СОДЕРЖАНИЕ

ВВЕДЕНИЕ

По данным компании Netmarketshare на конец марта 2021 года операционная система (ОС) Microsoft Windows была установлена на 89.21% рабочих станций, из которых 74,9% установок приходилось на Windows 10. Таким образом, Windows до сих пор остается самой распространенной клиентской корпоративной операционной системой.

По этой причине было принято решение сосредоточиться на информационной системе только для корпоративного сегмента в среде Windows. ОС Windows имеет ряд преимуществ, таких как поддержка большого количества программ, в т. ч. профессиональных, совместимость с большей частью актуального оборудования, а также широкие возможности для централизованного управления ОС, что делает ее самой популярной ОС в том числе для предприятий.

Целью разработки информационной системы (ИС) является автоматизация процесса инвентаризации аппаратного и программного обеспечения рабочих станций доменной сети предприятия. Одним из способов сбора подобной информации является ручная обработка данных и внесение информации в электронную таблицу или базу данных, но этот способ является очень трудозатратным и могущим привести к ошибкам.

Компанией Microsoft разработаны определенные технологии, позволяющие автоматизировать управление компьютерной инфраструктурой предприятия, такие как служба каталогов Active Directory (AD) и технология WMI – Windows Management Instrumentation. Эти технологии позволяют иметь единую групповую и локальную политики безопасности сети, ограничение времени работы учетных записей пользователей и прочие параметры, а также при помощи различных инструментов получать всю необходимую информацию как обо всех компьютерах сети, так и о конкретных рабочих станциях.

Например, доступ к данным AD и WMI возможно получить при помощи PowerShell – мощного средства автоматизации, состоящего из оболочки с

интерфейсом командной строки и сопутствующего языка сценариев. Но PowerShell не обладает удобным графическим интерфейсом, поэтому было принято решение разработать информационную систему на базе технологий Microsoft.

Разрабатываемая система предназначена для автоматизации и упрощения доступа к информации о компьютерной инфраструктуре предприятия, построенной на базе ОС Windows и доменов Active Directory, и должна обеспечивать получение информации при наличии соответствующего уровня доступа и представление ее в удобном для пользователя графическом виде.

Актуальность разработки информационной системы связана с большим количеством (около 1500) персональных компьютеров, используемых на предприятии, необходимостью их учета и быстрого доступа к информации о конфигурации конкретного компьютера.

1. ХАРАКТЕРИСТИКА ПРЕДМЕТНОЙ ОБЛАСТИ И ОБОСНОВАНИЕ АКТУАЛЬНОСТИ

1.1. Сущность и цели инвентаризации компьютеров

«Компьютерная техника давно стала неотъемлемой и важной частью в функционировании предприятия. Однако, периодически перед компанией встают вопросы о правильном учете и инвентаризации этой категории имущества.

Функционирующий персональный компьютер (ПК) – это технически сложный объект, состоящий из системного блока, источника питания, монитора, клавиатуры и мыши, т.е. он представляет собой комплект оборудования.

Но зачастую составляющие ПК закупаются по отдельности в разное время и в разном количестве. Это – комплектующие, они легко монтируются, имеют разные сроки полезного использования, могут заменяться и перемещаться с одного рабочего места на другое, т. е. комплектующие части компьютера не образуют единого объекта и не формируют комплекс конструктивно соединенных предметов. В этом случае можно учитывать отдельные мониторы и системные блоки в составе основных средств (ОС), а комплектующие (например, источники бесперебойного питания, клавиатуры, мыши, жесткие диски) – в составе материально-производственных запасов (МПЗ). При передаче ОС в эксплуатацию (например, при замене вышедших из строя) их учитывают в качестве самостоятельного объекта со своим сроком использования. При этом важно своевременно оформлять соответствующие документы, чтобы исключить случаи, когда комплектующие одновременно числятся в составе МПЗ и ОС.

Компьютерная техника, как и остальные категории ОС, подлежит инвентаризации. Эта процедура, хоть и отличается от проверок состояния других активов компании, но также регулируется законом № 402-ФЗ от 06.12.2011 «О бухучете», Методическими рекомендациями по проведению инвентаризации (Приказ Минфина РФ от 13.06.1995 № 49) и Положением по ведению бухучета (Приказ Минфина России от 29.07.1998 № 34н).

Как правило, проверку проводят раз в год накануне формирования финансовой отчетности компании. Однако предприятие вправе установить и дополнительные проверки. Например, крупные организации инвентаризируют оргтехнику ежеквартально, а также практикуют проведение внеплановых инвентаризаций после модернизации или ремонта. Это целесообразно, поскольку позволяет компании не только контролировать наличие техники, но и выявлять бездействующие объекты, оптимизировать расходы по закупке компьютеров и своевременно ремонтировать или списывать износившиеся и вышедшие из строя объекты.

Если корпоративная оргтехника размещается в различных точках и насчитывает более 30–50 единиц, процесс ее настройки, учета и контроля становится весьма затруднительным и требует автоматизации этого процесса.

С развитием информационных технологий (ИТ) заметно изменились алгоритмы проведения инвентаризации компьютерной техники. При использовании специальных программ, организующих удаленный доступ ко всем узлам локальной сети предприятия, этапы проверки автоматизируются, а время, затраченное на эту процедуру, сводится к минимуму, не предполагая высвобождение работников для физической проверки каждого объекта.

Любая программа инвентаризации компьютеров, позиционируемая на рынке, позволяет оперативно провести проверку их наличия. Необходимо лишь, чтобы вся техника фирмы была подключена к единой сети, а специализированное программное обеспечение установлено на сервер компании. Программа самостоятельно просмотрит работу всех сетевых устройств, систематизирует данные и сформирует инвентарные отчеты.

Применение программы дает возможность использования типовых или разработанных предприятием отчетных форм и получения стопроцентной точности полученных данных. Кроме того, при осуществлении такой процедуры колоссальна и экономия времени. Инвентаризация компьютеров по сети проводится по всему объему данных по компьютерам, комплектующим и

программным продуктам. Особенностью подобных программ является то, что сканирование данных можно планировать по графику, что дает возможность компании узнать о состоянии техники на конкретную дату.» [1]

1.2. Обзор существующих решений

В какой-то момент любая компания сталкивается с потребностью в наведении порядка в ИТ-инфраструктуре предприятия. Системным администраторам необходимо иметь возможность получать информацию о текущих конфигурациях рабочих станций, чтобы планировать модернизацию и понимать, какие комплектующие требуются для конкретных компьютеров. Целесообразным является централизованное получение и дальнейшее хранение подобной информации, для чего существуют различные средства автоматизации.

На рынке существует достаточно большое количество программного обеспечения (ПО) для корпоративного сегмента, позволяющего провести инвентаризацию компьютерной инфраструктуры. Использование подобных программ имеет ряд преимуществ, таких как широкий функционал, относительная легкость внедрения, при этом большая часть ПО является платной при использовании на среднем и крупном предприятии, а функционал может быть избыточен.

При анализе существующих средств автоматизации было принято решение создать собственную систему с учетом нужд конкретного предприятия и возможностью доработки и поддержки системы, а также не требующую дополнительных материальных затрат.

1.2.1. IT Invent

Программа (Рисунок 1) обладает следующим функционалом:

1. инвентаризация компьютеров;
2. учет программного обеспечения;

3. учет различного оборудования;

4. контроль расхода и перезаправки картриджей;

5. контроль расходных материалов;

6. инвентаризация компьютеров в сети;

7. инвентаризация компьютерной техники с использованием этикеток и сканера штрих-кодов.

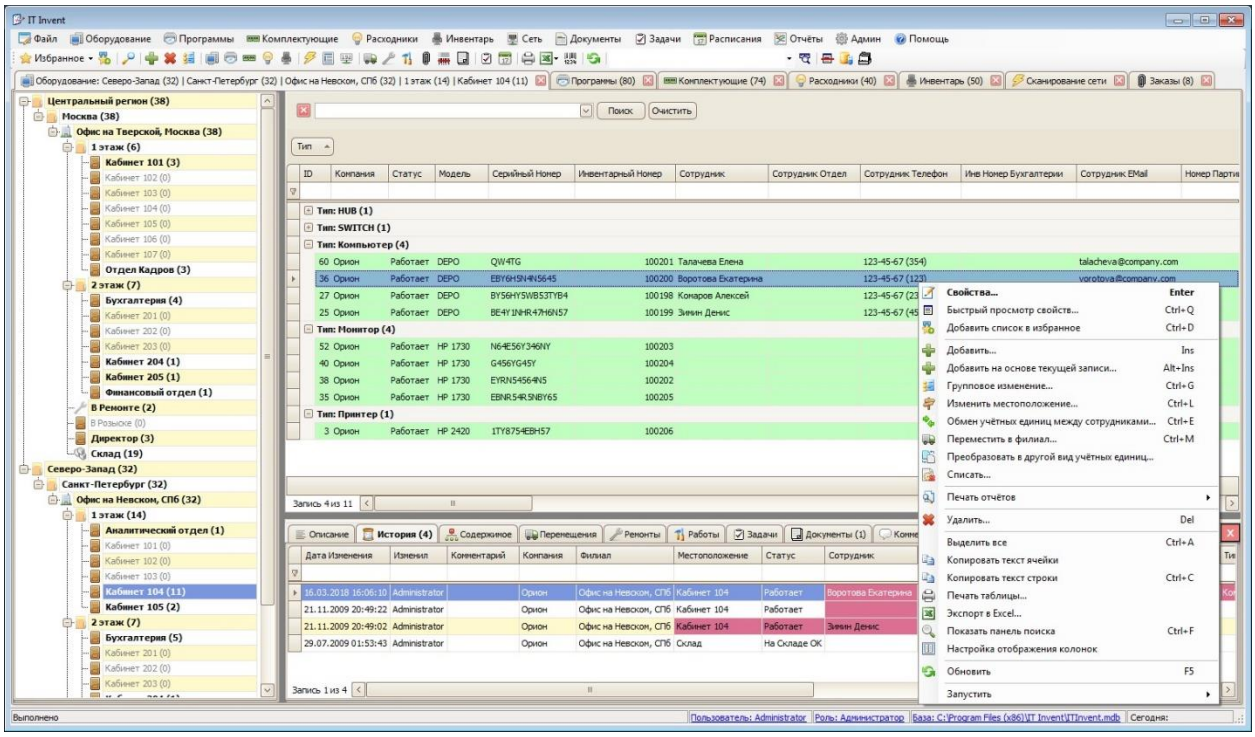


Рисунок 1. Окно программы IT Invent

В зависимости от количества единиц, подлежащих учету, у программы имеются различные типы лицензий, чья стоимость (Таблица 1) варьируется от бесплатного варианта с ограничением в 200 единиц до неограниченной лицензии стоимостью 156 тыс. руб.

Таблица 1. Лицензии программы IT Invent

Лицензия	Количество каждого вида единиц учета	Стоимость, тыс. руб.
Free	200	Бесплатно
Simple	500	16
Extended	1000	32
Professional	3000	48
Premium	6000	72
Smart	10000	96
VIP	20000	120
Unlimited	Неограниченно	156

1.2.2. 10-Страйк Инвентаризация Компьютеров

Программа (Рисунок 2) обладает следующим функционалом:

1. сбор информации об установленном ПО и мониторинг изменений, контроль лицензий и ключей;

2. оповещения и печать отчетов;

3. сбор информации об аппаратном обеспечении и мониторинг изменений;

4. мониторинг состояния жестких дисков.;

5. библиотека ПО на более чем 100 тысяч наименований (Pro);

6. расширенный функционал аудита ПО и дополнительные отчеты (Pro);

7. дополнительные возможности по интеграции с данными (Pro);

8. использование сетевой базы данных (Pro).

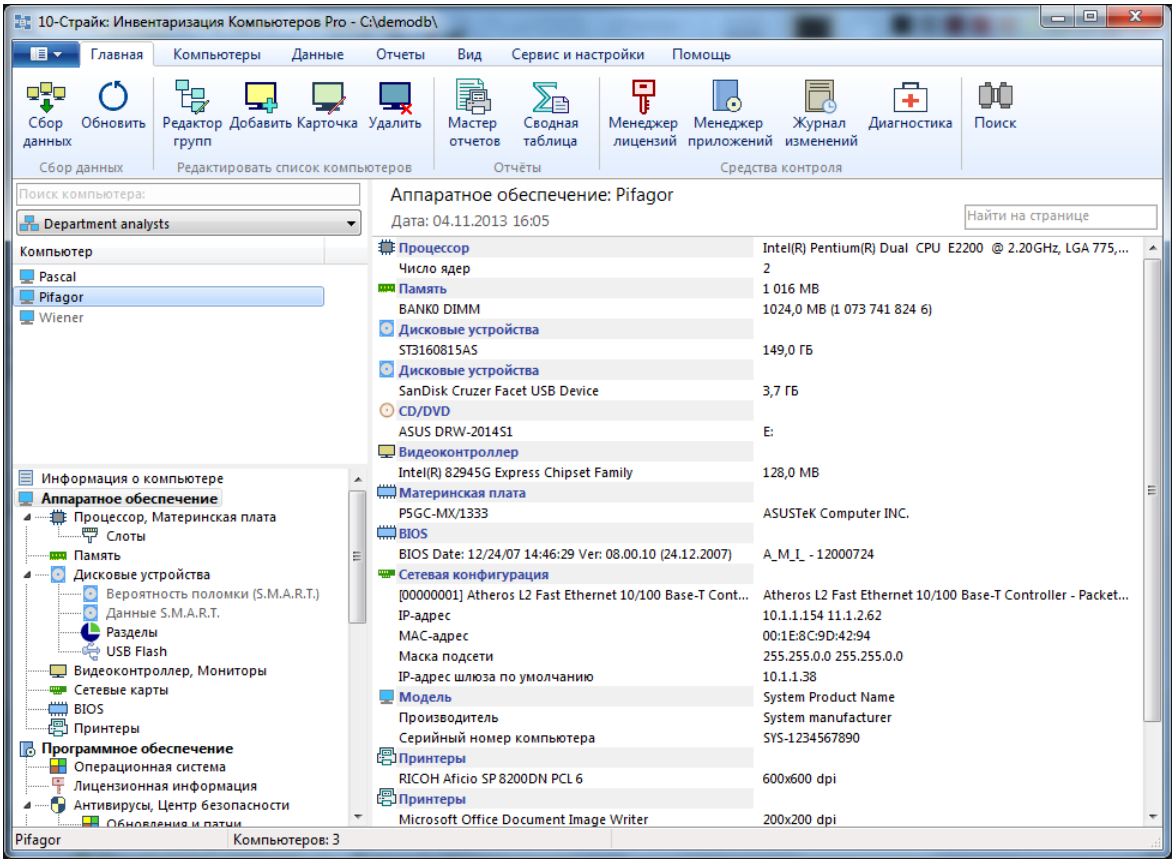


Рисунок 2. Окно программы 10-Страйк Инвентаризация Компьютеров

У программы имеются два типа версий – обычная и Pro, стоимость лицензий (Таблица 2) зависит от количества компьютеров и версии.

Таблица 2. Лицензии программы 10-Страйк Инвентаризация Компьютеров

Количество компьютеров	Стоимость, тыс. руб.	
	Обычная версия	Версия Pro
50	10	15
100	15	25
200	25	40
300	30	50
500	40	60
1000	50	80
Неограниченно	80	100

1.2.3. Total Network Inventory

Программа (Рисунок 3) обладает следующим функционалом:

1. удаленное сканирование различных операционных систем и устройств
2. сканирование компьютеров резидентным агентом;
3. инвентаризация и учет оборудования и ПО;
4. настраиваемые отчеты любой сложности;
5. планировщик сканирования сети;
6. журнал изменений оборудования и ПО;
7. бессрочная лицензия;
8. аудит программного обеспечения;
9. модуль учета лицензий ПО (Pro);
10. вычисление статуса лицензий и хранение лицензионных ключей (Pro);
11. создание подробной карты сети (Pro).

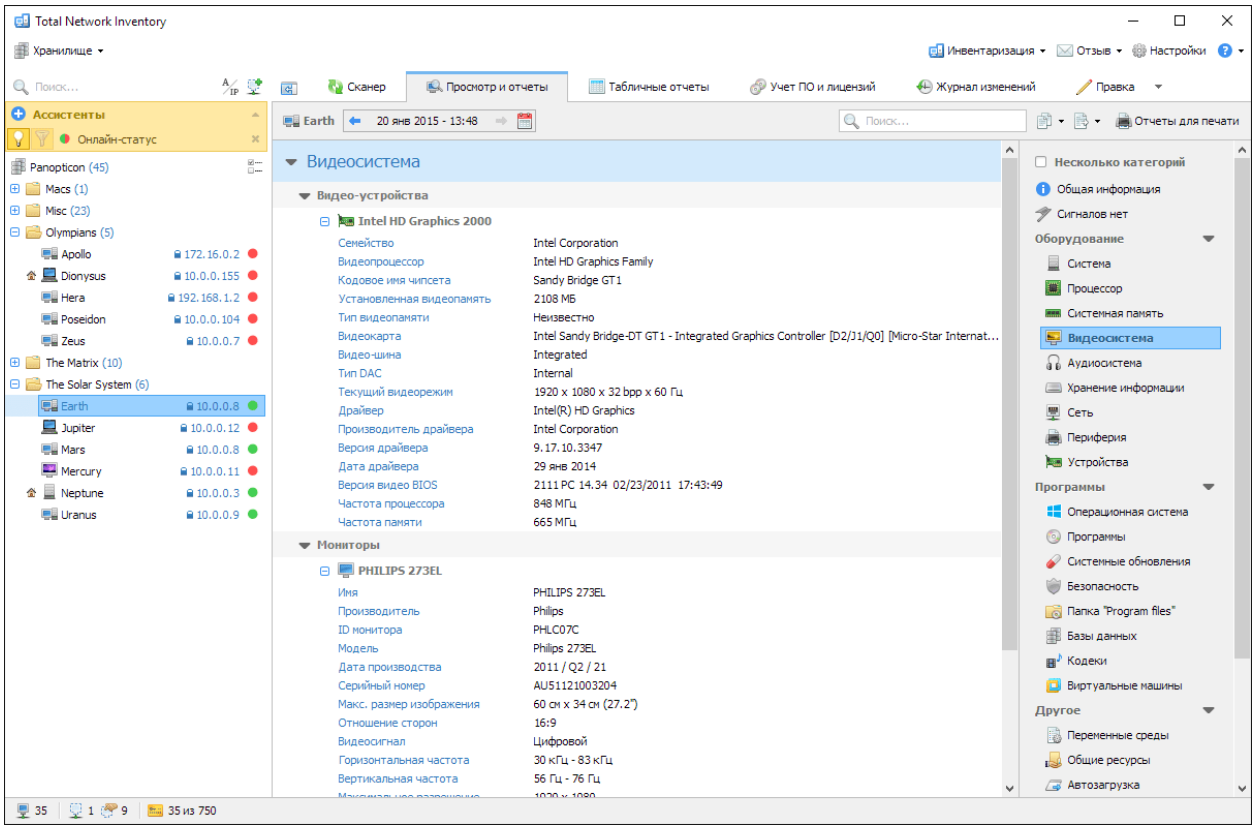


Рисунок 3. Окно программы Total Network Inventory

Программа имеет две версии – Стандартная и Профессиональная, стоимость лицензии (Таблица 3) зависит от количества компьютеров и версии.

Таблица 3. Лицензии программы Total Network Inventory

Количество компьютеров	Стоимость, тыс. руб.	
	Стандартная	Профессиональная
25	3,4	5
50	6,5	9,7
100	11	16,6
150	14,9	22,3
250	21,1	31,7
350	24,2	36,4
500	30,2	45,4
750	38	56,9
1000	45,4	68
1500	57,1	85,7
2000	68,7	103
Неограниченно	95,5	143,3

1.2.4. Microsoft Assessment and Planning Toolkit

«Программное обеспечение Microsoft Assessment and Planning Toolkit (MAP) – это набор инструментов, позволяющих проводить оценку ИТ-инфраструктуры организации.

Инструментарий Microsoft Assessment and Planning (MAP) Toolkit позволяет проводить мониторинг аппаратного и программного обеспечения клиента без установки программ-агентов. MAP генерирует отчеты для получения данных о состоянии серверов и рабочих станциях, их оперативной памяти, процессорах и уровне загрузки, объеме дискового пространства и установленных операционных системах и другого ПО.» [2]

Функционал программы заключается в следующем:

1) общее управление ИТ-инфраструктурой;

2) инвентаризация (аудит программного и аппаратного обеспечения) в гибридной серверной среде;

3) отслеживание использования ПО;

4) формирование отчетов:

a) мастера метрики производительности;

b) об уровне готовности к миграции на ПО Майкрософт;

c) о возможности консолидации серверов с использованием виртуализации.

Программа свободного распространяется компанией Майкрософт, что делает ее доступным средством для проведения инвентаризации, но т.к. основной функцией утилиты является инвентаризация серверов и оценка возможностей миграции и виртуализации, функционал в области инвентаризации рабочих станций является достаточно ограниченным. При этом несомненным плюсом программы является возможность инвентаризации не только компьютеров на базе Windows, но и множества других устройств, в т.ч. компьютеров на базе Linux/UNIX (Рисунок 4).

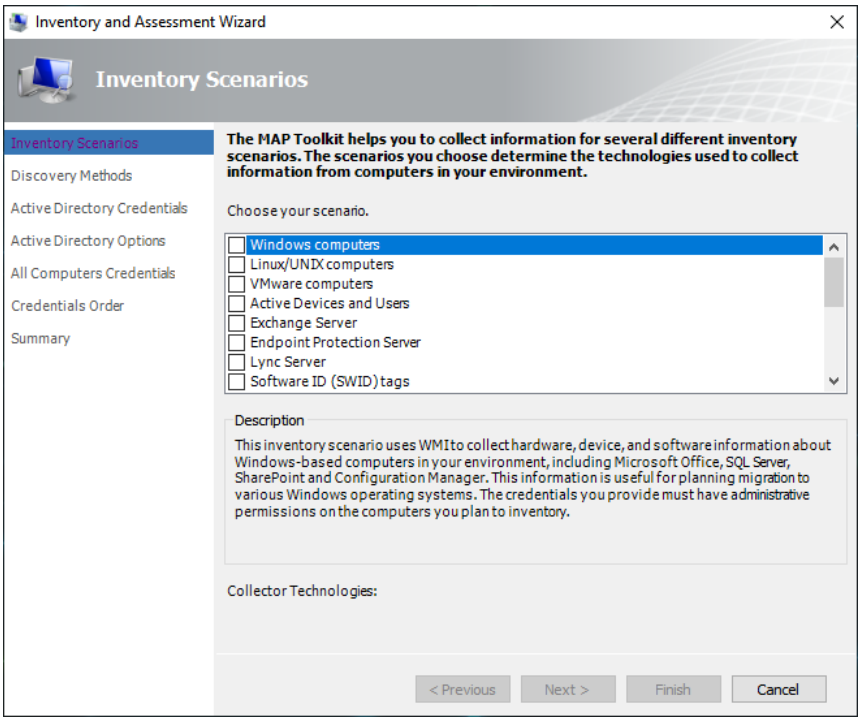


Рисунок 4. Окно выбора инвентаризуемого оборудования

Также возможно получать список целевых компьютеров различными способами (Рисунок 5) -получением списка компьютеров домена, сканированием IP-адресов, ручным вводом имен, импортом списка имен из файла и т.д.

