Федеральное агентство по рыболовству

ФГБОУ ВО «Калининградский государственный технический университет»

Балтийская государственная академия рыбопромыслового флота

Кафедра информационной безопасности

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

на тему:

«Разработка средств резервирования файлов с обеспечением защиты на основе операционной системы серверного кластера Инспекции ФНС №8 по г. Калининграду»

Калининград – 2017

Федеральное агентство по рыболовству

ФГБОУ ВО «Калининградский государственный технический университет»

Балтийская государственная академия рыбопромыслового флота

Радиотехнический факультет

Кафедра информационной безопасности

Допустить к защите

И.О. декана \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Жестовский А.Г./\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (фамилия, инициалы) (дата)

Заведующий кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Великите Н.Я./\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (фамилия, инициалы) (дата)

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

по \_10.05.03 Информационная безопасность автоматизированных систем\_

код и наименование специальности (направления подготовки)

на тему: Разработка средств резервирования файлов с обеспечением защиты на основе операционной системы серверного кластера Инспекции ФНС №8 по г. Калининграду

Пояснительная записка

(ДР. БГАРФ.10.05.03.ИБ51.17 ПЗ)

|  |  |  |  |
| --- | --- | --- | --- |
| Разработал студент гр. ИБ-51 | \_\_\_\_\_\_\_\_\_\_\_ | /Кутняк Д.А./ | .06.2017 |
| Руководитель доцент, к.т.н. | \_\_\_\_\_\_\_\_\_\_\_ | /Зензин А.С./ | .06.2017 |
| Нормоконтролер старший преподаватель | \_\_\_\_\_\_\_\_\_\_\_  (подпись) | /Подтопельный В.В./  (фамилия, инициалы) | .06.2017  (дата) |

Калининград – 2017

# Реферат

Кутняк Д.А. Разработка средств резервирования файлов с обеспечением защиты на основе операционной системы серверного кластера Инспекции ФНС №8 по г. Калининграду: ДР.БГАРФ.10.05.03.ИБ51.17 ПЗ: ВКР/БГАРФ, каф. ИБ; рук. А.С. Зензин. – Калининград, 2017. – ПЗ 96 с., 25 рис., 3 табл., 13 источников, 3 прил.

СИСТЕМЫ РЕЗЕРВНОГО КОПИРОВАНИЯ, ВОССТАНОВЛЕНИЕ ДАННЫХ, КЛИЕНТ-СЕРВЕРНАЯ АРХИТЕКТУРА, ТОПОЛОГИЯ РЕЗЕРВНОГО КОПИРОВАНИЯ, PYQT, MICROSOFT SQL SERVER, СИСТЕМА КОНТРОЛЯ РЕЗЕРВНОГО КОПИРОВАНИЯ.

Объект исследования – современные средства резервного копирования данных в ИС.

Цель ВКР – Разработать систему контроля резервного копирования данных в информационной системе Инспекции ФНС №8 по г. Калининграду, как средство улучшения информационной безопасности данной ИС и повышения удобства эксплуатации.

В ходе написания выпускной квалификационной работы были изучены и проанализированы различные системы резервного копирования данных, выявлены особенности их работы, а также рассмотрены положительные и отрицательные стороны различных систем.

Разработана система контроля резервного копирования данных на основе инфраструктуры Инспекции ФНС №8 по г. Калининграду, позволяющая повысить оперативность выявления проблем при резервном копировании данных и удобство аудита работы информационной системы Инспекции ФНС №8 по г. Калининграду.

Кроме того, использование предлагаемой программы позволяет уменьшить стоимость системы контроля резервного копирования данных и повысить эффективность использования рабочего времени сотрудников.

# Содержание

[Введение 8](#_Toc485804946)

[1 Особенности современных подсистем резервного копирования данных 10](#_Toc485804947)

[1.1 Понятие резервного копирования данных 10](#_Toc485804948)

[1.2 Значение резервного копирования данных 10](#_Toc485804949)

[2 Обзор технологий резервного копирования 13](#_Toc485804950)

[2.1 Полное резервное копирование 13](#_Toc485804951)

[2.2 Инкрементное резервное копирование 14](#_Toc485804952)

[2.3 Дифференциальное резервное копирование 15](#_Toc485804953)

[2.4 Технологии хранения резервных копий и данных 16](#_Toc485804954)

[2.5 Хранение резервных копий 22](#_Toc485804955)

[2.6 Восстановление данных из резервных копий 23](#_Toc485804956)

[2.7 Топология резервного копирования 24](#_Toc485804957)

[3 Средства резервного копирования, используемые в Инспекции ФНС №8 по г. Калининграду 30](#_Toc485804958)

[3.1 Kaspersky Security Center. Утилита klbackup. 32](#_Toc485804959)

[3.2 Microsoft SQL Server 34](#_Toc485804960)

[4 Разработка системы контроля резервного копирования данных 38](#_Toc485804961)

[4.1 Изучение и анализ аналогичных систем резервного копирования данных 38](#_Toc485804962)

[4.2 Требования к разрабатываемой системе контроля резервного копирования данных 41](#_Toc485804963)

[4.3 Понятие клиент-серверного приложения 43](#_Toc485804964)

[4.4 Модульная архитектура системы контроля резервного копирования данных в информационной системе Инспекции ФНС №8 по г. Калининграду 47](#_Toc485804965)

[4.5 Описание работы системы контроля резервного копирования данных в информационной системе Инспекции ФНС №8 по г. Калининграду 52](#_Toc485804966)

[5 Испытание системы контроля системы резервного копирования данных 62](#_Toc485804967)

[Заключение 65](#_Toc485804968)

[Cписок использованных источников 67](#_Toc485804969)

[Приложение А 69](#_Toc485804970)

[Приложение Б 75](#_Toc485804971)

[Приложение В 90](#_Toc485804972)

# Список обозначений и сокращений

|  |  |  |  |
| --- | --- | --- | --- |
| SQL | | Structured Query Language | |
| РК | | Резервное копирование | |
| АРМ | | Автоматизированное рабочее место | |
| АСУ | | Автоматизированные системы управления | |
| БД | | База данных | |
| ИКТ | | Информационные и коммуникационные технологии | |
| ИП | | Информационное пространство | |
| ИпС | | Информационная подсистема | |
| ИС | | Информационная система | |
| ИТ | | Информационные технологии | |
| КТС | | Комплекс технических средств | |
| ОС | | Операционная система | |
| СУБД | | Система управления базами данных | |
| ТЗ | | Техническое задание | |
| ФЗ | | Федеральный закон | |
| ПК | | Персональный компьютер | |
| ИБ | | Информационная безопасность | |
| HDD | | Hard Disk Drive, накопитель на жестких магнитных дисках | |
| Backup | | Резервное копирование данных | |
| HotSwap | | Горячая замена носителей информации | |
| RAID | | Redundant Array of Independent Disks, избыточный массив независимых дисков | |
| SCSI | | Small Computer System Interface, стандарт для физического подключения и передачи данных | |
| SSD | | Solid State Drive, твердотельный накопитель | |
| SLC | | Single Level Cells, одноуровневая система ячеек | |
| MLC | | Multi-Level Cells, многоуровневая система ячеек | |
| NAS | | Network Attached Storage, сетевая система хранения данных | |
| SAN | | Storage Area Network, сеть хранения данных | |
| LAN | | Local Area Network, локальная сеть | |
| LUN | | Local Unit Number, адрес дискового устройства хранения | |
| DAS | | Directed Attached Storage, система хранения данных прямого подключения | |
| РКД | | Резервные копии данных | |

# Введение

Резервное копирование - это процесс создания когерентной (непротиворечивой) копии данных. РК становится все актуальнее на фоне значительного увеличения объема данных в компьютерной индустрии. Подсистема резервного копирования - очень важная часть любой ИС. При правильной ее организации она эффективно защищает весь спектр важных данных от утери, а также организует быструю замену одного ПК на другой в случае необходимости, то есть, фактически обеспечить бесперебойную работу офисных сотрудников и как вследствие целой организации.

Для проведения процедуры РК обычно используются специальные программно-аппаратные подсистемы, называемые подсистемами резервного копирования. Они как раз и предназначены как для проведения регулярного автоматического копирования системных и пользовательских данных, так и для оперативного восстановления данных. РК предоставляет возможность эффективного и дешевого способа восстановления информации (документов, программ, настроек и т.д.) в случае утери её утери по какой-либо причине.

Также благодаря подсистеме РК решаются смежные проблемы:

* дублирование данных;
* передача данных и работа с общими документами.

Политика защиты информации подразумевает систему РК и архивирования, как неотъемлемую свою часть. От правильного выбора аппаратно-программных средств РК зависит надежность защиты, восстановления данных, и функционирования организации. Особую роль в защите информации играют автоматизированные системы РК и архивирования информации во всех необходимых серверных системах и ПК защищаемого сегмента сети Инспекции ФНС №8 по городу Калининграду.

Значение использование РК растет с каждым годом. Практически во всех коммерческих и государственных организациях наблюдается тенденция роста заинтересованности к подсистемам РК и восстановления данных. Компании постепенно понимают ценность информации и отдают предпочтение комплексным системам ИБ. В них обеспечение защиты информации на предприятии РК и ВД выполняет одну из основных ролей. Резервное копирование минимизирует убытки из-за простоя серверов, помогая избежать затраты на привлечения сторонних организаций и лиц к восстановлению утерянной информации, а также на содержание избыточного штата специалистов.

При внедрении системы РК, организация сталкивается со сложными задачами оценки ее текущих потребностей, планирования будущих объемов данных, выбора технологий и архитектур, которые максимально соответствуют:

* требованиям ИБ;
* возможности будущей модификации;
* удовлетворению техническим требованиям скорости записи, чтения и восстановления данных.

Выявление оптимального решение сложная задача, в которой учитывается многообразие существующих путей реализации РК. Построение высокоэффективной системы хранения данных, отвечающих реальным требованиям организации, а также выбор максимально подходящей системы резервного копирования – процесс весьма трудоемкий и требующий тщательного анализа.

# 1 Особенности современных подсистем резервного копирования данных

## 

## 1.1 Понятие резервного копирования данных

РК - это процесс создания когерентной (непротиворечивой) копии данных. Основные задачи резервного копирования данных:

* защита всего спектра важных данных от утери;
* организация быстрой миграции с одного ПК на другой в случае необходимости и обеспечение бесперебойной работы организации.

При условии выполнения данных пунктов можно говорить об эффективной работе РК.

Программно-аппаратные подсистемыРКпредназначаются для проведения регулярного автоматического копирования системных и пользовательских данных, для оперативного восстановления данных. [2] Хранение информации отдельно от системных файлов является обязательным правилом:

* в случае классического пользователя это означает, как минимум, разделение HDD на три логических диска: для системы, для приложений, для данных;
* в случае корпоративного сотрудника с большим объемом конфиденциальной информации - размещение информации на других, не системных физических дисках.

Данная мера облегчает и саму операцию архивирования данных. Принцип раздельного хранения информации непосредственно связан и с файловыми архивами, и к образам дисков. Их необходимо также хранить как минимум на несистемных разделах одного HDD. В случае корпоративного пользователя принцип раздельного хранения информации должен реализовываться еще жестче: как минимум одна из копий должна храниться в отдельном месте, чтобы не потерять корпоративную информацию в случае чрезвычайных ситуаций.

## 1.2 Значение резервного копирования данных

По данным исследования[[1]](#footnote-1) 49 процентов опрошенных пользователей ПК теряли важные данные в результате ошибочных действий, аварий или заражения вредоносными программами, поскольку они не использовали подсистемы РК. Риск данных потерь еще выше, как для небольших компаний, так и для крупных организаций. Основная часть пользователей приходит к пониманию важности резервного копирования файлов, когда происходят непредвиденные ситуации, но они не знают, как это делается на практике.

РКД [[2]](#footnote-2)— это полное или частичное сохранение данных, находящихся на одном носителе (жестком диске, компакт-диске, твердотельном накопителе и т. д.), на другой носитель, для создания аварийной копии на случай чрезвычайных ситуаций. Возможно создание дубликата файлов без внешнего устройства хранения информации, например, в другом разделе жесткого диска, но данный вариант не эффективен в случае серьезных неполадок носителей информации. [3]

Данные страдают от внедренных в компьютеры вредоносных программ: вирусов, червей и троянских коней. Такой вредоносный код может вызвать неустранимые повреждения. Некоторые вирусные утилиты, такие как вирус Gigger и червь W32/Vote-B, могут пытаться отформатировать жесткий диск. При условии защиты системы от внешних атак, безопасность системы не может быть гарантирована. По данным исследования, проведенного ассоциацией Clusif[[3]](#footnote-3), 39% случаев потери или искажения данных связаны с человеческими ошибками.

Компьютерные файлы не обладают большой стойкостью. Для их повреждения достаточно:

* неверного перемещения или его отмены до окончания процесса;
* ошибки в программном обеспечении;
* неправильного выключения системы;
* нецеленаправленного удаления файла;
* закрытия программы без сохранения изменений;
* потери пароля и др.

Вероятность оказаться в таких обстоятельствах увеличивается, при использовании ПК для профессиональной деятельности. Но не обязательно хранить контракты, счета или конфиденциальные документы, чтобы переживать о потере данных, так как порча личных данных, таких, например, как курсовые и дипломные работы, электронные письма, фотографии или музыкальная коллекция, могут причинить массу неудобств и проблем. Создание РКД на съемных носителях позволяет взять данные с собой. Необходимо также создавать резервные копии данных при выполнении тонких операций.[[4]](#footnote-4)

При работе в течение одного дня с несколькими файлами, нет необходимости совершать полный «backup» всех данных. «Поэтапное» резервное копирование (измененных данных) или может производиться только для наиболее важных элементов, таких как электронный почтовый ящик, производственные документы, системные параметры, специальные шаблоны или особые словари для приложений.

# 2 Обзор технологий резервного копирования

Частота использования и степень важности хранимой информации определяют вид процедуры резервного копирования данных:

* полное резервное копирование (Full backup);
* дифференциальное резервное копирование (Differential backup);
* инкрементное резервное копирование (Incremental backup).

## 2.1 Полное резервное копирование

Основным способов создания резервных копий, при котором используемый массив данных копируется полностью. Это наиболее целостный и надежный вид РК, хотя и самый ресурсно-затратный. В случае потребности сохранить несколько копий данных общий хранимый объем будет увеличиваться сравнимо их количеству. Полное резервное копирование незаменимо в случае, когда нужно подготовить резервную копию для быстрого восстановления системы из исходного состояния.

К достоинствам метода относят:

* удобный поиск файлов, так как выполняется РКД, содержащихся на устройстве, для поиска необходимого файла отсутствует необходимость просматривать несколько носителей;
* нахождение текущей резервной копии всей системы на одном носителе или наборе носителей.

Недостатки метода:

* избыточная защита данных, так как большинство файлов системы изменяются достаточно редко, то каждая последующая полная РКД представляет собой, сохраненных в ходе первого полного резервного копирования. Для полного резервного копирования требуется большой объём носителя;
* большое время полного резервного копирования, в особенности, если для хранения выбраны устройства в сети. [3]

## 2.2 Инкрементное резервное копирование

В отличие от полного РК в этом случае копируются не все данные, а только те, что были модифицированы с момента последнего копирования. Для выяснения времени копирования могут применяться различные методы, например, в системах под управлением операционных систем семейства Windows используется соответствующий атрибут файла (архивный бит), который устанавливается, когда файл был изменен, и сбрасывается программой резервного копирования. В других системах может использоваться дата изменения файла.

При полном восстановлении системы нужно провести её восстановление из последней копии, созданной полным РК, а потом поочередно восстановить данные из инкрементных копий в порядке их создания. Данный вид РК используется для того, чтобы в случае создания архивных копий сократить расходуемые объемы на устройствах хранения информации. Также это позволит минимизировать время выполнения заданий РК, что может быть крайне важно в условиях, когда ПК работает постоянно. У инкрементного копирования есть одна особенность, заключающаяся в том, что при поэтапном восстановлении возвращаются и нужные, и удаленные файлы за период восстановления. Например, по выходным дням выполняется полное копирование, а по будням инкрементное. Пользователь в понедельник создал файл, во вторник его изменил, в среду переименовал, а в четверг удалил. При последовательном поэтапном восстановлении данных за недельный период в этой ситуации будут сформированы два файла: со старым именем за вторник до переименования, и с новым именем, созданным в среду. Это произошло потому, что в разных инкрементных копиях хранились разные версии одного и того же файла, и в итоге будут восстановлены все варианты. Поэтому при последовательном восстановлении данных из архива «как есть» имеет смысл резервировать больше диско

вого пространства, чтобы в нем смогли поместиться, в том числе, и удаленные файлы.

Достоинства метода:

* эффективное использование носителей, так как сохраняются только файлы, измененные с момента последнего полного или инкрементального РК, резервные копии занимают меньше места;
* меньшее время резервного копирования и восстановления, в сравнении с полным и дифференциальным РК.

Недостаток метода:

* данные резервного копирования сохраняются на нескольких носителях, и поэтому восстановление устройства после аварии может занять больше времени. Кроме того, для эффективного восстановления работоспособности системы носители должны обрабатываться в определенном порядке.

## 2.3 Дифференциальное резервное копирование

Дифференциальное резервное копирование отличается от инкрементного тем, что выполнение копирования данных происходит с последнего момента выполнения полного РК. В системах семейства Windows этот эффект достигается тем, что архивный бит при дифференциальном копировании не сбрасывается, поэтому измененные данные попадают в архивную копию, пока полное копирование не обнулит архивные биты. Каждая новая копия, созданная таким образом, содержит данные из предшествующей, это более уместно для полного восстановления данных на момент происшествия. Для этого необходимы только две копии:

* полная;
* последняя из дифференциальных.

