 182,66
4	0	50 000	7 303,00	42 697,00	1 317,00	44 014,00	0,659	28 993,39
5	0	50 000	7 303,00	42 697,00	1 317,00	44 014,00	0,593	26 120,17
6	0	50 000	7 303,00	42 697,00	1 317,00	44 014,00	0,535	23 531,68
7	0	50 000	7 303,00	42 697,00	1 317,00	44 014,00	0,482	21 199,71
8	0	50 000	7 303,00	42 697,00	1 317,00	44 014,00	0,434	19 098,84
9	0	50 000	7 303,00	42 697,00	1 317,00	44 014,00	0,391	17 206,16
10	0	50 000	7 303,00	42 697,00	1 317,00	44 014,00	0,352	15 501,05
11	0	50 000	7 303,00	42 697,00	1 317,00	44 014,00	0,317	13 964,91
12	0	50 000	7 303,00	42 697,00	1 317,00	44 014,00	0,286	12 581,00
13	0	50 000	7 303,00	42 697,00	1 317,00	44 014,00	0,258	11 334,23
14	0	50 000	7 303,00	42 697,00	1 317,00	44 014,00	0,232	10 211,02

Продолжение таблицы 4.8

№ ме- ся-ца	Инвести- ции, руб.	Текущие эксплуатационные расходы, руб.			Амор- тиза-ция, руб.	Денеж-ный поток, руб.	Ко- эффи- циент дис-кон- тиро- вания	ЧДД, руб.
		До внед- рения	Пос-ле внед-рения	Эконо-мия				
15	0	50 000	7 303,00	42 697,00	1 317,00	44 014,00	0,209	9 199,12
16	0	50 000	7 303,00	42 697,00	1 317,00	44 014,00	0,188	8 287,49
17	0	50 000	7 303,00	42 697,00	1 317,00	44 014,00	0,170	7 466,21
18	0	50 000	7 303,00	42 697,00	1 317,00	44 014,00	0,153	6 726,32
19	0	50 000	7 303,00	42 697,00	1 317,00	44 014,00	0,138	6 059,74
20	0	50 000	7 303,00	42 697,00	1 317,00	44 014,00	0,124	5 459,23
21	0	50 000	7 303,00	42 697,00	1 317,00	44 014,00	0,112	4 918,22
22	0	50 000	7 303,00	42 697,00	1 317,00	44 014,00	0,101	4 430,83
23	0	50 000	7 303,00	42 697,00	1 317,00	44 014,00	0,091	3 991,74
24	0	50 000	7 303,00	42 697,00	1 317,00	44 014,00	0,082	3 596,16
	ИТОГО							159 150,21

ЧДД положительный, поэтому инвестиционный проект жизнеспособен. Индекс рентабельности (доходности) определяется отношением суммы приведенных эффектов к размеру капитальных вложений:

$$IP = \frac{159\,150}{152\,178} = 1.04$$

Если $IP > 1$, то проект эффективен, иначе – нет. Индекс рентабельности проекта равен 1,04, следовательно, можно сделать вывод, что данный инвестиционный проект эффективен.

Срок окупаемости – это период времени, который потребуется для возмещения инвестиций. Он равен отношению затрат на разработку к сумме месячной экономии.

$$T_{OK} = \frac{152\,178}{42\,697} = 3,5 \text{ мес.}$$

Срок окупаемости составляет 3,5 месяца. Дальнейшее использование системы несет заказчику чистую прибыль.

Исходя из полученных результатов, можно сделать вывод, что внедренная и модернизированная система жизнеспособна и эффективна при использовании, ее внедрение и эксплуатация дает реальную экономию за счет сокращения времени на сбор и обработку информации, система окупается за 3,5 месяца, а дальнейшая ее эксплуатация приносит заказчику чистую прибыль. Общая стоимость проекта будет составлять 152 178 рублей с учетом прибыли организации-исполнителя. Отчисления страховых взносов составляют 26 106 рублей. Внедрение программы будет иметь экономический эффект из-за того, что индекс рентабельности больше единицы.

					ПГУ 1.090501.07.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		99

5 Гигиена труда в системе мероприятий по сохранению высокой трудоспособности пользователя пэвм

Гигиена труда – раздел гигиены, изучающий трудовые процессы и производственную среду, их воздействие на организм человека и разрабатывающий практические мероприятия по созданию наиболее здоровых условий труда.

Гигиена труда изучает: формы и методы организации труда и отдыха, состояние организма в процессе трудовой деятельности, характер и особенности рабочих движений, положений тела при работе, инструменты и орудия труда, применяемое сырье, технологию процессов, техническое оборудование, готовые и промежуточные продукты, отходы производства с точки зрения их воздействия на работающих и окружающее население; физические, химические и биологические факторы производственной среды и физиологические изменения у работающих под влиянием этих факторов и трудовых процессов. Наряду с практическими мероприятиями по оздоровлению условий труда гигиена труда разрабатывает и научные основы для регламентирования санитарных условий труда на производстве.

Гигиена труда в системе мероприятий по сохранению высокой работоспособности пользователя ПЭВМ включает в себя следующие параметры:

организация рабочего места;

освещенность;

параметры микроклимата;

методы защиты от шума;

Изм.	Лист	№ докум.	Подпись	Дата

5.1.1 Организация рабочего места пользователя ПЭВМ

Рабочее место - это часть пространства, в котором работник осуществляет трудовую деятельность, и проводит большую часть рабочего времени. Рабочее место, хорошо приспособленное к трудовой деятельности, правильно и целесообразно организованное, в отношении пространства, формы, размера обеспечивает удобное положение при работе и высокую производительность труда при наименьшем физическом и психическом напряжении.

При правильной организации рабочего места производительность труда пользователя возрастает с 8 до 20 процентов.

Согласно ГОСТ 12.2.032-78 конструкция рабочего места и взаимное расположение всех его элементов должно соответствовать антропометрическим, физическим и психологическим требованиям. Большое значение имеет также характер работы. В частности, при организации рабочего места должны быть соблюдены следующие основные условия:

оптимальное размещение оборудования, входящего в состав рабочего места;

достаточное рабочее пространство, позволяющее осуществлять все необходимые движения и перемещения;

необходимо естественное и искусственное освещение для выполнения поставленных задач;

уровень акустического шума не должен превышать допустимого значения.

Главными элементами рабочего места пользователя ПЭВМ являются письменный стол и кресло. Основным рабочим положением является

положение сидя. Рабочее место для выполнения работ в положении сидя организуется в соответствии с ГОСТ 12.2.032-78.

Рабочая поза сидя вызывает минимальное утомление. Рациональная планировка рабочего места предусматривает четкий порядок и постоянство размещения предметов, средств труда и документации. То, что требуется для выполнения работ чаще, расположено в зоне легкой досягаемости рабочего пространства.

Моторное поле - пространство рабочего места, в котором могут осуществляться двигательные действия человека.

Максимальная зона досягаемости рук - это часть моторного поля рабочего места, ограниченного дугами, описываемыми максимально вытянутыми руками при движении их в плечевом суставе.

Оптимальная зона - часть моторного поля рабочего места, ограниченного дугами, описываемыми предплечьями при движении в локтевых суставах с опорой в точке локтя и с относительно неподвижным плечом.

При проектировании письменного стола следует учитывать следующее:

высота стола должна быть выбрана с учетом возможности сидеть свободно, в удобной позе, при необходимости опираясь на подлокотники;

нижняя часть стола должна быть сконструирована так, чтобы пользователь мог удобно сидеть, не был вынужден поджимать ноги;

поверхность стола должна обладать свойствами, исключающими появление бликов в поле зрения работника;

конструкция стола должна предусматривать наличие выдвижных ящиков (не менее 3 для хранения документации, листингов, канцелярских принадлежностей, личных вещей).

Параметры рабочего места выбираются в соответствии с антропометрическими характеристиками. При использовании этих данных в расчетах следует исходить из максимальных антропометрических характеристик (М+2).

При работе в положении сидя рекомендуются следующие параметры рабочего пространства:

- ширина не менее 700 мм;
- глубина не менее 400 мм;
- высота рабочей поверхности стола над полом 700-750 мм.