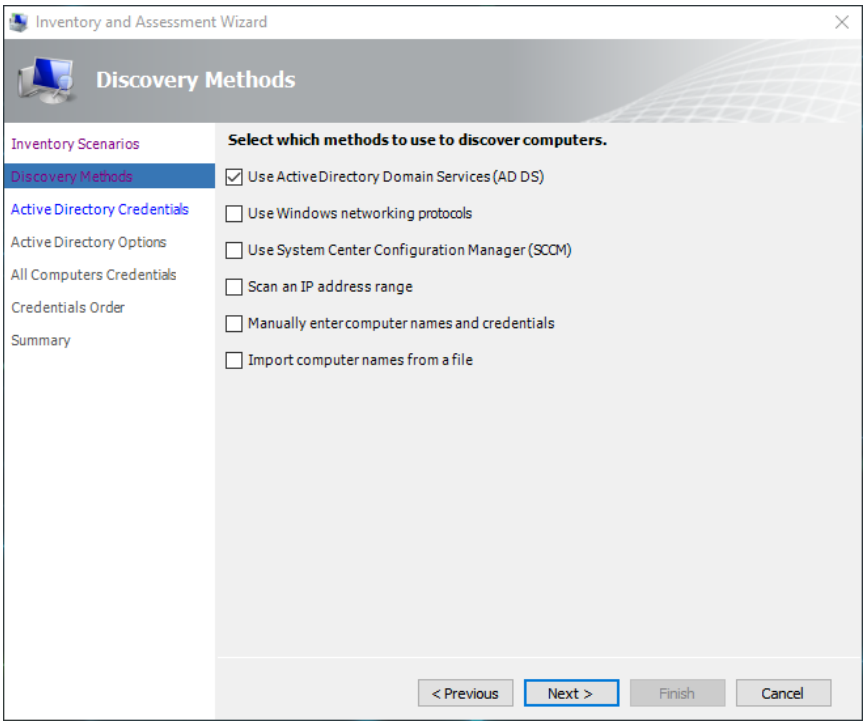
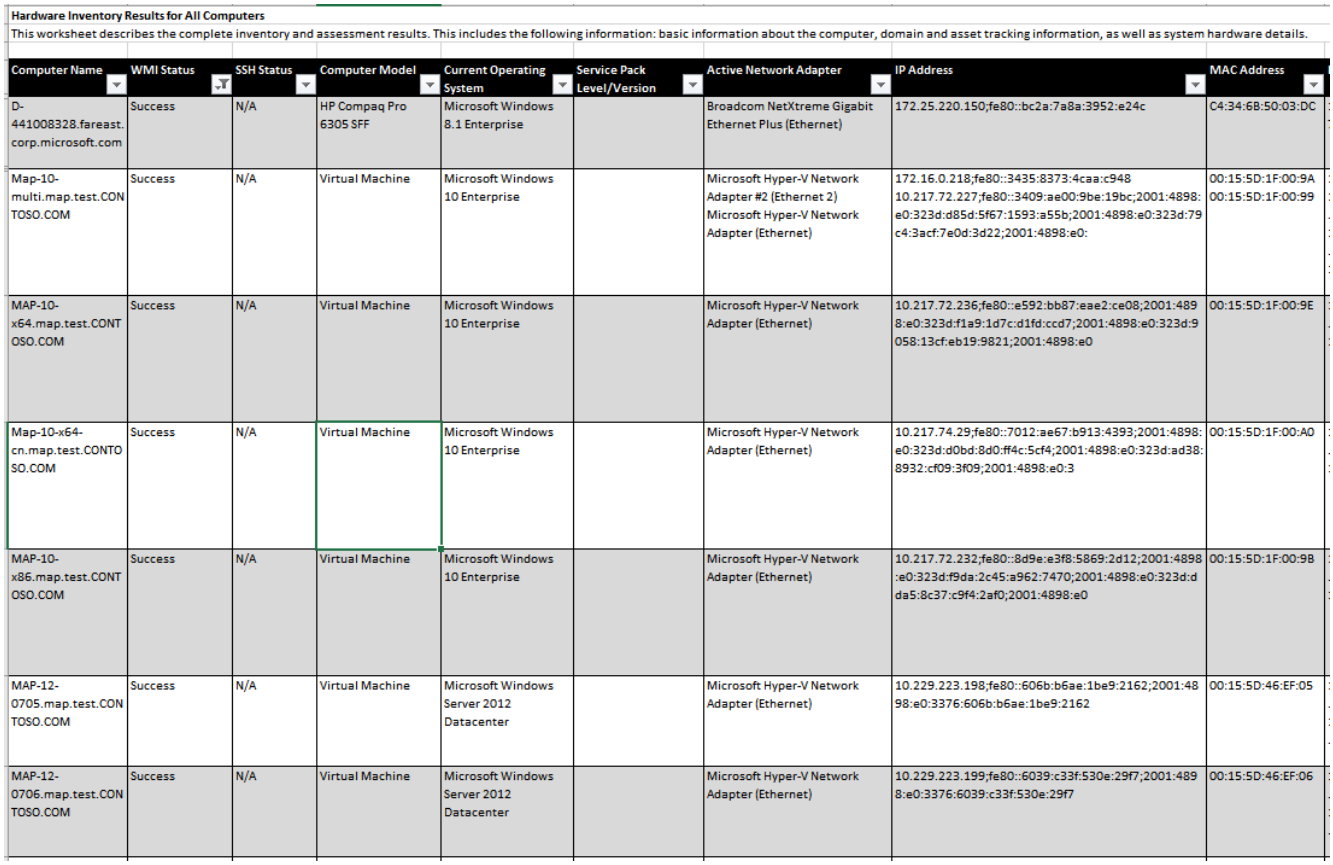


Рисунок 5. Окно выбора способов получения списка компьютеров

Недостатком программы можно назвать то, что результаты инвентаризации предоставляются в виде файла Excel (Рисунок 6), что требует дополнительной обработки данных в целях анализа информации и преобразования в более удобный для человека вид.



Hardware Inventory Results for All Computers

This worksheet describes the complete inventory and assessment results. This includes the following information: basic information about the computer, domain and asset tracking information, as well as system hardware details.

Computer Name	WMI Status	SSH Status	Computer Model	Current Operating System	Service Pack Level/Version	Active Network Adapter	IP Address	MAC Address
D-441008328.fareast.corp.microsoft.com	Success	N/A	HP Compaq Pro 6305 SFF	Microsoft Windows 8.1 Enterprise		Broadcom NetXtreme Gigabit Ethernet Plus (Ethernet)	172.25.220.150;fe80::bc2a:7a8a:3952:e24c	C4:34:6B:50:03:DC
Map-10-multi.map.test.CONTOSO.COM	Success	N/A	Virtual Machine	Microsoft Windows 10 Enterprise		Microsoft Hyper-V Network Adapter #2 (Ethernet 2) Microsoft Hyper-V Network Adapter (Ethernet)	172.16.0.218;fe80::3435:8373:4caa:c948 10.217.72.227;fe80::3409:ae00:9be:19bc;2001:4898:e0:323d:d85d:5f67:1593:a55b;2001:4898:e0:323d:79c4:3acf:7e0d:3d22;2001:4898:e0:	00:15:5D:1F:00:9A 00:15:5D:1F:00:99
MAP-10-x64.map.test.CONTOSO.COM	Success	N/A	Virtual Machine	Microsoft Windows 10 Enterprise		Microsoft Hyper-V Network Adapter (Ethernet)	10.217.72.236;fe80::e592:bb87:eae2:ce08;2001:4898:e0:323d:f1a9:1d7c:d1fd:ccd7;2001:4898:e0:323d:9058:13cf:eb19:9821;2001:4898:e0	00:15:5D:1F:00:9E
Map-10-x64-cn.map.test.CONTOSO.COM	Success	N/A	Virtual Machine	Microsoft Windows 10 Enterprise		Microsoft Hyper-V Network Adapter (Ethernet)	10.217.74.29;fe80::7012:ae67:b913:4393;2001:4898:e0:323d:d0bd:8d0:ff4c:5cf4;2001:4898:e0:323d:ad38:8932:cf09:3f09;2001:4898:e0:3	00:15:5D:1F:00:A0
MAP-10-x86.map.test.CONTOSO.COM	Success	N/A	Virtual Machine	Microsoft Windows 10 Enterprise		Microsoft Hyper-V Network Adapter (Ethernet)	10.217.72.232;fe80::8d9e:e3f8:5869:2d12;2001:4898:e0:323d:f9da:2c45:a962:7470;2001:4898:e0:323d:dda5:8c37:c9f4:2af0;2001:4898:e0	00:15:5D:1F:00:9B
MAP-12-0705.map.test.CONTOSO.COM	Success	N/A	Virtual Machine	Microsoft Windows Server 2012 Datacenter		Microsoft Hyper-V Network Adapter (Ethernet)	10.229.223.198;fe80::606b:b6ae:1be9:2162;2001:4898:e0:3376:606b:b6ae:1be9:2162	00:15:5D:46:EF:05
MAP-12-0706.map.test.CONTOSO.COM	Success	N/A	Virtual Machine	Microsoft Windows Server 2012 Datacenter		Microsoft Hyper-V Network Adapter (Ethernet)	10.229.223.199;fe80::6039:c33f:530e:29f7;2001:4898:e0:3376:6039:c33f:530e:29f7	00:15:5D:46:EF:06

Рисунок 6. Пример отчета программы в файле Excel

Таким образом, программа подходит для первичной инвентаризации ИТ-инфраструктуры, но неудобна для постоянного использования, т.к. функционал инвентаризации именно рабочих станций является достаточно урезанным.

1.2.5. Решение с открытым исходным кодом

Решение с открытым исходным кодом System Information [3] разработано на языке C#, имеет две версии – для просмотра информации о локальном (Рисунок 7) и об удаленном компьютере (Рисунок 8).

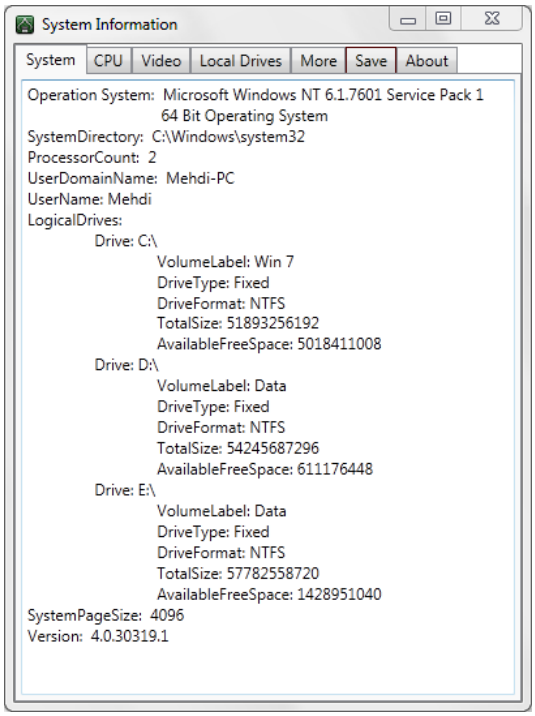


Рисунок 7. Просмотр информации о локальном компьютере

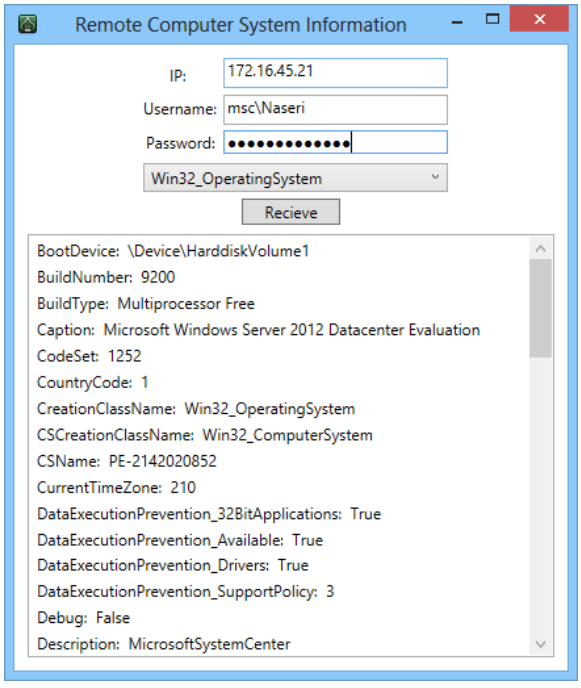


Рисунок 8. Просмотр информации об удаленном компьютере

Среди достоинств программы можно отметить простой интерфейс и возможность получения доступа ко всем возможным параметрам (Рисунок 9), а для локальной версии – сохранение полученной информации в текстовый файл (Рисунок 10).

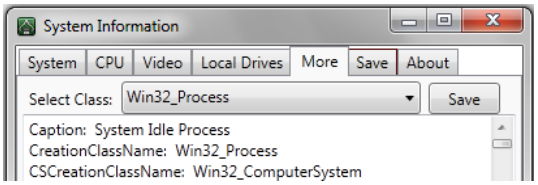


Рисунок 9. Выбор параметров

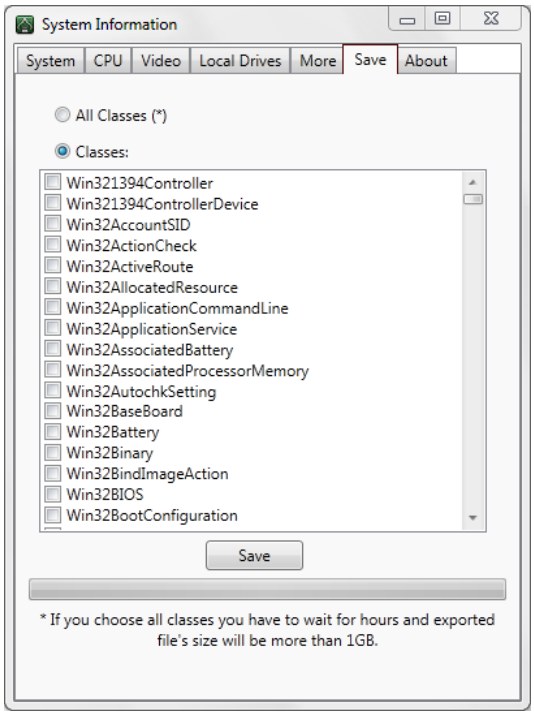


Рисунок 10. Сохранение в текстовый файл

Минусом программы является то, что для каждого удаленного компьютера необходимо делать отдельный запрос, а также отсутствует возможность сохранения информации в базу данных.

1.2.6. Автоматизация PowerShell

Произвести инвентаризацию сети предприятия возможно при помощи PowerShell – мощного средства автоматизации, состоящего из оболочки с интерфейсом командной строки и сопутствующего языка сценариев.

К плюсам автоматизации с помощью PowerShell можно отнести гибкость настраиваемых параметров, развитый язык и средства для работы с ним, а также отсутствие материальных затрат на использование.

Среди недостатков использования PowerShell можно отметить отсутствие графического интерфейса и необходимость изучения скриптового языка.

Рассмотрим ряд скриптов [4], выполняющих инвентаризацию доменных компьютеров.

1. Скрипт, импортирующий список компьютеров домена domain.local в файл ComputersList.csv

```
import-module activedirectory
get-ADcomputer -Filter * |
Where-Object {$a=$_.name; $_.DistinguishedName -ne
"CN=$a,OU=Disable,DC=domain,DC=local"} |
Sort-Object name | Select-Object name | Export-csv D:\ComputersList.csv -
NoTypeInformation
```

2. Скрипт, проверяющий доступность компьютеров из файла ComputersList.csv

```
import-csv D:\ ComputersList.csv | foreach {
$a=$_.name
if ((Test-connection $a -count 2 -quiet) -eq "True"){
if ((Get-WmiObject -computername $a Win32_OperatingSystem) -eq $null){}
}}
```

3. Скрипт, получающий информацию об операционной системе

```
"Компьютер" | out-file D:\$a.txt
Get-WmiObject -computername $a Win32_OperatingSystem |
select-object csname, caption, Serialnumber, csdVersion |
ft @{Label="Сетевое имя"; Expression={$_.CSname}},
@{label="Наименование"; Expression={$_.caption}},
@{label="Версия"; Expression={$_.csdVersion}},
@{label="Серийный номер"; Expression={$_.SerialNumber}} -auto -wrap |
```

```
out-file D:\$a.txt -append
```

4. Скрипт, получающий информацию о процессоре

```
Get-WmiObject -computername $a Win32_ComputerSystemProduct | select-object UUID |
ft UUID -autosize | out-file D:\$a.txt -append
"Процессор" | out-file D:\$a.txt -append
Get-WmiObject -computername $a Win32_Processor | select-object name,
SocketDesignation, Description |
ft @{label="Имя"; Expression={$_.name}},
@{label="Разъем"; Expression={$_.SocketDesignation}},
@{label="Описание"; Expression={$_.Description}} -auto -wrap | out-file D:\$a.txt
-append
```

5. Скрипт, получающий информацию о материнской плате

```
"Материнская плата" | out-file D:\$a.txt -append
Get-WmiObject -computername $a Win32_BaseBoard | select-object Manufacturer,
Product, SerialNumber |
ft @{label="Производитель"; Expression={$_.manufacturer}},
@{label="Модель"; Expression={$_.Product}},
@{label="Серийный номер"; Expression={$_.SerialNumber}} -auto -wrap |
out-file D:\$a.txt -append
```

6. Скрипт, получающий информацию о жестких дисках

```
"Жесткие диски" | out-file D:\$a.txt -append
Get-WmiObject -computername $a Win32_DiskDrive | select-object Model, Partitions,
Size, interfacetype |
ft @{Label="Модель"; Expression={$_.Model}},
@{Label="Количество разделов"; Expression={$_.Partitions}},
@{Label="Размер (гб)"; Expression={($_.Size/1GB).tostring("F00")}},
@{Label="Интерфейс"; Expression={$_.interfaceType}} -auto -wrap |
out-file D:\$a.txt -append
```

7. Скрипт, получающий информацию о логических дисках

```
"Логические диски" | out-file D:\$a.txt -append
Get-WmiObject -computername $a Win32_LogicalDisk -Filter "DriveType=3" | select-
object DeviceID, FileSystem, Size, FreeSpace |
ft @{Label="Наименование"; Expression={$_.DeviceID}},
@{Label="Файловая система"; Expression={$_.FileSystem}},
@{Label="Размер (гб)"; Expression={($_.Size/1GB).tostring("F00")}},
@{Label="Свободное место (гб)"; Expression={($_.FreeSpace/1GB).tostring("F00")}} -
auto -wrap |
out-file D:\$a.txt -append
```

8. Скрипт, получающий информацию ою оперативной памяти

```
"Оперативная память" | out-file D:\$a.txt -append
Get-WmiObject -computername $a Win32_Physicalmemory | Select-Object capacity,
DeviceLocator |
ft @{Label="Размер (мб)"; Expression={($_.capacity/1MB).tostring("F00")}},
@{Label="Расположение"; Expression={$_.DeviceLocator}} -auto -wrap |
out-file D:\$a.txt -append
```

9. Скрипт, получающий информацию о видеокарте

```
"Видеокарта" | out-file D:\$a.txt -append
Get-WmiObject -computername $a Win32_videoController |
Select-Object name, AdapterRAM, VideoProcessor |
ft @{Label="Наименование"; Expression={$_.name}},
@{Label="Объем памяти (мб)"; Expression={($_.AdapterRAM/1MB).tostring("F00")}},
@{Label="Видеопроцессор"; Expression={$_.VideoProcessor}} -auto -wrap |
out-file D:\$a.txt -append
```

10. Скрипт, получающий информацию о сетевом адаптере

```
"Сетевая карта" | out-file D:\$a.txt -append
$OS=Get-WmiObject -computername $a Win32_OperatingSystem | foreach {$_.caption}
if ($OS -eq "Microsoft Windows 2000 Professional")
{
Get-WmiObject -computername $a Win32_NetworkAdapterConfiguration -Filter
"DHCPEnabled=True" |
Select-Object caption,MACaddress |
ft @{Label="Наименование"; Expression={$_.caption}},
@{Label="MAC адрес"; Expression={$_.MACAddress}} -auto -wrap |
out-file D:\$a.txt -append
}
else
{
Get-WmiObject -computername $a Win32_NetworkAdapter -Filter
"NetConnectionStatus>0" |
Select-Object name, AdapterType, MACAddress |
ft @{Label="Наименование"; Expression={$_.name}},
@{Label="MAC адрес"; Expression={$_.MACAddress}},
@{Label="Тип"; Expression={$_.AdapterType}} -auto -wrap |
out-file D:\$a.txt -append
}
```

2. ОПИСАНИЕ ИСПОЛЬЗУЕМЫХ ТЕХНОЛОГИЙ И ВЫБОР СРЕДСТВ РАЗРАБОТКИ

2.1. Описание используемых технологий

При разработке информационной системы были использованы технологии Active Directory (AD) и Windows Management Instrumentation (WMI).

Домены на базе AD позволяют централизованно управлять всеми ресурсами, включая пользователей, файлы, периферийные устройства, доступ к службам, сетевым ресурсам, веб-узлам, базам данных и т. д. Каталог AD поддерживает иерархическое пространство имен для учетной информации о пользователях, группах и компьютерах, а также о других каталогах, что в конечном счете позволяет снизить административные издержки, связанные с поддержкой нескольких пространств имен.

WMI представляет собой интерфейс, обеспечивающий взаимодействие с компонентами системы. Эта технология позволяет получать информацию о компьютерной инфраструктуре и управлять ее компонентами, выполнять мониторинг и администрирование. WMI позволяет получить программный доступ к большому объему информации, касающейся аппаратных и программных средств локального или удаленного компьютера.

Все средства, обладающие сходным с разрабатываемой системой функционалом, используют технологии AD и WMI, т.к. именно они предоставляют доступ к необходимой информации.

2.1.1. Описание технологии Active Directory

«Служба каталогов Active Directory (AD) обеспечивает эффективную работу сложной корпоративной среды, предоставляя следующие возможности:

1. Единая регистрация в сети – пользователи могут регистрироваться в сети с одним именем и паролем и получать при этом доступ ко всем сетевым ресурсам

и службам (службы сетевой инфраструктуры, службы файлов и печати, серверы приложений и баз данных и т. д.).

2. Безопасность информации: средства аутентификации и управления доступом к ресурсам, встроенные в службу Active Directory, обеспечивают централизованную защиту сети.

3. Централизованное управление. Администраторы могут централизованно управлять всеми корпоративными ресурсами.

4. Администрирование с использованием групповых политик. При загрузке компьютера или регистрации пользователя в системе выполняются требования групповых политик; их настройки хранятся в объектах групповых политик (GPO) и применяются ко всем учетным записям пользователей и компьютеров, расположенных в сайтах, доменах или организационных подразделениях;

5. Интеграция с DNS. Функционирование служб каталогов полностью зависит от работы службы DNS. В свою очередь серверы DNS могут хранить информацию о зонах в базе данных Active Directory;

6. Расширяемость каталога. Администраторы могут добавлять в схему каталога новые классы объектов или добавлять новые атрибуты к существующим классам;

7. Масштабируемость. Служба Active Directory может охватывать как один домен, так и множество доменов, объединенных в дерево доменов, а из нескольких деревьев доменов может быть построен лес;

8. Репликация информации. В службе Active Directory используется репликация служебной информации в схеме со многими ведущими (multi-master), что позволяет модифицировать БД Active Directory на любом контроллере домена. Наличие в домене нескольких контроллеров обеспечивает отказоустойчивость и возможность распределения сетевой нагрузки;

9. Гибкость запросов к каталогу. БД Active Directory может использоваться для быстрого поиска любого объекта AD, используя его свойства (например, имя

пользователя или адрес его электронной почты, тип принтера или его местоположение и т. п.);

10. Стандартные интерфейсы программирования. Для разработчиков программного обеспечения служба каталогов предоставляет доступ ко всем возможностям (средствам) каталога и поддерживает принятые стандарты и интерфейсы программирования (API).

Служба каталогов Active Directory организована в виде иерархической структуры, построенной из различных компонентов, которые представляют элементы корпоративной сети. В этой структуре есть, например, пользовательские объекты, компьютерные объекты, и различные контейнеры. Способ организации этих элементов представляет собой логическую структуру Active Directory в корпоративной сети. Логическая структура Active Directory включает в себя леса, деревья, домены и организационные подразделения.

Основной единицей системы безопасности Active Directory является домен. Домен формирует область административной ответственности. База данных домена содержит учетные записи пользователей, групп и компьютеров. Большая часть функций по управлению службой каталогов работает на уровне домена (аутентификация пользователей, управление доступом к ресурсам, управление службами, управление репликацией, политики безопасности).

Контроллеры домена – специальные серверы, хранящие соответствующую данному домену часть базы данных Active Directory.

Основные функции контроллеров домена:

1. хранение БД Active Directory (организация доступа к информации, содержащейся в каталоге, включая управление этой информацией и ее модификацию);

2. синхронизация изменений в AD (изменения в базу данных AD могут быть внесены на любом из контроллеров домена, любые изменения, осуществляемые на одном из контроллеров, будут синхронизированы с копиями, хранящимися на других контроллерах);

3. аутентификация пользователей (любой из контроллеров домена осуществляет проверку полномочий пользователей, регистрирующихся на клиентских системах).

Дерево является набором доменов, которые связаны отношениями «дочерний»/ «родительский», а также используют связанные (смежные, или прилегающие) пространства имен. При этом дочерний домен получает имя от родительского. Между доменами автоматически устанавливаются двухсторонние транзитивные доверительные отношения. Это означает, что доверительные отношения могут быть использованы всеми другими доменами данного леса для доступа к ресурсам данного домена.

Лес – это одно или несколько деревьев, которые разделяют общую схему, серверы Глобального каталога и конфигурационную информацию. В лесу между всеми доменами установлены двухсторонние транзитивные доверительные отношения, что позволяет пользователям любого домена получать доступ к ресурсам всех остальных доменов, если они имеют соответствующие разрешения на доступ. По умолчанию, первый домен, создаваемый в лесу, считается его корневым доменом, в корневом домене хранится схема AD.