Поэтому вернуть к жизни данные можно гораздо эффективнее, чем последовательно восстанавливать все инкременты.

Достоинства метода:

* меньшее время для восстановления, так как дифференциальное РК занимает меньше времени, чем полное резервное копирование. Восстановление после аварии выполняется быстрее, поскольку для полного восстановления устройства необходимы только последняя полная резервная копия и дифференциальная резервная копия.

Недостаток метода:

* избыточная защита данных, поскольку происходит сохранение всех файлов, измененных с момента последнего инкрементального резервного копирования. Таким образом, создаются избыточные резервные копии. [3]

## 2.4 Технологии хранения резервных копий и данных

В ходе выполнения РКД может возникнуть проблема выбора технологии хранения резервных копий и данных. В настоящее время используются следующие виды носителей:

* дисковые накопители;
* твердотельные накопители.

Носители на жестких магнитных дисках (Hard Disk Drive, HDD) являются самыми распространёнными устройствами хранения информации. Для современных одиночных накопителей характерны объемы до нескольких терабайт при времени доступа 5-15 мс и скорости передачи данных 30-100 Мбайт/с. Относительно расположения (непосредственно в корпусе сервера, либо вне его) различают внутренние и внешние накопители. Накопители первого типа намного дешевле, но их максимальное количество ограничивается числом свободных отсеков корпуса, мощностью и количеством соответствующих разъемов блока питания сервера. Установка и замена стандартных внутренних накопителей требует отключения системы, что в некоторых случаях недопустимо. Внутренние накопители с возможностью "горячей" замены (Hot Swap) представляют собой винчестеры, установленные в специальные кассеты с разъемами. Для типовых корпусов существуют недорогие приспособления [[5]](#footnote-5) обеспечивающие быструю замену стандартных винчестеров. Внешние накопители имеют собственные корпуса и блоки питания, их предельное количество определяется возможностями интерфейса. Обслуживание внешних накопителей может производиться и при работающем сервере, хотя может требовать прекращения доступа к части дисков сервера. Для больших объемов, хранимых данных применяются блоки внешних накопителей - дисковые массивы и стойки, представляющие собой сложные устройства с собственными интеллектуальными контроллерами, обеспечивающими, кроме обычных режимов работы, диагностику и тестирование своих накопителей. Более сложными и надежными устройствами хранения являются RAID-массивы[[6]](#footnote-6). Для пользователя RAID представляет собой один (обычно SCSI) диск, в котором производится одновременная распределенная избыточная запись (считывание) данных на несколько физических накопителей (типично 4-5) по правилам, определяемым уровнем реализации (0-10). Например, RAID Level 5 позволяет при считывании исправлять ошибки и осуществлять замену любого диска без остановки обращения к данным.

К достоинствам таких накопителей относится:

* скоростной доступ к данным;
* параллельный доступ без значительной потери скорости.

Недостатки дисковых накопителей:

* невозможность обеспечения высокой безопасности копий;
* относительно крупные размеры.

SSD (Solid State Drive - накопитель на твердотельной памяти, твердотельный накопитель) - накопитель информации, основанный на чипах энергонезависимой памяти, которые сохраняют данные после отключения питания. Являются относительно новым видом носителей информации, а первое проявление и развитие, чипы энергонезависимой памяти получили от Flash накопителей и обычной RAM памяти. Содержит такие же интерфейсы ввода-вывода, как и современные жёсткие диски. В SSD не используются движущиеся части и элементы как в электромеханических устройствах (жёсткие диски, дискеты), что исключает вероятность износа механическим путём.

Архитектура и функционирование твердотельных накопителей SSD накопители, которые были построенные на основе энергонезависимой памяти[[7]](#footnote-7) , появились относительно недавно, но в связи с гораздо более низкой стоимостью (от 2 долларов США за гигабайт) начали успешное покорение рынка. На настоящий момент скорость записи \ чтение SSD накопителей многократно превышает производительность HDD. Главным недостатком накопителей RAM SSD, это высокая стоимость (до 2-20 у. е. за 1 Гб). RAM SSD накопители зачастую используются в системах управления баз данных и графических центрах. Для сохранения данных при отключении питания применяются аккумуляторы, либо система РК.

Большинство современных твердотельных накопителей основаны на энергонезависимой NAND-памяти. Повсеместное применение SSD накопителей в современных условиях происходит по нескольким причинам:

* во-первых, цены на современные SSD стали гораздо ниже, по сравнению с первыми серийными SSD;
* во-вторых, твердотельные накопители выпускаются во всех форм-факторах, присущих жестким дискам;
* в-третьих, любой HDD можно заменить на SSD - необходимо только, чтобы совпадал интерфейс подключения накопителя.

Замена HDD на SSD для пользователя практически не несёт изменений, то есть такие манипуляции как форматирование, деление на разделы, установку и переустановку операционной системы можно проводить точно также, как обычно это делалось с использованием жесткого диска. Последняя причина является основным условием быстрого внедрения SSD на рынок компьютерной техники.

Отличным показателем скорости работы современных SSD можно считать 450 - 550 Мб/с на чтение и запись, что уже почти соответствует верхнему порогу пропускной способности интерфейса SATA 6 Gb/s. Такое быстродействие сегодня способны обеспечить достаточно большое количество современных твердотельных накопителей, при этом цена этого класса накопителей стремительно снижается.

Максимальным быстродействием для SSD с интерфейсом SATA 6 Gb/s сегодня обладают сразу несколько серий SSD: Corsair Neutron Series GTX, OCZ Vertex 4, Plextor M5S, Intel 520 Series, Corsair Force GS/GT, Silicon Power Velox V60, Kingston HyperX, OCZ Vertex 3. Такие SSD способны обеспечить непревзойденную производительность вашему компьютеру или ноутбуку. Самым быстрым из указанных выше твердотельных накопителей является SSD Corsair Neutron Series GTX.[[8]](#footnote-8)

В серверостроении традиционно применялись дорогостоящие SSD накопители на основе SLC (Single Level Cell, ячейка с одним уровнем) чипов памяти, ввиду их большей надежности и долговечности, являющихся важными факторами для серверов. Сегодня MLC (Multi Level Cell, ячейка с несколькими уровнями) чипы памяти ни в чем не уступают SLC чипам, поэтому даже серверные SSD сегодня производятся с применением данного типа чипов памяти. В целях экономии, а также благодаря таким технологиям, как MaxIQ (Adaptec) и Cash Cade (LSI), реализованных в RAID контроллерах некоторых производителей, SSD можно использовать в качестве кэш памяти массива из сравнительно дешевых и емких HDD.

К положительным качествам таких накопителей можно отнести:

* быстрый доступ к данным;
* отсутствие шума от движущихся частей;
* высокая механическая стойкость;
* широкий диапазон рабочих температур;
* возможность параллельного доступа к данным без потери скорости;
* более низкое энергопотребление относительно HDD;
* малые габариты устройств;
* высокая надёжность.

Недостатки дисковых накопителей:

* высокая стоимость.

Сетевое хранение данных построено на трех фундаментальных компонентах:

* коммутации;
* хранении;
* файлах.

Все продукты хранения можно представить в виде комбинации функций данных компонентов. Поначалу это может вызвать замешательство: поскольку продукты хранения разрабатывались по совершенно разным направлениям, функции часто перекрывают друг друга.

Термин «коммутация» применяется ко всему программному и аппаратному обеспечению и к службам, которые обеспечивают транспортировку хранения и управление ею в сетевом хранилище. Сюда входят такие различные элементы, как разводка кабелей, сетевые контроллеры ввода-вывода, коммутаторы, концентраторы, аппаратура выборки адресов, контроль связи данных, транспортные протоколы, безопасность и резервы ресурсов. В сетевых хранилищах все еще широко используются технологии шин данных SCSI и ATA, и, скорее всего, они будут использоваться еще долго. Фактически продукты SCSI и ATA сегодня применяются гораздо чаще в технологии NAS[[9]](#footnote-9). Существуют два важных различия между сетями хранения SAN[[10]](#footnote-10) и обычными локальными сетями LAN. Сети хранения SAN автоматически синхронизируют данные между отдельными системами и хранилищами. В сетевых хранилищах необходимы компоненты высокой степени точности для обеспечения надежной и предсказуемой среды. Несмотря на ограничения по расстоянию, параллельная SCSI -- чрезвычайно надежная и предсказуемая технология. Если новые технологии коммутации, такие как Fibre Channel, Ethernet и InfiniBand, сменят SCSI, они должны будут показать схожий или лучший уровень надежности и предсказуемости. Имеется и такая точка зрения, которая рассматривает коммутацию как канал хранилища. Сам термин «канал», берущий свое начало в среде больших вычислительных машин, предполагает высокую надежность и работоспособность.

Хранение в основном затрагивает блочные операции адресного пространства, включая создание виртуальной среды, когда адреса логического блока хранения отображаются из одного адресного пространства в другое. Вообще говоря, в сетевых хранилищах функция хранения почти не изменилась, если не считать двух заметных отличий:

* возможность нахождения технологий виртуализации устройства, например, управление устройством внутри оборудования сетевого хранения. Этот вид функции иногда называют контроллером домена хранения или виртуализацией LUN[[11]](#footnote-11);
* продукты хранения, такие как подсистемы хранения, имеют значительно больше контроллеров/интерфейсов, чем предыдущие поколения шинной технологии, а также намного больший объем хранения, поэтому можно говорить о масштабируемости.

Кроме упомянутых технологий хранения данных NAS и SAN, ориентированных на крупные и глобальные сети, в небольших локальных сетях лидирующее положение занимает технология DAS[[12]](#footnote-12) в соответствии с которой хранилище находится внутри сервера, обеспечивающего объем хранилища и необходимую вычислительную мощность.

Элементарным примером DAS может служить накопитель на жестком диске внутри ПК, подключенный к единственному серверу. Запросы ввода-вывода (называемые также командами или протоколами передачи данных) напрямую обращаются к этим устройствам. Такие системы плохо масштабируются, и компании с целью увеличения объема хранилища вынуждены приобретать дополнительные серверы. Эта архитектура очень дорогая и может использоваться только для создания малых по объему хранилищ данных. [6]

## 2.5 Хранение резервных копий

После создания резервных копий данных, необходимо определить условия их хранения. Для правильного определения места хранения копий, необходимо учитывать особенности эксплуатации создаваемых РК, выделяются три основные ситуации:

* восстановление единичных файлов по запросу пользователей;
* глобальное восстановление при чрезвычайной ситуации;
* организация архивного хранения.

Между первым и вторым вариантом существуют несовместимые противоречия. При случайном удалении данных пользователем, бывают обстоятельства, когда его необходимо возвратить немедленно. Из этого следует, что резервный носитель должен быть не дальше нескольких метров от компьютера, на котором должны быть восстановлены данные. В случае чрезвычайных ситуаций и аварий требуется выполнить полное восстановление одного или нескольких компьютеров в центре данных, а если произошедший сбой будет иметь физический характер, он разрушит не только компьютеры, но и все РК, хранящиеся рядом. Для решения этих разных задач могут быть выбраны различные подходы, в зависимости от потребностей организации. Первый вариант заключается в хранении копий за несколько дней на рабочем месте, а затем их перенос в безопасное удалённое хранилище. Другой подход заключается в поддержке двух наборов носителей:

* в центре данных, используемый исключительно для восстановления отдельных данных по запросу;
* для удалённого хранения и восстановления в случае чрезвычайных ситуаций.

Наличие двух вариантов подразумевает необходимость делать все резервные копии дважды или копировать их, но двойное РК может занять продолжительное время, а для хранения РК могут потребоваться несколько устройств.

## 2.6 Восстановление данных из резервных копий

Резервное копирование, как правило, выполняется ежедневно. В свою очередь процессы восстановления запускается намного реже, но в них обязательно возникнет необходимость. Для успешного восстановления данных из резервных копий необходимо проанализировать две важные ситуации:

* восстановление данных на компьютере с новой операционной системой;
* проверка актуальности резервных копий.

В первом случае - это процесс восстановления полной копии системы на компьютере, на котором отсутствуют данные. Выделяются два основных подхода к восстановлению на новом компьютере:

* переустановка, за которой следует восстановление, то есть базовая операционная система устанавливается таким же образом, как и на совершенно новый компьютер;
* ОС установлена и правильно настроена, оставшиеся диски можно подключить и отформатировать, а затем восстановить все копии с резервных носителей;

Для восстановления системы можно использовать загрузочный носитель данных, который содержит минимальное системное окружение и позволяет выполнять самые основные административные задачи. Окружение восстановления содержит необходимые системные приложения для разбиения на разделы и форматирования дисков, драйверы устройств, необходимые для обращения к устройству с резервными копиями, и программы, необходимые для восстановления данных с резервных носителей.

Все типы копий необходимо контролировать, чтобы убедиться в их читаемости и актуальности. Чаще всего потеря или повреждение данных обнаруживается только при возникновении необходимости в резервной копии. Причины этого могут быть самыми разными:

* смещение головки стримера;
* неправильно настроенная программа резервного копирования;
* ошибка оператора и т.д.

Администратор не может быть уверен в том, что действительно есть резервные копии, с которых когда-нибудь позже можно будет восстановить данные, поэтому необходима регулярная проверка РКД. [3]

## 2.7 Топология резервного копирования

#### 2.7.1 Децентрализованная схема

Ядром децентрализованной схемы является общий сетевой ресурс (см. рис. 1)[[13]](#footnote-13). В таком случае необходим набор программ для резервного копирования, время от времени выгружающих информацию с серверов и рабочих станций на этот ресурс. Данные программы установлены на каждом сервере и работают независимо друг от друга. Несомненным плюсом является простота реализации этой схемы и ее низкая цена. В качестве программ копирования подойдут штатные средства, встроенные в ОС, или программное обеспечение, такое как СУБД. Например, это может быть программа ntbackup для семейства Windows, программа tar для UNIX-like операционных систем или набор скриптов, содержащих встроенные команды SQL-сервера для выгрузки баз данных в файлы резервных копий. Еще одним плюсом является возможность использования различных программ и систем, лишь бы все они могли получить доступ к целевому ресурсу для хранения резервных копий.

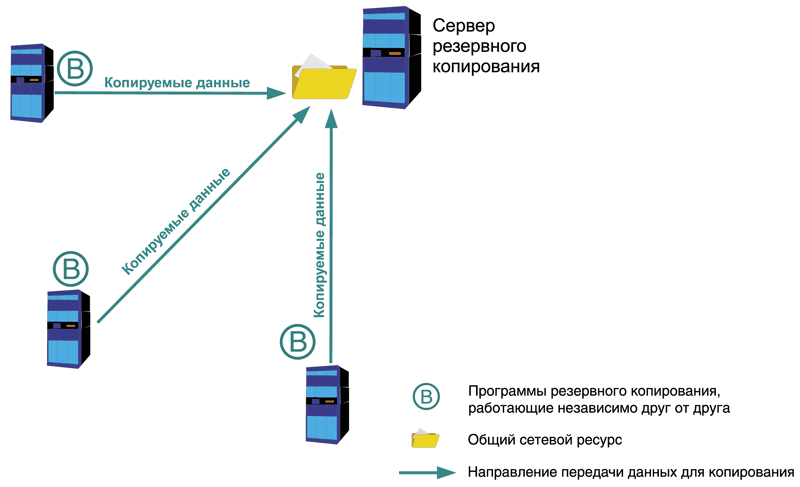
[](http://samag.ru/uploads/articles/2010/11/060_066_Backup/Pic1_decenter_scheme.gif)

Рисунок 1-Децентрализованная схема резервного копирования

Отрицательным качеством децентрализованной схемы является её инертность. Так как программы установлены независимо друг от друга, то и настраивать приходится каждую по отдельности. Из-за отсутствия единого интерфейса довольно тяжело учитывать особенности расписания и распределять временные интервалы, чтобы избежать конкуренции за целевой ресурс. Мониторинг также затруднен, процесс копирования с каждого сервера приходится отслеживать отдельно от других, что в свою очередь может привести к высоким трудозатратам.

Поэтому данная схема применяется в небольших сетях, а также в ситуации, когда невозможно организовать централизованную схему резервного копирования имеющимися средствами. [8]

#### 2.7.2 Централизованное резервное копирование

В данной схеме используется иерархическая модель, работающая по принципу «клиент-сервер». В классическом варианте на каждый компьютер устанавливаются специальные программы-агенты, а на центральный сервер – серверный модуль программного пакета. Эти системы также имеют специализированную консоль управления серверной частью. Схема управления выглядит следующим образом: с консоли создаются задания для копирования, восстановления, сбора информации о системе, диагностики и так далее, а сервер дает агентам необходимые инструкции для выполнения указанных операций[[14]](#footnote-14)(см. рис. 2).