Оптимальными размерами стола являются:

- высота 710 мм;
- длина стола 1300 мм;
- ширина стола 650 мм.

Поверхность для письма должна иметь не менее 40 мм в глубину и не менее 600 мм в ширину.

Под рабочей поверхностью должно быть предусмотрено пространство для ног:

- высота не менее 600 мм;
- ширина не менее 500 мм;
- глубина не менее 400 мм.

Важным элементом рабочего места является кресло. Оно выполняется в соответствии с ГОСТ 21.889-76. При проектировании кресла исходят из того, что при любом рабочем положении пользователя, его поза должна быть физиологически правильно обоснованной, т.е. положение частей тела должно быть оптимальным. Для удовлетворения требований физиологии, вытекающих из анализа положения тела человека в положении сидя, конструкция рабочего сидения должна удовлетворять следующим основным требованиям:

- допускать возможность изменения положения тела, т.е. обеспечивать свободное перемещение корпуса и конечностей тела друг относительно друга;
- допускать регулирование высоты в зависимости от роста работающего человека (в пределах от 400 до 550 мм);
- иметь слегка вогнутую поверхность,
- иметь небольшой наклон назад.

Исходя из вышесказанного, приведем параметры рабочего стола:

- высота стола 710 мм;
- длина стола 1300 мм;
- ширина стола 650 мм;
- глубина стола 400 мм.

Поверхность для письма:

- в глубину 40 мм;
- в ширину 600 мм.

Создание благоприятных условий труда и правильное эстетическое оформление рабочих мест на производстве имеет большое значение как для

облегчения труда, так и для повышения его привлекательности, положительно влияющей на производительность труда. Окраска помещений и мебели должна способствовать созданию благоприятных условий для зрительного восприятия, хорошего настроения. В служебных помещениях, в которых выполняется однообразная умственная работа, требующая значительного нервного напряжения и большого сосредоточения, окраска должна быть спокойных тонов - малонасыщенные оттенки холодного зеленого или голубого цветов.

При разработке оптимальных условий труда необходимо учитывать освещенность, шум и микроклимат.

5.1.2 Освещенность рабочего места

Освещение на рабочем месте пользователя ПЭВМ должно быть таким, чтобы работник мог без напряжения зрения выполнять свою работу. Утомляемость органов зрения зависит от ряда причин:

- недостаточность освещенности;
- чрезмерная освещенность;
- неправильное направление света.

Недостаточность освещения приводит к напряжению зрения, ослабляет внимание, приводит к наступлению преждевременной утомленности. Чрезмерно яркое освещение вызывает ослепление, раздражение и резь в глазах. Неправильное направление света на рабочем месте может создавать резкие тени, блики, дезориентировать работающего. Все эти причины могут привести к несчастному случаю или профзаболеваниям, поэтому столь важен правильный расчет освещенности.

Расчет освещенности рабочего места сводится к выбору системы освещения, определению необходимого числа светильников, их типа и

размещения. Процесс работы пользователя в таких условиях, когда естественное освещение недостаточно или отсутствует. Искусственное освещение выполняется посредством электрических источников света двух видов: ламп накаливания и люминесцентных ламп. Будем использовать люминесцентные лампы, которые по сравнению с лампами накаливания имеют существенные преимущества:

- по спектральному составу света они близки к дневному, естественному освещению;
- обладают более высоким КПД (в 1.5-2 раза выше, чем КПД ламп накаливания);
- обладают повышенной светоотдачей (в 3-4 раза выше, чем у ламп накаливания);
- более длительный срок службы.

5.1.3 Параметры микроклимата на рабочем месте

Микроклимат рабочего помещения - метеорологические условия внутренней среды помещений, которые определяются действующими на организм человека сочетаниями температуры, влажности, скорости движения воздуха и теплового излучения; комплекс физических факторов, оказывающих влияние на теплообмен человека с окружающей средой, на тепловое состояние человека и определяющих самочувствие, работоспособность, здоровье и производительность труда. Показатели микроклимата: температура воздуха и его относительная влажность, скорость его движения.

Параметры микроклимата могут меняться в широких пределах, в то время как необходимым условием жизнедеятельности человека является поддержание постоянства температуры тела благодаря свойству

терморегуляции, т.е. способности организма регулировать отдачу тепла в окружающую среду.

Основной принцип нормирования микроклимата - создание оптимальных условий для теплообмена тела человека с окружающей средой. В санитарно-эпидемиологических правилах и нормативах СанПиН 2.2.2/2.4.1340-03 установлены величины параметров микроклимата, создающие комфортные условия. Эти нормы устанавливаются в зависимости от времени года, характера трудового процесса и характера производственного помещения (значительные или незначительные тепловыделения). Для рабочих помещений с избыточным тепловыделением до 20 ккал/м³ допустимые и оптимальные значения параметров микроклимата приведены в таблице:

Таблица 5.1 - допустимые значения микроклимата

Время года	Зона	Температура воздуха, °С	Относительная влажность, %	Скорость движения воздуха, м/с
Холодный период	Оптимальная	18 - 21	60 - 40	< 0.2
Переходный период	Допустимая	17 - 21	< 75	< 0.3
Теплый период года (t > 10°С)	Оптимальная	20 - 25	60 - 40	< 0.3
	Допустимая	< 28	< 75	< 0.5

В настоящее время для обеспечения комфортных условий используются как организационные методы, так и технические средства. К числу организационных относятся рациональная организация проведения работ в зависимости от времени года и суток, а также организация правильного чередования труда и отдыха. В связи с этим рекомендуется на территории предприятия организовывать зеленую зону со скамейками для отдыха и водоемом (бассейны, фонтаны). Технические средства включают вентиляцию, кондиционирование воздуха, отопительную систему.

5.1.4 Методы защиты от шума

На рабочем месте источниками шума, как правило, являются технические средства, как то - компьютер, принтер, вентиляционное оборудование, а также внешний шум. Они издают довольно незначительный шум, поэтому в помещении достаточно использовать звукопоглощение. Уменьшение шума, проникающего в помещение извне, достигается уплотнением по периметру притворов окон и дверей. Под звукопоглощением понимают свойство акустически обработанных поверхностей уменьшать интенсивность отраженных ими волн за счет преобразования звуковой энергии в тепловую. Звукопоглощение является достаточно эффективным мероприятием по уменьшению шума. Наиболее выраженными звукопоглощающими свойствами обладают волокнисто-пористые материалы: фибролитовые плиты, стекловолокно, минеральная вата, полиуретановый поропласт, пористый поливинилхлорид и др. К звукопоглощающим материалам относятся лишь те, коэффициент звукопоглощения которых не ниже 0.2.

Звукопоглощающие облицовки из указанных материалов (например, маты из супертонкого стекловолокна с оболочкой из стеклоткани нужно разместить на потолке и верхних частях стен). Максимальное

звукопоглощение будет достигнуто при облицовке не менее 60% общей площади ограждающих поверхностей помещения.

5.2 Вывод

В данной главе были изложены требования к гигиене труда в системе мероприятий по сохранению высокой трудоспособности пользователя ПЭВМ. На основании нормативных документов были указаны требования к организации рабочего места, требования к освещению, параметры микроклимата на рабочем месте, а также методы защиты от шума в рабочем помещении

Заклучение

В ходе выполнения дипломного проекта была разработана мобильное приложение для настройки сетевого оборудования Mikrotik, было сделано:

1. Анализ предметной области;
2. Рассмотрены технологии систем различных производителей;
3. Рассмотрены существующие операционные системы и был сделан анализ, в котором были выявлены достоинства и недостатки;
4. Разработан алгоритм работы и был сделан выбор программного обеспечения для разработки приложения;
5. Разработана база данных, позволяющая хранить в себе массивы данных;
6. Разработано программное обеспечение для оборудования Mikrotik.

Разработанная программа может быть внедрена для тестирования оборудования и рутинной настройки его в рамках локальной сети.