Организационные подразделения (Organizational Units, OU) – контейнеры внутри AD, которые создаются для объединения объектов в целях делегирования административных прав и применения групповых политик в домене. ОП существуют только внутри доменов и могут объединять только объекты из своего домена. ОП могут быть вложенными друг в друга, что позволяет строить внутри домена сложную древовидную иерархию из контейнеров и осуществлять более гибкий административный контроль. Кроме того, ОП могут создаваться для отражения административной иерархии и организационной структуры компании.

Глобальный каталог является перечнем всех объектов, которые существуют в лесу Active Directory. По умолчанию, контроллеры домена содержат только информацию об объектах своего домена. Сервер Глобального каталога является

контроллером домена, в котором содержится информация о каждом объекте (хотя и не обо всех атрибутах этих объектов), находящемся в данном лесу.

Учетные записи (accounts) пользователей, компьютеров и групп – один из главных элементов управления доступом к сетевым ресурсам, а значит, и всей системы безопасности сети в целом.

В среде Active Directory существует 3 главных типа пользовательских учетных записей:

1. Локальные учетные записи пользователей. Эти учетные записи существуют в локальной базе данных SAM (Security Accounts Manager) на каждой системе, работающей под управлением Windows. Эти учетные записи создаются с использованием инструмента Local Users and Groups (Локальные пользователи и группы) консоли Computer Management (Управление компьютером).

2. Учетные записи пользователей домена. Эти учетные записи хранятся в Active Directory и могут использоваться для входа в систему и доступа к ресурсам по всему лесу AD. Учетные записи этого типа создаются централизованно при помощи консоли «Active Directory Users and Computers» («Active Directory – пользователи и компьютеры»).

3. Встроенные учетные записи. Эти учетные записи создаются самой системой и не могут быть удалены. По умолчанию любая система, будь то изолированная (отдельно стоящая) или входящая в домен, создает две учетные записи – Administrator (Администратор) и Guest (Гость). По умолчанию учетная запись Гость отключена.

Доменные учетные записи пользователей (а также компьютеров и групп) хранятся в специальных контейнерах AD. Это могут быть либо стандартные контейнеры Users для пользователей и Computers для компьютеров, либо созданное администратором Организационное подразделение (ОП). Исключение составляют учетные записи контроллеров домена, они всегда хранятся в ОП с названием Domain Controllers.

Учетные записи групп, как и учетные записи пользователей, могут быть созданы либо в локальной базе SAM компьютера (сервера или рабочей станции), либо в доменной базе данных Active Directory.

Локальные группы простого сервера-члена домена или рабочей станции могут включать в себя и локальные учетные записи данного компьютера, и глобальные учетные записи любого пользователя или компьютера всего леса, а также доменные локальные группы «своего» домена и глобальные и универсальные группы всего леса.

В Active Directory группы различаются по типу (группы безопасности и группы распространения) и по области действия (локальные в домене, глобальные и универсальные).

Группы безопасности – каждая группа данного типа, так же, как и каждая учетная запись пользователя, имеет идентификатор безопасности (Security Identifier, или SID), поэтому группы безопасности используются для назначения разрешений при определении прав доступа к различным сетевым ресурсам.

Группы распространения – группы этого типа не имеют идентификатора безопасности, поэтому не могут использоваться для назначения прав доступа, их главное назначение – организация списков рассылки для почтовых программ (например, для Microsoft Exchange Server).

Локальные в домене могут содержать глобальные группы из любого домена, универсальные группы, глобальные учетные записи пользователей из любого домена леса, используются при назначении прав доступа только к ресурсам «своего» домена;

Глобальные могут содержать только глобальные учетные записи пользователей «своего» домена, используются при назначении прав доступа к ресурсам любого домена в лесу;

Универсальные могут содержать другие универсальные группы всего леса, глобальные группы всего леса, глобальные учетные записи пользователей из

любого домена леса, используются при назначении прав доступа к ресурсам любого домена в лесу.

Управление рабочими станциями, серверами, пользователями в большой организации – очень трудоемкая задача. Механизм Групповых политик (Group Policy) позволяет автоматизировать данный процесс управления. С помощью групповых политик (ГП) можно настраивать различные параметры компьютеров и пользовательской рабочей среды сразу в масштабах сайта AD, домена, организационного подразделения (детализацию настроек можно проводить вплоть до отдельного компьютера или пользователя).

Каждый объект групповых политик (GPO, Group Policy Object) состоит из двух частей: контейнера групповых политик (GPC, Group Policy Container), хранящегося в БД Active Directory, и шаблона групповых политик (GPT, Group Policy Template).

Каждый объект политик содержит два раздела: конфигурация компьютера и конфигурация пользователя. Параметры этих разделов применяются соответственно либо к настройкам компьютера, либо к настройкам среды пользователя.

При загрузке компьютера и аутентификации в домене к нему применяются компьютерные разделы всех привязанных политик. При входе пользователя в систему к пользователю применяется пользовательский раздел всех групповых политик. Политики, привязанные к некоторому уровню иерархии объектов AD (сайта, домена, подразделения) наследуются всеми объектами AD, находящимися на более низких уровнях. Порядок применения политик:

1. локальная политика;
2. политики сайта Active Directory;
3. политики домена;
4. политики организационных подразделений.

Если в процессе применения политик какие-либо параметры определяются в различных политиках, то действующими значениями параметров будут значения, определенные позднее.

Групповые политики могут использоваться для установки прикладных программ в масштабах всего домена или отдельного организационного подразделения.

Протокол аутентификации Kerberos, используемый в среде Active Directory, предлагает механизм взаимной аутентификации клиента и сервера перед установлением связи между ними, причем в протоколе учтен тот факт, что начальный обмен информацией между клиентом и сервером происходит в незащищенной среде, а передаваемые пакеты могут быть перехвачены и модифицированы. Таким образом, протокол идеально подходит для применения в Интернет и аналогичных сетях.» [5]

2.1.2. Описание технологии WMI

Технология WMI (Windows Management Instrumentation) – одна из базовых технологий Microsoft для централизованного управления и слежения за работой различных частей компьютерной сети под управлением Windows.

«По своей сути WMI – это расширенная реализация модели управления предприятием на базе Web (WBEM – WebBased Enterprise Management). Задачей WBEM является разработка таких стандартов удаленного управления информационной средой предприятия, которые не зависят от конкретного оборудования, сетевой инфраструктуры, операционной системы, файловой системы и т.д.

В основе структуры представления данных в стандарте WBEM лежит CIM (Common Information Model – модель информации общего типа), реализующая объектно-ориентированный подход к представлению компонентов систем как классов со своим набором свойств и методов, а также принципов наследования.

Основное средство для описания новых элементов модели CIM – это синтаксис языка Managed Object Format (MOF), который является текстовым и легко понятным человеку. Таким образом, любое приложение или драйвер в операционной системе, которая поддерживает стандарт WBEM, может добавить к системной модели CIM свой набор классов. Такое расширение модели CIM позволяет легко интегрировать в единую систему мониторинга и управления все новые и новые приложения. Для этой интеграции приложение должно лишь зарегистрировать свои классы в существующей модели CIM и обеспечить стандартные вызовы встроенных процедур для создания объектов этих классов и наполнения их свойствами и методами. Набор этих процедур оформляется, как WMI Provider – специальная библиотека, являющаяся мостом между любым приложением и ядром службы WMI. Многие производители программного и аппаратного обеспечения ведут разработку ПО в соответствии со стандартом WBEM. Как следствие, это ПО совместимо и с WMI, а значит, может управляться через единый и удобный интерфейс.» [6]

«Архитектура WMI состоит из трех частей:

1. Управляемые объекты/ресурсы (managed resources) – любые логические или физические компоненты информационной системы, доступ к которым может быть получен с помощью WMI. В качестве управляемых ресурсов могут выступать, например, файлы на жестком диске, запущенный экземпляр приложения, системное событие, предоставленный в общее пользование ресурс, сетевой пакет или установленный в компьютере процессор.

2. Ядро WMI (WMI infrastructure). Это связующее звено архитектуры WMI, отвечающее за связь управляющих программ с управляемыми объектами. Ядро WMI, в свою очередь, можно разделить на три части: менеджер объектов CIM (Common Information Model Object Manager, CIMOM), репозиторий (хранилище классов и объектов) CIM и провайдеры WML.

3. Управляющие программы (management applications), которые являются потребителями сервисов WMI. В качестве потребителей могут выступать

полновесные Win32-приложения, Web-приложения, сценарии WSH или другие инструменты администрирования, с помощью которых происходит доступ к управляемым объектам посредством WMI.

Задачей менеджера объектов CIM (CIMOM) является обеспечение взаимодействия между потребителями сервисов WMI (управляющими приложениями) и провайдерами WMI. CIMOM обрабатывает все запросы, которые поступают от управляющих приложений к WMI, и обеспечивает доставку к этим приложениям информации, полученной в результате выполнения таких запросов.

Провайдеры WMI (Таблица 4) обеспечивают связь между менеджером объектов CIM и управляемыми ресурсами: провайдеры предоставляют для CIMOM данные об управляемом объекте, обрабатывают запросы от управляющих программ и генерируют сообщения о наступлении определенных событий.

При этом провайдер WMI общается с управляемым объектом с помощью специфического API этого объекта, а с CIMOM – посредством стандартного интерфейса прикладного программирования WMI (WMI API). Таким образом, провайдеры скрывают детали внутренней реализации управляемых объектов, позволяя CIMOM обращаться к этим объектам единообразно, используя один и тот же WMI API. Провайдеры WMI, которые скрывают детали внутренней реализации управляемых объектов, позволяя CIMOM обращаться к этим объектам единообразно, используя WMI API.

Таблица 4. Некоторые стандартные провайдеры WMI

Провайдер	DLL-файл	Описание
Провайдер каталога Active Directory (Active Directory provider)	Dsprov.dll	Позволяет обращаться к объектам Active Directory как к объектам WMI
Провайдер журнала событий (Event Log provider)	Ntevt.dll	Обеспечивает управление журналом событий (выборка по определенному критерию записей для чтения, создание резервных копий и очистка журнала, изменение настроек и т. д.). Также этот провайдер позволяет обрабатывать события, генерируемые журналом (например, добавление в журнал записи определенного типа)

Провайдер	DLL-файл	Описание
Провайдер системных счетчиков производительности (Perfomance Counter provider)	Wbemperf.dll	Обеспечивает доступ к счетчикам производительности, т. е. к данным, позволяющим численно оценивать производительность системы
Провайдер реестра (Registry provider)	Stdprov.dll	Позволяет читать данные из реестра, создавать и модифицировать там ключи и разделы. Кроме этого, провайдер обеспечивает генерацию события WMI при изменении определенного ключа или ветви реестра
Провайдер SNMP-устройств (SNMP provider)	Snmpincl.dll	Является шлюзом для доступа к системам и устройствам, которые управляются с помощью протокола SNMP (Simple Network Management Protocol)
Провайдер драйверов устройств (WDM provider)	Wmiprov.dll	Позволяет получить доступ к информации низкого уровня о драйверах устройств Windows Driver Model (WDM); в качестве таких устройств могут выступать, например, порты ввода/вывода или сетевые платы
Провайдер подсистемы Win32 (Win32 provider)	Cimwin32.dll	Обеспечивает доступ к информации о компьютере, операционной системе, под-системе безопасности, дисках, перифе-рийных устройствах, файловых системах, файлах, папках, сетевых ресурсах, прин-терах, процессах, сервисах и т. п.
Провайдер инсталлированных программных продуктов (Windows Installer provider)	Msiprov.dll	Позволяет получить информацию об инсталлированном программном обеспечении

Основной идеей, на которой базируется WMI, является возможность представить информацию о состоянии любого управляемого объекта в виде стандартной схемы. В качестве такой схемы выступает информационная модель CIM, которая является репозиторием (хранилищем) объектов и классов, моделирующих различные компоненты компьютерной системы.

Таким образом, CIM можно считать хранилищем классов, где класс – это модель (шаблон) управляемого объекта (в качестве управляемых объектов могут выступать самые различные логические и физические компоненты компьютерной системы: жесткие диски, журналы событий, сетевые карты, файлы и папки, процессы, сервисы, процессоры и т. д.). С этой точки зрения CIM похожа на другие каталоги, которые используются в Windows (например, каталог файловой системы

содержит объекты-файлы и объекты-папки, а каталог Active Directory – объекты-домены, объекты-пользователи, объекты-принтеры и т. д.). Однако важной особенностью CIM является то, что хранящиеся в ней классы чаще всего соответствуют динамически изменяемым ресурсам, поэтому объекты-экземпляры таких классов не хранятся постоянно в CIM, а создаются провайдером по запросу потребителя WML Связано это с тем, что состояние большинства WMI-совместимых устройств меняется очень быстро и постоянное обновление информации в CIM может значительно снизить общую производительность системы.

Классы, составляющие CIM, имеют свойства и методы и находятся в иерархической зависимости друг от друга – классы-потомки могут наследовать или переопределять свойства родительских классов, а также добавлять собственные свойства. Свойства описывают конфигурацию и текущее состояние управляемого ресурса, а методы позволяют выполнить над этим ресурсом определенные действия.

Классы CIM группируются в пространства имен (namespaces), которые упорядочены иерархически (корневое пространство имен обозначается через Root). Пространство имен – это группа логически связанных друг с другом классов, которые относятся к какой-либо определенной технологии или области управления. Например, одно из наиболее часто используемых на практике пространств имен CIMV2 (Таблица 5) содержит классы, которые описывают компьютер и операционную систему.

Таблица 5. Некоторые классы из пространства имен Root\CIMV2

Класс	Описание
Win32_BaseBoard	Описывает системную (материнскую) плату. С помощью экземпляра этого класса можно, например, узнать серийный номер материнской платы
Win32_Bus	Описывает физические шины (например, шины PCI или USB) с точки зрения операционной системы Win32
Win32_Processor	Представляет процессоры, т.е. устройства, способные обрабатывать наборы машинных команд в системе Win32. В мультипроцессорной системе для каждого из процессоров существует отдельный экземпляр этого класса

Класс	Описание
Win32_DiskPartition	Позволяет получать информацию об имеющихся в системе разделах жестких дисков. Для каждого из разделов создается свой экземпляр этого класса
Win32_BIOS	Свойства этого класса представляют атрибуты базовой системы ввода/вывода (BIOS): компания-производитель, версия, номер сборки и т. д.
Win32_OperatingSystem	Описывает установленную на компьютере операционную систему. В свойствах экземпляра этого класса хранятся номер сборки системы, используемая по умолчанию кодовая страница, время последней перезагрузки, число пользовательских лицензий и т. д.
Win32_Directory	Экземпляры этого класса соответствуют каталогам файловой системы. Свойства и методы класса Win32_Directory практически совпадают со свойствами и методами класса CIM_DataFile
Win32_Desktop	В свойствах экземпляров класса Win32_Desktop хранятся характеристики рабочих столов пользователей: частота мигания курсора, имя исполняемого файла заставки, частота вызова заставки, имя файла с рисунком рабочего стола и т. д.
Win32_Share	Экземпляры этого класса соответствуют общим ресурсам, имеющимся в системе (общие папки, принтеры, именованные каналы и т. д.)
Win32_Service	Экземпляры данного класса позволяют получать информацию о службах (services) операционной системы и управлять ими (запускать, останавливать и т. д.)
Win32_Process	Каждому запущенному в системе процессу соответствует экземпляр класса Win32_Process. Свойства этого класса позволяют получить полную информацию о процессе (имя, идентификатор, время создания, приоритет и т. д.), ас помощью методов данного класса можно создавать новые процессы, менять приоритеты, завершать процессы и т. д

Всякому ресурсу, управляемому с помощью WMI, соответствует специальный класс WMI; каждый класс имеет четко определенную структуру и содержит свойства, методы и квалификаторы (свои квалификаторы могут быть также у свойств и методов). Классы описываются с помощью специального языка MOF (Managed Object Format), который, в свою очередь, базируется на языке IDL (Interface Definition Language), применяемом для описания интерфейсов COM-объектов. После определения структуры класса с помощью MOF разработчик может добавить откомпилированное представление этого класса в репозиторий CIM с помощью стандартной утилиты mofcomp.exe.» [7]

2.2. Выбор средств разработки

В качестве технологий для реализации информационной системы были выбраны следующие инструменты:

1. Язык C# и платформа .NET – предоставляют широкий набор функций, в том числе для работы с AD и WMI, а также создания графического интерфейса, C# является объектно-ориентированным языком с достаточно простым и удобным синтаксисом.
2. Среда разработки Microsoft Visual Studio 2019 – является основной средой разработки для языка C#, используется свободно распространяемая версия Community.
3. Технология WPF (Windows Presentation Foundation) – подсистема для построения графических интерфейсов платформы .NET.
4. Система управления базами данных (СУБД) Microsoft SQL Server 2016 – СУБД, использующая реляционную модель организации баз данных, используется свободно распространяемая версия Express.
5. Entity Framework Core – позволяет абстрагироваться от самой базы данных и ее таблиц и работать с данными независимо от типа хранилища, т.е. с объектами и их коллекциями.

2.2.1. Язык C# и платформа .NET

«C# (произносится «си шарп») – объектно-ориентированный язык программирования. Разработан в 1998-2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как язык разработки приложений для платформы Microsoft .NET Framework и впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов

явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщенные типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Переняв многое от своих предшественников - языков C++, Java, Delphi, Модула и Smalltalk – C#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# не поддерживает множественное наследование классов (в отличие от C++).» [8]

«Microsoft .NET Framework – это интегрированная в ОС Windows, согласованная и всесторонне развитая модель программирования. Она используется для создания ПО высокого качества, обеспеченного превосходным интерфейсом, но и отличается защищенностью и прозрачностью сетевых связей.

Ядром работы платформы является многоязычная среда программирования под названием Common Language Runtime (CLR) как ответ на популярную на то время платформу Java от компании Sun.

Microsoft .NET Framework состоит из двух частей:

1. среда исполнителя;

2. подключаемая библиотека.

Главной составляющей можно назвать Common Language Runtime (CLR). Она может выполнять программную часть обычных приложений или серверных, являясь исполняющей средой.

Framework Class Library (FCL) – это библиотека классов, в которой содержится достаточно много элементов для обращения с:

1. базами данных;

2. сетями;

3. интерфейсами;

4. файлами;

5. вводом и выводом данных.

Все это дает разработчику возможность использовать готовые классы для создания программ, обходя низкоуровневую часть.» [9]

2.2.2. Microsoft Visual Studio

«Microsoft Visual Studio – линейка продуктов компании Майкрософт, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в собственном, так и в управляемом кодах для всех платформ, поддерживаемых Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework и Microsoft Silverlight.

Visual Studio включает в себя редактор исходного кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и как отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования или инструментов для прочих аспектов цикла разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

Интегрированная среда разработки программ Microsoft Visual Studio предназначена для создания программных приложений и позволяет:

1. создавать приложения с использованием языков Visual Basic, Visual C#, Visual C++, Visual J#;

2. создавать Windows- и Web-приложения, включая приложения для портативных устройств;

3. создавать клиент-серверные приложения;

4. создавать корпоративные Web-приложения масштаба предприятия.

Microsoft Visual Studio увеличивает производительность труда разработчиков при создании приложений. Включает, например, Visual Studio Team System – интегрированный и расширяемый набор инструментов для управления программными проектами на всех этапах разработки и тестирования, что обеспечивает взаимодействие и совместную работу коллектива разработчиков.

Microsoft Visual Studio содержит среду выполнения .NET Framework, которая состоит из двух частей: единой среды исполнения (Common Language Runtime, CLR) и библиотеки классов. Библиотека классов является всесторонней, объектно-ориентированной коллекцией типов, которую можно использовать для разработки приложений, начиная с традиционных приложений с командной строкой и с использованием графического пользовательского интерфейса и заканчивая приложениями, использующими Web-формы и XML Web-сервисы. Класс – это тип, описывающий устройство объектов, в то время как объект является конкретным представителем определенного класса. Таким образом, каждый объект является экземпляром определенного класса. Единая среда исполнения управляет кодом во время его выполнения.

Элементы графического интерфейса среды разработки Visual Studio, называемого сокращенно IDE (Integrated Development Environment) (меню, панели инструментов, диалоговые окна), характерны для среды Windows. Среда разработки Visual Studio является интегрированной, так как в ней можно выполнять различные действия при разработке программного продукта, такие как проектирование графического интерфейса приложения, редактирование программного кода, компиляция всех элементов приложения и сборка в выполняемый файл, а также пошаговая отладка приложения. Пользователь может изменять расположение и форму окон IDE, а также сворачивать их, чтобы сделать

доступными и видимыми на экране необходимые элементы среды программирования. Главное окно Visual Studio после запуска программы содержит несколько основных объектов: главное меню и стандартная панель инструментов, окна Start Page (Начальная страница), Solution Explorer (Обозреватель решений), Toolbox (Инструментарий), Properties (Свойства), Object Browser (Просмотр объектов) и Dynamic Help (Динамическая справка). Диалоговое окно Start Page (Начальная страница) позволяет открывать недавно использовавшиеся проекты, осуществляет поиск примеров, как из справочной системы, так и Интернета. В окне Solution Explorer (Обозреватель решений) размещаются проекты и файлы текущего решения. К средству, призванному облегчить разработку, относится также окно Toolbox (Инструментарий), отображающее элементы, используемые в проектах. Окно Properties (Свойства) предназначено для отображения и настройки свойств объектов решения, включая форму и размещенные в ней объекты. Для получения подробной информации об объектах используется диалоговое окно Object Browser (Просмотр объектов). Оно позволяет искать и исследовать элементы, их свойства, методы, события, находящиеся в проектах и ссылках на них, как бы представляя собой внутреннюю библиотеку. Для удобства разработки используется окно Dynamic Help (Динамическая справка). Во время работы оно постоянно обновляется и в нем предлагается справочная информация, относящаяся к текущим действиям.

Решения и проекты – это контейнеры, которые Visual Studio использует для размещения и группировки кода, который пишется в интегрированной среде. Контейнеры – это объекты, внутри которых размещены другие объекты. Решения – это виртуальные контейнеры; они группируют свойства, относящиеся к одному (или нескольким) содержащимся в решении проектам. Решения не подвергаются обработке компилятором. Внутри интегрированной среды можно настроить несколько свойств на уровне решения. Проекты имеют и виртуальный, и физический характер. Помимо функционирования в качестве организационных единиц для создающегося кода они также однозначно соответствуют результатам,

получаемым на выходе компилятора. Иначе говоря, Visual Studio превращает проекты в откомпилированный код (эквивалентную программу на машинном языке).

Каждый проект приводит к созданию .NET-компонента (такого как файл с расширением dll – динамическая библиотека, или exe – исполняемый файл приложения).» [10]

2.2.3. WPF (Windows Presentation Foundation)

«Технология WPF – часть экосистемы платформы .NET, представляющая собой подсистему для построения графических интерфейсов.

Если при создании традиционных приложений на основе WinForms за отрисовку элементов управления и графики отвечали такие части ОС Windows, как User32 и GDI+, то приложения WPF основаны на DirectX. В этом состоит ключевая особенность рендеринга графики в WPF: используя WPF, значительная часть работы по отрисовке графики, как простейших кнопочек, так и сложных 3D-моделей, ложиться на графический процессор на видеокарте, что также позволяет воспользоваться аппаратным ускорением графики.

Преимущества WPF:

1. Использование традиционных языков .NET-платформы – C# и VB.NET для создания логики приложения.

2. Возможность декларативного определения графического интерфейса с помощью специального языка разметки XAML, основанном на xml и представляющем альтернативу программному созданию графики и элементов управления, а также возможность комбинировать XAML и C#/VB.NET.

3. Независимость от разрешения экрана: поскольку в WPF все элементы измеряются в независимых от устройства единицах, приложения на WPF легко масштабируются под разные экраны с разным разрешением.

4. Новые возможности, которых сложно было достичь в WinForms, например, создание трехмерных моделей, привязка данных, использование таких элементов, как стили, шаблоны, темы и др.

5. Хорошее взаимодействие с WinForms, благодаря чему, например, в приложениях WPF можно использовать традиционные элементы управления из WinForms.

6. Богатые возможности по созданию различных приложений: это и мультимедиа, и двухмерная и трехмерная графика, и богатый набор встроенных элементов управления, а также возможность самим создавать новые элементы, создание анимаций, привязка данных, стили, шаблоны, темы и многое другое

7. Аппаратное ускорение графики – вне зависимости от того, идет ли работа с 2D или 3D, графикой или текстом, все компоненты приложения транслируются в объекты, понятные Direct3D, и затем визуализируются с помощью процессора на видеокарте, что повышает производительность, делает графику более плавной.