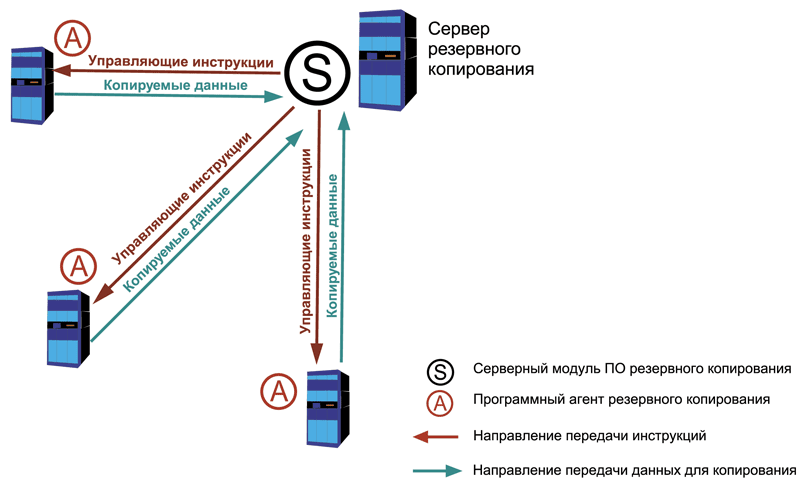
[](http://samag.ru/uploads/articles/2010/11/060_066_Backup/Pic2_center_scheme.gif)

Рисунок 2-Централизованная схема резервного копирования

Для небольших компаний в некоторых случаях есть возможность использовать упрощенный вариант централизованной схемы резервного копирования без применения программ-агентов (см. рис. 3). Также эта схема может быть задействована, если не реализован специальный агент для используемого ПО резервные копирования. Вместо этого серверный модуль будет использовать уже существующие службы и сервисы. Данная схема имеет весьма существенные ограничения, связанные с проблемами сохранения файлов, открытых для записи. В результате подобных действий открытые файлы будут либо пропущены и не попадут в РК, либо скопированы с ошибками. Существуют методы обхода данных проблем, например, повторный запуск задания с целью скопировать только ранее открытые файлы. Поскольку надежных вариантов исключения таких ошибок, такая схема подходит для применения только в определенных ситуациях. Например, в небольших организациях, работающих в режиме 5х8, с дисциплинированными сотрудниками, которые сохраняют изменения и закрывают файлы перед уходом домой (см. рис.3).

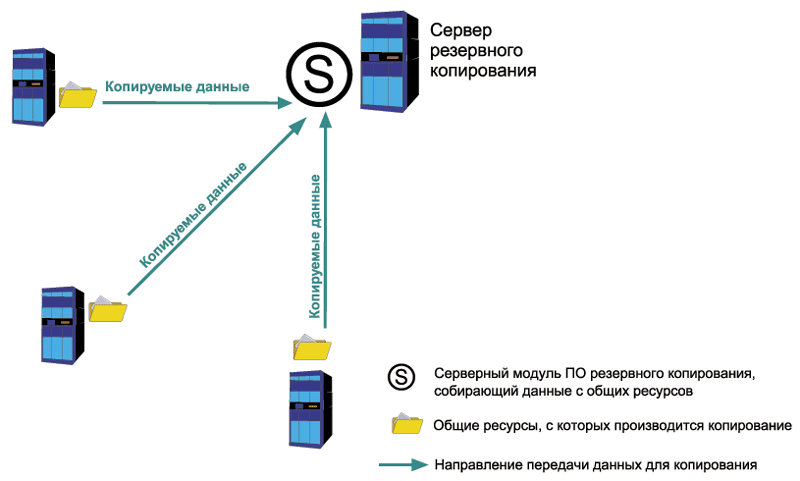
[](http://samag.ru/uploads/articles/2010/11/060_066_Backup/Pic3_simple_center_scheme.gif)

Рисунок 3-Упрощенная централизованная схема резервного копирования

Существует смешанная схема резервного копирования (см. рис. 4). Например, с серверов, для которых есть в наличии программы-агенты РК, данные собираются посредством этих агентов. Для всех остальных ресурсов используется децентрализованная схема, когда локальные программы складывают копии данных на общий ресурс сервера с установленным агентом, заносящим в общее хранилище резервных копий данную информацию.

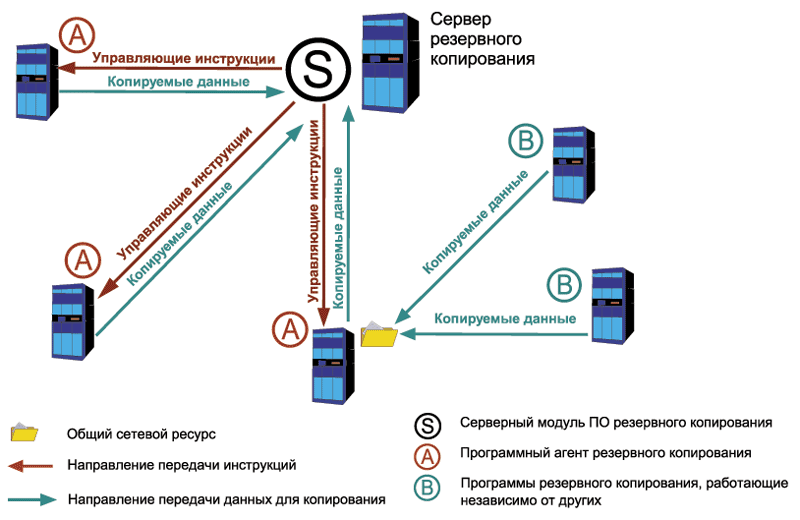
[](http://samag.ru/uploads/articles/2010/11/060_066_Backup/Pic4_combination_scheme.gif)

Рисунок 4- Смешанная схема резервного копирования

При событиях, способных вызвать отрицательные события для ИТ-инфраструктуры и бизнеса в целом [[15]](#footnote-15), одним из способов избежать потери информации в таких ситуациях является хранение резервных копий в месте, удаленном от основного расположения серверного оборудования. При этом необходимо предусмотреть быстрый способ доступа к данным, необходимым для восстановления. Описываемый метод называется off-site (хранение копий за территорией предприятия). В основном используются два метода организации этого процесса:

* запись данных на съемные носители и их физическое перемещение. В этом случае необходимо позаботиться о средствах быстрой доставки носителей обратно в случае сбоя. Например, хранить их в соседнем здании. Плюсом такого метода является возможность организовать этот процесс без каких-либо затруднений. Минусом являются сложность возврата носителей и сама необходимость передачи информации на хранение, а также риск повредить носители при перевозке;
* копирование данных в другое расположение по сетевому каналу. Например, с использованием VPN-туннеля через Интернет. Плюсом в этом случае является то, что нет нужды везти куда-то носители с информацией, минусом – необходимость использования достаточного широкого канала (как правило, это весьма недешево) и защиты передаваемых данных (например, с помощью того же VPN). Возникающие сложности передачи больших объемов данных можно значительно снизить, используя алгоритмы сжатия или технологию дедупликации.

Необходимо помнить о мерах ИБ при организации хранения данных. В начале необходимо позаботиться о том, чтобы носители с данными находились в охраняемом помещении, и о мерах, препятствующих прочтению данных посторонними лицами. Например, использовать систему шифрования, заключить договора о неразглашении. Если задействованы съемные носители, данные на них должны быть зашифрованы. Используемая система маркировки при этом не должна помогать злоумышленнику в анализе данных. Необходимо применять безликую номерную схему маркировки носителей названий передаваемых файлов. При передаче данных по сети необходимо (как уже писалось выше) использовать безопасные методы передачи данных, например, VPN-туннель. [3]

# 3 Средства резервного копирования, используемые в Инспекции ФНС №8 по г. Калининграду

Главной задачей федеральной налоговой службы является контроль выполнения налогового законодательства Российской Федерации, включая выполнение следующих функций:

* учет налогоплательщиков;
* контроль уплаты налогов и сборов;
* выявление налоговых нарушений;
* поддержка судебных исков, информационное обслуживание налогоплательщиков;
* налоговое планирование и др.

При этом ФНС имеет разветвленную структуру из следующих компонентов:

* центральный аппарат ФНС;
* межрегиональные инспекции федеральных округов и специализированные межрегиональные инспекции;
* 82 управления ФНС по субъектам Российской Федерации;
* около 1500 районных, межрайонных и городских налоговых инспекций.

ФНС осуществляет информационное взаимодействие с различными министерствами и ведомствами Российской Федерации, органами местного самоуправления, а также налогоплательщиками – физическими и юридическими лицами. ФНС располагает одной из самых мощных информационных систем среди государственных организаций (более 50 тысяч ПК, более 6000 серверов, единая мультисервисная сеть передачи данных), которая активно используется для решения задач ведомства.

В Инспекции ФНС №8 насчитывается свыше двухсот ПК и 6 серверов. Двое из них отвечают за резервное копирование, в свою очередь один серверов является «основным», а второй «зеркальным». Используется централизованная схема резервного копирования, описанная в предыдущей главе. Ниже показано схематичное представление устройства информационной сети (см.рис.5).

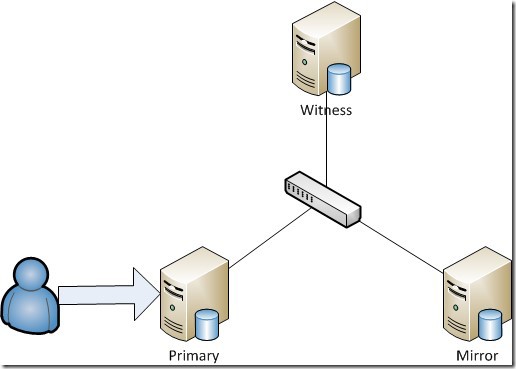


Рисунок 5-Схема зеркального резервного копирования

На главном сервере (Primary) находится база данных, с которой в обычном режиме работают пользователи. На резервном сервере (Mirror) располагается зеркальная копия базы данных. При защищенном синхронном режиме зеркалирования в схему может быть добавлен следящий сервер (Witness), задача которого осуществление мониторинга зеркального отображения и в случае отказа основного сервера или базы данных осуществить автоматический перевод в рабочее состояние зеркальной базы данных.

К преимуществам зеркалирования можно отнести:

* обеспечение отказоустойчивости дисковых массивов;
* возможность использования ресурсов резервного сервера;
* возможность создания географически распределенного кластера.

Недостатки:

* высокая стоимость решения (необходим резервный сервер с дисковой подсистемой, а также следящий сервер). [3]

В Инспекции ФНС №8 по городу Калининграду для создания резервных копий программного комплекса используется решение, разработанное «Лабораторией Касперского», Kaspersky Security Center. Проведение резервирования персональных данных налогоплательщиков и сотрудников Инспекции ФНС №8 осуществляется с помощью системы управления реляционными базами данных Microsoft SQL Server.

## 3.1 Kaspersky Security Center. Утилита klbackup.

В Инспекции ФНС №8 по городу Калининграду используется Kaspersky Security Center для администрирования и создания политики безопасности. Данная программа управляет системой предотвращения вторжений и настройкой параметров безопасности для отдельных устройств. Kaspersky Security Center, кроме того, обеспечивает контроль всех параметров IT-безопасности, а также дает централизованный доступ к широкому спектру функций управления системами, таких как:

* политика контроля, предотвращающая запуск нежелательных программ;
* управление правами доступа всех устройств, подключаемых пользователями к сети;
* контроль доступа к интернету: ограничение или запрет доступа к определенным сайтам или категориям сайтов;
* централизованное управление всеми параметрами шифрования данных для:
  + жестких дисков (шифрование файлов и папок или полное шифрование диска);
  + съемных устройств (шифрование файлов и папок или полное шифрование диска);
* согласование политик шифрования с параметрами Контроля программ и Контроля устройств;
* обнаружение уязвимостей при автоматическом сканировании всей корпоративной сети и назначении приоритетов их устранения;
* автоматическое распространение исправлений и обновлений (для продуктов компании Microsoft и другого ПО);
* автоматический учет аппаратного обеспечения, в том числе обнаружение гостевых устройств;
* автоматический учет программного обеспечения, который упрощает контроль использования программ и управление лицензиями;
* централизованное развертывание операционных систем;
* ролевое управление доступом, которое позволяет назначать разным администраторам различные обязанности по управлению безопасностью и системами;
* резервное копирование с помощью утилиты klbackup.

Утилита klbackup предназначена для выполнения РК и последующего восстановления БД. При РК сохранению подлежат:

* информационная база Сервера администрирования (политики, задачи, настройки приложения, сохраненные на Сервера администрирования события);
* конфигурационная информация о структуре логической сети и клиентских компьютерах;
* хранилище дистрибутивов программ для удаленной установки.

Во время выполнения резервного копирования базы данных доступ к Серверу администрирования невозможен. Это занимает от 10 до 15 минут. После завершения процесса РК доступ Серверу администрирования восстанавливается.

Утилита klbackup сохраняется в каталоге установки (C:\Program Files\Kaspersky Lab\Kaspersky Security Center) и может быть запущена из командной строки с набором ключей или в интерактивном режиме. Таким образом указание параметров работы утилиты будет выполняться через графический интерфейс утилиты.

## 3.2 Microsoft SQL Server

Microsoft SQL Server — система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Для эффективной работы с данными SQL Server имеет целый набор специальных инструментов. Можно выделить следующие характеристики основного инструмента Microsoft SQL Server:

* + SQL Setup – используется для установки нового или модификации установленного программного обеспечения. Программа Setup также может быть использована для изменения опций сетевой поддержки, подключения языка, перестройка базы данных Master и установки опций доступа к данным;
  + SQL Service – используется для старта и остановки служб SQL Server Manager (SQL Server и SQL Executive);
  + iSQL/w – позволяет вводить выражения и хранимые процедуры Transact-SQL в графическом интерфейсе запросов;
  + SQL Security – позволяет управлять бюджетами пользователей серверов Manager SQL;
  + SQL Client – устанавливает информацию соединения Server для утилиты конфигурирования клиентов;
  + SQL Transfer – обеспечивает легкий графический способ переноса Manager объектов и данных с одного Server на другой;
  + SQL Trace – графическая утилита, позволяющая администраторам и разработчикам отслеживать и фиксировать активность клиентских приложений, обращающихся к Microsoft SQL Server;
  + SQL Trace может в реальном времени отображать все аспекты обращений к серверу или использовать фильтры, отображающие информацию о действиях конкретных пользователей, приложений или машин.

Одним из самых используемых способов запуска РК в Microsoft SQL сервер является процедурное расширение языка SQL— Transact-SQL (T-SQL), созданное компанией Microsoft (для Microsoft SQL Server) и Sybase (для Sybase ASE).

Для более быстрого и эффекьтивного решения различных задач SQL был расширен такими дополнительными возможностями как:

* + управляющие операторы,
  + локальные и глобальные переменные,
  + различные дополнительные функции для обработки строк, дат, математики и т. п.,
  + поддержка аутентификации Microsoft Windows.

Язык Transact-SQL является основой к использованию MS SQL Server. Все приложения, взаимодействующие с экземпляром MS SQL Server, независимо от их реализации и пользовательского интерфейса, отправляют серверу инструкции Transact-SQL.

Пример использования Transact-SQL для создание полной резервной копии базы данных, выполняется в два этапа.

*В качестве первого этапа* рассматривается создание полной резервной копии базы данных. При этом задается имя базы данных для создания резервной копии и указывается устройство резервного копирования, на которое записывается полная резервная копия базы данных.

Базовая структура синтаксиса Transact-SQL для полного резервного копирования базы данных может быть представлена следующим образом:

BACKUP DATABASE *database*

TO *backup\_device* [ **,**...*n* ]

[ WITH *with\_options* [ **,**...*o* ] ] ;

Таблица 1-Параметры Transact-SQL

| Параметр | Описание |
| --- | --- |
| *База данных* | База данных для резервного копирования. |
| *backup\_device* [ **,**...*n* ] | Указывает список от 1 до 64 устройств резервного копирования, используемых для создания резервной копии. Можно указать как физическое устройство резервного копирования, так и соответствующее логическое устройство, если оно уже определено. Для указания физического устройства резервного копирования используйте параметр DISK или TAPE. { DISK | TAPE } **=***имяфизическогоустройства\_резервного \_копирования* Дополнительные сведения см. в разделе Устройства резервного копирования (SQL Server). |
| WITH *with\_options* [ **,**...*o* ] | При необходимости можно указать один или несколько дополнительных параметров, *o*. Сведения о некоторых основных параметрах см. в пункте 2. |

*На втором этапе* при необходимости указывается один или несколько параметров WITH. Здесь описываются некоторые основные параметры WITH.

Ниже представлены основные параметры WITH резервного набора данных:

{ COMPRESSION | NO\_COMPRESSION }  
Только в версии SQL Server 2008 Enterprise и выше указано, выполняется ли команда backup compression для этой резервной копии, переопределяя значение по умолчанию на уровне сервера.

ШИФРОВАНИЕ (АЛГОРИТМ, СЕРТИФИКАТ СЕРВЕРА |АСИММЕТРИЧНЫЙ КЛЮЧ)  
Только для SQL Server 2014 и выше указывается используемый алгоритм шифрования, а также сертификат или асимметричный ключ для шифрования.

DESCRIPTION **=** { **'***text***'** | **@***text\_variable* }-Данная команда задает произвольное текстовое описание резервного набора данных. В этой строке может содержаться до 255 символов.

NAME **=** { *backup\_set\_name* | **@***backup\_set\_name\_var* } -указывает имя резервного набора данных. Длина имени не может превышать 128 символов. Если параметр NAME не указан, то имя является пустым.

Основные параметры WITH резервного набора данных:

По умолчанию команда BACKUP добавляет резервную копию в существующий набор носителей, сохраняя существующие резервные наборы данных. Чтобы явно указать это, необходимо использовать параметр NOINIT. Сведения о присоединении к существующим резервным наборам данных приведены в разделе «Наборы носителей, семейства носителей и резервные наборы данных (SQL Server)».

Чтобы отформатировать носитель резервной копии используется команда FORMAT:

FORMAT [ **,** MEDIANAME**=** { *media\_name* | **@***media\_name\_variable* } ] [ **,** MEDIADESCRIPTION **=** { *text* | **@***text\_variable* } ]  
Используется FORMAT при первом использовании носителя или при необходимости перезаписать существующие данные.