Список используемых источников

1. mikrotik.com официальный сайт MikroTik
2. Григорьев В.М Книга - Виртуальная лаборатория по компьютерным сетям М: Букинист, 2018 г.
3. Мальков И.В. Начальная настройка Mikrotik Router OS. Установка с CD, Сетевая установка СПб: Книжный двор, 2017 г.
4. Арно И.В., Крени С.В. Начальная настройка Mikrotik Router OS. Подключение к провайдеру, различные схемы подключения. М: Библиоглобус, 2015 г.
5. Владленов Г.В. Начальная настройка Mikrotik Router OS. Раздача интернета пользователям, различные схемы раздачи. М: Литрес, 2015 г.
6. Началов С.И., Смирнова В.Ю. Начальная настройка Mikrotik Router OS. Пробрасывание портов через Mikrotik (Port-Mapping), Новосибирск, Велес, 2012 г.

Приложение А
(обязательное)
ЛИСТИНГ

					ПГУ 1.090501.07.001 ПЗ	Лист
						112
Изм.	Лист	№ докум.	Подпись	Дата		

APiConnection.java

```
import java.util.Map;
import javax.net.SocketFactory;
public abstract class ApiConnection implements AutoCloseable {
    /**
     * default TCP port used by Mikrotik API
     */
    public static final int DEFAULT_PORT = 8728;
    /**
     * default TCP TLS port used by Mikrotik API
     */
    public static final int DEFAULT_TLS_PORT = 8729;
    /**
     * default connection timeout to use when opening the connection
     */
    public static final int DEFAULT_CONNECTION_TIMEOUT = 60000;
    /**
     * default command timeout used for synchronous commands
     */
    public static final int DEFAULT_COMMAND_TIMEOUT = 60000;
    public static ApiConnection connect(SocketFactory fact, String host, int port, int
timeout) throws MikrotikApiException {
        return ApiConnectionImpl.connect(fact, host, port, timeout);
    }
    public static ApiConnection connect(String host) throws MikrotikApiException {
        return connect(SocketFactory.getDefault(), host, DEFAULT_PORT,
DEFAULT_COMMAND_TIMEOUT);
    }
    public abstract boolean isConnected();
    /**
     *
     */
    public abstract void login(String username, String password) throws MikrotikApiException;
    /**
     * execute a command and return a list of results.
     *
     */
    public abstract List<Map<String, String>> execute(String cmd) throws MikrotikApiException;
    public abstract String execute(String cmd, ResultListener lis) throws
MikrotikApiException;
    public abstract void cancel(String tag) throws MikrotikApiException;
    public abstract void setTimeout(int timeout) throws MikrotikApiException;
    public abstract void close() throws ApiConnectionException;
}
```

XMLElements.java

```
import java.io.InputStream;
import java.io.OutputStream;
import java.io.UnsupportedEncodingException;
import java.nio.charset.Charset;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.List;
import me.legrange.mikrotik.ApiConnectionException;

final class XMLElements {
    static void write(Command cmd, OutputStream out) throws UnsupportedEncodingException,
IOException {
        encode(cmd.getCommand(), out);
        for (Parameter param : cmd.getParameters()) {
            encode(String.format("=%s=%s", param.getName(), param.hasValue() ?
param.getValue() : ""), out);
        }
        String tag = cmd.getTag();
        if ((tag != null) && !tag.equals("")) {
            encode(String.format(".tag=%s", tag), out);
        }
        List<String> props = cmd.getProperties();
        if (!props.isEmpty()) {
            StringBuilder buf = new StringBuilder("=.proplist=");
            for (int i = 0; i < props.size(); ++i) {
                if (i > 0) {
                    buf.append(",");
                }
                buf.append(props.get(i));
            }
            encode(buf.toString(), out);
        }
        for (String query : cmd.getQueries()) {
            encode(query, out);
        }
        out.write(0);
    }

    static String decode(InputStream in) throws ApiDataException, ApiConnectionException {
        StringBuilder res = new StringBuilder();
        decode(in, res);
        return res.toString();
    }

    private static void decode(InputStream in, StringBuilder result) throws ApiDataException,
```

```

ApiConnectionException {
    try {
        int len = readLen(in);
        if (len > 0) {
            byte buf[] = new byte[len];
            for (int i = 0; i < len; ++i) {
                int c = in.read();
                if (c < 0) {
                    throw new ApiDataException("Truncated data. Expected to read more
bytes");
                }
                buf[i] = (byte) (c & 0xFF);
            }
            String res = new String(buf, Charset.forName("UTF-8"));
            if (result.length() > 0) {
                result.append("\n");
            }
            result.append(res);
            decode(in, result);
        }
    } catch (IOException ex) {
        throw new ApiConnectionException(ex.getMessage(), ex);
    }
}

static String hashMD5(String s) throws ApiDataException {
    MessageDigest algorithm = null;
    try {
        algorithm = MessageDigest.getInstance("MD5");
    } catch (NoSuchAlgorithmException nsae) {
        throw new ApiDataException("Cannot find MD5 digest algorithm");
    }
    byte[] defaultBytes = new byte[s.length()];
    for (int i = 0; i < s.length(); i++) {
        defaultBytes[i] = (byte) (0xFF & s.charAt(i));
    }
    algorithm.reset();
    algorithm.update(defaultBytes);
    byte messageDigest[] = algorithm.digest();
    StringBuilder hexString = new StringBuilder();
    for (int i = 0; i < messageDigest.length; i++) {
        String hex = Integer.toHexString(0xFF & messageDigest[i]);
        if (hex.length() == 1) {
            hexString.append('0');
        }
        hexString.append(hex);
    }
}

```

```

    }
    return hexString.toString();
}
static String hexStrToStr(String s) {
    StringBuilder ret = new StringBuilder();
    for (int i = 0; i < s.length(); i += 2) {
        ret.append((char) Integer.parseInt(s.substring(i, i + 2), 16));
    }
    return ret.toString();
}
private static void encode(String word, OutputStream out) throws
UnsupportedEncodingException, IOException {
    byte bytes[] = word.getBytes("UTF-8");
    int len = bytes.length;
    if (len < 0x80) {
        out.write(len);
    } else if (len < 0x4000) {
        len = len | 0x8000;
        out.write(len >> 8);
        out.write(len);
    } else if (len < 0x20000) {
        len = len | 0xC00000;
        out.write(len >> 16);
        out.write(len >> 8);
        out.write(len);
    } else if (len < 0x10000000) {
        len = len | 0xE0000000;
        out.write(len >> 24);
        out.write(len >> 16);
        out.write(len >> 8);
        out.write(len);
    } else {
        out.write(0xF0);
        out.write(len >> 24);
        out.write(len >> 16);
        out.write(len >> 8);
        out.write(len);
    }
    out.write(bytes);
}
private static int readLen(InputStream in) throws IOException {
    int c = in.read();
    if (c > 0) {
        if ((c & 0x80) == 0) {
        } else if ((c & 0xC0) == 0x80) {

```

```

        c = c & ~0xC0;
        c = (c << 8) | in.read();
    } else if ((c & 0xE0) == 0xC0) {
        c = c & ~0xE0;
        c = (c << 8) | in.read();
        c = (c << 8) | in.read();
    } else if ((c & 0xF0) == 0xE0) {
        c = c & ~0xF0;
        c = (c << 8) | in.read();
        c = (c << 8) | in.read();
        c = (c << 8) | in.read();
    } else if ((c & 0xF8) == 0xF0) {
        c = in.read();
        c = (c << 8) | in.read();
        c = (c << 8) | in.read();
        c = (c << 8) | in.read();
        c = (c << 8) | in.read();
    }
}
return c;
}
}

```

Parser.java

```

import java.util.LinkedList;
import java.util.List;
import me.legrange.mikrotik.impl.Scanner.Token;
class Parser {
    static Command parse(String text) throws ParseException {
        Parser parser = new Parser(text);
        return parser.parse();
    }
    private Command parse() throws ParseException {
        command();
        while (!is(Token.WHERE, Token.RETURN, Token.EOL)) {
            param();
        }
        if (token == Token.WHERE) {
            where();
        }
        if (token == Token.RETURN) {
            returns();
        }
        expect(Token.EOL);
    }
}