8. Создание приложений под множество ОС семейства Windows – от Windows XP до Windows 10

В тоже время WPF имеет определенные ограничения. Несмотря на поддержку трехмерной визуализации, для создания приложений с большим количеством трехмерных изображений, прежде всего игр, лучше использовать другие средства - DirectX или специальные фреймворки, такие как Monogame или Unity.

Также стоит учитывать, что по сравнению с приложениями на Windows Forms объем программ на WPF и потребление ими памяти в процессе работы в среднем несколько выше. Но это с лихвой компенсируется более широкими графическими возможностями и провышенной производительностью при отрисовке графики.

При разработке графического интерфейса на основе технологии WPF используется XAML (eXtensible Application Markup Language) – язык разметки,

используемый для инициализации объектов в технологиях на платформе .NET. Применительно к WPF данный язык используется прежде всего для создания пользовательского интерфейса декларативным путем, аналогично HTML в веб-программировании.

Использование XAML несет некоторые преимущества:

1. Возможность отделить графический интерфейс от логики приложения, благодаря чему над разными частями приложения могут относительно автономно работать разные специалисты: над интерфейсом – дизайнеры, над кодом логики – программисты.

2. Компактность, понятность, код на XAML относительно легко поддерживать.

При компиляции приложения в Visual Studio код в xaml-файлах также компилируется в бинарное представление кода xaml, которое называется BAML (Binary Application Markup Language). И затем код baml встраивается в финальную сборку приложения - exe или dll-файл.» [11]

2.2.4. Microsoft SQL Server

«MS SQL Server – одна из наиболее популярных систем управления базами данных в мире. Данная СУБД подходит для самых различных проектов: от небольших приложений до больших высоконагруженных проектов.

Для организации баз данных MS SQL Server использует реляционную модель. Она предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая строка хранит отдельный объект, а в столбцах размещаются атрибуты этого объекта.

Для идентификации каждой строки в рамках таблицы применяется первичный ключ (primary key). В качестве первичного ключа может выступать один или несколько столбцов.

Через ключи одна таблица может быть связана с другой, а сама таблица может быть представлена в виде отношения (relation).

Для взаимодействия с базой данных применяется язык SQL (Structured Query Language). Клиент (например, внешняя программа) отправляет запрос на языке SQL посредством специального API. СУБД должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения.» [12]

2.2.5. Entity Framework Core

«Entity Framework Core (EF Core) представляет собой объектно-ориентированную, легковесную и расширяемую технологию от компании Microsoft для доступа к данным. EF Core является ORM-инструментом (object-relational mapping – отображения данных на реальные объекты). То есть EF Core позволяет работать базами данных, но представляет собой более высокий уровень абстракции: EF Core позволяет абстрагироваться от самой базы данных и ее таблиц и работать с данными независимо от типа хранилища.

Entity Framework Core поддерживает множество различных систем баз данных. На данный момент Microsoft предоставляет ряд встроенных провайдеров: для работы с MS SQL Server, для SQLite, для PostgreSQL. Также имеются провайдеры от сторонних поставщиков, например, для MySQL.

Как технология доступа к данным Entity Framework Core может использоваться на различных платформах стека .NET. Это и стандартные платформы типа Windows Forms, консольные приложения, WPF, UWP и ASP.NET Core. При этом кроссплатформенная природа EF Core позволяет задействовать ее не только на ОС Windows, но и на Linux и Mac OS X.

Центральной концепцией Entity Framework является понятие сущности или entity. Сущность определяет набор данных, которые связаны с определенным объектом. Поэтому данная технология предполагает работу не с таблицами, а с объектами и их коллекциями.

Любая сущность, как и любой объект из реального мира, обладает рядом свойств. Например, если сущность описывает человека, то мы можем выделить такие свойства, как имя, фамилия, рост, возраст. Свойства необязательно

представляют простые данные типа int или string, но могут также представлять и более комплексные типы данных. И у каждой сущности может быть одно или несколько свойств, которые будут отличать эту сущность от других и будут уникально определять эту сущность. Подобные свойства называют ключами.

При этом сущности могут быть связаны ассоциативной связью один-ко-многим, один-ко-одному и многие-ко-многим, подобно тому, как в реальной базе данных происходит связь через внешние ключи.

Все сущности, с которыми работает Entity Framework Core, определяются в виде классов моделей. При этом EF Core использует ряд условностей для сопоставления классов моделей с таблицами. Например, названия столбцов должны соответствовать названиям свойств и т.д. В этом случае Entity Framework сможет сопоставить столбцы таблицы и свойства модели.

Отличительной чертой Entity Framework Core, как технологии ORM, является использование запросов LINQ для выборки данных из БД. С помощью LINQ мы можем создавать различные запросы на выборку объектов, в том числе связанных различными ассоциативными связями. А Entity Framework при выполнении запроса транслирует выражения LINQ в выражения, понятные для конкретной СУБД (как правило, в выражения SQL).» [13]

3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ

3.1. Проектирование базы данных

Целью разработки любой базы данных является хранение и использование информации о какой-либо предметной области.

Логическая модель описывает понятия предметной области, их взаимосвязь, а также ограничения на данные, налагаемые предметной областью. Она строится в терминах информационных единиц, но без привязки к конкретной СУБД. Основным средством разработки логической модели данных в настоящий момент являются различные варианты ER-диаграмм (Entity-Relationship, диаграммы сущность-связь).

Физическая модель данных находится на более низком уровне. Физическая модель данных описывает данные средствами конкретной СУБД. Отношения, разработанные на стадии формирования логической модели данных, преобразуются в таблицы, атрибуты становятся столбцами таблиц, для ключевых атрибутов создаются уникальные индексы, домены преображаются в типы данных, принятые в конкретной СУБД.

3.1.1. Классы WMI

В процессе разработки программы был выделен ряд классов WMI и их свойств, представляющих интерес для целей инвентаризации. На основе этих классов были созданы таблицы базы данных.

Класс Win32_BaseBoard (Таблица 6) представляет основную плату (которую также называют материнской или системной платой).

Таблица 6. Win32_BaseBoard

Свойство	Описание свойства	Тип данных
Manufacturer	Название производителя	String
Product	Название модели	String

Класс Win32_DiskDrive (Таблица 7) представляет физический дисковый накопитель с точки зрения компьютера с операционной системой Win32.

Таблица 7. Win32_DiskDrive

Свойство	Описание свойства	Тип данных
InterfaceType	Тип интерфейса физического диска	String
MediaType	Тип носителя	String
Model	Имя модели диска	String
Size	Размер диска	UInt64

Класс Win32_NetworkAdapterConfiguration (Таблица 8) представляет атрибуты и поведение сетевого адаптера.

Таблица 8. Win32_NetworkAdapterConfiguration

Свойство	Описание свойства	Тип данных
IPAddress	IP адрес(а) адаптера	String[]
MACAddress	MAC адрес адаптера	String

Класс Win32_OperatingSystem (Таблица 9) представляет операционную систему, установленную на компьютерной системе Win32.

Таблица 9. Win32_OperatingSystem

Свойство	Описание свойства	Тип данных
Caption	Имя	String
Version	Пакет обновления	String

Класс Win32_PhysicalMemory (Таблица 10) представляет физическое устройство памяти, установленное в компьютере, с точки зрения операционной системы.

Таблица 10. Win32_PhysicalMemory

Свойство	Описание свойства	Тип данных
BankLabel	Имя слота оперативной памяти	String
Capacity	Емкость оперативной памяти	UInt64
DeviceLocator	Расположение модуля на материнской плате	String
FormFactor	Форм-фактор модуля	UInt16
Manufacturer	Изготовитель памяти	String
SMBIOSMemoryType	Тип памяти	UInt32
Speed	Скорость памяти	UInt32

Класс Win32_Processor (Таблица 11) представляет устройство, способное интерпретировать последовательность машинных инструкций в системе Win32. В мультипроцессорной системе для каждого из процессоров существует отдельный экземпляр этого класса.

Таблица 11. Win32_Processor

Свойство	Описание свойства	Тип данных
Manufacturer	Изготовитель процессора	String
Name	Модель процессора	String
NumberOfCores	Количество ядер	UInt32
NumberOfEnabledCore	Количество включенных ядер	UInt32
NumberOfLogicalProcessors	Количество логических процессоров	UInt32
SocketDesignation	Сокет процессора	String
VirtualizationFirmwareEnabled	Виртуализация включена	Boolean

Класс Win32_UserProfile (Таблица 12) содержит данные о профилях пользователей в системе Windows.

Таблица 12. Win32_UserProfile

Свойство	Описание свойства	Тип данных
LocalPath	Путь к профилю пользователя на локальном компьютере	String
SID	ИД безопасности пользователя	String

Класс Win32_LogicalDisk (Таблица 13) представляет источник данных, которым является локальное устройство хранения данных на компьютере с системой Win32. Класс возвращает как локальные, так и подключенные логические диски.

Таблица 13. Win32_LogicalDisk

Свойство	Описание свойства	Тип данных
DriveType	Тип логического диска	UInt32
FileSystem	Тип файловой системы	String
FreeSpace	Объем в байтах доступного дискового пространства	UInt64
Name	Путь к логическому устройству	String
Size	Размер диска в байтах	UInt64

Класс Win32_ComputerSystem (Таблица 14) описывает компьютерную систему, работающую в среде Win32.

Таблица 14. Win32_ComputerSystem

Свойство	Описание свойства	Тип данных
TotalPhysicalMemory	Общий размер физической памяти	Uint64

3.1.2. Логическая и физическая модели базы данных

На основе приведенных в разделе 3.1.1 классов WMI выделены следующие сущности логической модели (Рисунок 11):

1. основная информация;
2. информация о материнской плате;
3. информация о процессоре;
4. информация о сетевом адаптере;
5. информация об операционной системе;
6. общая информация об оперативной памяти;
7. подробная информация об оперативной памяти;
8. информация о логических дисках;
9. информация о физических дисках;
10. информация о профилях пользователей компьютера.

Основной таблицей является MainInfo, с ней связаны все остальные сущности (связями один-ко-многим и один-к-одному). В качестве ID для таблицы используется доменное имя компьютера, т. к. в рамках домена каждое имя компьютера является уникальным. ID остальных таблиц генерируется автоматически на уровне базы данных.

Так как физическая модель ориентирована на СУБД, в ней появляются таблицы (Таблица 15), реальные типы данных, ограничения и т. д.

Таблица 15. Соответствие между логической и физической моделями базы данных

Сущность логической модели	**Таблица физической модели**
Основная информация	MainInfo
Информация о материнской плате	BaseBoard
Информация о процессоре	Processor
Информация о сетевом адаптере	NetworkAdapter
Информация об операционной системе	OperatingSystem
Общая информация об оперативной памяти	TotalMemory
Подробная информация об оперативной памяти	PhysicalMemory
Информация о логических дисках	LogicalDisk
Информация о физических дисках	DiskDrive
Информация о профилях пользователей компьютера	UserProfile

Скрипт генерации базы данных на языке T-SQL представлен в приложении 1 на стр.76. Физическая модель данных представлена в приложении 2 на стр. 78.

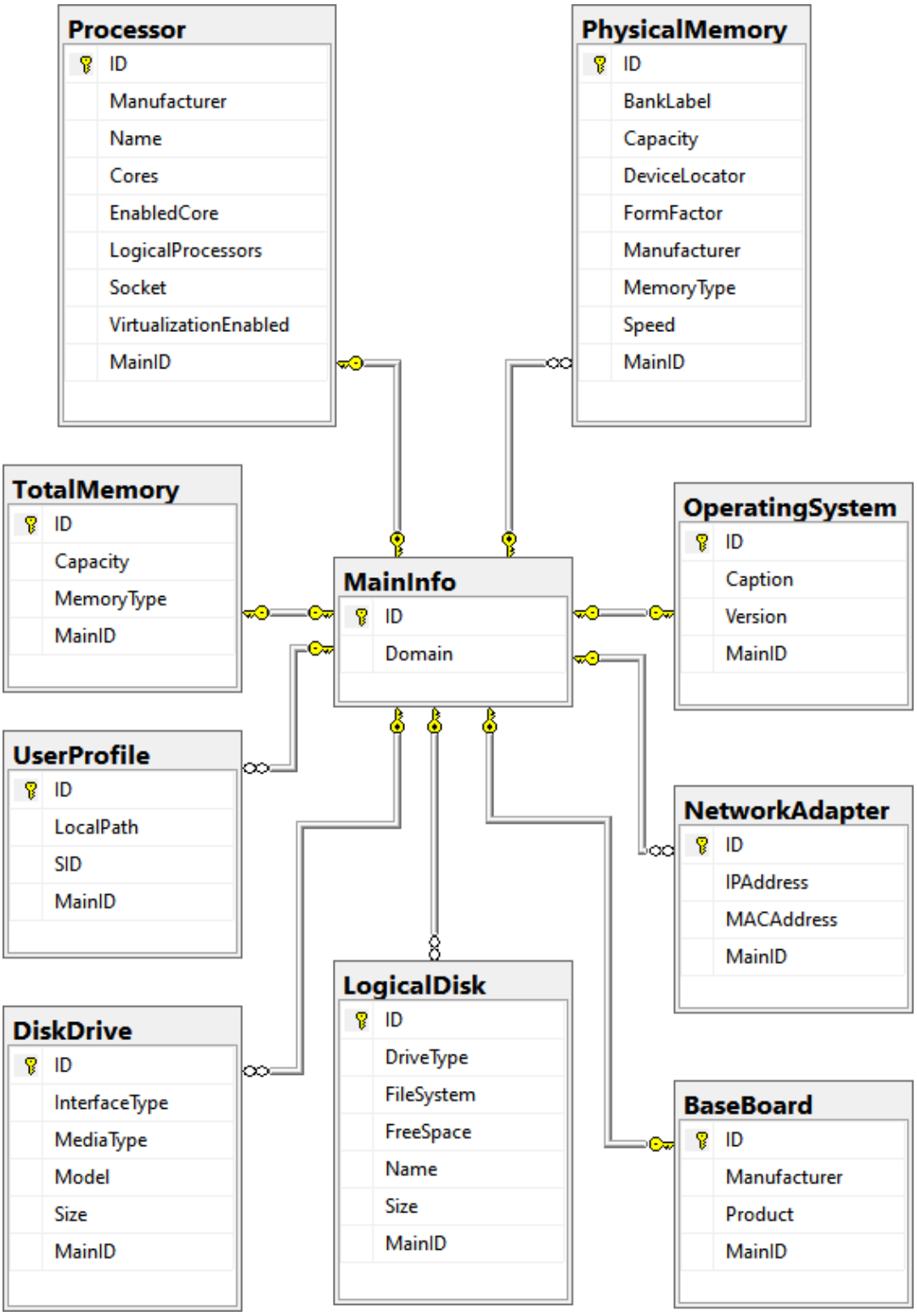


Рисунок 11. Логическая модель базы данных

3.2. Подключение проекта к базе данных

Для начала работы с Entity Framework Core необходимо добавить в проект пакет Entity Framework Core. Для добавления пакетов можно использовать Консоль диспетчера пакетов. Для этого в меню Visual Studio необходимо перейти к пункту Средства – Диспетчер пакетов NuGet – Консоль диспетчера пакетов (Рисунок 12).

В окне консоли диспетчера пакетов нужно ввести команду:

```
Install-Package Microsoft.EntityFrameworkCore.SqlServer
```

После выполнения этой команды требуется выполнить вторую команду:

```
Install-Package Microsoft.EntityFrameworkCore.Tools
```

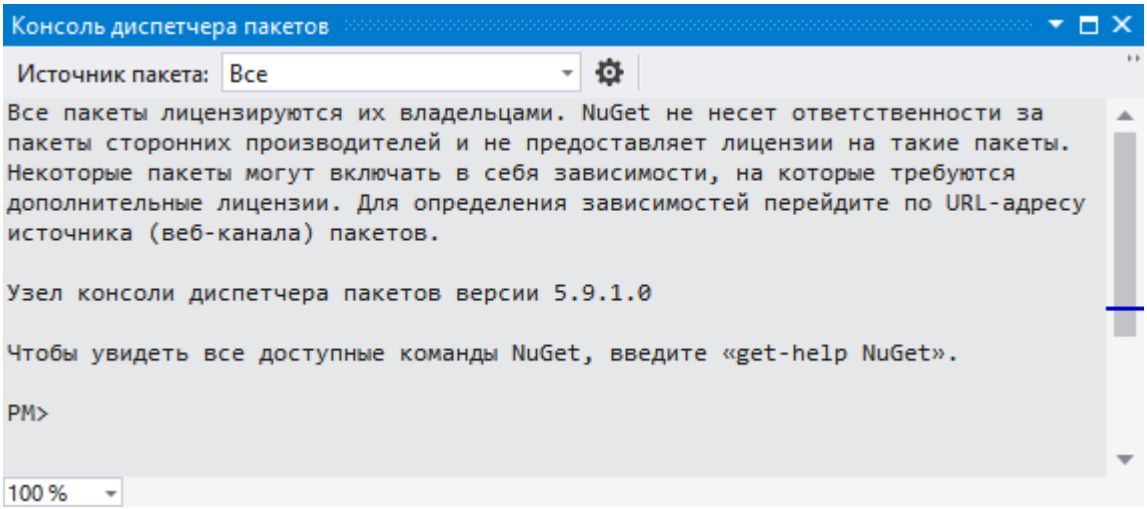


Рисунок 12. Консоль диспетчера пакетов

При работе с EF Core можно как создать базу напрямую из кода, добавив в проект класс контекста данных и классы моделей, которые соответсвуют базе данных и определениям таблиц, так и подключиться к существующей базе.

При разработке проекта был выбран вариант подключения к существующей базе, предварительно созданной при помощи MS SQL Server Management Studio (SSMS).

Для этого нужно в консоли диспетчера пакетов выполнить следующую команду:

```
Scaffold-DbContext
"Server=(localdb)\mssqllocaldb;Database=Computers;Trusted_Connection
=True;" Microsoft.EntityFrameworkCore.SqlServer
```

Здесь в качестве параметра команде Scaffold-DbContext передается строка подключения с указанием сервера и названием базы данных.

Созданная в SSMS база данных располагается локально в папке текущего пользователя (%USERPROFILE%) и имеет имя Computers.mdf. В строке подключения имя базы указывается без расширения «.mdf».

После этого в проекте появятся автоматически созданные классы моделей и класс контекста данных. Структура классов моделей соответствует структуре базы данных. Для удобства навигации в проекте они были перенесены в папку DB (Рисунок 13).

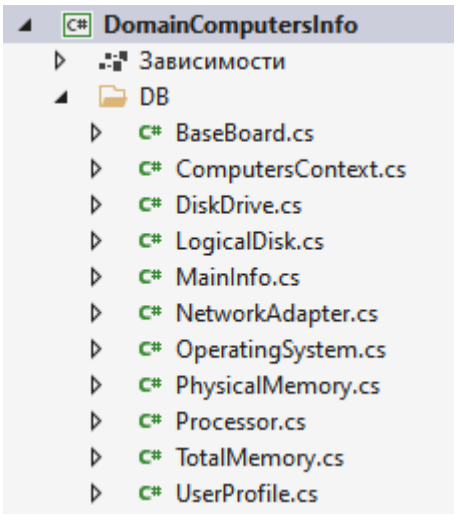


Рисунок 13. Классы моделей и класс контекста данных

Диаграммы сгенерированных классов приведены в приложении 3 на стр.79.

3.3. Разработка логики программы

Для обработки данных были выделены следующие действия:

1. получение списка компьютеров из Active Directory;
2. получение информации из WMI и запись ее в базу;
3. получение информации из базы данных;
4. вычисление сводной статистики.

Для реализации этих действий были разработаны соответствующие классы, исходный код которых приведен в приложении 4 на стр.80.

3.3.1. Получение списка компьютеров

Для получения списка компьютеров из был разработан класс GetComputers (Рисунок 14), содержащий два метода – GetFromAD для получения списка из Active Directory и GetManual для ввода имен компьютеров вручную.

Для работы с AD необходимо добавить в проект пакет System.DirectoryServices.AccountManagement. Это пространство имен обеспечивает унифицированный доступ и управление участниками безопасности (пользователь, компьютер, группа), находящимися в доменных службах Active Directory.

Метод GetFromAD получает строку с именем домена, а возвращает список строк с именами компьютеров. Пользователь, запускающий программу, должен иметь права на доступ к этому домену.

В методе используются следующие классы пространства имен System.DirectoryServices.AccountManagement:

1. PrincipalContext – инкапсулирует сервер или домен, в отношении которого выполняются все операции, контейнер, используемый в качестве базы этих операций, и учетные данные, используемые для выполнения операций.

2. ComputerPrincipal – инкапсулирует участников, которые являются учетными записями компьютера.

3. PrincipalSearcher – инкапсулирует методы и шаблоны поиска, используемые для выполнения запросов к базовому хранилищу участников.

Также используется метод FindAll класса PrincipalSearcher, возвращающий результат поиска участника, который содержит коллекцию всех объектов-участников, соответствующих участнику, указанному в свойстве фильтра запроса.

После создания экземпляра класса PrincipalSearcher с помощью метода FindAll происходит поиск среди всех элементов, и, если элемент соответствует условию «Включено», он добавляется в список строк. После окончания цикла поиска метод возвращает полученный список.

Метод GetManual получает строки с именем домена и именем компьютера, а также список имен. Далее происходит обработка строки с именем компьютера для соответствия формату, после чего строка добавляется в список и возвращает список.

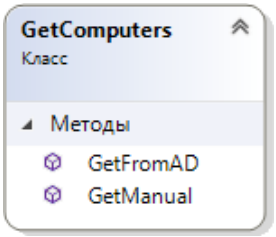


Рисунок 14. Диаграмма класса GetComputers

3.3.2. Получение информации из WMI и работа с базой данных

Для получения информации из WMI и работы с базой были разработаны классы GetFromWMI и WorkWithDB (Рисунок 15). Для работы с WMI необходимо добавить в проект пакет System.Management. Это пространство имен предоставляет доступ к обширному набору сведений и событий управления, относящихся к системе, устройствам и приложениям, поддерживающим инфраструктуру WMI. Для доступа используется класс ManagementObject, представляющий экземпляр WMI – объект управления.

При необходимости пересоздать базу данных (например, в целях тестирования) используется метод RecreateDB класса WorkWithDB. Для удаления конкретной записи базы данных используется метод DeleteEntry.

Для доступа к конкретным классам WMI имена этих классов выведены в поле Path класса WMI, которое представляет собой список строк с этими именами.

Метод GetAllInfo класса WorkWithDB получает список компьютеров, полученный из Active Directory, а также имя пользователя, имеющего администраторские привилегии на этих компьютерах, и его пароль. В методе создается подключение для доступа к WMI. После этого определяется использование контекста базы данных.

Далее для каждого имени компьютера программа пытается установить подключение к этому компьютеру. Если этого не происходит, срабатывает обработка исключений – для случаев невозможности установить соединение (например, компьютер отключен от сети) и для случаев отсутствия привилегированного доступа у пользователя, и программа переходит к следующему компьютеру из списка.

При удачном подключении к компьютеру в базу добавляется сначала элемент таблицы MainInfo, состоящий из полей имя компьютера (оно же служить внешним ключом для других таблиц) и названия домена.

После этого для каждого класса WMI в цикле создается экземпляр класса управления CIM, и для каждого объекта управления происходит выбор таблицы базы данных, в которую будет записана информация. Для этого предусмотрено соответствие класса WMI и таблицы БД, что обеспечивается циклом со счетчиком for и условной конструкцией switch/case – значение счетчика соответствует варианту условной конструкции.

Далее происходит вызов соответствующего метода класса GetFromWMI и запись полученного значения в базу. Все методы GetFromWMI класса действуют по одному принципу – получают имя объекта управления и имя компьютера,

создают экземпляр класса модели базы данных, заполняют его интересующими данными из WMI и возвращают этот экземпляр класса.

Также предусмотрена обработка исключений – в таком случае данные сначала записываются в поля специально созданного статического класса, либо при получении исключения заполняются пустой строкой или нулем, а после этого уже создается экземпляр класса модели БД.

Важным моментом является то, что данные в БД заносятся в том виде, в котором получены из WMI, т.е. не являются нормализованными.

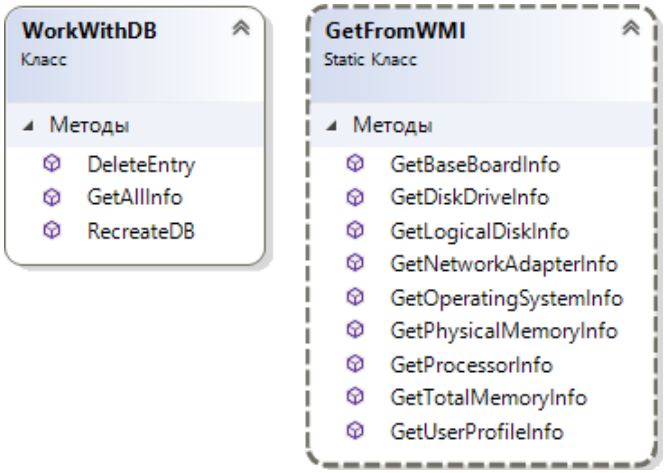


Рисунок 15. Диаграммы классов WorkWithDB и GetFromWMI

3.3.3. Получение информации из базы данных

Для получения данных из базы и вывода их пользователю были разработаны классы GetFromDB и Convert (Рисунок 16).