При резервном копировании с помощью Windows PowerShell используется команда Backup-SqlDatabase. Чтобы явно указать, что это полная резервная копия базы данных, параметр -BackupAction со значением по умолчанию Database не является необязательным для полных резервных копий баз данных. В следующем примере создается полная резервная копия базы данных MyDB в заданном по умолчанию расположении резервного копирования на экземпляре сервера Computer\Instance. Дополнительно в этом примере указывается параметр **-**BackupAction Database:

Backup-SqlDatabase -ServerInstance Computer\Instance -Database MyDB -BackupAction Database

# 4 Разработка системы контроля резервного копирования данных

Просмотр отчета об ошибках и выполнении резервного копирования является необходимой ежедневной задачей. Этот процесс занимает много времени, но он необходим, так как от этого зависит надежность работающей системы резервного копирования. Проблемы при РК, как правило, возникают «лавинообразно». Единичный сбой может повлечь за собой последовательность, на первый взгляд даже не связанных между собой затруднений. Например, задание резервного копирования может либо "зависнуть", либо не запуститься из-за каких-либо неполадок. Выполнение данного процесса не позволило закончить резервное копирование в установленный расписанием срок. Ответственный системный администратор из-за своей халатности своевременно не смог проинформировать администратора резервного копирования о сбоях в подсистеме РК, что привело к возникновению проблем при совершении дальнейшего восстановления системы. Естественно, для успешного решения возникающих проблем необходима согласованная работа системных, сетевых администраторов и администраторов баз данных.

В ходе прохождения преддипломной практики была поставлена задача по разработке средства контроля системы резервного копирования данных. Для достижения поставленной цели необходимо провести анализ уже существующих систем и на основе этих данных составить концепцию разрабатываемой подсистемы.

## 4.1 Изучение и анализ аналогичных систем резервного копирования данных

В настоящее время существует большой выбор систем резервного копирования данных (продукты компаний EMC, IBM, HP, Symantec, Dell, NetApp, CA Technologies, CommVault, Acronis и Paragon Software Group). Их решения особенно актуальны в связи с политикой экономии многих компаний, которые начинают уделять особое внимание показателям «цена/качество».

Главной системой компании EMC является EMC Avamar. Это комплексное программно-аппаратное решение для резервного копирования и восстановления данных, позволяющее работать как реальными, так и виртуальными устройствами. Поддерживается функция дедупликации сегментов переменной длины, которая осуществляется на устройстве клиента. Проводится оптимизации передачи данных при работе с разными типами сетей. Восстановление производится за один шаг.

В сфере резервного копирования компания IBM представлена продуктом IBM Tivoli Storage Manager. Это программный продукт, который занимается созданием резервных копий и управлением устройствами хранения. IBM Tivoli Storage Manager совместим с большим количеством различных систем хранения данных. Он обеспечивает работу в локальных (LAN), глобальных (WAN) сетях и развивающихся сейчас сетях хранения данных (SAN). Tivoli Storage Manager включает в себя средства для удалённого администрирования из любой точки сети или через Интернет. Администратор может автоматизировать различные этапы работы системы резервного копирования и восстановления данных за счет создания политик хранения. Набор политик, настроенный администратором, позволяет не только проводить штатные операции, но и оперативно реагировать на возникновение различных незапланированных ситуаций.

Компания Symantec предоставляет два продукта для РК и ВД - это Symantec Backup Exe и Symantec NetBackup, которые предназначены для работы в средних и крупных сетях. Данные приложения включают в себя весь набор необходимых для резервного копирования функций (дедупликация, автоматическая целевая репликация, миграция между компьютерами, работа с физическими и виртуальными устройствами, работа в гетерогенной среде и т.д.). Данные продукты могут работать с различными системами хранения данных. Для удобства работы с системами резервного копирования и восстановления данных Symantec также выпустила на рынок программно-аппаратные устройства Symantec Backup Exec 3600, Symantec NetBackup 5230. Одним из преимуществ их использования является минимальное время для их развертывания на предприятии. Утверждается, что администратору понадобится 20-30 минут, чтобы устройства начали работать и полноценно выполнять свои функции.

Компания HP представляет на рынке большой набор различных решений. Программное обеспечение HP Data Protector предназначено для обеспечения автоматического резервного копирования и восстановления данных, а также централизованного управления соответствующими политиками и процессами. Оно поддерживает большинство известных платформ и приложений и позволяет выполнять следующие функции:

* дедупликацию данных в соответствии с технологией HP StoreOnce Federated Deduplication как на клиентах (source based), так и на отдельно выделенном сервере (backup server) или же на системах хранения данных (target based);
* резервное копирование и восстановление виртуальных сред, включая защиту как целиком виртуальных машин, так и отдельных данных внутри них;
* интеграцию с функциональностью дисковых массивов по созданию аппаратных снимков (snapshots).

Для крупных предприятий и компаний Dell предлагает решение NetVault для организации резервного копирования всей инфраструктуры, а для компаний малого и среднего бизнеса более простое решение Appasure. Российские разработчики подразделения Dell представляют также специализированные решения Dell Software для гранулярного восстановления AD и Exchаnge, а также уникальную технологию автоматизированного восстановления Active Directory при потере данных.

Компания Acronis предоставляет целую линейку программного обеспечения для организации резервного копирования и восстановления данных. Для домашнего использования предназначено приложение Acronis True Image, для малых предприятий используется Acronis Backup & Recovery Server for Windows, а для крупных предприятий - Acronis Backup & Recovery Advanced Server for Windows. Наиболее функциональным является корпоративный продукт Acronis Backup & Recovery Advanced Server, позволяющий выполнять большое количество функций: файловое резервное копирование и резервное копирование на основе образа.

Рассмотренные средства РК соответствуют всем современным требованиям, эффективно и успешно справляются с поставленными перед ними задачами. После ознакомления с коммерческим программным обеспечением для организации резервного копирования, были составлены необходимые требования для разрабатываемой подсистемы.

## 4.2 Требования к разрабатываемой системе контроля резервного копирования данных

Система резервного копирование в Инспекции ФНС №8 реализована с помощью современных средств, но при этом не лишена недостатков, вызванных её функциональными особенностями:

* РК осуществляется с использованием двух программных средств MS SQL Server и klbackup, следовательно, отображение журналов отчетов резервного копирования происходит в разных пользовательских интерфейсах. При этом отсутствует единообразие в последовательности проверки результатов РК;
* обзор логов ошибок и отчета о результате резервного копирования занимает много времени и является ежедневной задачей, так как непосредственно влияет на надёжность действующей системы;
* возможность проверки результатов резервного копирования с любого компьютера, зарегистрированного в системе контроля резервного копировании данных.

По требованиям заказчика Инспекции ФНС №8 по городу Калининграду и по данным полученным при анализе других систем РК был выделен ряд элементов, которые были приняты к рассмотрению и привели к разработке средства контроля системы резервного копирования. Ключевыми задачами стали:

* обеспечение консолидации, хранения и обработки журнала резервного копирования;
* обеспечение возможности удобного удаленного доступа к информации и ограничения доступа к информации для каждого пользователя системы;
* создание эффективного программного продукта, предъявляющего низкие требования к аппаратному обеспечению, имеющего удобный и функциональный пользовательский интерфейса, высокую скорость работы и отсутствие влияния на основную систему.

В Инспекции ФНС №8 по городу Калининграду в основе системы резервного копирования используется Microsoft SQL Server. Это современная система, которая предусматривает всестороннюю поддержку со стороны корпорации Microsoft и на протяжении последних 7 лет считается самой безопасной платформой.

В настоящее время продукт компании Microsoft имеет ряд особенностей, которые будут использоваться при разработке системы контроля РК.

Во-первых, это поддержка и развитие среды программирования Qt, Python.  
Пользователи получили возможность запускать Python-скрипты, а также встраивать код Python напрямую в базы данных SQL Server, включая его как хранимую процедуру T-SQL. Это позволяет коду Python развертываться в рабочей среде с данными, которые он будет обрабатывать.  При этом предустанавливать Python не требуется. Администраторы БД смогут устанавливать ограничения в поведении Python и блокировать скрипты, угрожающие безопасности — изолируя процессы, управляя привилегиями систем и сетевым доступом. Что касается ученых, то нововведение позволит им строить модели, используя полные своды данных на SQL Server, отказавшись от сокращенных выборок. Также они получают возможность пользоваться любыми пакетами Python для машинного обучения.

Во-вторых, данный продукт имеет развитое сообщество разработчиков, в связи с чем найти квалифицированного сотрудника для обслуживания и ремонта системы не составит труда. Также доступна быстрая модернизация всей системы, и при этом продукт обладает высокой отказоустойчивостью и возможностью интеграции платформы практически в любую действующую систему.

## 4.3 Понятие клиент-серверного приложения

Разрабатываемая программа контроля резервного копирования должна состоять из двух основных частей: клиентской и серверной. Это определяется архитектурой система резервного копирования Заказчика. В процессе разработки программы были рассмотрены существующие типы «клиент-серверной» архитектуры, произведен их анализ, на основе которого выбрана конфигурация создаваемой подсистемы контроля резервного копирования данных в ИФНС №8 по городу Калининграду.

В абстрактном сетевом приложении можно выделить три группы функций, ориентированных на решение различных подзадач:

* функции ввода и отображения данных (обеспечивают взаимодействие с пользователем);
* прикладные функции, характерные для данной предметной области;
* функции управления ресурсами (файловой системой, базой данных и т.д.)



Рисунок 6-Компоненты сетевого приложения

Выполнение этих функций в основном обеспечивается программными средствами, которые можно представить в виде взаимосвязанных компонентов (рис. 6).

Компонент представления отвечает за пользовательский интерфейс, прикладной компонент реализует алгоритм решения конкретной задачи, а компонент управления ресурсом обеспечивает доступ к необходимым ресурсам.

Архитектура «клиент-сервер» определяет общие принципы взаимодействия в сети, где имеются серверы, узлы-поставщики некоторых специфичных функций (сервисов) и клиенты, потребители этих функций.

Практические реализации такой архитектуры называются клиент-серверными технологиями. Каждая технология определяет собственные или использует имеющиеся правила взаимодействия между клиентом и сервером, которые называются протоколом обмена (протоколом взаимодействия).

Существует три основных типа клиент-серверной архитектуры:

* двухзвенная;
* трехзвенная;
* многозвенная.

В любой сети (даже одноранговой), построенной на современных сетевых технологиях, присутствуют элементы клиент-серверного взаимодействия, чаще всего на основе двухзвенной архитектуры. Двухзвенной (two-tier, 2-tier) она называется из-за необходимости распределения трех базовых компонентов между двумя узлами (клиентом и сервером).

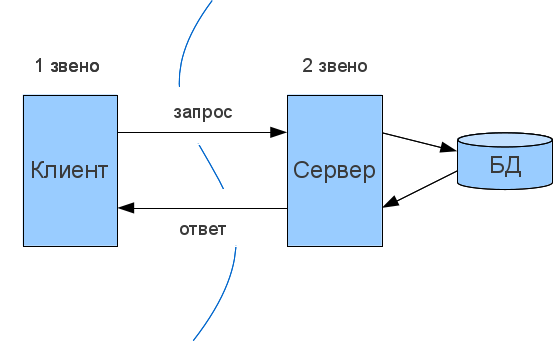


Рисунок 7-Двухзвенная клиент-серверная архитектура

Двухзвенная архитектура используется в клиент-серверных системах, где сервер отвечает на клиентские запросы напрямую и в полном объеме, при этом используя только собственные ресурсы. Т.е. сервер не вызывает сторонние сетевые приложения и не обращается к сторонним ресурсам для выполнения какой-либо части запроса (рис. 7)

Расположение компонентов на стороне клиента или сервера определяет следующие основные модели их взаимодействия в рамках двухзвенной архитектуры:

* сервер терминалов — распределенное представление данных;
* файл-сервер — доступ к удаленной базе данных и файловым ресурсам;
* сервер БД — удаленное представление данных;
* сервер приложений — удаленное приложение.

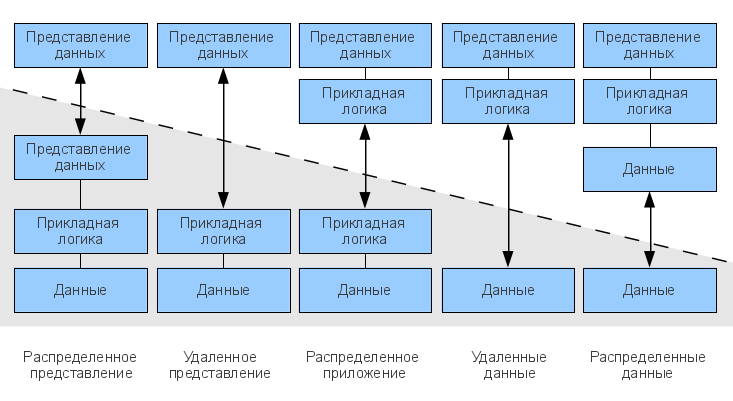


Рисунок 8-Модель клиент-серверного взаимодействия

С появлением персональных компьютеров и локальных сетей, была реализована модель файлового сервера, представлявшего доступ файловым ресурсам, в том числе и к удаленной базе данных. В этом случае выделенный узел сети является файловым сервером, на котором размещены файлы БД. На клиентах выполняются приложения, в которых совмещены компонент представления и прикладной компонент, использующие подключенную удаленную базу как локальный файл. Протоколы обмена при этом представляют набор низкоуровневых вызовов операций файловой системы.

Такая модель показала свою неэффективность ввиду того, что при активной работе с таблицами БД возникает большая нагрузка на сеть. Частичным решением является поддержка тиражирования (репликации) таблиц и запросов. В этом случае, например, при изменении данных, обновляется не вся таблица, а только модифицированная ее часть.

С появлением специализированных СУБД появилась возможность реализации другой модели доступа к удаленной базе данных — модели сервера баз данных. В этом случае ядро СУБД функционирует на сервере, прикладная программа на клиенте, а протокол обмена обеспечивается с помощью языка SQL. Такой подход по сравнению с файловым сервером ведет к уменьшению загрузки сети и унификации интерфейса «клиент-сервер». Однако, сетевой трафик остается достаточно высоким, кроме того, по-прежнему невозможно удовлетворительное администрирование приложений, поскольку в одной программе совмещаются различные функции.

С разработкой и внедрением на уровне серверов баз данных механизма хранимых процедур появилась концепция активного сервера БД. В этом случае часть функций прикладного компонента реализованы в виде хранимых процедур, выполняемых на стороне сервера. Остальная прикладная логика выполняется на клиентской стороне. Протокол взаимодействия — соответствующий диалект языка SQL.

Преимущества такого подхода очевидны:

* возможно централизованное администрирование прикладных функций;
* снижение стоимости владения системой (TOC, total cost of ownership) за счет аренды сервера, а не его покупки;
* значительное снижение сетевого трафика (т.к. передаются не SQL-запросы, а вызовы хранимых процедур).

Реализация прикладного компонента на стороне сервера представляет следующую модель — клиент-серверных приложений. Перенос функций прикладного компонента на сервер снижает требования к конфигурации клиентов и упрощает администрирование, но представляет повышенные требования к производительности, безопасности и надежности сервера. В настоящее время намечается тенденция возврата к тому, с чего начиналась клиент-серверная архитектура — к централизации вычислений на основе модели терминал-сервера. В современной реинкарнации терминалы отличаются от своих алфавитно-цифровых предков тем, что, имея минимум программных и аппаратных средств, представляют мультимедийные возможности (в том числе графический пользовательский интерфейс). Работу терминалов обеспечивает высокопроизводительный сервер, куда вынесено все, вплоть до виртуальных драйверов устройств, включая драйверы видеоподсистемы.

## 4.4 Модульная архитектура системы контроля резервного копирования данных в информационной системе Инспекции ФНС №8 по г. Калининграду

Система контроля создания резервных копий данных - представляет собой клиент-серверное приложение, реализующие функции проверки сохранности копий ценной информации (в данном случае документации, персональных данных и программных средств Инспекции ФНС №8 по городу Калининграду).

Сервер — основная программа, предоставляющая сервис по консолидации отчетной информации, о хранящихся на сервере файлах РК и ее предоставлении приложениям клиентам. Пользовательская часть приложения отображает информацию в удобном для пользователя графическом виде. Предусмотрена возможность фильтрации и поиска информации в логах РК. Для удобства работы с все записи разделены по категориям. Сервер также предоставляет возможность создания новых учетных записей для клиентов.

Система реализована на языке Python, версии 3.6, с использованием фреймворка PyQt5, версии 5.8, сетевая составляющая реализована средствами Qt — собственными классами работы с сетью по протоколу TCP/IP.

Выбор языка Python определили следующие его достоинства:

* официальная поддержка со стороны Microsoft SQL Server;
* интеграция с большинством графических и технических приложений;
* интерпретатор Python реализован на всех платформах и операционных системах;
* наличие большого числа подключаемых модулей, дающих дополнительные возможности (Например, Numerical Python, Tkinker, OpenGL);
* возможность эффективного решения сложных задач;
* удобное сопровождение программного кода;
* переносимость кода между платформами;
* поддержка процедурного, функционального и объектного стилей программирования;
* поддержка Unicode.

Использования PyQt определили следующие его возможности:

* большая база готовых графических компонентов;
* возможность создания собственных графических элементов;
* совместимость с современными графическими редакторами.

В процессе разработки системы был проведен анализ технических требования к ней, на основе которого выделены следующие основные функции подсистемы контроля резервного копирования данных:

* добавление, удаление учетных записей клиентов;
* возможность установки времени отслеживания успешности резервного копирования;
* разделение записей в клиенте по категориям;
* сортировка доступности записей в клиенте в зависимости от пользователя;
* контроль успешности резервного копирования данных;
* аутентификация пользователей с учетом их предварительной регистрации на сервере.