```

```

        return cmd;
    }
    private void command() throws ParseException {
        StringBuilder path = new StringBuilder();
        do {
            expect(Token.SLASH);
            path.append("/");
            next();
            expect(Token.TEXT);
            path.append(text);
            next();
        } while (token == Token.SLASH);
        cmd = new Command(path.toString());
    }
    private void param() throws ParseException {
        String name = text;
        next();
        if (token == Token.EQUALS) {
            next();
            StringBuilder val = new StringBuilder();
            if (token == Token.PIPE) { // handle cases like hotspot=!auth
                val.append(token);
                next();
            }
            expect(Token.TEXT);
            val.append(text);
            next();
            while (is(Token.COMMA, Token.SLASH)) {
                val.append(token);
                next();
                expect(Token.TEXT);
                val.append(text);
                next();
            }
            cmd.addParameter(new Parameter(name, val.toString()));
        } else {
            cmd.addParameter(new Parameter(name));
        }
    }
    private void where() throws ParseException {
        next(); // swallow the word "where"
        expr();
    }
    private void expr() throws ParseException {
        expect(Token.NOT, Token.TEXT, Token.LEFT_BRACKET);
    }

```

Изм.	Лист	№ докум.	Подпись	Дата

```

switch (token) {
    case NOT:
        notExpr();
        break;
    case TEXT: {
        String name = text;
        next();
        expect(Token.EQUALS, Token.LESS, Token.MORE, Token.NOT_EQUALS);
        switch (token) {
            case EQUALS:
                eqExpr(name);
                break;
            case NOT_EQUALS:
                notExpr(name);
                break;
            case LESS:
                lessExpr(name);
                break;
            case MORE:
                moreExpr(name);
                break;
            default:
                hasExpr(name);
        }
    }
    break;
    case LEFT_BRACKET:
        nestedExpr();
        break;
}

// if you get here, you had a expression, see if you want more.
switch (token) {
    case AND:
        andExpr();
        break;
    case OR:
        orExpr();
        break;
}
}

private void nestedExpr() throws ParseException {
    expect(Token.LEFT_BRACKET);
    next();
    expr();
    expect(Token.RIGHT_BRACKET);
}

```

Изм.	Лист	№ докум.	Подпись	Дата

```

        next();
    }
    private void andExpr() throws ParseException {
        next(); // eat and
        expr();
        cmd.addQuery("?#&");
    }
    private void orExpr() throws ParseException {
        next(); // eat or
        expr();
        cmd.addQuery("?#|");
    }
    private void notExpr() throws ParseException {
        next(); // eat not
        expr();
        cmd.addQuery("?#!");
    }
    private void eqExpr(String name) throws ParseException {
        next(); // eat =
        expect(Token.TEXT);
        cmd.addQuery(String.format("'%s'=%s", name, text));
        next();
    }
    private void lessExpr(String name) throws ScanException {
        next(); // eat <
        cmd.addQuery(String.format("'%s'<%s", name, text));
        next();
    }
    private void notExpr(String name) throws ScanException {
        next(); // eat !=
        cmd.addQuery(String.format("'%s'!=%s", name, text));
        cmd.addQuery("?#!");
        next();
    }
    private void moreExpr(String name) throws ScanException {
        next(); // eat >
        cmd.addQuery(String.format("'%s'>%s", name, text));
        next();
    }
    private void hasExpr(String name) {
        cmd.addQuery(String.format("'%s'", name));
    }
    private void returns() throws ParseException {
        next();
        expect(Token.TEXT);
    }