Для вывода основной информации класс GetFromDB содержит метод GetMainInfo, принимающий экземпляр ListView – элемента управления WPF, отображающего информацию на множестве строк и столбцов, т.е. в табличном виде. В этом методе сначала происходит запрос к базе данных на языке LINQ, который для каждого элемента MainInfo (т.е. для каждого компьютера в базе) получает связанные по ключу данные из других таблиц.

Метод GetAllInfo класса GetFromDB предназначен для вывода подробной информации о конкретном компьютере в элементы управления ListBox. Метод

обращается к другим методам (GetMbProcessor и т.д.), которые принимают соответствующие элементы ListBox и контекст базы данных, производят запрос к базе на языке LINQ и выводят полученные данные в элементы ListBox.

Так как данные записываются в базу в том виде, в котором они представлены во WMI, они не всегда представлены в удобном для пользователя виде (например, объем оперативной памяти и жестких дисков отображаются в байтах), поэтому для вывода данных пользователю они предварительно преобразуются методами класса Convert.

В классе Convert производятся следующие преобразования данных:

1. Конвертация байтов в гигабайты.
2. Конвертация числовых значений типов и форм-фактора оперативной памяти, а также типов дисков в общепринятые обозначения.
3. Конвертация номеров сборок Windows 10 в кодовые имена.

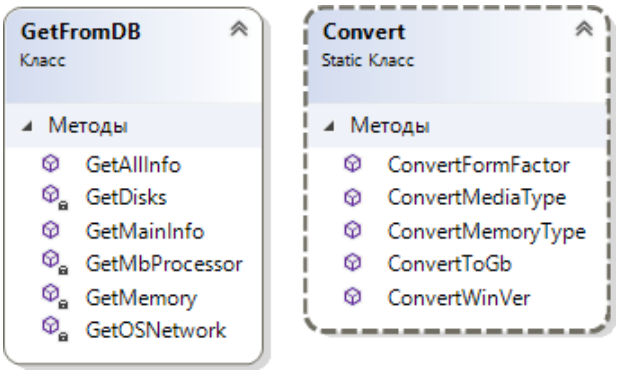


Рисунок 16. Диаграммы классов DisplayInfo, GetAllFromDB и Convert

3.3.4. Вычисление сводной статистики

На практике администраторам было бы полезно знать знать сводную статистику компонентов, используемых на предприятии, например, соотношение различных версий Windows или моделей процессоров.

Было установлено, что предприятии в основном используются:

1. процессоры Intel Core i3, i5, i7 и i9 различных поколений;
2. сокеты процессоров LGA1156, LGA1155, LGA1150 и LGA1151;
3. функция аппаратной виртуализации;

4. оперативная память DDR3 и DDR4 емкостью 4, 8, 16, 32 или 64 Гб;

5. операционные системы Windows 7 и Windows 10 редакций «Профессиональная» и «Корпоративная».

На основании этой информации был разработан класс Statistics (Рисунок 17), в котором производится подсчет статистики.

Метод GetFromDB этого класса производит запрос к базе данных на языке LINQ, создает списки соответствующих типов и добавляет в них полученные данные. После этого происходит обращение к методам получения статистики. Выделены методы для статистики по процессорам (модели процессоров, сокеты и виртуализация), по оперативной памяти (емкость и тип) и по операционной системе (версия и редакция).

Методы строятся по следующему принципу: создается кортеж целочисленных переменных, их начальные значения инициализируются нулями, после чего происходит инкрементация этих переменных в зависимости от того, какое значение обнаружено в списке, куда помещены значения из базы данных. Для строкового типа используется метод Contains, возвращающий true, если строка содержит определенную подстроку, для числовых значений происходит непосредственно сравнение, а для булевого типа он используется как условие. После метод возвращает список со подсчитанными значениями.

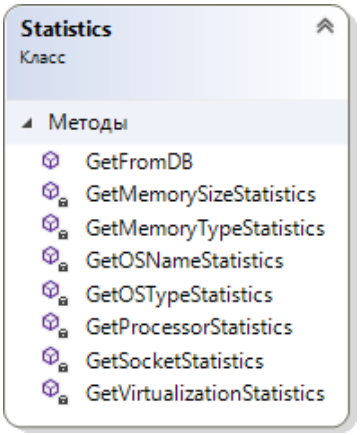


Рисунок 17. Диаграмма класса Statistics

3.4. Разработка интерфейса программы

При разработке информационной системы был создан ряд окон (Рисунок 18), навигация между которыми осуществляется при помощи кнопок или контекстного меню. При запуске программы открывается окно «Информация о компьютере», из которого можно открыть три окна: «Работа с базой данных», «Общая информация о компьютерах» и «Вывод статистики».

При открытии окна «Работа с базой данных» можно перейти к окнам «Подтверждение удаления» и «Выбор способа работы с базой», из которого в зависимости от выбора можно перейти либо к окну «Список компьютеров» и после к окну «Учетные данные», либо сразу к окну «Учетные данные».

Окно «Общая информация о компьютерах» предоставляет возможность перейти к окну «Детальная информация о компьютере» с помощью контекстного меню.

Из окна «Вывод статистики» возможно открыть окна «Статистика по процессорам», «Статистика по операционным системам» и «Статистика по оперативной памяти».

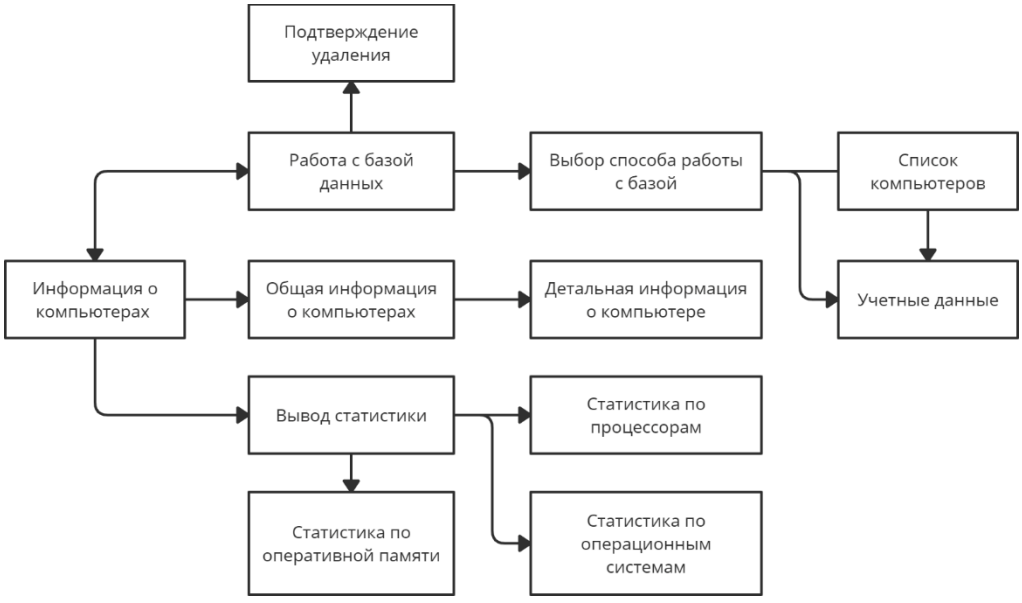


Рисунок 18. Общая структура окон приложения

Исходный код классов графического интерфейса приведен в приложении 5 на стр.93.

3.4.1. Информация о компьютерах

Главное окно программы (Рисунок 19) имеет название «Информация о компьютерах» и три кнопки (Работа с базой данных, Общая информация и Вывод статистики), при нажатии на которые открываются соответствующие окна.

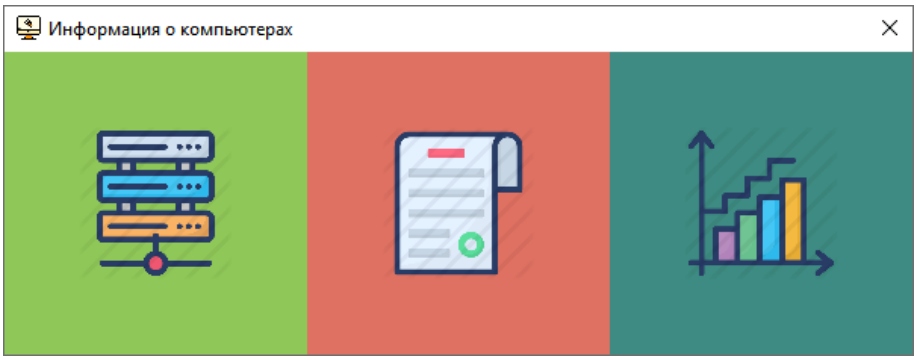


Рисунок 19. Окно «Информация о компьютерах»

Так как при проектировании графического интерфейса в качестве ориентира был выбран стиль оформления Metro UI, отличительной особенностью которого являются элементы управления в виде плоских прямоугольных плиток, то каждая кнопка имеет всплывающую подсказку о действии, которое выполняется при нажатии (Рисунок 20, Рисунок 21, Рисунок 22).

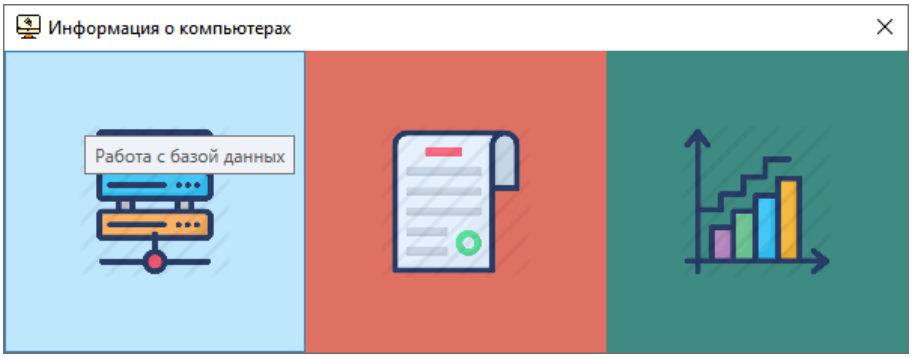


Рисунок 20. Всплывающая подсказка кнопки «Работа с базой данных»

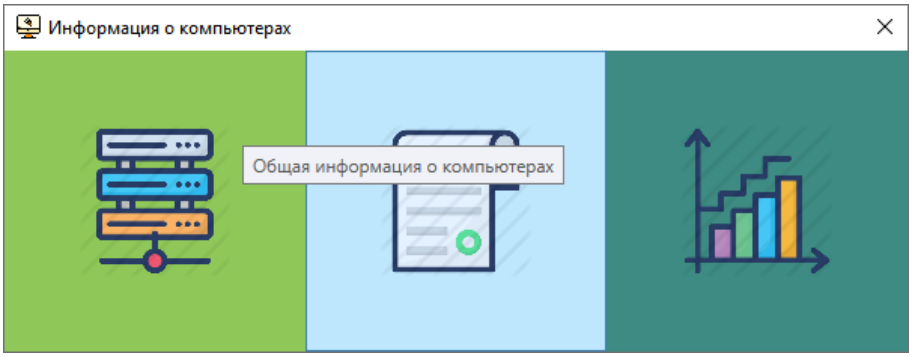


Рисунок 21. Всплывающая подсказка кнопки «Общая информация о компьютерах»

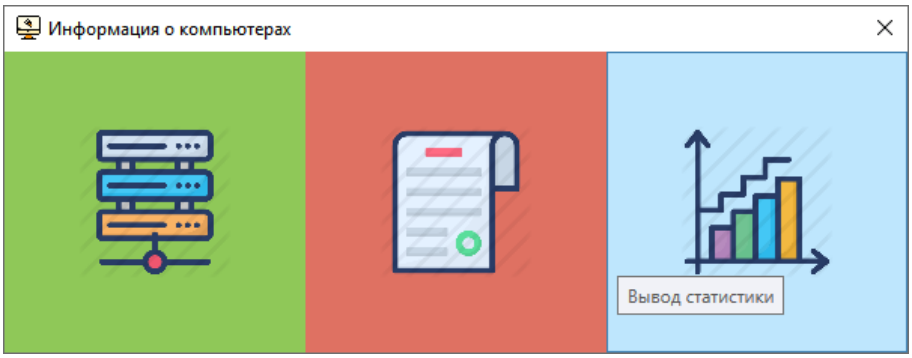


Рисунок 22. Всплывающая подсказка кнопки «Вывод статистики»

3.4.2. Работа с базой данных

Окно работы с базой данных (Рисунок 23) имеет название «Работа с базой данных» и имеет две кнопки («Создать базу данных» и «Заполнить базу данных»), при нажатии на которые открываются соответствующие окна.

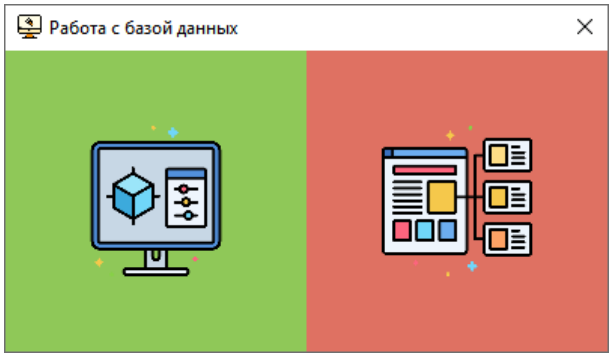


Рисунок 23. Окно «Работа с базой данных»

Каждая кнопка также имеет всплывающую подсказку о действии, которое выполняется при нажатии (Рисунок 24, Рисунок 25).

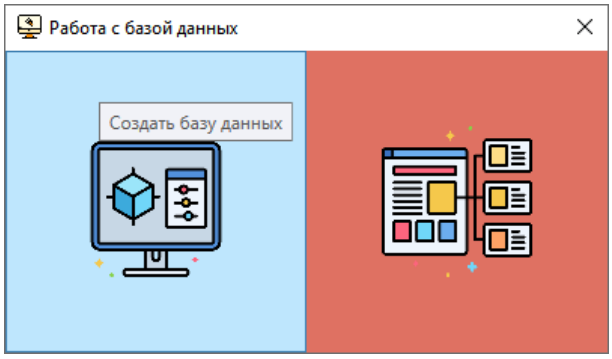


Рисунок 24. Всплывающая подсказка кнопки «Создать базу данных»

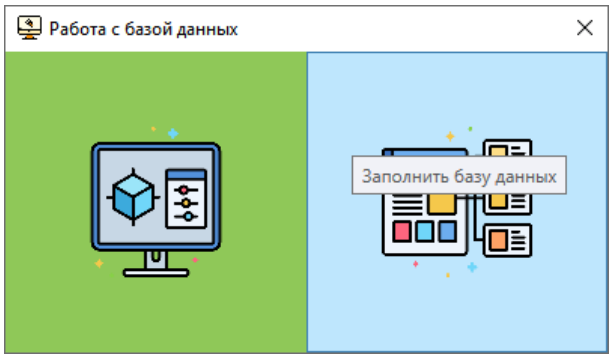


Рисунок 25. Всплывающая подсказка кнопки «Заполнить базу данных»

При нажатии на кнопку «Создать базу данных» появляется окно «Подтверждение» (Рисунок 26), где нужно подтвердить удаление, т.к. если база уже существует, при повторном создании вся информация будет удалена.

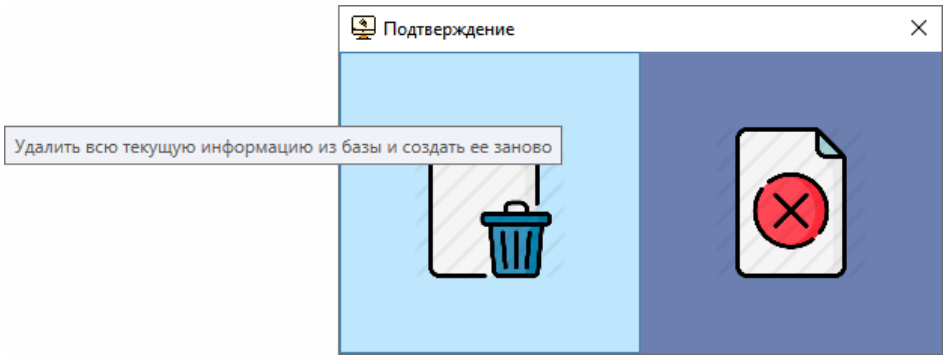


Рисунок 26. Подтверждение удаления

При нажатии на кнопку «Заполнить базу данных» откроется окно (Рисунок 27), в котором можно выбрать способ заполнения базы.

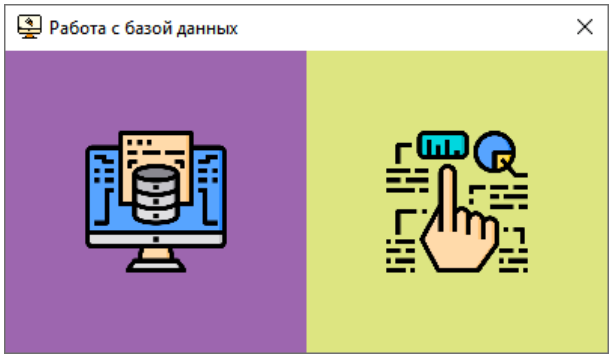


Рисунок 27. Окно «Заполнение базы данных»

Предусмотрено два варианта заполнения базы данных – импортировать список имен компьютеров из домена Active Directory (Рисунок 28) либо ввести список компьютеров вручную (Рисунок 29).

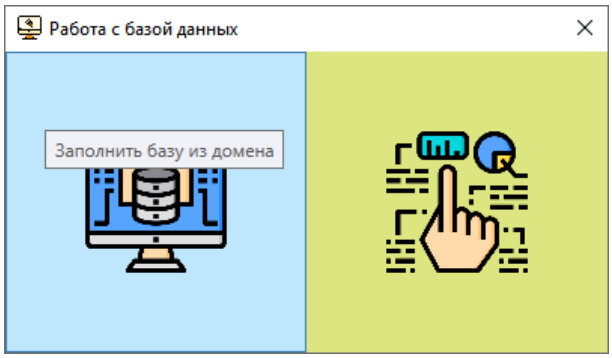


Рисунок 28. Кнопка «Заполнить базу из домена»

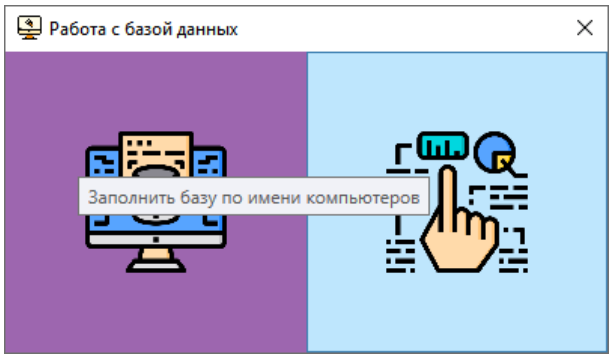


Рисунок 29. Кнопка «Заполнить базу по имени компьютеров»

Если выбрать кнопку «Заполнить базу по имени компьютеров», программа предложить ввести имя компьютера и его домен в окне «Список компьютеров» (Рисунок 30).

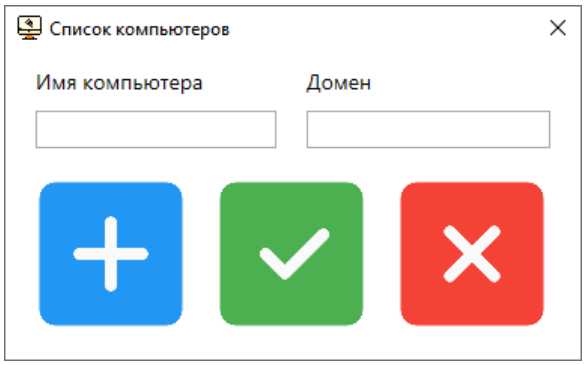


Рисунок 30. Окно ввода имени компьютера

После ввода данных нужно нажать кнопку «Добавить компьютер в список» (Рисунок 31), и после этого либо продолжить вводить имена компьютеров», либо нажать кнопку «Внести компьютеры в базу» (Рисунок 32). Также предусмотрена отмена действия (Рисунок 33).

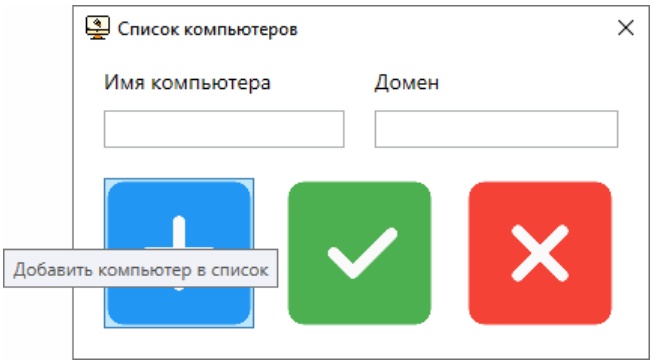


Рисунок 31. Добавление компьютера в список

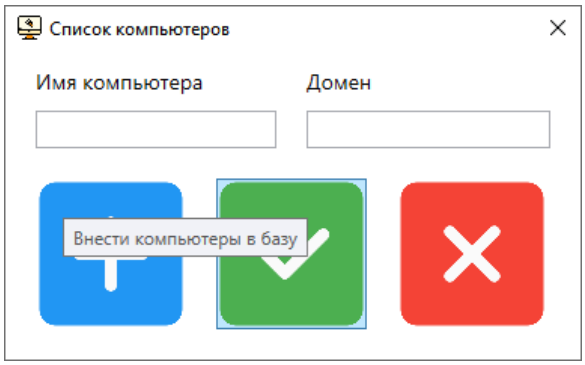


Рисунок 32. Заполнение базы по введенному списку компьютеров

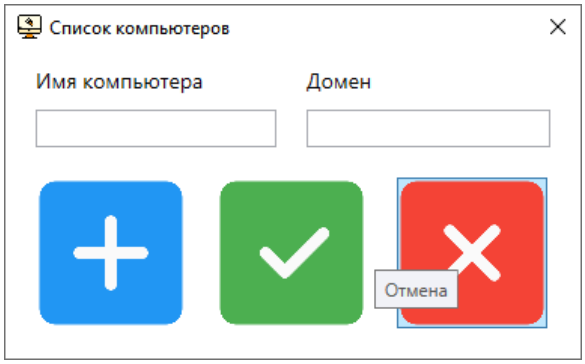


Рисунок 33. Отмена действия

Окно ввода учетных данных (Рисунок 34) открывается либо после нажатия кнопки «Внести компьютеры в базу» окна «Список компьютеров», либо после нажатия кнопки «Заполнить базу из домена» окна «Работа с базой данных».

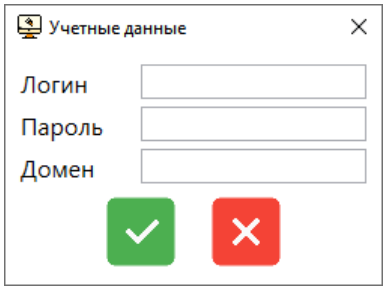


Рисунок 34. Окно ввода учетных данных

Программа запрашивает имя пользователя (логин), пароль и интересующий домен. Учетная запись должна иметь администраторский доступ к домену и права локального администратора на целевых копьютерах.

После успешного заполнения базы появляется уведомляющее окно (Рисунок 35).