На основе анализа функциональных связей элементов системы разработана структурная схема средства контроля системы резервного копирования, представленная на рис. 9.

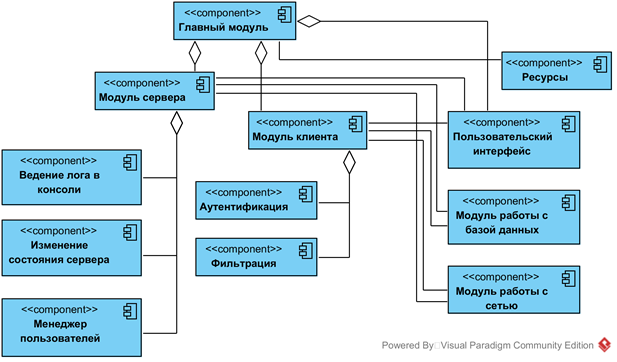


Рисунок 9-Структурная схема средства контроля системы резервного копирования

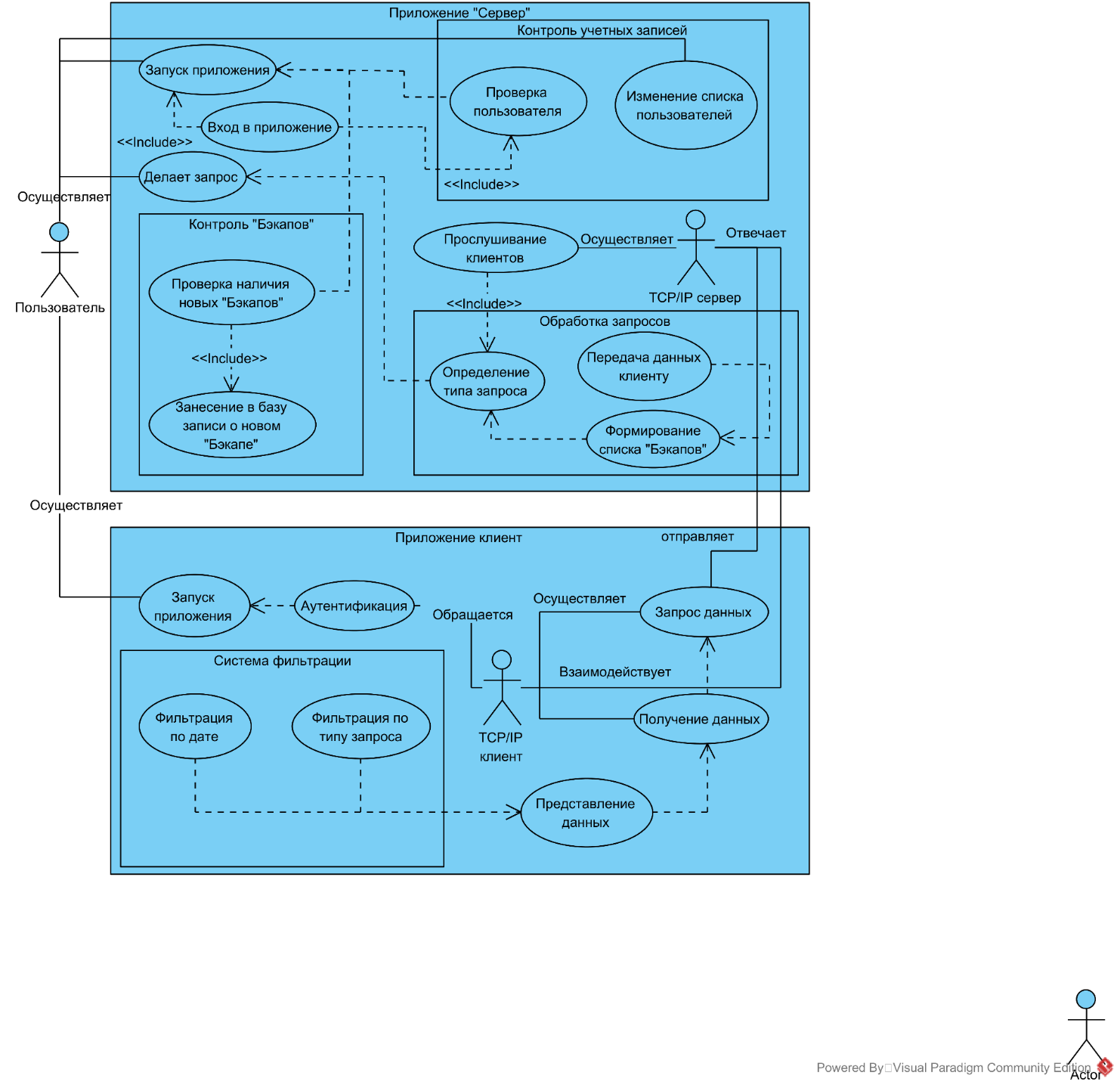


Рисунок 10-Диаграмма вариантов использования

Диаграмма вариантов использования — это диаграмма, на которой изображаются отношения между сущностями и вариантами использования.

ГОСТ 34.601-90[[16]](#footnote-16) подразумевает возможность создания и использования диаграмма вариантов использования, которая является исходным концептуальным представлением или концептуальной моделью системы в процессе ее проектирования и разработки. Для её составления использовалась среда объектно-ориентированного проектирования на языке UML, распространяемая бесплатно для учебной деятельности, Visual Paradigm for UML. Создание диаграммы вариантов использования имеет следующие цели:

* определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы;
* сформулировать общие требования к функциональному поведению проектируемой системы;
* разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;
* подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Назначение данной диаграммы состоит в следующем: проектируемая программная система представляется в форме так называемых вариантов использования, с которыми взаимодействуют внешние сущности. Ими может быть человек, техническое устройство, программа или любая другая система, которая служит источником воздействия на моделируемую систему так, как определит разработчик. 

Благодаря данной диаграмме вариантов использования, появляется возможность перейти к этапу написания программного кода средства контроля системы резервного копирования данных, совершенно точно зная, какой функционал программного продукта необходимо реализовать.

## 4.5 Описание работы системы контроля резервного копирования данных в информационной системе Инспекции ФНС №8 по г. Калининграду

Общая работа серверного приложения системы контроля резервного копирования Инспекции ФНС №8 по городу Калининграду представлена в виде диаграммы активности[[17]](#footnote-17) программы на рисунке 11.



Рисунок 11-Диаграмма деятельности приложения сервера системы контроля резервного копирования

При запуске сервера открывается консольное окно. В верхней части формы находится индикатор включения сервера, IP-адрес и номер порта (192.168.0.109: 9090), количество подключенных клиентов (на момент запуска:0), «кнопка» конфигурации пользователей и настройки времени проверки отчетов о РК. Визуальное представление показано на рисунке 12.

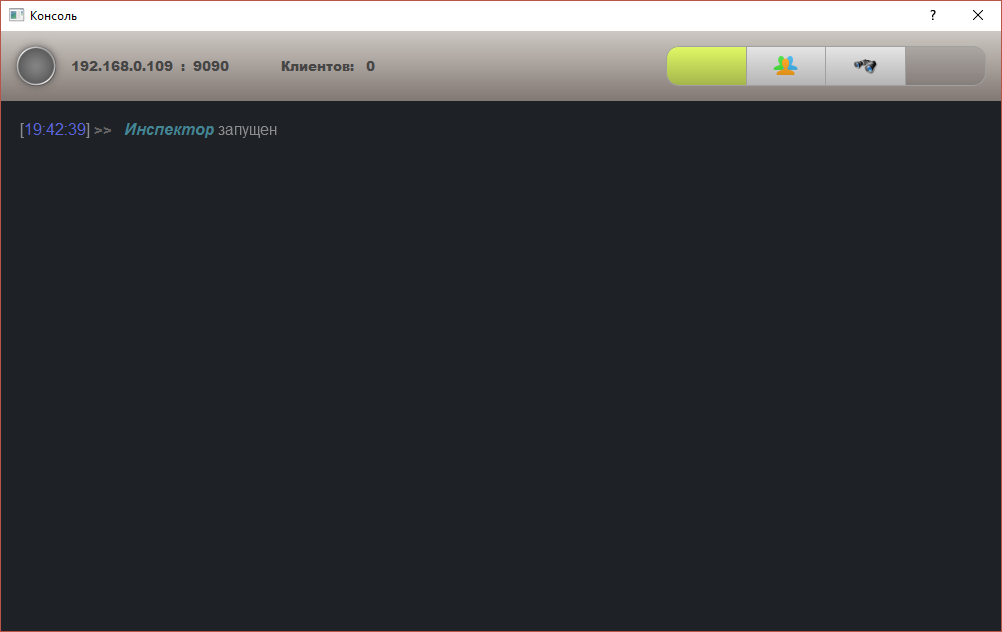


Рисунок 12-Начальное окно приложения сервера.

Пользовательский интерфейс формы администрирования пользователей представлен на рисунке 13. На нём отображаются кнопки «добавления» и «удаления пользователей», таблица с логином и паролем в зашифрованном виде.

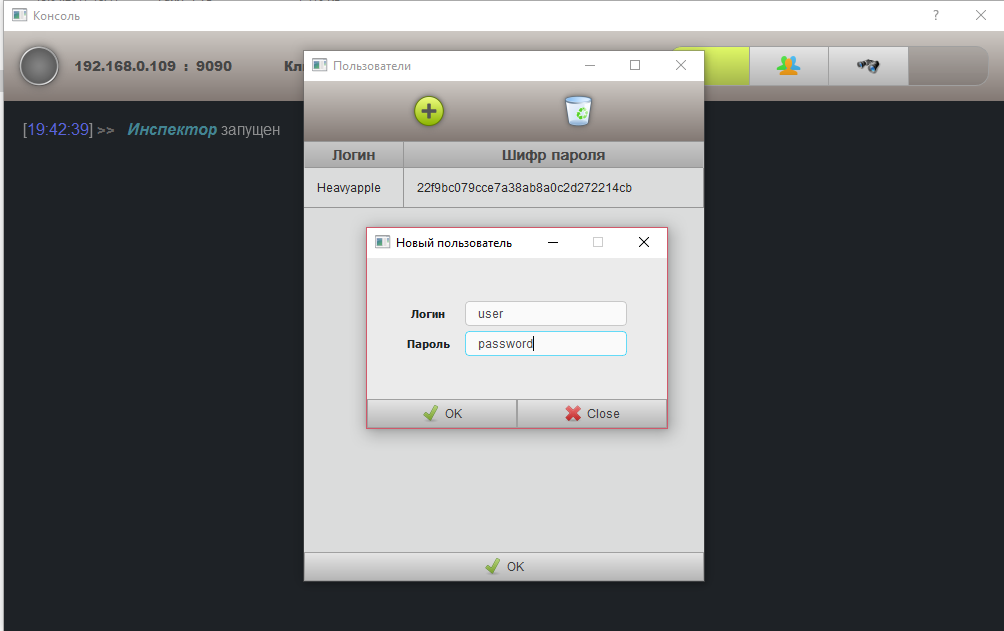


Рисунок 13-Форма администрирования пользователей

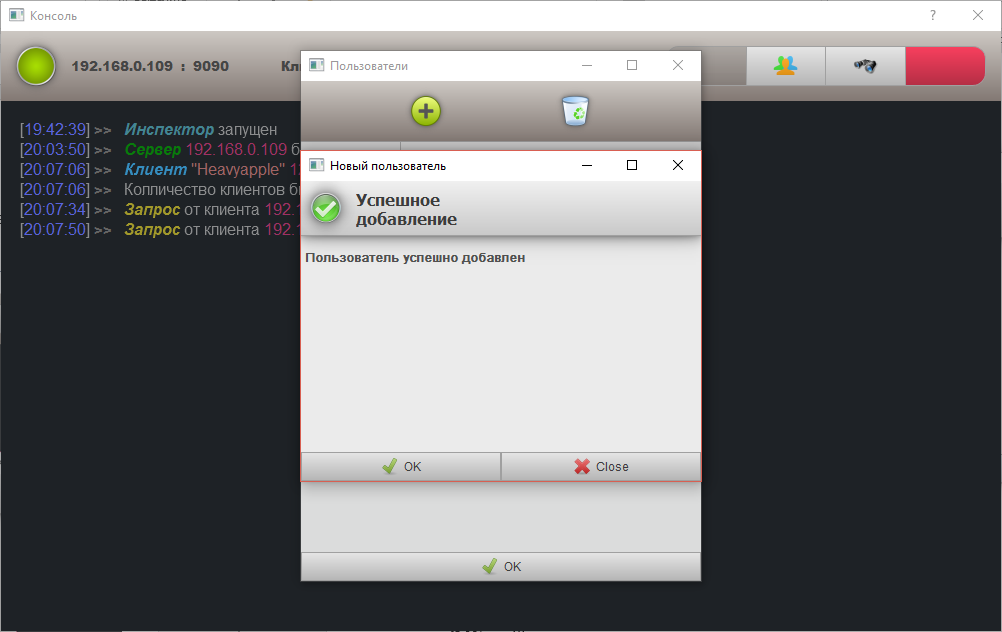


Рисунок 14-Сообщение об успешном добавлении пользователя

На диалоговом окне присутствуют поля для ввода имени пользователя и пароля. Имя пользователя должно совпадать с именем персонального компьютера, на котором будет запущен клиент. Иначе, аутентификация будет невозможна. Пароль автоматически шифруется по технологии хеширования переменной разрядности, разработанный группой авторов во главе с Йоаном Дайменом, под названием SHA3, также известным, как Keccak. Также на рисунке 14 показано диалоговое окно, в котором сообщается об успешном добавлении пользователя. Наиболее важная часть исходного кода по шифрованию пароля представлена ниже.

Для шифрования строк предназначен модуль hashlib. Прежде чем использовать функции из этого модуля, необходимо подключить модуль с помощью инструкции:

import hashlib

class PasswordEncoder(QObject):

def \_\_init\_\_(self):

super(PasswordEncoder, self).\_\_init\_\_()

def encode(self, password=''):

password = str.encode(password)

Модуль предоставляет следующие функции: md5(), sha1(), sha224(), sha256(), sha384, sha512() и sha3(). В качестве необязательного параметра функциям можно передать шифруемую последовательность байтов.

hash = hashlib.sha3(password)

hex = hash.hexdigest()

return hex

Получить зашифрованную последовательность байтов и строку позволяет метод —hexdigest():

def isRight(self, pasw='', hex=''):

pasw = str.encode(pasw)

hash = hashlib.sha3(pasw)

pasw\_hex = hash.hexdigest()

return pasw\_hex == hex

Пароль будет отображен в пользовательском графическом интерфейсе в зашифрованном виде:

if \_\_name\_\_ == '\_\_main\_\_':

pe = PasswordEncoder()

hex = pe.encode('123')

print(pe.isRight('123', hex))

Форма настройки параметров сканирования представлена на рисунке 15. На ней задается время для проверки журанала резервного копирования и директрорию расположения файлов резервного копирования данных.

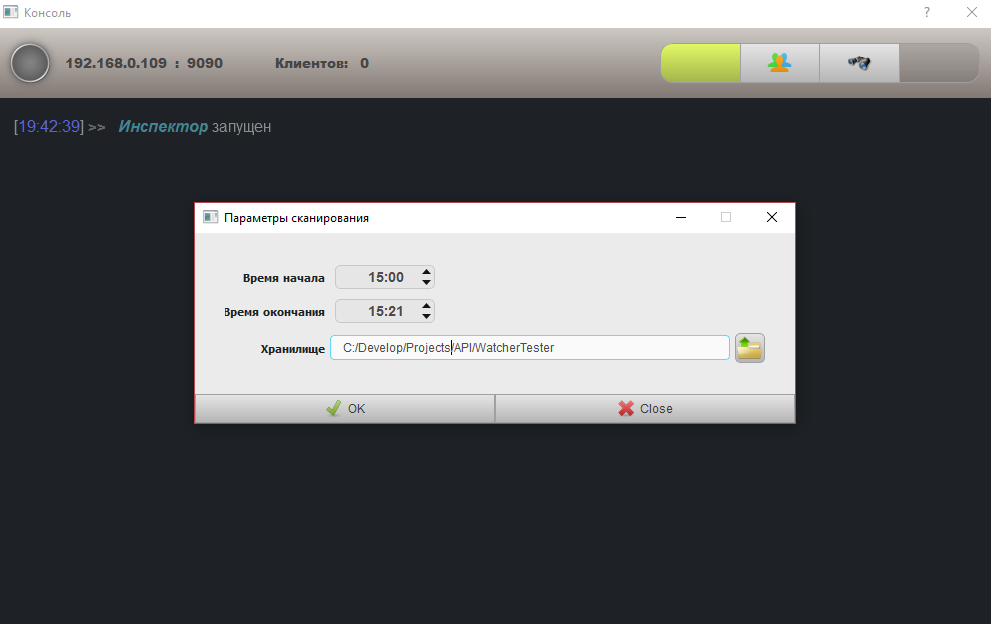


Рисунок 15-Форма настройки параметров сканирования

После проведения необходимых настроек, которые сохраняются в памяти сервера, производится запуск с помощью зеленой кнопки на начальной форме серверной части приложения. В консоле сервера появится запись о том, что сервис активирован, его IP-адрес (192.168.0.109) и назначенный порт (9090).