```

Изм.	Лист	№ докум.	Подпись	Дата

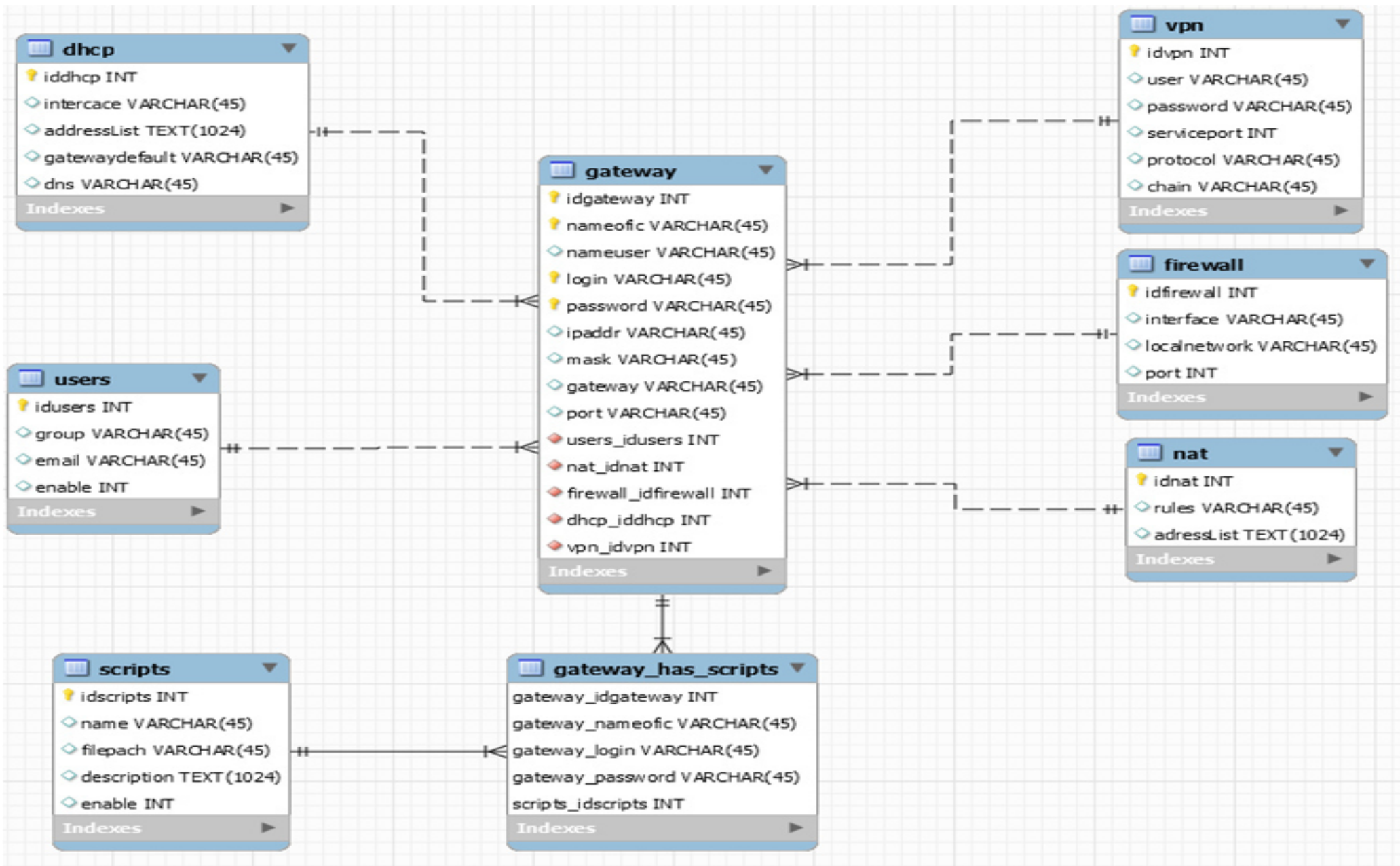
```

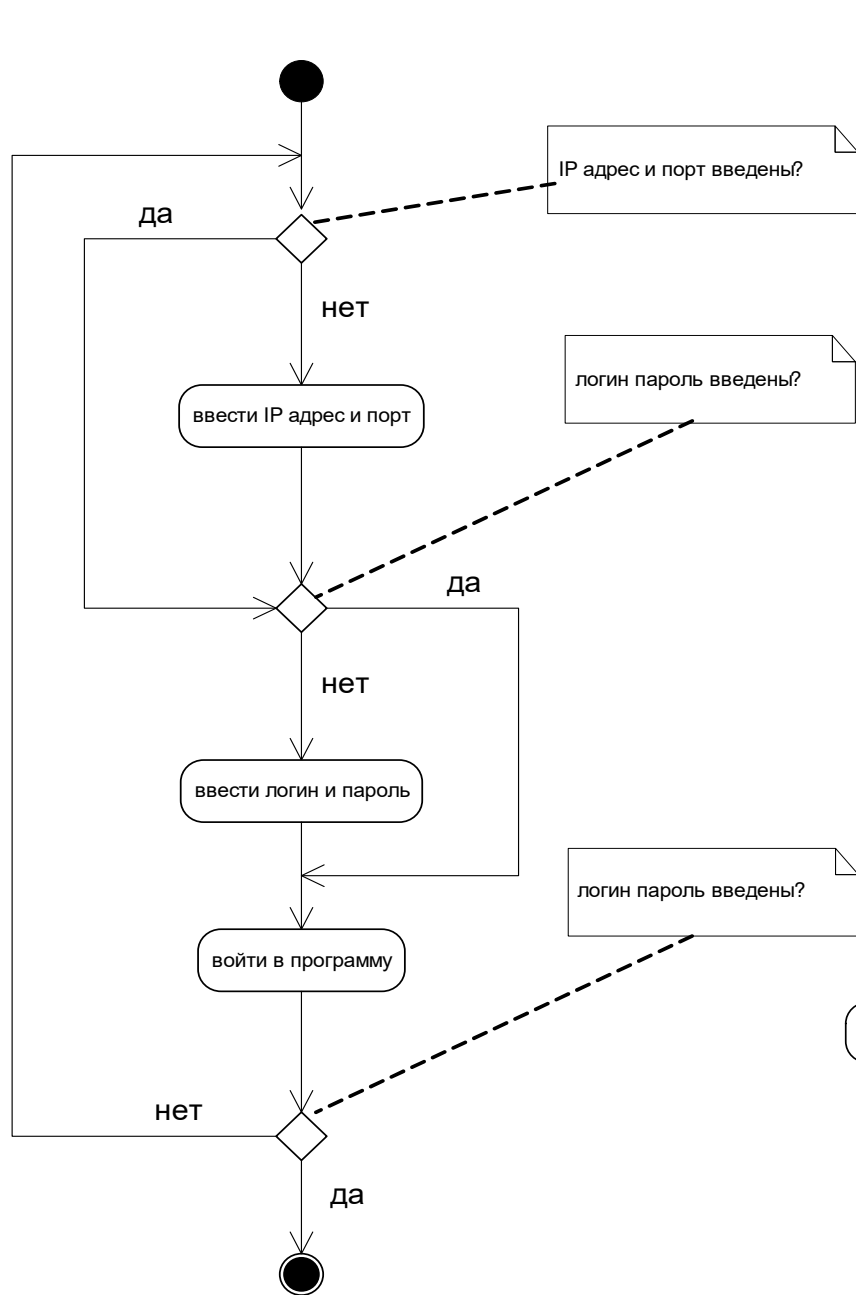
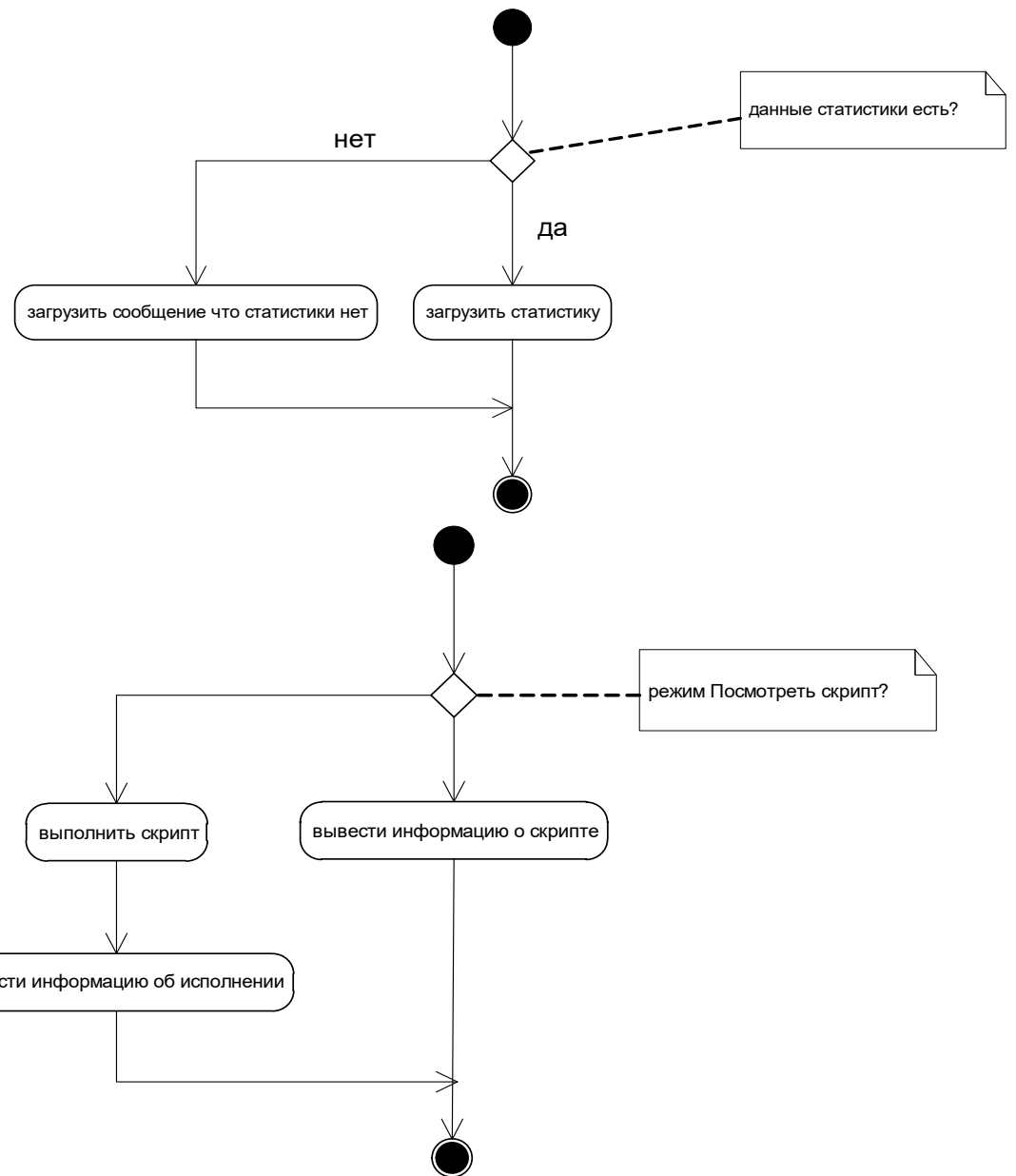
        List<String> props = new LinkedList<>();
        while (!(token == Token.EOL)) {
            if (token != Token.COMMA) {
                props.add(text);
            }
            next();
        }
        cmd.addProperty(props.toArray(new String[props.size()]));
    }
    private void expect(Token... tokens) throws ParseException {
        if (!is(tokens))
            throw new ParseException(String.format("Expected %s but found %s at position %d",
Arrays.asList(tokens), this.token, scanner.pos()));