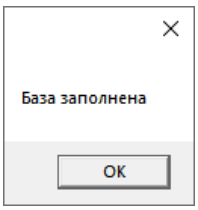
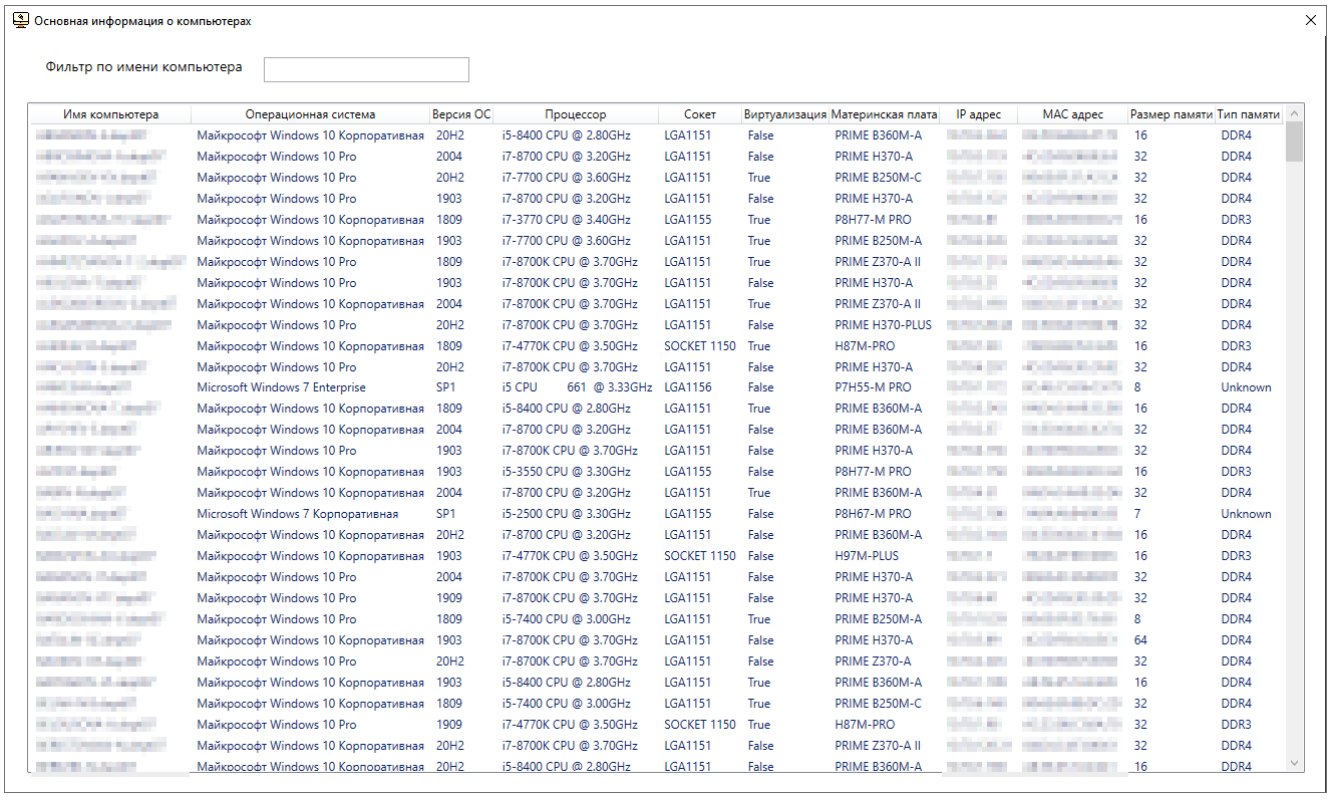


Рисунок 35. Подтверждающее сообщение

3.4.3. Общая информация о компьютерах

Окно «Общая информация о компьютерах» (Рисунок 36) содержит таблицу DataGrid, куда выводятся основные данные обо всех компьютерах в базе.



Имя компьютера	Операционная система	Версия ОС	Процессор	Сокет	Виртуализация	Материнская плата	IP адрес	MAC адрес	Размер памяти	Тип памяти
[illegible]	Майкрософт Windows 10 Корпоративная	20H2	i5-8400 CPU @ 2.80GHz	LGA1151	False	PRIME B360M-A	[illegible]	[illegible]	16	DDR4
[illegible]	Майкрософт Windows 10 Pro	2004	i7-8700 CPU @ 3.20GHz	LGA1151	False	PRIME H370-A	[illegible]	[illegible]	32	DDR4
[illegible]	Майкрософт Windows 10 Pro	20H2	i7-7700 CPU @ 3.60GHz	LGA1151	True	PRIME B250M-C	[illegible]	[illegible]	32	DDR4
[illegible]	Майкрософт Windows 10 Pro	1903	i7-8700 CPU @ 3.20GHz	LGA1151	False	PRIME H370-A	[illegible]	[illegible]	32	DDR4
[illegible]	Майкрософт Windows 10 Корпоративная	1809	i7-3770 CPU @ 3.40GHz	LGA1155	True	P8H77-M PRO	[illegible]	[illegible]	16	DDR3
[illegible]	Майкрософт Windows 10 Корпоративная	1903	i7-7700 CPU @ 3.60GHz	LGA1151	True	PRIME B250M-A	[illegible]	[illegible]	32	DDR4
[illegible]	Майкрософт Windows 10 Pro	1809	i7-8700K CPU @ 3.70GHz	LGA1151	True	PRIME Z370-A II	[illegible]	[illegible]	32	DDR4
[illegible]	Майкрософт Windows 10 Pro	1903	i7-8700K CPU @ 3.70GHz	LGA1151	False	PRIME H370-A	[illegible]	[illegible]	32	DDR4
[illegible]	Майкрософт Windows 10 Корпоративная	2004	i7-8700K CPU @ 3.70GHz	LGA1151	True	PRIME Z370-A II	[illegible]	[illegible]	32	DDR4
[illegible]	Майкрософт Windows 10 Pro	20H2	i7-8700K CPU @ 3.70GHz	LGA1151	False	PRIME H370-PLUS	[illegible]	[illegible]	32	DDR4
[illegible]	Майкрософт Windows 10 Корпоративная	1809	i7-4770K CPU @ 3.50GHz	SOCKET 1150	True	H87M-PRO	[illegible]	[illegible]	16	DDR3
[illegible]	Майкрософт Windows 10 Pro	20H2	i7-8700K CPU @ 3.70GHz	LGA1151	False	PRIME H370-A	[illegible]	[illegible]	32	DDR4
[illegible]	Microsoft Windows 7 Enterprise	SP1	i5 CPU 661 @ 3.33GHz	LGA1156	False	P7H55-M PRO	[illegible]	[illegible]	8	Unknown
[illegible]	Майкрософт Windows 10 Корпоративная	1809	i5-8400 CPU @ 2.80GHz	LGA1151	True	PRIME B360M-A	[illegible]	[illegible]	16	DDR4
[illegible]	Майкрософт Windows 10 Корпоративная	2004	i7-8700 CPU @ 3.20GHz	LGA1151	False	PRIME B360M-A	[illegible]	[illegible]	32	DDR4
[illegible]	Майкрософт Windows 10 Pro	1903	i7-8700K CPU @ 3.70GHz	LGA1151	False	PRIME H370-A	[illegible]	[illegible]	32	DDR4
[illegible]	Майкрософт Windows 10 Корпоративная	1903	i5-3550 CPU @ 3.30GHz	LGA1155	False	P8H77-M PRO	[illegible]	[illegible]	16	DDR3
[illegible]	Майкрософт Windows 10 Корпоративная	2004	i7-8700 CPU @ 3.20GHz	LGA1151	True	PRIME B360M-A	[illegible]	[illegible]	32	DDR4
[illegible]	Microsoft Windows 7 Корпоративная	SP1	i5-2500 CPU @ 3.30GHz	LGA1155	False	P8H67-M PRO	[illegible]	[illegible]	7	Unknown
[illegible]	Майкрософт Windows 10 Корпоративная	20H2	i7-8700 CPU @ 3.20GHz	LGA1151	False	PRIME B360M-A	[illegible]	[illegible]	16	DDR4
[illegible]	Майкрософт Windows 10 Корпоративная	1903	i7-4770K CPU @ 3.50GHz	SOCKET 1150	False	H97M-PLUS	[illegible]	[illegible]	16	DDR3
[illegible]	Майкрософт Windows 10 Pro	2004	i7-8700K CPU @ 3.70GHz	LGA1151	False	PRIME H370-A	[illegible]	[illegible]	32	DDR4
[illegible]	Майкрософт Windows 10 Pro	1909	i7-8700K CPU @ 3.70GHz	LGA1151	False	PRIME H370-A	[illegible]	[illegible]	32	DDR4
[illegible]	Майкрософт Windows 10 Pro	1809	i5-7400 CPU @ 3.00GHz	LGA1151	True	PRIME B250M-A	[illegible]	[illegible]	8	DDR4
[illegible]	Майкрософт Windows 10 Корпоративная	1903	i7-8700K CPU @ 3.70GHz	LGA1151	False	PRIME H370-A	[illegible]	[illegible]	64	DDR4
[illegible]	Майкрософт Windows 10 Pro	20H2	i7-8700K CPU @ 3.70GHz	LGA1151	False	PRIME Z370-A	[illegible]	[illegible]	32	DDR4
[illegible]	Майкрософт Windows 10 Корпоративная	1903	i5-8400 CPU @ 2.80GHz	LGA1151	True	PRIME B360M-A	[illegible]	[illegible]	16	DDR4
[illegible]	Майкрософт Windows 10 Корпоративная	1809	i5-7400 CPU @ 3.00GHz	LGA1151	True	PRIME B250M-C	[illegible]	[illegible]	32	DDR4
[illegible]	Майкрософт Windows 10 Pro	1909	i7-4770K CPU @ 3.50GHz	SOCKET 1150	True	H87M-PRO	[illegible]	[illegible]	32	DDR3
[illegible]	Майкрософт Windows 10 Корпоративная	20H2	i7-8700K CPU @ 3.70GHz	LGA1151	False	PRIME Z370-A II	[illegible]	[illegible]	32	DDR4
[illegible]	Майкрософт Windows 10 Корпоративная	20H2	i5-8400 CPU @ 2.80GHz	LGA1151	False	PRIME B360M-A	[illegible]	[illegible]	16	DDR4

Рисунок 36. Окно информации о компьютерах

В таблице вывода данных предусмотрена фильтрация по имени компьютера (Рисунок 37) для поиска нужного элемента.

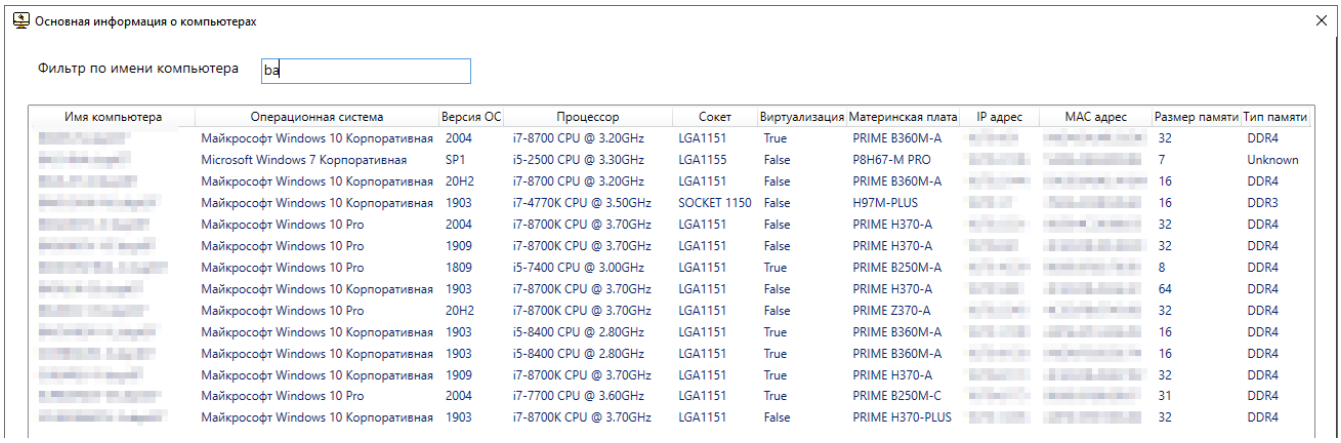


Рисунок 37. Фильтрация по имени компьютера

При выборе конкретной строки в таблице можно открыть контекстное меню (Рисунок 38), где доступно копирование имени компьютера, удаление записи из базы и переход к детальной информации.

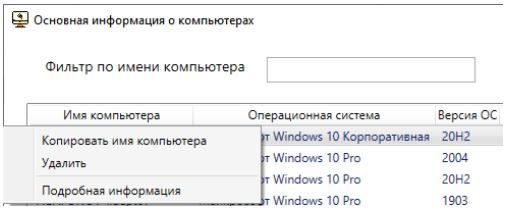


Рисунок 38. Контекстное меню

3.3.5. Детальная информация о компьютере

Окно детальной информации о компьютере содержит имя компьютера в качестве заголовка окна и вкладки с текстовыми блоками, куда выводится информация из базы – «Материнская плата и процессор» (Рисунок 39), «Операционная система и сетевой адаптер» (Рисунок 40), «Логические и физические диски» (Рисунок 41) и «Оперативная память» (Рисунок 42).

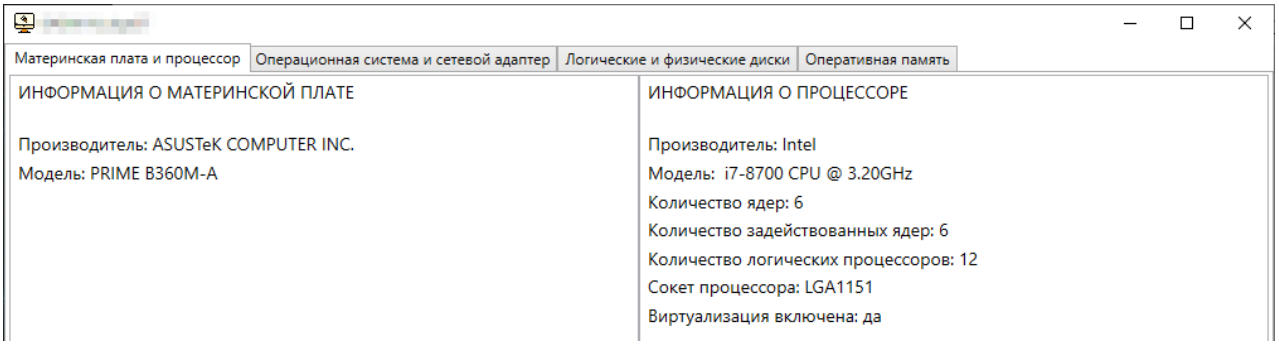


Рисунок 39. Детальная информация о материнской плате и процессоре

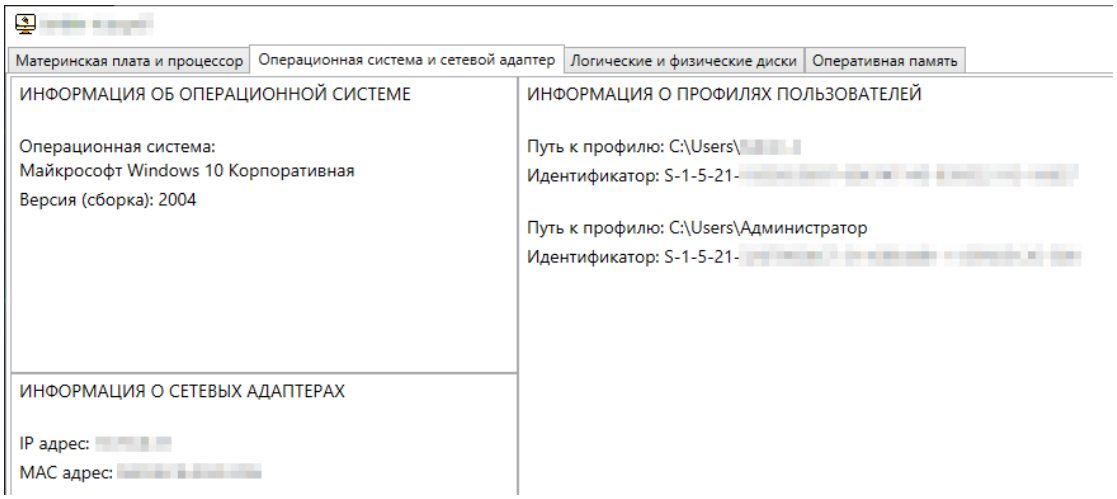


Рисунок 40. Детальная информация об операционной системе и сетевом адаптере

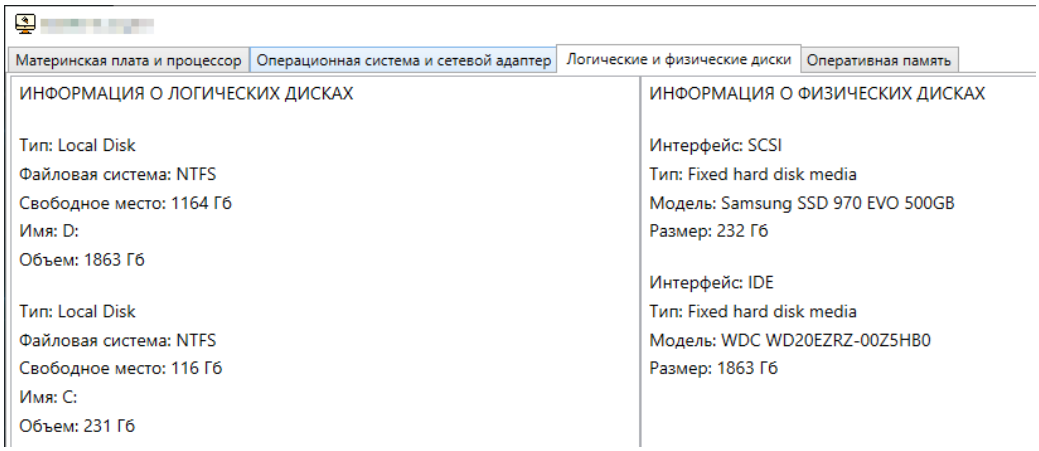


Рисунок 41. Детальная информация о логических и физических дисках

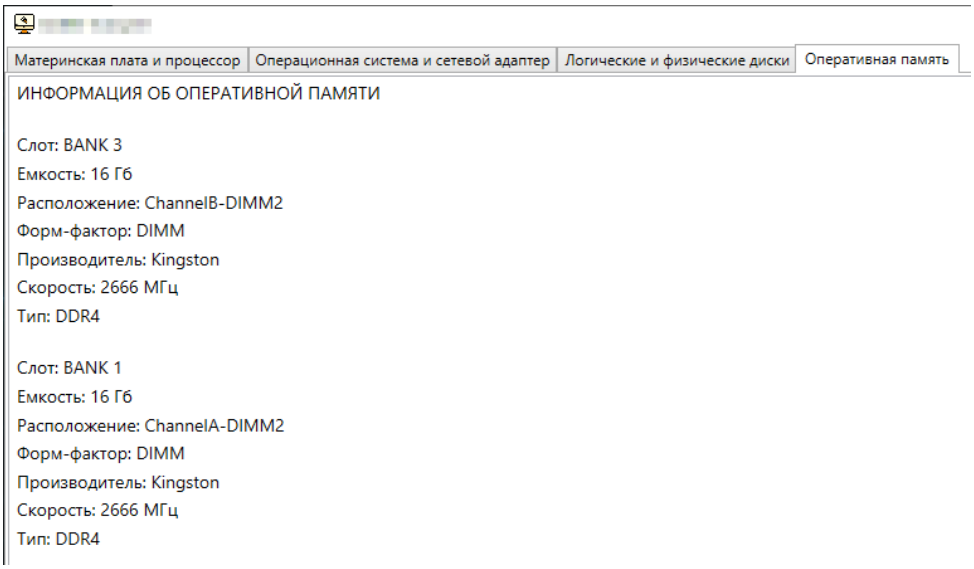


Рисунок 42. Детальная информация об оперативной памяти

3.3.6. Вывод статистики

Окно вывода статистики (Рисунок 43) имеет три кнопки (Статистика по операционным системам, Статистика по оперативной памяти и Статистика по процессорам), при нажатии на которые открываются соответствующие окна.

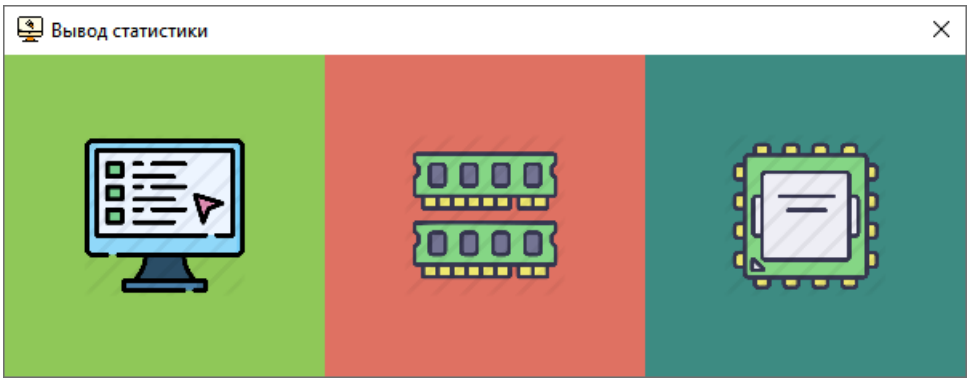


Рисунок 43. Окно «Вывод статистики»

Каждая кнопка имеет всплывающую подсказку о действии, которое выполняется при нажатии (Рисунок 44, Рисунок 45, Рисунок 46). Статистика вычисляется при помощи класса Statistics и выводится в виде столбчатых диаграмм.

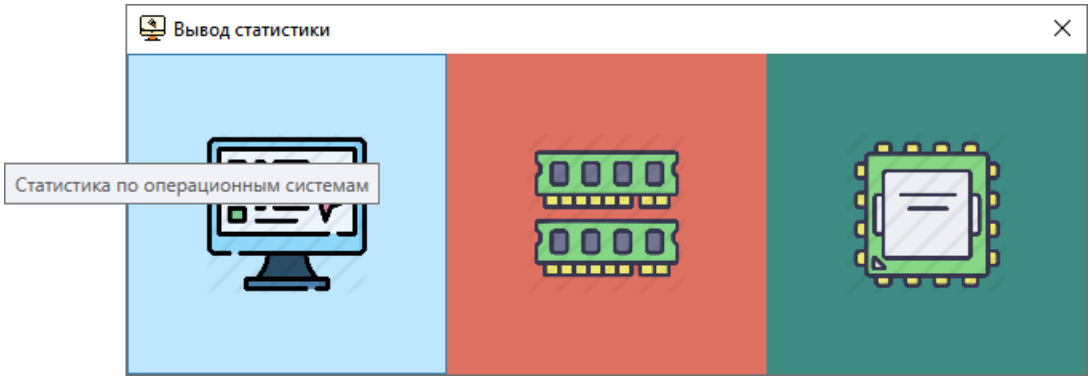


Рисунок 44. Всплывающая подсказка кнопки «Статистика по операционным системам»

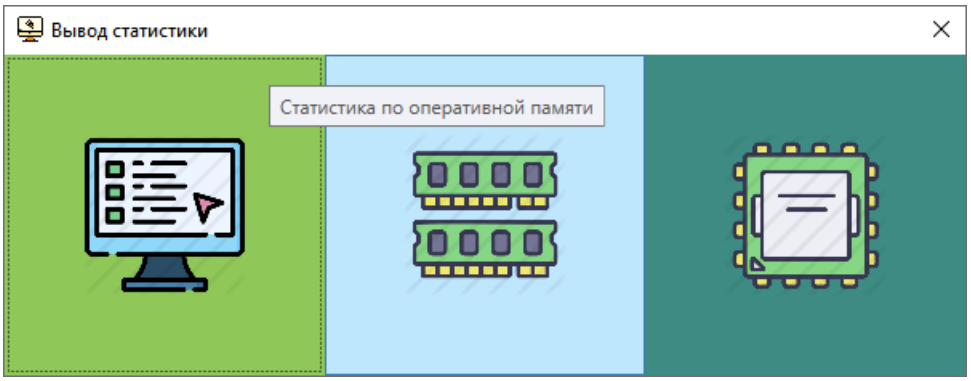


Рисунок 45. Всплывающая подсказка кнопки «Статистика по оперативной памяти»

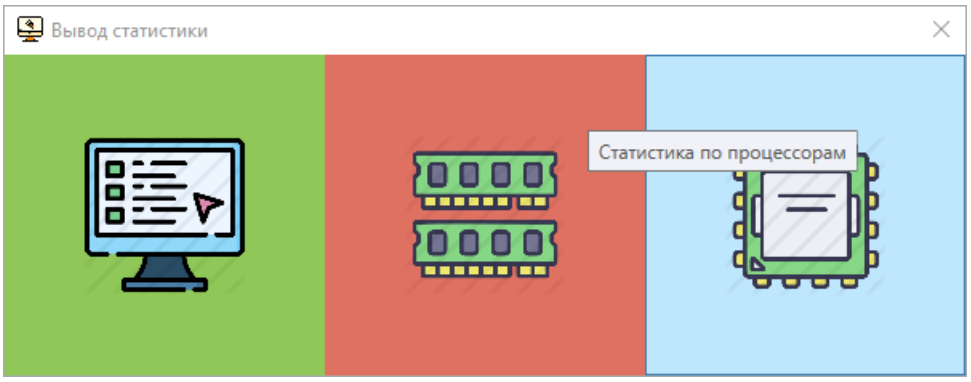


Рисунок 46. Всплывающая подсказка кнопки «Статистика по процессорам»

Окно статистики по операционным системам (Рисунок 47) содержит две диаграммы – «Операционная система» и «Редакция операционной системы».