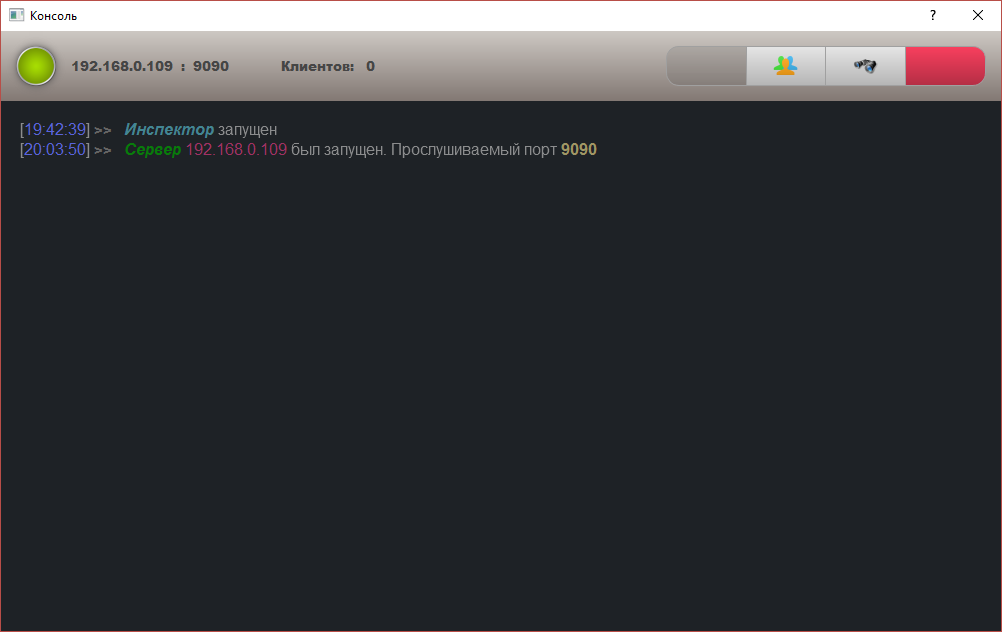


Рисунок 16-Состояние формы при запущенном сервере

Общая работа клиентского приложение показана в виде диаграммы деятельности на рисунке 17.



Рисунок 17-Диаграмма деятельности клиентского приложения

При запуске клиентской части программы введем неверный пароль в форму входа, после данного действия появится сообщение об ошибке, и на сервере отобразится, что был произведен запрос на авторизацию (см.рис.17).

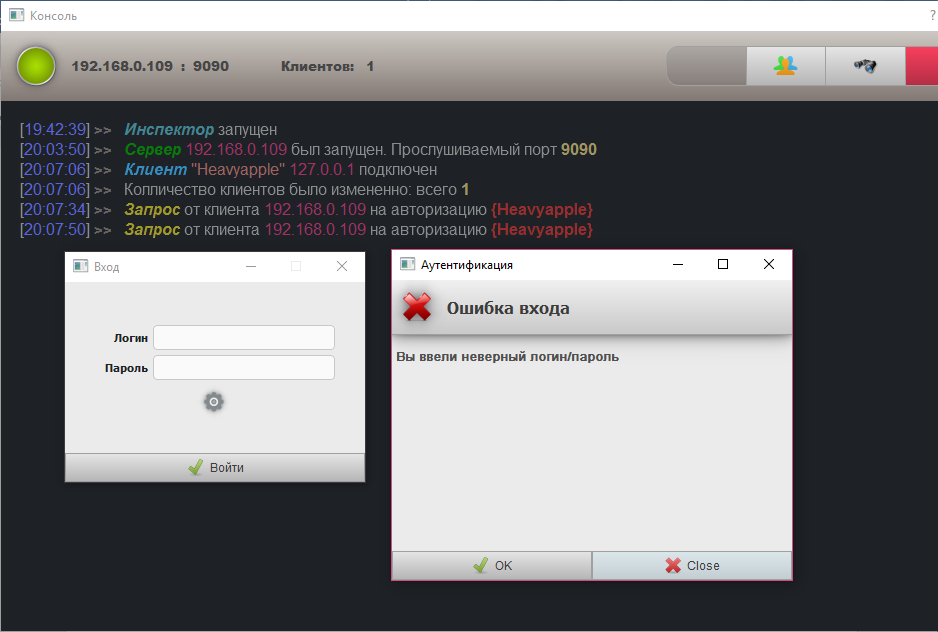


Рисунок 18-Форма входа и сообщение об ошибке аутентификации

Для подключения клиента к серверу необходимо провести настройки сетевой конфигурации клиента, пример настройки показан на рисунке 19.

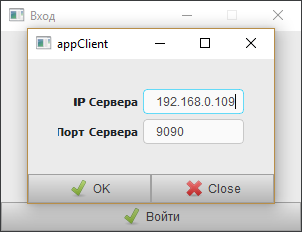


Рисунок 19-Форма настройки IP-адреса и порта подключения к серверу.

При изменении настроек клиента и его подключения к серверу все процессы отображаются в окне консоли (см.рис.20).

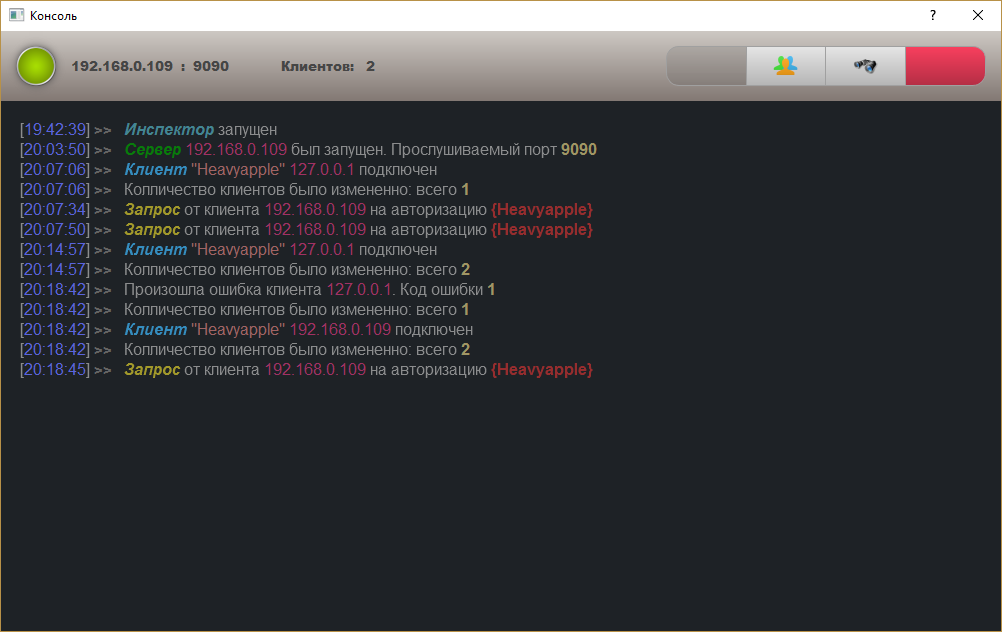


Рисунок 20-Отображение действий настройки клиента

На рисунке 21 представлена форма запущенного клиента. На ней отображается окно выбора типа отчетов о резервном копировании данных:

* персональные данные налогоплательщиков;
* персональные данные сотрудников ФНС;
* программное обеспечение;
* документы, используемые в работе ФНС.

Кроме того, клиентская часть приложения представлена визуальным интерфейсом с меню выбора дат, отображением запрашиваемых данных и непосредственным именем пользователя, запустившего клиент, время/дату резервного копирования, его успешность, а также сортировку по выполнению резервного копирования данных и сортировку журнала отображения.

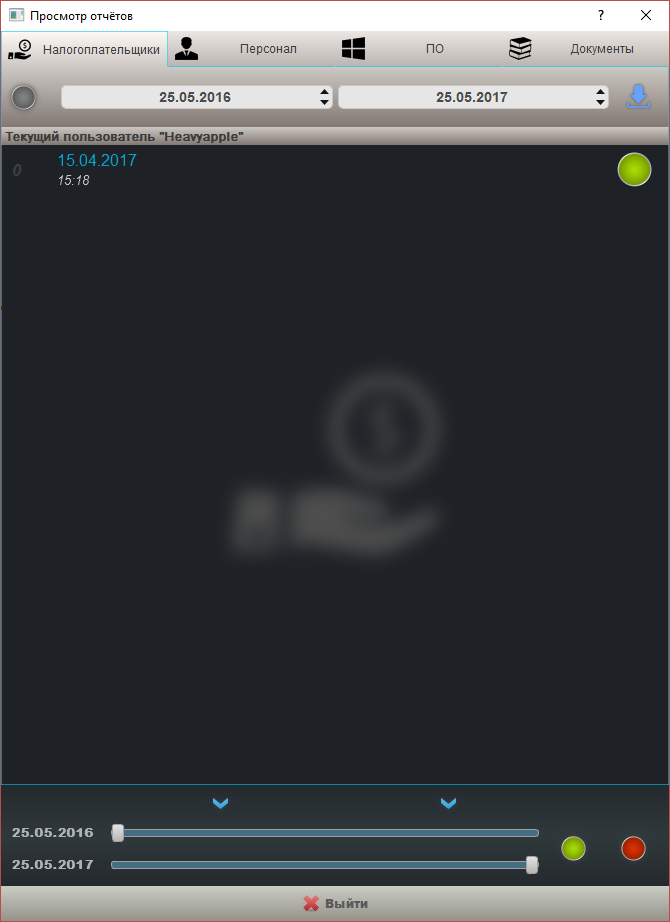


Рисунок 21-Запущенное клиентское приложение

На рисунках 22-23 представлено сообщение о результатах резервного копирование данных в виде:

* индикатор будет «гореть» зеленым, значит копирование состоялось успешно (см.рис.22);
* если индикатор «горит» красным светом, значит резервное копирование осуществлено с ошибками (см.рис.23).

Проверка осуществляется с помощью проверки контрольных битов архива, содержащих данные резервного копирования.

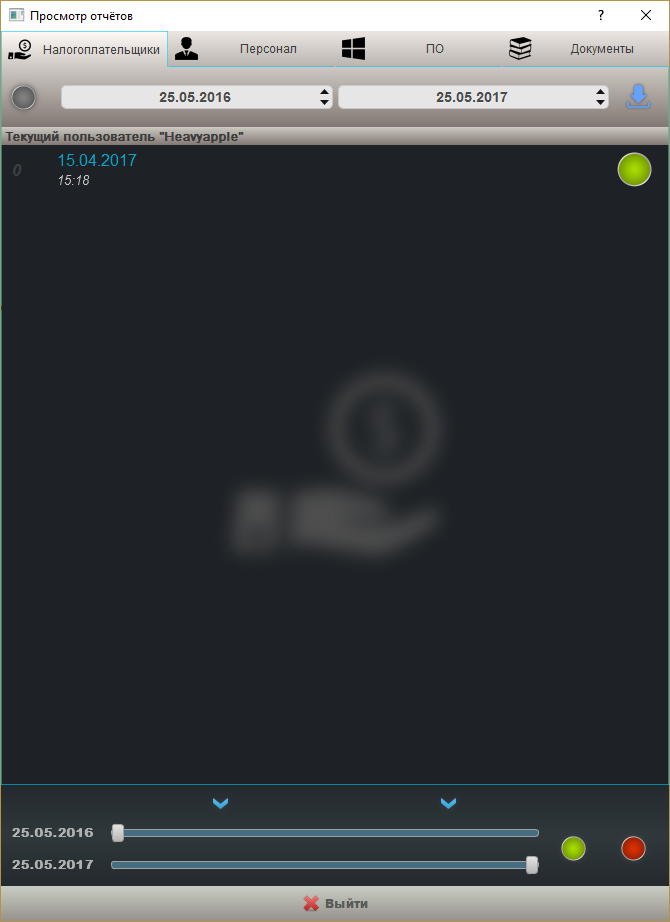


Рисунок 22-Успешное резервное копирование

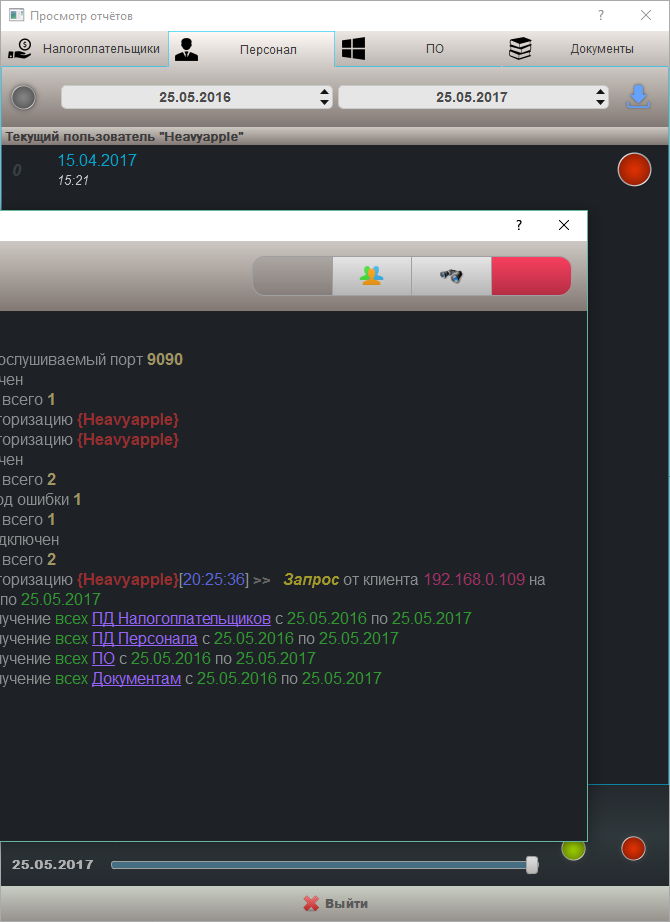


Рисунок 23-Резервное копирование не состоялось или завершено с ошибками.

После завершения работы с клиентским модулем необходимо выйти из программы с помощью кнопки в нижней части формы (см.рис.24).

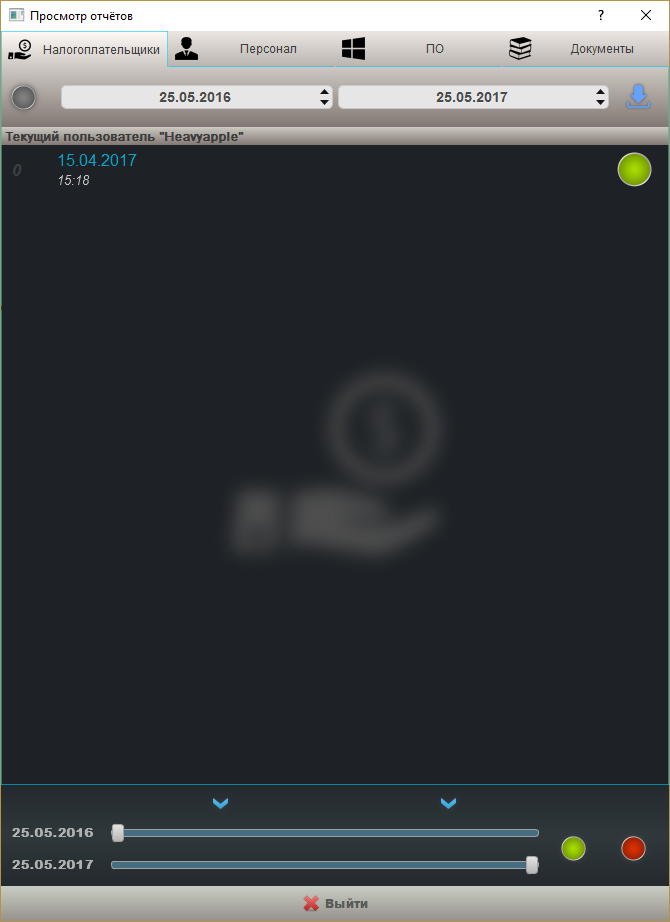


Рисунок 24-Выход из клиентского приложения

# 5 Испытание системы контроля системы резервного копирования данных

После проведения испытаний с использованием виртуальных сетей, а также непосредственного тестирования в системе Инспекции ФНС №8 по г. Калининграду, были получены результаты, говорящие об успешности разработанного программного продукта. Экспериментально были выяснены минимальные системные требования к аппаратно-программной платформе.

Чтобы осуществлять корректную работу с серверной частью приложения может быть использована следующая конфигурация аппаратно-программных средств:

* операционная система: Windows (с пакетом обновлений новее 2009 года), Unix/Linux и MacOS (поддерживающая Python версии 3.5 и выше);
* центральный процессор: x64/86, выпущенный позднее 2007 года;
* оперативная память: для Windows-100 Мб, для Unix/Linux и MacOS- 50 Мб;
* устройства ввода/вывода: компьютерная мышь или сенсорная панель, клавиатура.

Для работы с клиентской частью приложения необходима конфигурация оборудования аналогичная, но с уменьшенным количеством ОЗУ:

* операционная система: Windows (с пакетом обновлений новее 2009 года), Unix/Linux и MacOS (поддерживающая Python версии 3.5 и выше);
* центральный процессор x64/86: выпущенный позднее 2007 года
* оперативная память: для Windows-50 Мб, для Unix/Linux и MacOS-20 Мб;
* устройства ввода/вывода: компьютерная мышь или сенсорная панель, клавиатура.

Проведение испытаний системы контроля РК позволили провести ряд расчётов по вычислению времени, затраченного пользователем на проверку результатов резервного копирования данных.