    }
    private boolean is(Token... tokens) {
        for (Token want : tokens) {
            if (this.token == want) return true;
        }
        return false;
    }
    private void next() throws ScanException {
        token = scanner.next();
        while (token == Token.WS) {
            token = scanner.next();
        }
        text = scanner.text();
    }
    private Parser(String line) throws ScanException {
        line = line.trim();
        scanner = new Scanner(line);
        next();
    }
    private final Scanner scanner;
    private Token token;
    private String text;
    private Command cmd;
}

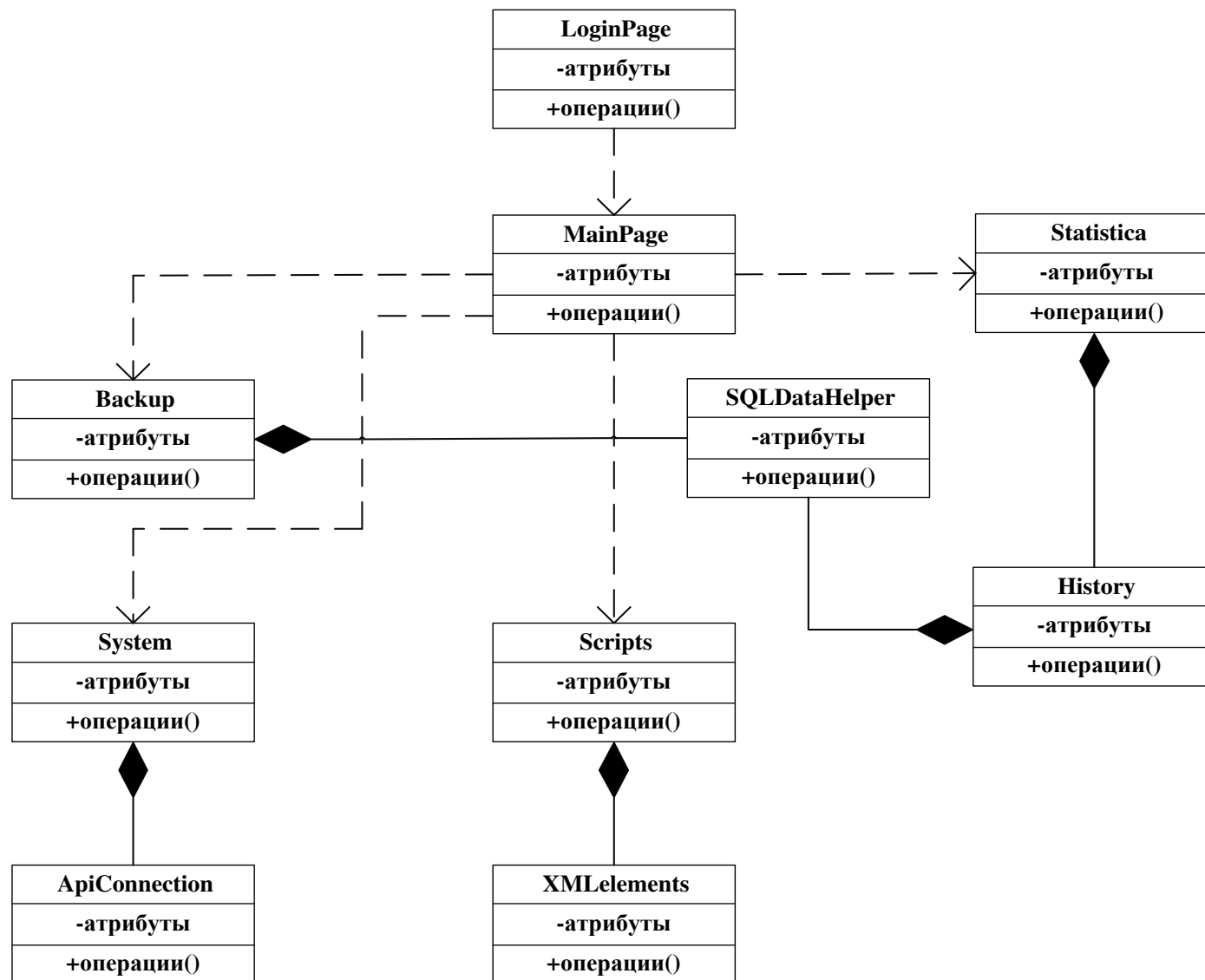
```