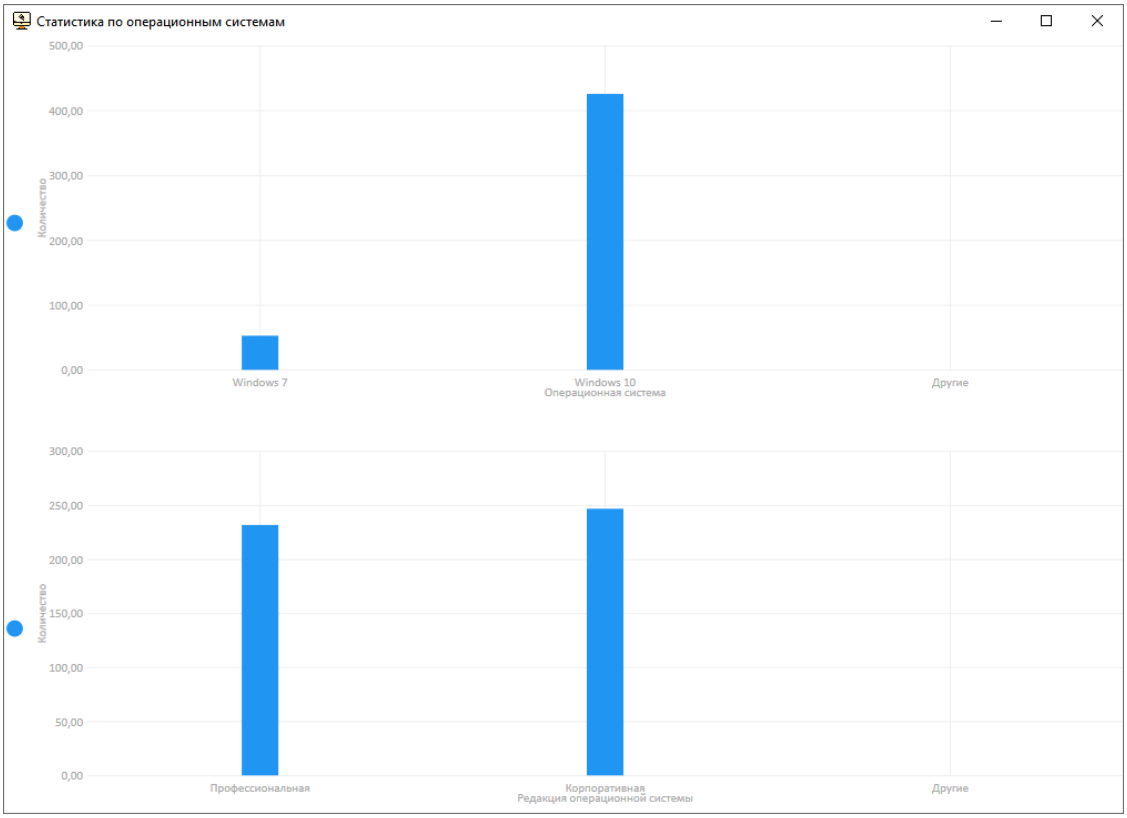


Рисунок 47. Окно статистики по операционным системам

Диаграммы имеют подписи, а также всплывающую при наведении подсказку (Рисунок 48) с количеством учтенных элементов.

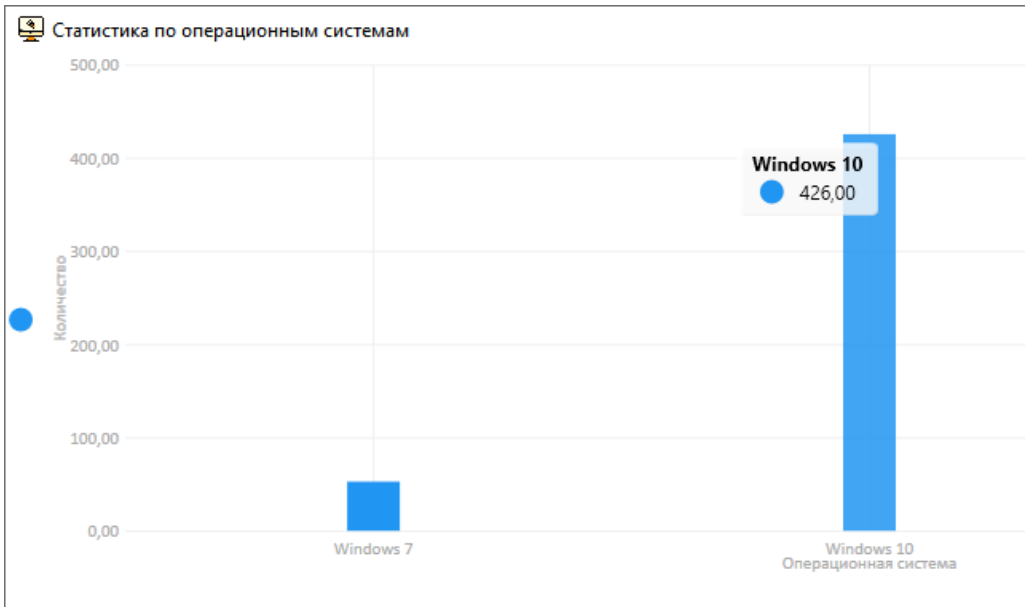


Рисунок 48. Всплывающая подсказка диаграммы

Окно статистики по оперативной памяти (Рисунок 49) содержит две диаграммы – «Размер оперативной памяти» и «Тип оперативной памяти». Диаграммы также имеют подписи и всплывающую при наведении подсказку с количеством учтенных элементов.

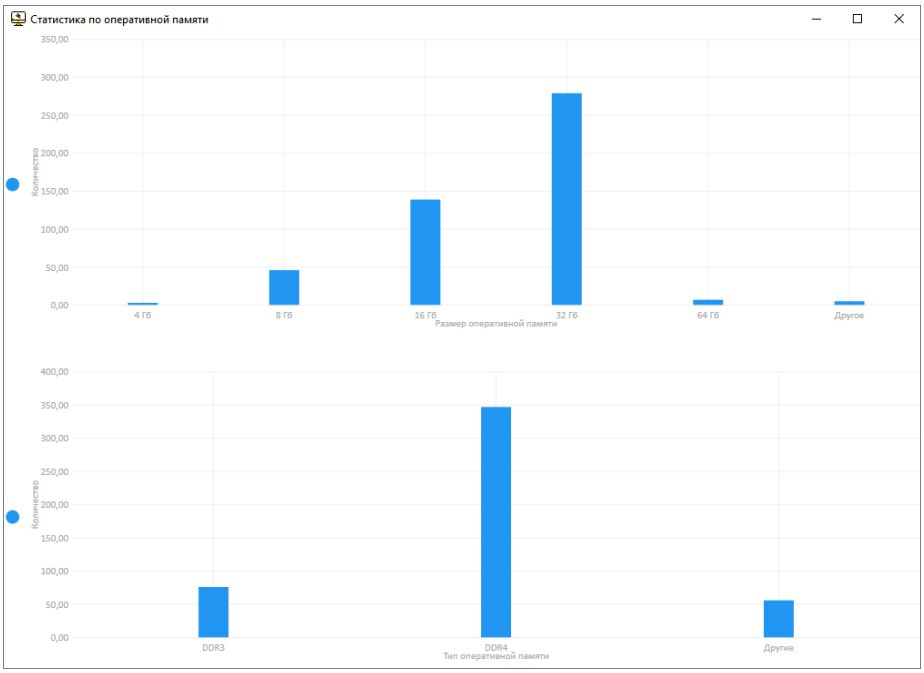


Рисунок 49. Окно статистики по оперативной памяти

Окно статистики по процессорам (Рисунок 50) содержит три диаграммы – «Семейство процессоров», «Сокет процессора» и «Виртуализация», у всех диаграмм также имеются подписи и всплывающие подсказки.

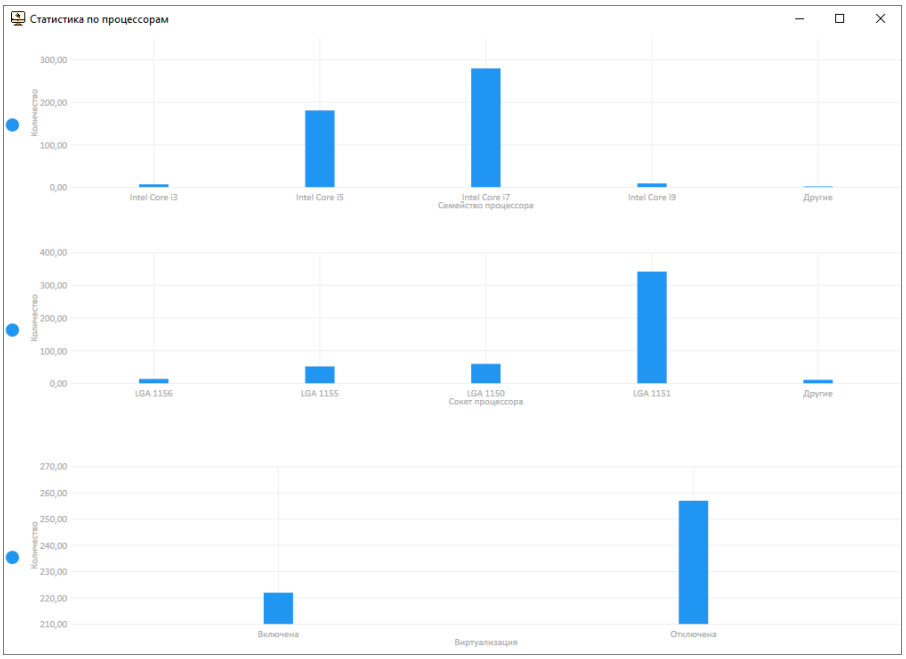


Рисунок 50. Окно статистики по процессорам

Пример исходного кода для вывода окна статистики по оперативной памяти приведен в приложении 5.

4. ЭКОНОМИЧЕСКАЯ ЭФФЕКТИВНОСТЬ РЕАЛИЗАЦИИ ИС

Согласно Налоговому кодексу Российской Федерации, «расходы на производство и реализацию подразделяются на:

1) прямые;

2) косвенные.

К прямым расходам могут быть отнесены:

1) на приобретение сырья и (или) материалов, используемых в производстве товаров (выполнении работ, оказании услуг) и (или) образующих их основу либо являющихся необходимым компонентом при производстве товаров (выполнении работ, оказании услуг);

2) на приобретение комплектующих изделий, подвергающихся монтажу, и (или) полуфабрикатов, подвергающихся дополнительной обработке у налогоплательщика;

3) расходы на оплату труда персонала, участвующего в процессе производства товаров, выполнения работ, оказания услуг, а также расходы на обязательное пенсионное страхование, на обязательное социальное страхование на случай временной нетрудоспособности и в связи с материнством, обязательное медицинское страхование, обязательное социальное страхование от несчастных случаев на производстве и профессиональных заболеваний, начисленные на указанные суммы расходов на оплату труда;

4) суммы начисленной амортизации по основным средствам, используемым при производстве товаров, работ, услуг.

К косвенным расходам относятся все иные суммы расходов, за исключением внереализационных расходов.

Налогоплательщик самостоятельно определяет в учетной политике для целей налогообложения перечень прямых расходов, связанных с производством товаров (выполнением работ, оказанием услуг).» [14]

Таким образом, при разработке информационной системы могут быть учтены следующие затраты:

1. стоимость использованных материалов;
2. оплата труда персонала и отчисления на социальные нужды;
3. затраты на содержание и эксплуатацию оборудования.

4.1. Расчет стоимости использованных материалов

При разработке информационной системы материальные затраты практически отсутствовали, поэтому к этой категории относятся лишь затраты на канцелярские товары.

Таблица 16. Расчет стоимости использованных материалов

№ п/п	Наименование материала	Расход, шт.	Цена, руб./шт.	Сумма, руб.
1	Тетрадь	2	70	140
2	Бумага А4 (пачка)	1	350	350
3	Карандаш	2	30	60
4	Ручка гелевая	2	80	160
5	Картридж для принтера	1	700	700
Итого				1410

4.2. Расчет оплаты труда персонала и отчислений на социальные нужды

Основная заработная плата ($ЗП_{осн}$) разработчика рассчитывается по формуле:

$$ЗП_{осн} = С_т \times Т_п, \quad (1)$$

где $С_т$ – часовая тарифная ставка,

$Т_п$ – трудоемкость разработки программы (130 часов).

Часовая тарифная ставка ($С_т$) рассчитывается по формуле:

$$С_т = \frac{О_м}{Ф_{рв}}, \quad (2)$$

где $О_м$ – месячный оклад разработчика (56000 руб.),

$Т_п$ – плановый фонд рабочего времени за месяц (176 часов).

Таким образом, согласно формуле 2 часовая тарифная ставка ($С_т$) составляет 318 руб 18 коп, из чего следует, что по формуле 1 основная заработная плата ($ЗП_{осн}$) составляет 41 363 руб. 40 коп.

Дополнительная заработная плата ($ЗП_{доп}$) рассчитывается согласно действующему на предприятии положению об оплате труда (в данном случае не предусмотрена).

Общая сумма затрат на оплату труда (ФОТ) разработчика программы рассчитывается по формуле:

$$ФОТ = ЗП_{осн} + ЗП_{доп}, \quad (3)$$

где $ЗП_{осн}$ – основная заработная плата,

$ЗП_{доп}$ – дополнительная заработная плата.

Таким образом, по формуле 3 общая сумма затрат на оплату труда составляет 41 363 руб. 40 коп.

Отчисления в Фонд социального страхования (ФСС) рассчитываются согласно действующей ставке и составляют 30% от ФОТ, из чего следует, что отчисления составляют 12 409 руб. 2 коп.

4.3. Расчет затрат на содержание и эксплуатацию оборудования

Затраты на содержание и эксплуатацию компьютерного оборудования состоят из затрат на электроэнергию и амортизационных отчислений.

Затраты на электроэнергию (Зэ) рассчитываются по формуле:

$$З_э = W \times t_{оп} \times Ц_э, \quad (4)$$

где W – потребляемая мощность единицы оборудования (0,6 кВт),

$t_{оп}$ – оперативное время работы оборудования,

$Ц_э$ –цена одного киловатт-часа (6,74098 руб.).

Из формулы 4 следует, что затраты на электроэнергию составляют 566 руб. 24 коп.

Норма амортизационных отчислений (На) рассчитывается по формуле:

$$\text{Н}_\text{а} = \frac{1}{N} \times\ 100\%, \quad (5)$$

где N – срок полезного использования (36 месяцев).

Из формулы 5 следует, что норма амортизационных отчислений составляет 2,78%.

Амортизационные отчисления рассчитываются по формуле:

$$\text{А} = \text{ПС} \times t_{\text{исп}} \times \frac{\text{Н}_\text{а}}{100}, \quad (6)$$

где ПС – первоначальная стоимость оборудования (42 000 руб.),

$t_{исп}$ – срок использования оборудования (3 месяца),

$Н_а$ –норма амортизационных отчислений.

Из формулы 6 следует, что амортизационные отчисления составляют 3 502 руб. 80 коп.

4.4. Расчет суммы прямых затрат

Таблица 17. Расчет суммы прямых затрат

№ п/п	Наименование статьи расходов	Сумма руб.
1	Материальные затраты	1410
2	Затраты на оплату труда	41363,4
3	Отчисления в фонды социального страхования	12409,02
4	Расходы на содержание и эксплуатацию оборудования	3502,8
ИТОГО		58685,22

Исходя из приведенных в таблице 17 расчетов, разработка информационной системы является экономически эффективной.

ЗАКЛЮЧЕНИЕ

Инвентаризация аппаратного обеспечения компьютеров, т.е. его диагностика и учет, играет важную роль в работе любой организации. Польза инвентаризации заключается в упрощении доступа к информации о конфигурациях и проведении анализа состава имеющегося оборудования.

В выпускной квалификационной работе была реализована информационная система, проводящая сбор информации о компьютерах в сети предприятия и сохранение полученной информации в базу данных, а также обеспечивающая вывод этих данных в удобном пользователю виде и подсчет статистики по основным параметрам.

В ходе выполнения работы был проведен анализ существующих решений и изучена технология WMI, после чего были определены требования к разрабатываемой системе, спроектирована база данных под управлением MS SQL Server, а также разработаны программные модули для работы с данными и представления пользовательского интерфейса.

Программа удовлетворяет всем поставленным требованиям, а также обладает широкими возможностями расширения функционала и доработки под нужды предприятия.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Совместный предприниматель [Электронный ресурс]. – Режим доступа: https://spmag.ru/articles/inventarizaciya-kompyuterov

2. Компания АЛАН [Электронный ресурс]. – Режим доступа: https://www.alan-it.ru/wkpages/service/SAM/MAP.aspx

3. CodeProject – For those who code [Электронный ресурс]. – Режим доступа: https://www.codeproject.com/Articles/362227/System-Information-2

4. Хабр [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/124386/

5. Бобынцев, Д. О. Основы администрирования информационных систем: учебное пособие / Д. О. Бобынцев [и др.]. — Москва; Берлин: Директ-Медиа, 2021. — 200 с.

6. Леонтьев К. Вы все еще не используете WMI? Часть I // Системный администратор. 2006. № 1 (38). С. 4-11.

7. Попов А.В., Шикин Е.А. Администрирование Windows с помощью WMI и WMIC. СПб.: БХВ-Петербург, 2004 – 752 с.: ил.

8. Кузьмичев, А.Э. Программирование для Windows Phone для начинающих : курс лекций. Москва: Интуит НОУ, 2016. – 165 с.

9. WindowsTune – Установка и Настройка Windows [Электронный ресурс]. – Режим доступа: https://windowstune.ru/misc/microsoft-net-framework.html

10. Знакомство с интегрированной средой разработки программного обеспечения Microsoft Visual Studio C# [Электронный ресурс]. – Режим доступа: https://infourok.ru/znakomstvo-s-integrirovannoy-sredoy-razrabotki-programmnogo-obespecheniya-microsoft-visual-studio-c-razrabotka-proektov-konsolno-2702455.html

11. Руководство по WPF [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/wpf/1.php

12. Введение в MS SQL Server и T-SQL [Электронный ресурс]. – Режим доступа: https://metanit.com/sql/sqlserver/1.1.php

13. Руководство по Entity Framework Core [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/entityframeworkcore/1.1.php

14. Налоговый кодекс РФ. Статья 318. Порядок определения суммы расходов на производство и реализацию

15. Томас О., Маклин Й. Администрирование Windows Server 2008. Учебный курс Microsoft. М.: Издательство «Русская редакция», 2013. – 688 с.: ил.

16. Станек Уильям Р. Windows Server 2008. Справочник администратора. М.: Издательство «Русская редакция»; СПб.: БХВ-Петербург, 2008. – 688 с.: ил.

17. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. СПб.: Питер, 2013. – 896 с.: ил.

18. Фленов М.Е. Библия C#. СПб.: БХВ-Петербург, 2019. – 512 с.: ил.

19. Бен-Ган И. Microsoft SQL Server 2012. Основы T-SQL. М.: Эксмо, 2015 – 400 с.

20. Сарка Д. Microsoft SQL Server 2012. Реализация хранилищ данных. Учебный курс Microsoft. М.: Издательство «Русская редакция», 2014. – 816 с.: ил.

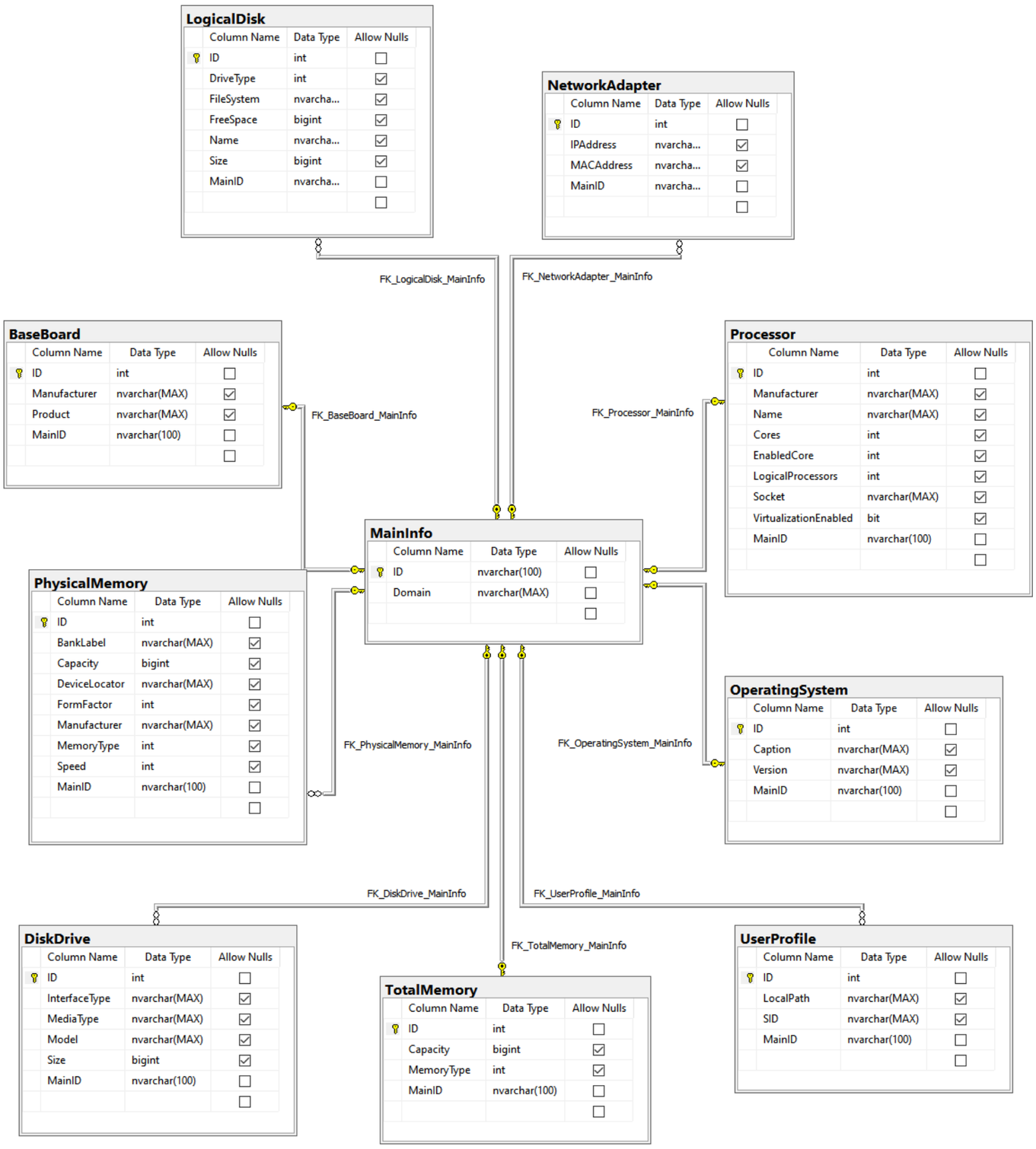
21. Натан А. WPF 4. Подробное руководство. СПб.: Символ-Плюс, 2011. – 880 с.: ил.