Во время преддипломной практики был произведен ряд экспериментов, позволивших установить, что среднее время, затраченное на проверку успешности одного направления резервного копирования, равняется 7 минутам. В свою очередь количеств данных подвергаемых процессу РК равняется четырём[[18]](#footnote-18). Инспекция ФНС №8 по городу Калининграду в 2017 году функционирует в течении 247 дней и продолжительность одного рабочего дня составляет 8 часов;

Таблица 2-Исходные данные

|  |  |  |
| --- | --- | --- |
| Данные | Условное обозначение | Величина |
| Время, затраченное на проверку одного направления | Tв | 7 |
| Количество направлений | N | 4 |
| Количество рабочих дней в 2017 году | Y | 247 |

На основе исходных данных необходимо произвести вычисления, отображающие время, затраченное на контроль результатов РК системным администратором ИФНС №8 по городу Калининграду:

Tв \* N\*Y=6916 минут

6916 минут уходит на элементарную операцию проверки успешности резервного копирования. 6916 минут равны 115,27 часа, что в свою очередь составляет 14,4 восьмичасовых рабочих дня. Использование разработанной подсистемы РК данных сокращает время в 3.5 раза и принимает значение 4.1 рабочий день, что значительно увеличивает эффективность распределения времени. Для наглядности этот результат представлен на диаграмме 1.

Рисунок 25-Диаграмма, показывающая сравнение времени, которое было затрачено на проверку результатов резервного копирования

# Заключение

При написании дипломной выпускной квалификационной работы были изучены информационные системы, используемые в Инспекции ФНС №8 по г. Калининграду, а также рассмотрены и проанализированные их аналоги. Получены практические навыки по использованию систем резервного копирования, оформлению отчетной документации с учетом требований ГОСТ 2.501—2013, ГОСТ Р ИСО/МЭК 27031-2012.

Результатом выполнения работы является разработанная система, реализованная на базе клиент-серверной архитектуры. Ключевая цель разработки данной системы -контроль системы резервного копирования данных на основе серверного кластера Инспекции ФНС №8 по г. Калининграду. Разработанная система характеризуется следующими параметрами:

* быстродействие;
* удобство использования;
* отображение всех отчетов РК в одной графической оболочке.

В большинстве случаев достигнутые характеристики системы превышают первоначально задаваемые на этапе проектирования требования. В ходе проведенных испытаний, также было выяснено, что данная подсистема резервного копирования значительно сокращает расход времени на проверку журнала событий, и в связи с этим повышает эффективность работы сотрудников, эксплуатирующих информационные системы, что свидетельствует об успешном повышении экономической эффективности процесса эксплуатации всей существующей подсистемы контроля РК данных в Инспекции ФНС №8 по городу Калининграду.

Серверная часть программы работает фоновым процессом, не нагружая используемые системы, так как необходимые аппаратный ресурс минимален. Клиентское приложение способно работать на персональных компьютерах с очень низкой вычислительной мощности. В программе используется алгоритм аутентификации, позволяющий избежать несанкционированного доступа. Шифрование пароля проводится с использованием относительно нового стандарта ГОСТ Р 34.11-2012, принятым в 2013году, алгоритм хеширования Кeccak считается надежным.

А в связке с уже существующими средствами защиты и принятой внутренней политикой безопасности Инспекции ФНС №8 по городу Калининграду полностью соответствует внутренним стандартам организации.

Таким образом, в ходе выполнения работы была полностью достигнута поставленная в начале дипломного проектирования цель. Разработана система контроля резервного копирования данных, работающая в составе информационной системы Инспекции ФНС №8 по г. Калининграду. Данное система, реализованная как клиент-серверное приложение, прошла испытания в реальной эксплуатационной среде организации. В приложении представлен исходный код программы контроля РК данных, написанный на языке Python, версии 3.6, с использованием фреймворка PyQt5, версии 5.8, сетевая составляющая реализована собственными классами работы с сетью по протоколу TCP/IP.

# Cписок использованных источников

1. ГОСТ Р ИСО/МЭК 27001–2006. Методы и средства обеспечения безопасности. Системы менеджмента информационной безопасности [Текст]. – М.: Федеральное агентство по техническому регулированию и метрологии, 2008. – 31 с.
2. Грушо А. А., Тимонина Е. Е. Теоретические основы защиты информации [Текст]. – М.: Агентство «Яхтсмен», 1996. – 192 с.
3. Гайдамакин Н. А. Теоретические основы компьютерной безопасности: учебное пособие. – Екатеринбург: УГУ им. А.М. Горького, 2008. – 212 с.
4. Генри С. Уоррен, мл. Глава 5. Подсчет битов // Алгоритмические трюки для программистов = Hacker'sDelight [Текст]. – М.: Вильямс, 2007. – 288 с.
5. Морелос-Сарагоса Р. Искусство помехоустойчивого кодирования. Методы, алгоритмы, применение / Перевод с англ. В. Б. Афанасьева [Текст]. – М.: Техносфера, 2006. – 320 с.
6. Шаньгин В. Ф. Защита компьютерной информации. Эффективные методы и средства [Текст]. - М.: ДМК Пресс, 2010. - 544 с.
7. Хорев П.Б. Программно-аппаратная защита информации. Учебное пособие [Текст]. – М.: Форум, 2009. – 351 с.
8. ГОСТ Р 34.11. Информационная технология. Криптографическая защита информации. Функция хэширования. – Взамен ГОСТ Р 34.11–94. – М.: Стандартинформ, 2012. – 38 с.
9. Лебедев П. Сравнение старого и нового стандартов РФ на криптографическую хэш-функцию на ЦП и графических процессорах Nvidia [Электронный ресурс] URL: https://www.streebog.net/ru/
10. ГОСТ P 34.10. Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи. // М.: Госстандарт России. – 2001. – 34 с.
11. Кнут Д. Е. Искусство программирования. –М.: «Вильямс», 2007. – 960 с.
12. Уоррен Г. Алгоритмические трюки для программистов. –М.: «Вильямс», 2013. – 512 с.
13. ГОСТ Р 19.701. Единая схема программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения, 1990. – 27 с.

# Приложение А

**Клиент**  
Модуль TCP/IP-клиент

*from PyQt5.QtCore import QByteArray*

*from PyQt5.QtCore import QDataStream*

*from PyQt5.QtCore import QIODevice*

*from PyQt5.QtCore import QObject*

*from PyQt5.QtCore import pyqtSignal*

*from PyQt5.QtNetwork import QTcpSocket*

*from API.EventsConverter import EventsConverter*

*from API.Types.Query.Report import DS\_ReportQuery*

*from API.Types.Query.Authentication import DS\_AuthenticationQuery*

*from Source.namespace import DPL*

*PORT = DPL.ConnectionAttributes.port*

*UINT32 = DPL.ConnectionAttributes.UINT32*

*QTDSVERSION = DPL.ConnectionAttributes.QtDataStreamVersion*

*SHOW\_DEBUG = True*

*class Client(QObject):*

*“*

*Данный класс отвечает за реализацию*

*нетворк клиента. Обеспечивает соединение с*

*сервером и отправку запросов на сервер*

*”*

*# сигналы, генерируемые во время*

*# исполнения следующих функций:*

*# \_readServerData()*

*# \_processServerData()*

*authenticationReceived = pyqtSignal(str) # result*

*reportReceived = pyqtSignal(tuple) # (report type, DS\_Events.toString)*

*errors = pyqtSignal(int)*

*connectedToServer = pyqtSignal(str) # server ip*

*def \_\_init\_\_(self, parent=None):*

*super(Client, self).\_\_init\_\_()*

*if parent: self.setParent(parent)*

*self.\_\_init\_attributtes()*

*self.\_\_init\_connects()*

*def \_\_init\_attributtes(self):*

*"""*

*Инициализация атрибутов класса*

*"""*

*self.\_\_socket = QTcpSocket()*

*self.\_\_querybytes = None*

*self.\_\_converter = EventsConverter()*

*self.\_\_serverIP = ''*

*self.\_\_serverPort = -1*

*def \_\_init\_connects(self):*

*"""*

*Инициализация коннектов класса*

*"""*

*self.\_\_socket.readyRead.connect(self.\_processServerData)*

*self.\_\_socket.error.connect(self.\_\_emit\_errors)*

*@property # сложный геттер*

*def serverIP(self):*

*return self.\_\_serverIP*

*@property*

*def serverPort(self):*

*return self.\_\_serverPort*

*def \_\_DEBUG(self, \*args):*

*if SHOW\_DEBUG:*

*print(\*args)*

*def \_\_emit\_errors(self, error):*

*e = int(error)*

*self.errors.emit(e)*

*def \_readServerData(self):*

*"""*

*Считывает данные сервера*

*"""*

*self.\_\_DEBUG('Client "{0}" started receiving server data'.format(self.\_\_socket.peerName()))*

*stream = QDataStream(self.\_\_socket)*

*stream.setVersion(QTDSVERSION)*

*data = ''*

*while True:*

*if self.\_\_socket.nextBlockSize == 0:*

*if self.\_\_socket.bytesAvailable() < UINT32:*

*break*

*self.\_\_socket.nextBlockSize = stream.readUInt32()*

*if self.\_\_socket.bytesAvailable() < self.\_\_socket.nextBlockSize:*

*break*

*partion = stream.readQString()*

*self.\_\_socket.nextBlockSize = 0*

*data += partion*

*self.\_\_DEBUG('Client finished receiving server data')*

*return data*

*# if SHOW\_DEBUG:*

*# print(' Events string: {0}'.format(events\_string))*

*# if SHOW\_DEBUG:*

*# print(' Data on "{0}" request type'.format(type))*

*# print(' Events converted to list of DS\_Events() by EventsConverter()')*

*# print(' Events count: {0}'.format(str(len(events))))*

*# if SHOW\_DEBUG:*

*# print(' Receiving finished.\n')*

*def \_processServerData(self):*

*# обрабатывает данные полученные от сервера*

*try:*

*self.\_\_DEBUG('Processing started')*

*data = self.\_readServerData()*

*if data.count('>') == 1: # если имеются данные*

*type, auth\_result = data.split('>')*

*self.authenticationReceived.emit(auth\_result)*

*elif data.count('>') == 2:*

*type, report, events = data.split('>')*

*self.\_\_converter.setEventsString(events)*

*events = self.\_\_converter.events*

*self.reportReceived.emit((report, events))*

*self.\_\_DEBUG('Processing finished')*

*except:*

*self.\_\_DEBUG('Processing was crashed')*

*def sendReportRequest(self, query=DS\_ReportQuery()):*

*"""*

*Отправляет запрос на авторизацию на сервере*

*"""*

*self.\_\_DEBUG('Client "{0}" started send "REPORT" request'.format(self.\_\_socket.peerName()))*

*self.\_\_DEBUG(' Query: {0}'.format(query.toString))*

*# преобразование данных в поток байт*

*self.\_\_querybytes = QByteArray()*

*stream = QDataStream(self.\_\_querybytes, QIODevice.WriteOnly)*

*stream.setVersion(QTDSVERSION)*

*stream.writeUInt32(0)*

*stream.writeQString(query.toString)*

*stream.device().seek(0)*

*stream.writeUInt32(self.\_\_querybytes.size() - UINT32)*

*self.\_\_socket.write(self.\_\_querybytes)*

*self.\_\_socket.nextBlockSize = 0*

*self.\_\_querybytes = None*

*#*

*self.\_\_DEBUG(' "REPORT" requesting is finished.\n')*

*def sendAuthenticationRequest(self, query=DS\_AuthenticationQuery()):*

*self.\_\_DEBUG('Client "{0}" started send "AUTHENTICATION" request'.format(self.\_\_socket.peerName()))*

*self.\_\_DEBUG(' Query: {0}'.format(query.toString))*

*#*

*self.\_\_querybytes = QByteArray()*

*stream = QDataStream(self.\_\_querybytes, QIODevice.WriteOnly)*

*stream.setVersion(QTDSVERSION)*

*stream.writeUInt32(0)*

*stream.writeQString(query.toString)*

*stream.device().seek(0)*

*stream.writeUInt32(self.\_\_querybytes.size() - UINT32)*

*self.\_\_socket.write(self.\_\_querybytes)*

*self.\_\_socket.nextBlockSize = 0*

*self.\_\_querybytes = None*

*#*

*self.\_\_DEBUG(' "AUTHENTICATION" requesting is finished.\n')*

*def connectToServer(self, ip, port=DPL.ConnectionAttributes.port):*

*self.\_\_serverIP = ip*

*self.\_\_serverPort = port*

*# подключение к серверу*

*if self.\_\_socket.isOpen(): # если ещё не подключен*

*self.\_\_socket.abort()*

*self.\_\_socket.connectToHost(ip, port)*

*self.\_\_DEBUG('The client tries to connect to the "{0}" "{1}" server'.format(ip, str(port)))*

# Приложение Б

**Сервер**

Модуль TCP/IP-сервера

*import socket*

*from PyQt5.QtCore import QByteArray*

*from PyQt5.QtCore import QDataStream*

*from PyQt5.QtCore import QIODevice*

*from PyQt5.QtCore import QObject*

*from PyQt5.QtCore import pyqtSignal*

*from PyQt5.QtNetwork import QHostAddress*

*from PyQt5.QtNetwork import QTcpServer*

*from API.Network.Server.ClientRequestProcessor import ClientRequestProcessor*

*from Source.namespace import DPL*

*PORT = DPL.ConnectionAttributes.port*

*UINT32 = DPL.ConnectionAttributes.UINT32*

*QTDSVERSION = DPL.ConnectionAttributes.QtDataStreamVersion*

*SHOW\_DEBUG = True*

*class Server(QObject):*

*clientConnected = pyqtSignal(str)*

*clientAuthenticationRequested = pyqtSignal(str)*

*clientReportRequested = pyqtSignal(str)*

*clientError = pyqtSignal(str, int) # ip error*

*clientCountChanged = pyqtSignal(int)*

*clientRemoved = pyqtSignal(str) # ip*

*serverStarted = pyqtSignal(str)*

*serverStoped = pyqtSignal(str)*

*def \_\_init\_\_(self, db):*

*super(Server, self).\_\_init\_\_()*

*self.\_clients = []*

*self.\_server = QTcpServer(self)*

*self.\_server.newConnection.connect(self.\_addClient)*

*self.\_\_processor = ClientRequestProcessor(db, self)*

*def \_\_DEBUG(self, \*args):*

*if SHOW\_DEBUG:*

*print(\*args)*

*@property*

*def loacalHost(self):*

*return socket.gethostbyname(socket.gethostname())*

*@property*

*def port(self):*

*return self.\_\_port*

*def setPort(self, port=DPL.ConnectionAttributes.port):*

*self.\_\_port = port*

*def start(self):*

*try:*

*self.\_server.listen(QHostAddress(QHostAddress.Any), PORT)*

*info = 'Сервер {0} был запущен. Прослушиваемый порт {1}\n'.format(self.loacalHost, self.\_server.serverPort())*

*print(info)*

*self.serverStarted.emit(info)*

*except:*

*pass*

*def stop(self):*

*try:*

*self.\_server.close()*

*for client in self.\_clients:*

*client.deleteLater()*

*print('Сервер был остановлен')*

*self.serverStoped.emit('Сервер был остановлен')*

*except:*

*pass*

*def \_addClient(self):*

*try:*

*self.\_\_DEBUG('Started client connection:')*

*client = self.\_server.nextPendingConnection()*

*client.nextBlockSize = 0*

*client.readyRead.connect(self.\_processClientData)*

*client.disconnected.connect(self.\_removeClient)*

*client.error.connect(self.\_clientError)*

*#*

*ip = client.localAddress().toString()[7:]*

*port = client.localPort()*

*name = socket.getnameinfo((ip, port), 0)[0]*

*client.setObjectName(name)*

*#*

*self.\_clients.append(client)*

*self.clientConnected.emit('Клиент "{0}" {1} подключен '.format(name, ip))*

*self.clientCountChanged.emit(len(self.\_clients))*

*#*

*self.\_\_DEBUG(' IP: {0}'.format(ip))*

*self.\_\_DEBUG(' port: {0}'.format(port))*

*self.\_\_DEBUG(' name: {0}'.format(name))*

*self.\_\_DEBUG('Connection is finished.\n')*

*except:*

*pass*

*def \_readClientData(self, client):*

*try:*

*if client.bytesAvailable() > 0:*

*stream = QDataStream(client)*

*stream.setVersion(QTDSVERSION)*

*#*

*if client.nextBlockSize == 0:*

*if client.bytesAvailable() < UINT32:*

*return ''*

*client.nextBlockSize = stream.readUInt32()*

*if client.bytesAvailable() < client.nextBlockSize:*

*return ''*

*#*

*data = stream.readQString()*

*return data*

*return ''*

*except:*

*pass*

*def \_processClientData(self):*

*try:*

*self.\_\_DEBUG('Started client data processing:')*

*client = self.sender()*

*data = self.\_readClientData(client)*

*self.\_\_DEBUG(' Data: {0}'.format(data))*

*#*

*self.\_\_processor.setQuery(data)*

*if self.\_\_processor.isReport:*

*self.\_\_DEBUG( 'Request is "REPORT"')*

*query = self.\_\_processor.reportQuery*

*events = self.\_\_processor.events()*

*self.clientReportRequested.emit(data)*

*self.\_sendDataToClient(query.type + '>' + query.report + '>' + events, client)*