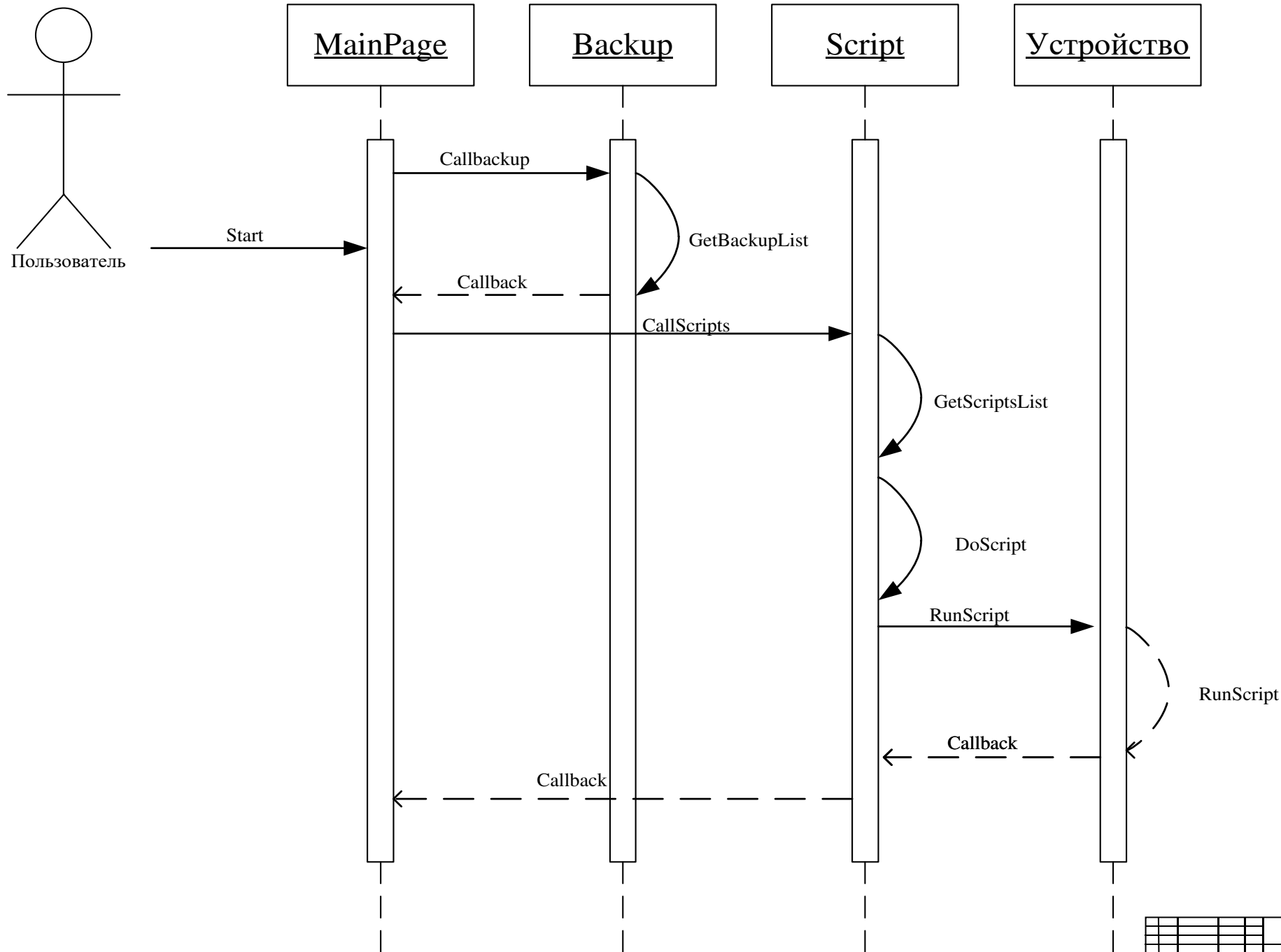
Приложение В
(Обязательное)
Схемы

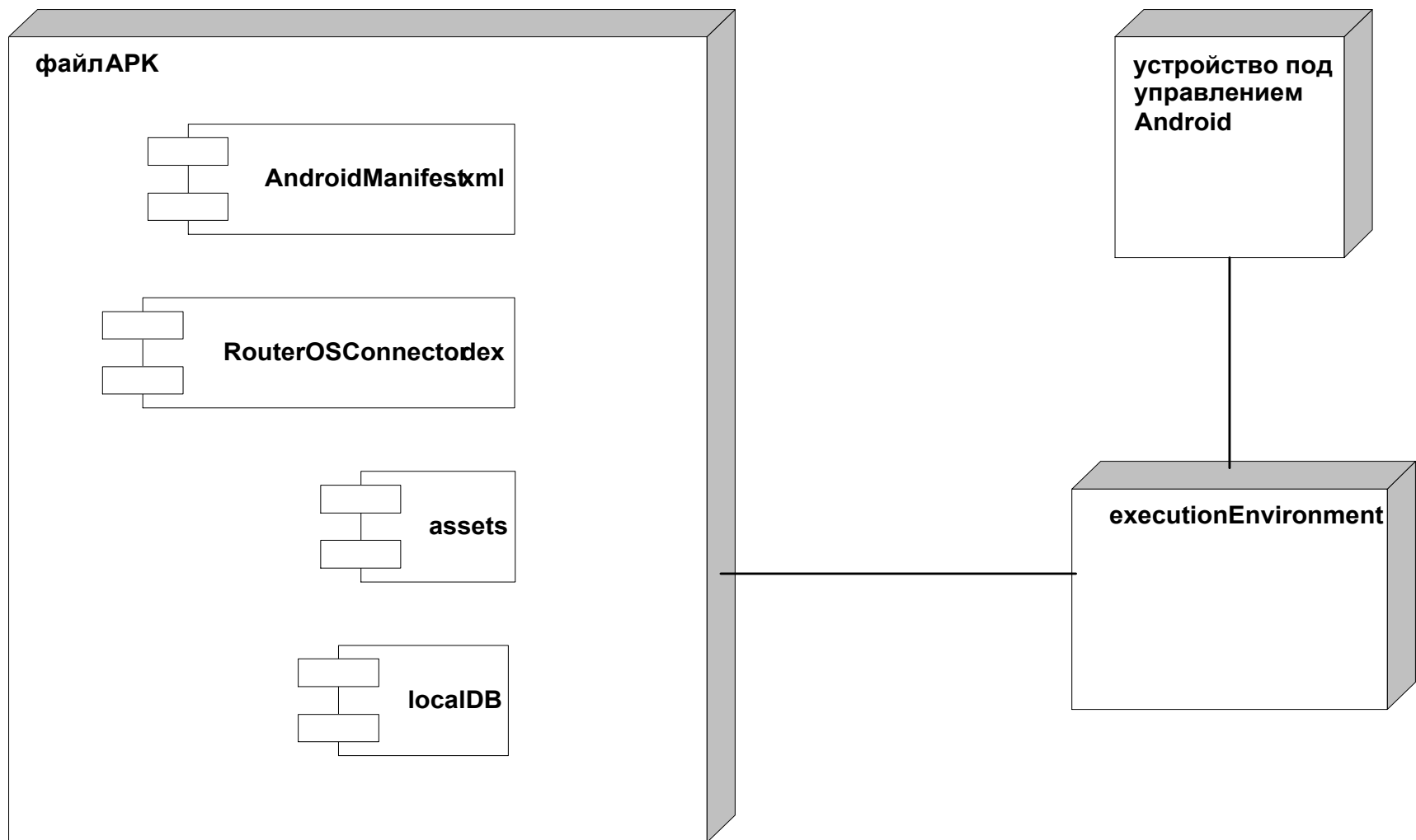
					ПГУ 1.090501.07.001 ПЗ	Лист
						122
Изм.	Лист	№ докум.	Подпись	Дата		











						ПГУ1.090501.07.002 Д05									
Изм.	Лист	№ докум.			Полном.	Дата	Структурная схема Графическая часть				Лист	Масш.	Масш.		
Разраб.		Копылов А.С.													
Проект.		Федосин Р.И.													
Н. контр.		Кузин А.В.													
Усть.		Ситников М.А.									ФВТ Каф. Вт. Гр. 15В02				

Приложение С
(обязательное)
Презентация

					ПГУ 1.090501.07.001 ПЗ	Лист
						128
Изм.	Лист	№ докум.	Подпись	Дата		

ТЕМА РАБОТЫ:

Программное обеспечение технической поддержки коммутационного оборудования

Автор работы: Студент группы 15ВО2 Кошляков А.С.

Руководитель ВКР: к.т.н., доцент каф. ВТ, Федюнин Р. Н.

2020

ЦЕЛИ И ЗАДАЧИ

- Цель работы – разработать мобильное приложение для рутинной настройки сетевого оборудования на базе ОС RouterOS.
- Задачи:
 - Изучить предметную область,
 - Выбрать средства и технологии для разработки,
 - Разработать мобильное приложение,
 - Проверить работоспособность приложения.

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.07.001 ПЗ

Лист

129

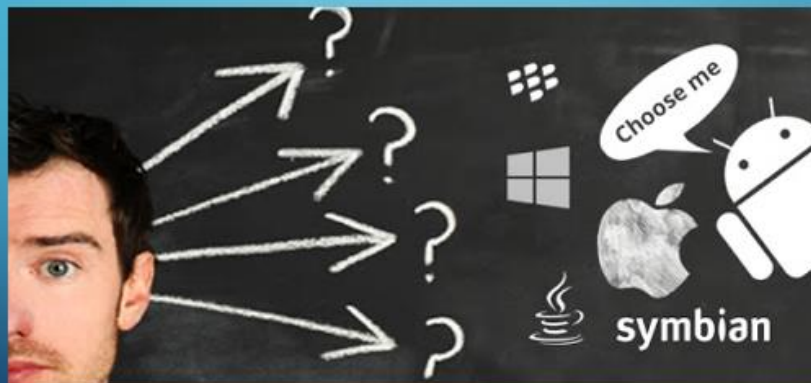
ОСНОВНЫЕ ТРЕБОВАНИЯ ДЛЯ МОБИЛЬНОГО ПРИЛОЖЕНИЯ

- Мониторинг состояния сетевого оборудования
- Ограничение скорости
- Блокировка сайтов
- Создание резервной копии

ВЫБОР ТЕХНОЛОГИЧЕСКОЙ ПЛАТФОРМЫ

Функционал	<u>RouterOS</u>	<u>DD-Wrt</u>	<u>OpenWrt</u>	LEDE	<u>VyOS</u>
Поддержка API	+	-	-	-	-
Блокировка сайтов	+	-	+	+	-
Мониторинг оборудования	+	-	-	-	+
Создание резервных копий	+	+	+	-	+
Ограничение скорости	+	+	-	+	-

IOS ИЛИ ANDROID?



ВЫБОР СУБД

- MySQL - самая популярная и распространённая СУБД;
- PostgreSQL - наиболее продвинутая СУБД;
- SQLite - очень мощная встраиваемая система управления.

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.07.001 ПЗ

Лист

131

ДИАГРАММА КЛАССОВ

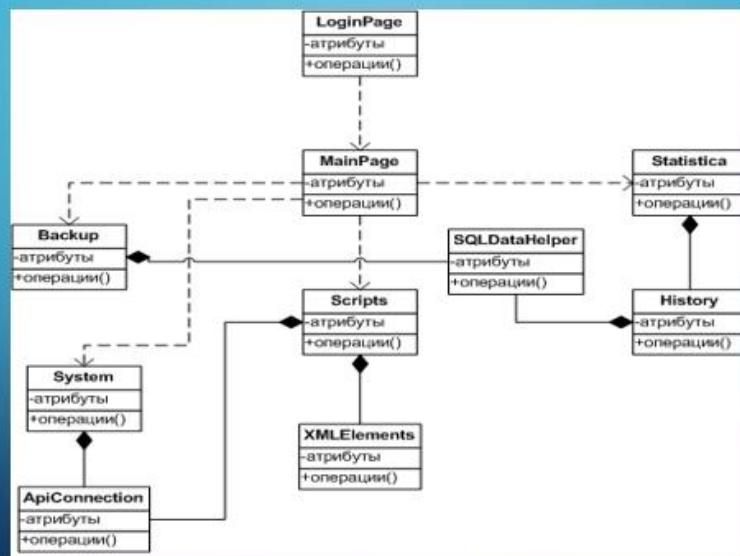
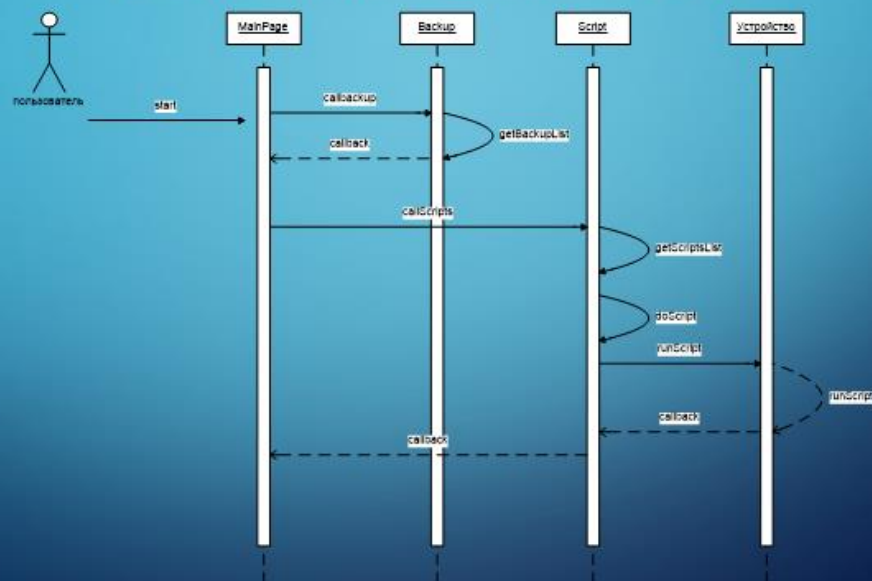
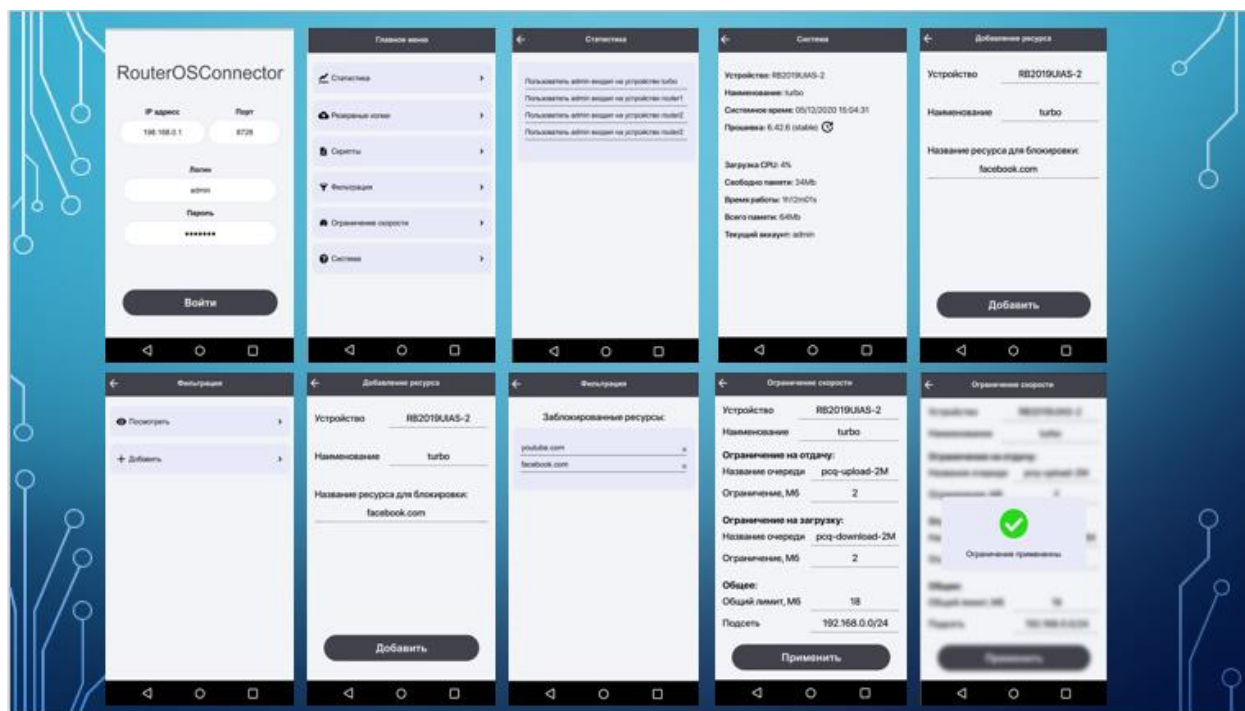


ДИАГРАММА ПОСЛЕДОВАТЕЛЬНОСТИ





ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта было разработано мобильное приложение для первичной настройки сетевого оборудования Microtik. В ходе работы выполнены следующие шаги:

- Анализ предметной области;
- Выбран стек технологий для реализации проекта;
- Изучена настройка коммутационного оборудования Microtik;
- Изучен и протестирован скриптовый язык работы с оборудованием Microtik;
- Реализовано мобильное приложение для настройки оборудования Microtik.

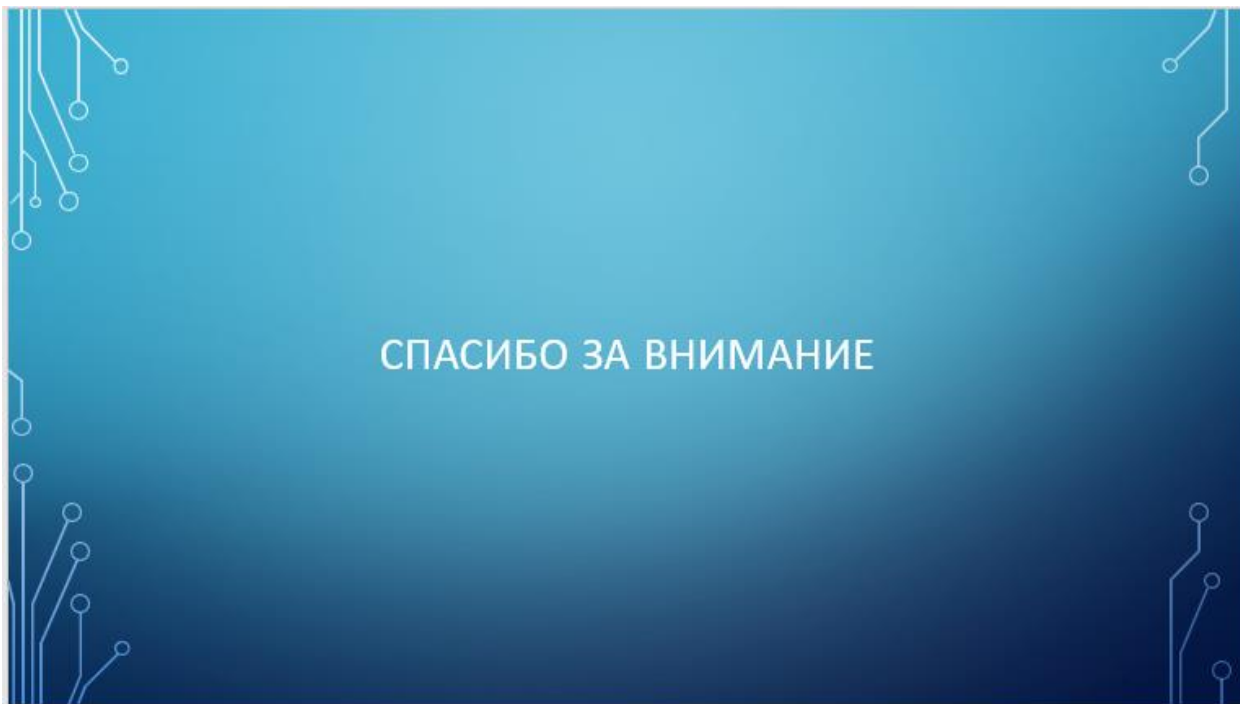
Данное программное обеспечение может быть использовано для тестирования и настройки коммутационного оборудования в рамках локальной сети.

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090501.07.001 ПЗ

Лист

133



					ПГУ 1.090501.07.001 ПЗ	Лист
						134
Изм.	Лист	№ докум.	Подпись	Дата		