*elif self.\_\_processor.isAuthentication:*

*self.\_\_DEBUG( 'Request is "AUTHENTICATION"')*

*query = self.\_\_processor.authenticationQuery*

*hasuser = str(self.\_\_processor.hasUser())*

*self.clientAuthenticationRequested.emit(data)*

*self.\_sendDataToClient(query.type + '>' + hasuser, client)*

*#*

*self.\_\_DEBUG('Client data processing is finished.\n')*

*except:*

*pass*

*def \_sendDataToClient(self, data, client):*

*try:*

*self.\_\_DEBUG(' Started send data to client:')*

*client.nextBlockSize = 0*

*array = QByteArray()*

*stream = QDataStream(array, QIODevice.WriteOnly)*

*stream.setVersion(QTDSVERSION)*

*stream.writeUInt32(0)*

*stream.writeQString(data)*

*stream.device().seek(0)*

*stream.writeUInt32(array.size() - UINT32)*

*client.write(array)*

*client.nextBlockSize = 0*

*self.\_\_DEBUG(' Sending data to client is finished.')*

*except:*

*pass*

*def \_removeClient(self):*

*try:*

*client = self.sender()*

*# print(' REMOVE ')*

*ip = client.localAddress().toString()[7:]*

*self.\_clients.remove(client)*

*self.clientRemoved.emit(ip)*

*self.clientCountChanged.emit(len(self.\_clients))*

*except:*

*pass*

*def \_clientError(self, error):*

*try:*

*# print(error)*

*# print(' ERROR ')*

*client = self.sender()*

*ip = client.localAddress().toString()[7:]*

*self.clientError.emit(ip, error)*

*except:*

*pass*

Модуль обработки запросов клиента

*from PyQt5.QtCore import QObject*

*from API.Database.Models.ReportModel import DBModel\_Report*

*from API.Database.Models.Users import DBModel\_Users*

*from API.EventsConverter import EventsConverter*

*from API.Types.Query.Report import DS\_ReportQuery*

*from API.Types.Query.Authentication import DS\_AuthenticationQuery*

*from Source.namespace import DPL*

*SHOW\_DEBUG = True*

*class ClientRequestProcessor(QObject):*

*“*

*Данный класс обеспечивает обработку данных,*

*полученных от клиента. Позволяет определить*

*тип запроса, обработать запрос и собрать*

*необходимую информацию.*

*”*

*def \_\_init\_\_(self, db, parent=None):*

*super(ClientRequestProcessor, self).\_\_init\_\_()*

*if parent: self.setParent(parent)*

*self.\_\_qauth = DS\_AuthenticationQuery()*

*self.\_\_qrep = DS\_ReportQuery()*

*self.\_\_reports = DBModel\_Report(DBModel\_Report.Table\_Staff, db, self)*

*self.\_\_users = DBModel\_Users(db, self)*

*self.\_\_converter = EventsConverter()*

*self.\_\_data = ''*

*def \_\_DEBUG(self, text):*

*if SHOW\_DEBUG:*

*print(text)*

*@property*

*def isAuthentication(self):*

*return self.\_\_qauth.isRightString(self.\_\_data)*

*@property*

*def isReport(self):*

*return self.\_\_qrep.isRightString(self.\_\_data)*

*@property*

*def authenticationQuery(self):*

*return self.\_\_qauth*

*@property*

*def reportQuery(self):*

*return self.\_\_qrep*

*def hasUser(self):*

*log = self.\_\_qauth.login*

*psw = self.\_\_qauth.password*

*if self.\_\_users.hasUser(log, psw):*

*return 1*

*else:*

*return 0*

*def setQuery(self, data=''):*

*self.\_\_qrep.clear()*

*self.\_\_qauth.clear()*

*self.\_\_data = data*

*if self.isAuthentication: self.\_\_qauth.setString(data)*

*elif self.isReport: self.\_\_qrep.setString(data)*

*def events(self):*

*self.\_\_DEBUG(' Started processing of request:')*

*query = self.reportQuery*

*self.\_\_DEBUG(' Query: {0}'.format(query.toString))*

*events = self.\_\_reports.recordsRange(query.min, query.max, query.state, query.report)*

*#*

*if SHOW\_DEBUG:*

*print(' Started events converting to string:')*

*self.\_\_converter.setEvents(events)*

*events\_str = self.\_\_converter.encodeToString()*

*if SHOW\_DEBUG:*

*print(' Events count = ', len(events))*

*# print(' Events string = ', events\_str)*

*print(' Events converting finished.')*

*print(' Request processing is finished.')*

*return events\_str*

Модуль инспекции резервного копирования

*import os*

*from PyQt5.QtCore import QFileInfo*

*from PyQt5.QtCore import QObject*

*from PyQt5.QtCore import QTimer*

*from PyQt5.QtCore import pyqtSignal*

*from API.Database.Models.ReportModel import DBModel\_Report*

*from API.Types.Date import Date*

*from API.Types.Event import DS\_Event*

*from API.Types.Time import Time*

*from Source.namespace import DPL*

*class ReportWatcher(QObject):*

*“*

*Данный класс обеспечивает отслеживание за*

*результатом выполнения бекапа. Если бекап не*

*выполнен в назначенное время генерируется сигнал*

*оповещающий, аналогично и для успешного бекапа*

*.*

*”*

*reportAccepted = pyqtSignal(str, Time) # report type, date, time*

*reportRejected = pyqtSignal(str, Time) # report type, date, time*

*scanningStarted = pyqtSignal()*

*scanningStoped = pyqtSignal()*

*scanningParametersChanged = pyqtSignal(Time, Time, str)*

*started = pyqtSignal()*

*stoped = pyqtSignal()*

*def \_\_init\_\_(self, db, parent=None):*

*super(ReportWatcher, self).\_\_init\_\_()*

*if parent: self.setParent(parent)*

*self.\_\_init\_attributes(db)*

*self.\_\_init\_connects()*

*def \_\_init\_attributes(self, db):*

*self.\_\_model = DBModel\_Report(DBModel\_Report.Table\_Documents, db, self)*

*self.\_\_iterMilisec = 1000*

*self.\_\_timer = QTimer(self)*

*self.\_\_timerange = [Time.currentTime(), Time.currentTime()]*

*self.\_\_scanningStarted\_emited = False*

*self.\_\_storage = ''*

*self.\_\_reportStates = {*

*DPL.ReportQueryTypes.Staff: False,*

*DPL.ReportQueryTypes.Taxpayers: False,*

*DPL.ReportQueryTypes.Documents: False,*

*DPL.ReportQueryTypes.Softwares: False*

*}*

*self.\_\_reportPatterns = {*

*DPL.ReportQueryTypes.Staff: 'ПДП', # pattern, dir*

*DPL.ReportQueryTypes.Taxpayers: 'ПДН', # pattern, dir*

*DPL.ReportQueryTypes.Documents: 'Д', # pattern, dir*

*DPL.ReportQueryTypes.Softwares: 'ПО' # pattern, dir*

*}*

*def \_\_init\_connects(self):*

*self.\_\_timer.timeout.connect(self.checkReportsLoop)*

*def \_\_tool\_addEvent(self, report, event):*

*self.\_\_model.addEventToSpecificTable(report, event)*

*def setStorage(self, path):*

*self.\_\_storage = path*

*def setPattern(self, report, pattr):*

*self.\_\_reportPatterns[report] = pattr*

*def setParameters(self, min, max, store):*

*self.stop()*

*self.\_\_timerange = [min, max]*

*self.\_\_storage = store*

*self.scanningParametersChanged.emit(min, max, store)*

*self.start()*

*def start(self):*

*self.\_\_timer.start(self.\_\_iterMilisec)*

*self.started.emit()*

*self.\_\_scanningStarted\_emited = False*

*def stop(self):*

*self.\_\_timer.stop()*

*self.stoped.emit()*

*def checkReportsLoop(self):*

*try:*

*if os.path.exists(self.\_\_storage):*

*ctime = Time.currentTime()*

*cdate = Date.currentDate()*

*#*

*if ctime >= self.\_\_timerange[0] and ctime <= self.\_\_timerange[1]:*

*if not self.\_\_scanningStarted\_emited:*

*self.scanningStarted.emit()*

*self.\_\_scanningStarted\_emited = True*

*for report in self.\_\_reportPatterns:*

*pattr = self.\_\_reportPatterns[report]*

*for fname in os.listdir(self.\_\_storage):*

*fpath = os.path.join(self.\_\_storage, fname)*

*date, title = fname.split('\_')*

*title = title.split('.')[0]*

*date = date.replace('-', '.')*

*date = Date().fromString(date)*

*if date == cdate:*

*if title == pattr:*

*info = QFileInfo(fpath)*

*t = Time()*

*t.setQTime(info.created().time())*

*if self.\_\_reportStates[report] == False:*

*event = DS\_Event()*

*event.setTime(t)*

*event.setDate(date)*

*event.setState(1)*

*self.\_\_tool\_addEvent(report, event)*

*self.reportAccepted.emit(report, t)*

*self.\_\_reportStates[report] = True*

*break*

*if self.\_\_reportStates[report] == False:*

*self.\_\_reportStates[report] = False*

*#*

*if self.\_\_scanningStarted\_emited:*

*if ctime > self.\_\_timerange[1]:*

*for report in self.\_\_reportStates:*

*result = self.\_\_reportStates[report]*

*if not result:*

*t = Time()*

*t.setQTime(ctime)*

*event = DS\_Event()*

*event.setTime(t)*

*event.setDate(Date().setCurrentDate())*

*event.setState(0)*

*self.\_\_tool\_addEvent(report, event)*

*self.reportRejected.emit(report, self.\_\_timerange[1])*

*self.scanningStoped.emit()*

*self.\_\_timer.stop()*

*# print(self.\_\_reportStates)*

*except:*

*print('ReportWatcher: checking is failed')*

*if \_\_name\_\_ == '\_\_main\_\_':*

*import sys*

*from PyQt5.QtWidgets import QApplication*

*from API.Database.database import Database*

*app = QApplication([])*

*db = Database()*

*s = Server(db)*

*s.start()*

*sys.exit(app.exec\_())*

*if \_\_name\_\_ == '\_\_main\_\_':*

*import sys*

*from PyQt5.QtWidgets import \**

*from API.Database.database import Database*

*app = QApplication([])*

*db = Database()*

*view = QTableView()*

*model = DBModel\_Report(DBModel\_Report.Table\_Documents, db)*

*# model.clearTable()*

*# model.testFilling()*

*view.setModel(model)*

*view.show()*

*sys.exit(app.exec\_())*

# Приложение В

**Модуль по работе с базой данных**

*from PyQt5.QtSql import QSqlTableModel*

*from API.Types.Date import Date*

*from API.Types.Event import DS\_Event*

*from API.Types.Time import Time*

*from Source.namespace import DPL*

*class DBModel\_Report(QSqlTableModel):*

*"""*

*Данный класс отвечает за потоки*

*ввода и вывода информации БД.*

*Класс-родитель - "QSqlTableModel" имеет*

*готовые методы для работы с SQL БД, что*

*позволяет избавиться от SQL-запросов.*

*"""*

*# НЕЙМСПЕЙС имена полей(столбцов) таблицы*

*Table\_Softwares = 'Softwares' # таблица софта*

*Table\_Staff = 'Staff' # таблица сотрудников*

*Table\_Taxpayers = 'Taxpayers' # таблица налогоплательщиков*

*Table\_Documents = 'Documents' # таьлица документов*

*def \_\_init\_\_(self, table, db, parent=None):*

*"""*

*:param table: <DBAccessModel.TableName> : имя таблицы из неймспейса данного класса*

*:param db: <QSqlDataBase> : подключенная к приложению БД*

*"""*

*super(DBModel\_Report, self).\_\_init\_\_(db=db)*

*if parent: self.setParent(parent)*

*self.setEditStrategy(self.OnRowChange)*

*self.setTable(table)*

*def record(self, row=0):*

*# возвращает запись*

*record = self.record(row)*

*date = record.value('Date')*

*time = record.value('Time')*

*state = record.value('State')*

*# установка параметров*

*# полученной записи(отчета)*

*event = DS\_Event()*

*event.setDateString(date)*

*event.setTimeString(time)*

*event.setState(int(state))*

*return event*

*def records(self):*

*# возвращает список записей*

*events = []*

*for row in range(self.rowCount()):*

*events.append(self.event(row))*

*return events*

*def addRecord(self, event):*

*# добавляет событие в БД*

*record = self.record(0)*

*record.clearValues()*

*# установка параметров записи(отчета)*

*record.setValue('Date', event.date.toString)*

*record.setValue('Time', event.time.toString)*

*record.setValue("State", str(event.state))*

*# добавление в БД*

*res = self.insertRecord(self.rowCount(), record)*

*self.select()*

*return res*

*def addEventToSpecificTable(self, report=DPL.ReportQueryTypes.Documents, event=DS\_Event):*

*# добавляет запись в определенную таблицу*

*# установка рабочей таблицы*

*if report == DPL.ReportQueryTypes.Documents:*

*self.setTable(self.Table\_Documents)*

*elif report == DPL.ReportQueryTypes.Staff:*

*self.setTable(self.Table\_Staff)*

*elif report == DPL.ReportQueryTypes.Softwares:*

*self.setTable(self.Table\_Softwares)*

*elif report == DPL.ReportQueryTypes.Taxpayers:*

*self.setTable(self.Table\_Taxpayers)*

*self.select()*

*# добавление записи*

*if event not in self.events():*

*self.addRecord(event)*

*def removeRecord(self, row=-1):*

*# удаляет запись сз БД*

*if row < 0:*

*row = self.rowCount() - 1*

*res = self.removeRow(row)*

*self.select() #*

*return res*

*def setTable(self, table):*

*# устанавливает рабочую таблицу*

*super(DBModel\_Report, self).setTable(table)*

*self.select()*

*def setHeaders(self, headers=[]):*

*for i in range(0, len(headers)):*

*self.setHeaderData(i, 1, str(headers[i]))*

*def recordsRange(self, start, end, state, report=None):*

*# возврящает список отчетов в*

*# определенный период*

*if not report:*

*records = []*

*self.select()*

*for row in range(self.rowCount()):*

*event = self.event(row)*

*if event.isState(state):*

*if event.inRange(start, end):*

*records.append(event)*

*return records*

*else:*

*current\_table = self.tableName()*

*self.setTable(self.table(report))*

*self.select()*

*records = []*

*self.select()*

*#*

*for row in range(self.rowCount()):*

*event = self.event(row)*

*if event.isState(state):*

*if event.inRange(start, end):*

*records.append(event)*

*self.setTable(current\_table)*

*self.select()*

*return records*

*def table(self, report=DPL.ReportQueryTypes.Documents):*

*# возвращает таблицу БД в зависимости от отчета*

*if report == DPL.ReportQueryTypes.Documents:*

*return self.Table\_Documents*

*elif report == DPL.ReportQueryTypes.Softwares:*

*return self.Table\_Softwares*

*elif report == DPL.ReportQueryTypes.Taxpayers:*

*return self.Table\_Taxpayers*

*elif report == DPL.ReportQueryTypes.Staff:*

*return self.Table\_Staff*

*else:*

*return self.Table\_Documents*

*if \_\_name\_\_ == '\_\_main\_\_':*

*import sys*

*from PyQt5.QtWidgets import \**

*from API.Database.database import Database*

*app = QApplication([])*

*db = Database()*

*view = QTableView()*

*model = DBModel\_Report(DBModel\_Report.Table\_Documents, db)*

*# model.clearTable()*

*# model.testFilling()*

*view.setModel(model)*

*view.show()*

*sys.exit(app.exec\_())*

*from PyQt5.QtWidgets import QApplication*

*from API.Database.database import Database*

*app = QApplication([])*

*db = Database()*

*s = Server(db)*

*s.start()*

*sys.exit(app.exec\_())*

*if \_\_name\_\_ == '\_\_main\_\_':*

*import sys*

*from PyQt5.QtWidgets import \**

*from API.Database.database import Database*

*app = QApplication([])*

*db = Database()*

*view = QTableView()*

*model = DBModel\_Report(DBModel\_Report.Table\_Documents, db)*

*# model.clearTable()*

*# model.testFilling()*

*view.setModel(model)*

*view.show()*

*sys.exit(app.exec\_())*

1. Которое было проведено компанией Iomega в 2015 году [↑](#footnote-ref-1)
2. Резервное копирование данных [↑](#footnote-ref-2)
3. Французская ассоциация информационной безопасности [↑](#footnote-ref-3)
4. Например, обновление реестра. [↑](#footnote-ref-4)
5. Mobile Rack [↑](#footnote-ref-5)
6. Redundant Array of Inexpensive Disks - избыточный массив недорогих дисков [↑](#footnote-ref-6)
7. NAND SSD [↑](#footnote-ref-7)
8. Несмотря на сравнительно скромные заявленные скорости на чтение и запись, SSD данной серии оказались лучшими по результатам независимого тестирования в лаборатории НИКС.  [↑](#footnote-ref-8)
9. Network Attached Storage, сетевая система хранения данных, сетевое хранилище [↑](#footnote-ref-9)
10. Storage Area Network, Сеть хранения данных [↑](#footnote-ref-10)
11. Logical Unit Number, адрес диска (в ранних версиях) или дискового устройства в сетях хранения [↑](#footnote-ref-11)
12. Direct-Attached Storage — система хранения данных с прямым подключением [↑](#footnote-ref-12)
13. Например, общая папка или FTP-сервер. [↑](#footnote-ref-13)
14. Именно по такому принципу работает большинство популярных систем резервного копирования, таких как Symantec Backup Exec, CA Bright Store ARCServe Backup, Bacula [↑](#footnote-ref-14)
15. Например, пожар в здании, прорыв батареи центрального отопления в серверной комнате, кража техники и комплектующих [↑](#footnote-ref-15)
16. Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания. [↑](#footnote-ref-16)
17. Activity diagram- UML-диаграмма, на которой показаны действия, состояния которых описано на диаграмме состояний. [↑](#footnote-ref-17)
18. Персональные данные налогоплательщиков и сотрудников налоговой инспекции, программного обеспечения, документы, создаваемые и используемые в ходе работы налоговой инспекции. [↑](#footnote-ref-18)