ПРИЛОЖЕНИЕ 1 – СКРИПТ ГЕНЕРАЦИИ БАЗЫ ДАННЫХ

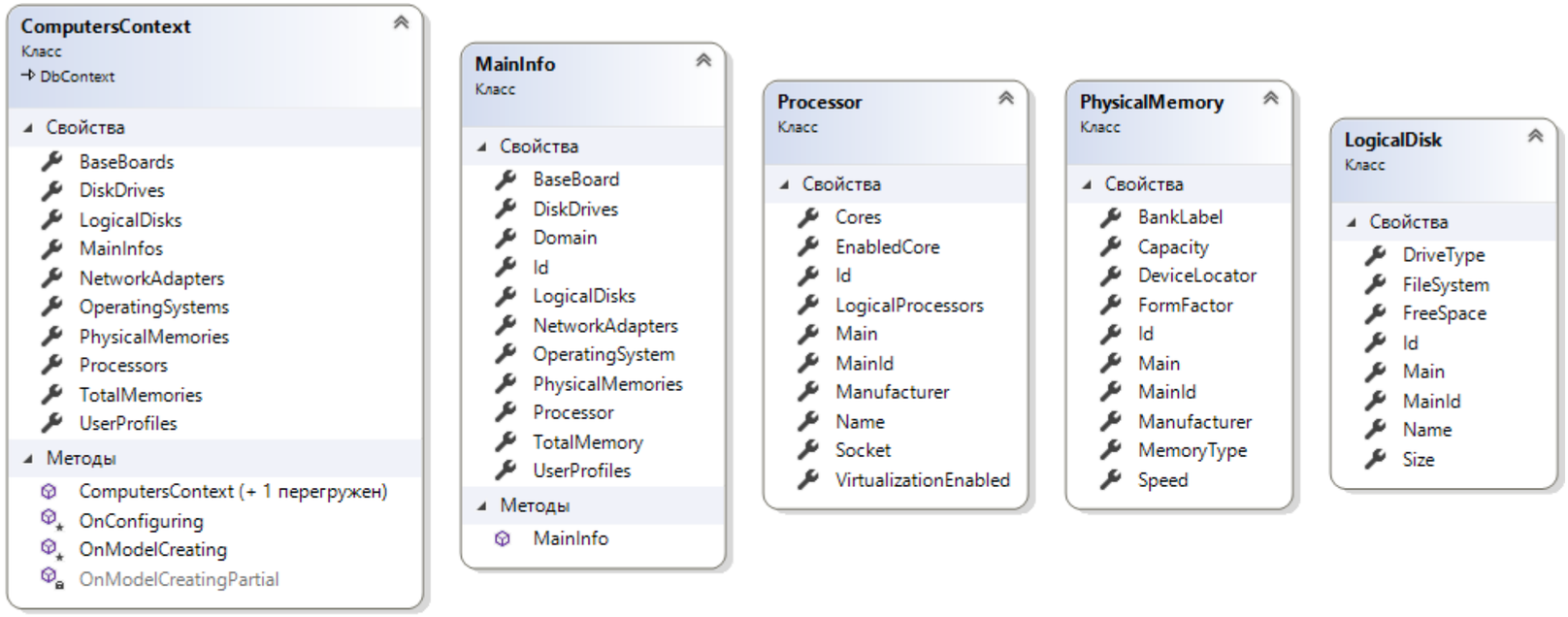
```
USE Computers
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE dbo.MainInfo(
             ID NVARCHAR(100) CONSTRAINT PK_MainInfo_ID PRIMARY KEY NOT NULL,
          Domain NVARCHAR(max) NOT NULL,
)
GO

CREATE TABLE dbo.DiskDrive(
            ID INT CONSTRAINT PK_DiskDrive_ID PRIMARY KEY IDENTITY,
           InterfaceType NVARCHAR(max) NULL,
          MediaType NVARCHAR(max) NULL,
         Model NVARCHAR(max) NULL,
        Size BIGINT NULL,
             MainID NVARCHAR(100) NOT NULL CONSTRAINT FK_DiskDrive_MainInfo FOREIGN KEY
REFERENCES dbo.MainInfo(ID)
         ON DELETE CASCADE
         ON UPDATE CASCADE,
)
GO

CREATE TABLE dbo.LogicalDisk(
            ID INT CONSTRAINT PK_LogicalDisk_ID PRIMARY KEY IDENTITY,
          DriveType INT NULL,
           FileSystem NVARCHAR(max) NULL,
          FreeSpace BIGINT NULL,
          Name NVARCHAR(max) NULL,
         Size BIGINT NULL,
             MainID NVARCHAR(100) NOT NULL CONSTRAINT FK_LogicalDisk_MainInfo FOREIGN KEY
REFERENCES dbo.MainInfo(ID)
         ON DELETE CASCADE
         ON UPDATE CASCADE,
)
GO

CREATE TABLE dbo.NetworkAdapter(
              ID INT CONSTRAINT PK_NetworkAdapter_ID PRIMARY KEY IDENTITY,
           IPAddress NVARCHAR(max) NULL,
           MACAddress NVARCHAR(max) NULL,
                MainID NVARCHAR(100) NOT NULL CONSTRAINT FK_NetworkAdapter_MainInfo FOREIGN KEY
REFERENCES dbo.MainInfo(ID)
         ON DELETE CASCADE
         ON UPDATE CASCADE,
 )
GO

CREATE TABLE dbo.PhysicalMemory(
             ID INT CONSTRAINT PK_PhysicalMemory_ID PRIMARY KEY IDENTITY,
          BankLabel NVARCHAR(max) NULL,
         Capacity BIGINT NULL,
          DeviceLocator NVARCHAR(max) NULL,
         FormFactor INT NULL,
          Manufacturer NVARCHAR(max) NULL,
         MemoryType INT NULL,
        Speed INT NULL,
       MainID   NVARCHAR(100)   NOT   NULL   CONSTRAINT   FK_PhysicalMemory_MainInfo   FOREIGN   KEY   REFERENCES
dbo.MainInfo(ID)
         ON DELETE CASCADE
         ON UPDATE CASCADE,
)
GO

CREATE TABLE dbo.UserProfile(
              ID INT CONSTRAINT PK_UserProfile_ID PRIMARY KEY IDENTITY,
           LocalPath NVARCHAR(max) NULL,
```

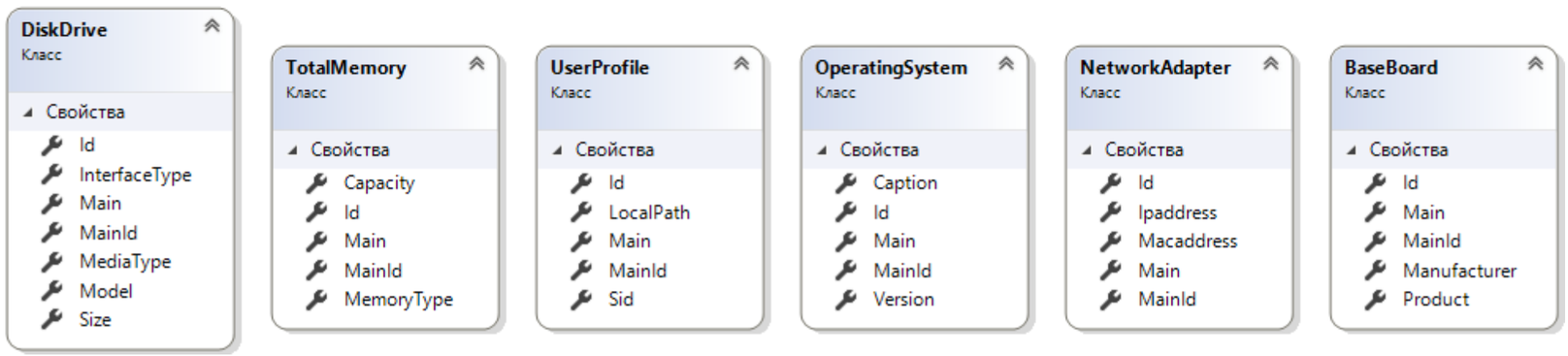
```
          SID NVARCHAR(max) NULL,
               MainID NVARCHAR(100) NOT NULL CONSTRAINT FK_UserProfile_MainInfo FOREIGN KEY
REFERENCES dbo.MainInfo(ID)
          ON DELETE CASCADE
          ON UPDATE CASCADE,
)
GO

CREATE TABLE dbo.TotalMemory(
              ID INT CONSTRAINT PK_TotalMemory_ID PRIMARY KEY IDENTITY,
          Capacity BIGINT NULL,
          MemoryType INT NULL,
               MainID NVARCHAR(100) NOT NULL UNIQUE CONSTRAINT FK_TotalMemory_MainInfo FOREIGN
KEY REFERENCES dbo.MainInfo(ID)
          ON DELETE CASCADE
          ON UPDATE CASCADE,
)
GO

CREATE TABLE dbo.BaseBoard(
              ID INT CONSTRAINT PK_BaseBoard_ID PRIMARY KEY IDENTITY,
           Manufacturer NVARCHAR(max) NULL,
           Product NVARCHAR(max) NULL,
               MainID NVARCHAR(100) NOT NULL CONSTRAINT UQ_BaseBoard_MainInfo UNIQUE CONSTRAINT
FK_BaseBoard_MainInfo FOREIGN KEY REFERENCES dbo.MainInfo(ID)
          ON DELETE CASCADE
          ON UPDATE CASCADE,
)
GO

CREATE TABLE dbo.OperatingSystem(
              ID INT CONSTRAINT PK_OperatingSystem_ID PRIMARY KEY IDENTITY,
           Caption NVARCHAR(max) NULL,
           Version NVARCHAR(max) NULL,
               MainID NVARCHAR(100) NOT NULL CONSTRAINT UQ_OperatingSystem_MainInfo UNIQUE
CONSTRAINT FK_OperatingSystem_MainInfo FOREIGN KEY REFERENCES dbo.MainInfo(ID)           ON DELETE CASCADE
          ON UPDATE CASCADE,
)
GO

CREATE TABLE dbo.Processor(
              ID INT CONSTRAINT PK_Processor_ID PRIMARY KEY IDENTITY,
            Manufacturer NVARCHAR(max) NULL,
           Name NVARCHAR(max) NULL,
          Cores INT NULL,
           EnabledCore INT NULL,
           LogicalProcessors INT NULL,
           Socket NVARCHAR(max) NULL,
           VirtualizationEnabled bit NULL,
               MainID NVARCHAR(100) NOT NULL CONSTRAINT UQ_Processor_MainInfo UNIQUE CONSTRAINT
FK_Processor_MainInfo FOREIGN KEY REFERENCES dbo.MainInfo(ID)
          ON DELETE CASCADE
          ON UPDATE CASCADE,
)
GO
```

ПРИЛОЖЕНИЕ 2 – ФИЗИЧЕСКАЯ МОДЕЛЬ БАЗЫ ДАННЫХ



ПРИЛОЖЕНИЕ 3 – ДИАГРАММЫ КЛАССОВ БАЗЫ ДАННЫХ





ПРИЛОЖЕНИЕ 4 – ИСХОДНЫЙ КОД КЛАССОВ ЛОГИКИ ПРОГРАММЫ

GetComputers.cs

```
using System;
using System.Collections.Generic;
using System.DirectoryServices.AccountManagement;

namespace DomainComputersInfo
{
    public class GetComputers
    {
        //=============================================================================================
        public static List<String> GetFromAD(String DomainName)
        {
            PrincipalContext CurrentPrincipalContext = new(ContextType.Domain, DomainName);
            ComputerPrincipal CurrentComputerPrincipal = new(CurrentPrincipalContext);
            PrincipalSearcher search = new(CurrentComputerPrincipal);

            List<String> PCnames = new();

            foreach (ComputerPrincipal result in search.FindAll())
            {
                if ((bool)result.Enabled)
                    PCnames.Add(result.Name + "." + DomainName);
            }
            search.Dispose();

            return PCnames;
        }
        //=============================================================================================
        public static List<String> GetManual(String DomainName, String PCName, List<String> PCNames)
        {
            PCNames.Add(PCName + "." + DomainName);

            return PCNames;
        }
        //=============================================================================================
    }
}
```

GetFromWMI.cs

```
using System;
using System.Management;
using WMIClasses;

namespace DomainComputersInfo
{
    public static class GetFromWMI
    {
        //=============================================================================================
        public static BaseBoard GetBaseBoardInfo(ManagementObject CurrentManagementObject, String PCName)
        {
            BaseBoard Info = new()
            {
                Manufacturer = CurrentManagementObject["Manufacturer"].ToString(),
                Product = CurrentManagementObject["Product"].ToString(),
                MainId = PCName
            };
            return Info;
        }
```

```
        //=============================================================================================
        public static DiskDrive GetDiskDriveInfo(ManagementObject CurrentManagementObject, String PCName)
        {
            DiskDrive Info = new()
            {
                InterfaceType = CurrentManagementObject["InterfaceType"].ToString(),
                MediaType = CurrentManagementObject["MediaType"].ToString(),
                Model = CurrentManagementObject["Model"].ToString(),
                Size = Convert.ConvertToGb(CurrentManagementObject["Size"]),
                MainId = PCName
            };
            return Info;
        }
        //=============================================================================================
        public static LogicalDisk GetLogicalDiskInfo(ManagementObject CurrentManagementObject, String
PCName)
        {
            try { WMILogicalDisk.FileSystem = CurrentManagementObject["FileSystem"].ToString(); }
            catch (NullReferenceException) { WMILogicalDisk.FileSystem = ""; }

            try { WMILogicalDisk.FreeSpace = (ulong)CurrentManagementObject["FreeSpace"]; }
            catch (NullReferenceException) { WMILogicalDisk.FreeSpace = 0; }

            try { WMILogicalDisk.ProviderName = CurrentManagementObject["ProviderName"].ToString(); }
            catch (NullReferenceException) { WMILogicalDisk.ProviderName = ""; }

            try { WMILogicalDisk.Size = (ulong)CurrentManagementObject["Size"]; }
            catch (NullReferenceException) { WMILogicalDisk.Size = 0; }

            LogicalDisk Info = new()
            {
                DriveType = (int?)(uint)CurrentManagementObject["DriveType"],
                FileSystem = WMILogicalDisk.FileSystem,
                FreeSpace = Convert.ConvertToGb(WMILogicalDisk.FreeSpace),
                Name = CurrentManagementObject["Name"].ToString(),
                Size = Convert.ConvertToGb(WMILogicalDisk.Size),
                MainId = PCName
            };
            return Info;
        }
        //=============================================================================================
        public static NetworkAdapter GetNetworkAdapterInfo(ManagementObject CurrentManagementObject,
String PCName)
        {
            String ip;

            try { ip = ((string[])CurrentManagementObject["IPAddress"])[0]; }
            catch (NullReferenceException) { ip = ""; }

            try { WMINetworkAdapterConfiguration.MACAddress =
CurrentManagementObject["MACAddress"].ToString(); }
            catch (NullReferenceException) { WMINetworkAdapterConfiguration.MACAddress = ""; }

            NetworkAdapter Info = new()
            {
                Ipaddress = ip,
                Macaddress = WMINetworkAdapterConfiguration.MACAddress,
                MainId = PCName
            };
            return Info;
        }
        //=============================================================================================
        public static OperatingSystem GetOperatingSystemInfo(ManagementObject CurrentManagementObject,
String PCName)
        {
            WMIOperatingSystem.Version = CurrentManagementObject["Version"].ToString();

            OperatingSystem Info = new()
            {
                Caption = CurrentManagementObject["Caption"].ToString(),
                Version = WMIOperatingSystem.Version,
                MainId = PCName
            };
            return Info;
        }
```

```
        //=============================================================================================
        public static PhysicalMemory GetPhysicalMemoryInfo(ManagementObject CurrentManagementObject,
String PCName)
        {
            try { WMIPhysicalMemory.SMBIOSMemoryType = (uint)CurrentManagementObject["SMBIOSMemoryType"];
}
            catch (ManagementException) { WMIPhysicalMemory.SMBIOSMemoryType = 0; }
            catch (System.Runtime.InteropServices.COMException) { WMIPhysicalMemory.SMBIOSMemoryType = 0;
}
            catch (NullReferenceException) { WMIPhysicalMemory.SMBIOSMemoryType = 0; }

            PhysicalMemory Info = new()
            {
                BankLabel = CurrentManagementObject["BankLabel"].ToString(),
                Capacity = (long?)Convert.ConvertToGb(CurrentManagementObject["Capacity"]),
                DeviceLocator = CurrentManagementObject["DeviceLocator"].ToString(),
                FormFactor = (ushort)CurrentManagementObject["FormFactor"],
                Manufacturer = CurrentManagementObject["Manufacturer"].ToString(),
                MemoryType = (int?)WMIPhysicalMemory.SMBIOSMemoryType,
                Speed = (int?)(uint)CurrentManagementObject["Speed"],
                MainId = PCName
            };
            return Info;
        }
        //=============================================================================================
        public static Processor GetProcessorInfo(ManagementObject CurrentManagementObject, String PCName)
        {
            try { WMIProcessor.NumberOfCores = (uint)CurrentManagementObject["NumberOfCores"]; }
            catch (ManagementException) { WMIProcessor.NumberOfCores = 0; }

            try { WMIProcessor.NumberOfEnabledCore = (uint)CurrentManagementObject["NumberOfEnabledCore"];
}
            catch (ManagementException) { WMIProcessor.NumberOfEnabledCore = 0; }

            try { WMIProcessor.SocketDesignation =
CurrentManagementObject["SocketDesignation"].ToString(); }
            catch (ManagementException) { WMIProcessor.SocketDesignation = ""; }

            try { WMIProcessor.VirtualizationFirmwareEnabled =
(bool)CurrentManagementObject["VirtualizationFirmwareEnabled"]; }
            catch (ManagementException) { WMIProcessor.VirtualizationFirmwareEnabled = false; }

            Processor Info = new()
            {
                Manufacturer = CurrentManagementObject["Manufacturer"].ToString(),
                Name = CurrentManagementObject["Name"].ToString().Replace("Intel(R) Core(TM)", ""),
                Cores = (int?)WMIProcessor.NumberOfCores,
                EnabledCore = (int?)WMIProcessor.NumberOfEnabledCore,
                LogicalProcessors = (int?)(uint)CurrentManagementObject["NumberOfLogicalProcessors"],
                Socket = WMIProcessor.SocketDesignation,
                VirtualizationEnabled = WMIProcessor.VirtualizationFirmwareEnabled,
                MainId = PCName
            };
            return Info;
        }
        //=============================================================================================
        public static UserProfile GetUserProfileInfo(ManagementObject CurrentManagementObject, String
PCName)
        {

            UserProfile Info = new()
            {
                LocalPath = CurrentManagementObject["LocalPath"].ToString(),
                Sid = CurrentManagementObject["SID"].ToString(),
                MainId = PCName
            };
            return Info;
        }
```

```
        //=============================================================================================
        public static TotalMemory GetTotalMemoryInfo(ManagementObject CurrentManagementObject, int t,
String PCName)
        {
            TotalMemory Info = new()
            {
                Capacity = (long?)Convert.ConvertToGb(CurrentManagementObject["TotalPhysicalMemory"]),
                MemoryType = t,
                MainId = PCName
            };
            return Info;
        }
        //=============================================================================================
    }
}
```

WorkWithDB.cs

```
using System;
using System.Collections.Generic;
using System.Management;
using System.Linq;

namespace DomainComputersInfo
{
    public class WorkWithDB
    {
        //=============================================================================================
        public static void RecreateDB()
        {
            using ComputersContext db = new();
            db.Database.EnsureDeleted();
            db.Database.EnsureCreated();
        }
        //=============================================================================================
        public static void DeleteEntry()
        {
            using ComputersContext db = new();

            var remove = db.MainInfos
                .Where(o => o.Id == Windows.CommonInfoWindow.ComputerName)
                .FirstOrDefault();

            db.MainInfos.Remove(remove);
            db.SaveChanges();
        }
        //=============================================================================================
        public static void GetAllInfo(List<String> arrComputers, String username, String password)
        {
            ConnectionOptions CurrentConnectionOptions = new()
            {
                Username = username,
                Password = password,
                Impersonation = ImpersonationLevel.Impersonate,
                EnablePrivileges = true
            };

            using ComputersContext db = new();

            foreach (string strComputer in arrComputers)
            {
                try
                {
                    ManagementScope CurrentManagementScope = new(string.Format("\\\\{0}\\root\\cimv2",
strComputer), CurrentConnectionOptions);
                    CurrentManagementScope.Connect();
                    ObjectGetOptions CurrentObjectGetOptions = new();

                    MainInfo mi = new()
                    {
                        Id = strComputer,
                        Domain = ""
                    };
```

```
                    db.MainInfos.Add(mi);

                    for (int i = 0; i < Logics.WMI.Path.Count; i++)
                    {
                        ManagementPath CurrentManagementPath = new(Logics.WMI.Path[i]);
                        ManagementClass CurrentManagementClass = new(CurrentManagementScope,
CurrentManagementPath, CurrentObjectGetOptions);

                        foreach (ManagementObject CurrentManagementObject in
CurrentManagementClass.GetInstances())
                        {
                        switch (i)
                        {
                        case 0:
                        db.BaseBoards.Add(GetFromWMI.GetBaseBoardInfo(CurrentManagementObject,
strComputer));
                        break;
                        case 1:
                        db.DiskDrives.Add(GetFromWMI.GetDiskDriveInfo(CurrentManagementObject,
strComputer));
                        break;
                        case 2:
                        db.LogicalDisks.Add(GetFromWMI.GetLogicalDiskInfo(CurrentManagementObject,
strComputer));
                        break;
                        case 3:
                        NetworkAdapter Info = new();
                        Info = GetFromWMI.GetNetworkAdapterInfo(CurrentManagementObject, strComputer);
                        db.NetworkAdapters.Add(Info);
                        break;
                        case 4:
                        db.OperatingSystems.Add(GetFromWMI.GetOperatingSystemInfo(CurrentManagementObject,
strComputer));
                        break;
                        case 5:
                        db.PhysicalMemories.Add(GetFromWMI.GetPhysicalMemoryInfo(CurrentManagementObject,
strComputer));
                        break;
                        case 6:
                        db.Processors.Add(GetFromWMI.GetProcessorInfo(CurrentManagementObject,
strComputer));
                        break;
                        case 7:
                        db.UserProfiles.Add(GetFromWMI.GetUserProfileInfo(CurrentManagementObject,
strComputer));
                        break;
                        case 8:
                        db.TotalMemories.Add(GetFromWMI.GetTotalMemoryInfo(CurrentManagementObject,
(int)WMIClasses.WMIPhysicalMemory.SMBIOSMemoryType, strComputer));
                        break;
                        }
                        }
                        db.SaveChanges();
                    }
                }
                catch (UnauthorizedAccessException) { }
                catch (System.Runtime.InteropServices.COMException) { }
                catch (System.Management.ManagementException) { }
                catch (System.OutOfMemoryException) { }
            }
        }
        //=============================================================================================
    }
}
```

GetFromDB.cs

```
using System.Collections.Generic;
using System.Linq;
using System.Windows.Controls;
using System.ComponentModel;
using System.Windows.Data;

namespace DomainComputersInfo.Logics
{
    class GetFromDB
    {
        //=============================================================================================
        public static void GetMainInfo(ListView listView)
        {
            using ComputersContext db = new();

            var computers = from main in db.MainInfos
                            join baseboard in db.BaseBoards on main.Id equals baseboard.MainId
                            join processor in db.Processors on main.Id equals processor.MainId
                            join os in db.OperatingSystems on main.Id equals os.MainId
                            join memory in db.TotalMemories on main.Id equals memory.MainId
                            join network in db.NetworkAdapters on main.Id equals network.MainId
                            where network.Ipaddress.StartsWith("10.70") ||
network.Ipaddress.StartsWith("10.50")
                            select new
                            {
                                pcname = main.Id,
                                osname = os.Caption,
                                osver = os.Version,
                                proc = processor.Name,
                                socket = processor.Socket,
                                virt = processor.VirtualizationEnabled,
                                product = baseboard.Product,
                                ipaddr = network.Ipaddress,
                                macaddr = network.Macaddress,
                                memory = memory.Capacity,
                                memtype = memory.MemoryType
                            };

            List<Logics.DisplayData> items = new();
            foreach (var p in computers)
            {
                items.Add(new Logics.DisplayData()
                {
                    PCname = p.pcname,
                    OSname = p.osname,
                    OSversion = Convert.ConvertWinVer(p.osver),
                    ProcName = p.proc,
                    Socket = p.socket,
                    VirtEn = p.virt.ToString(),
                    BBname = p.product,
                    Ipaddr = p.ipaddr,
                    Macaddr = p.macaddr,
                    Memory = (1 + p.memory).ToString(),
                    Memtype = Convert.ConvertMemoryType(p.memtype)
                });
            }
            listView.Items.Refresh();
            listView.ItemsSource = items;
            CollectionView view =
(CollectionView)CollectionViewSource.GetDefaultView(listView.ItemsSource);
            view.SortDescriptions.Add(new SortDescription("PCname", ListSortDirection.Ascending));
        }
        //=============================================================================================
        public static void GetAllInfo(ListBox ListBoxProcessor, ListBox ListBoxBaseBoard, ListBox
ListBoxOperatingSystem, ListBox ListBoxUserProfile, ListBox ListBoxNetworkAdapter, ListBox
ListBoxLogicalDisks, ListBox ListBoxPhysicalDisks, ListBox ListBoxMemory)
        {
            using ComputersContext db = new();
            GetMbProcessor(ListBoxProcessor, ListBoxBaseBoard, db);
            GetOSNetwork(ListBoxOperatingSystem, ListBoxUserProfile, ListBoxNetworkAdapter, db);
            GetDisks(ListBoxLogicalDisks, ListBoxPhysicalDisks, db);
            GetMemory(ListBoxMemory, db);
        }
```

```
        //=============================================================================================
        private static void GetMbProcessor(ListBox ListBoxProcessor, ListBox ListBoxBaseBoard,
ComputersContext db)
        {
            var Processor = from main in db.MainInfos
                            where main.Id == Windows.CommonInfoWindow.ComputerName
                            join processor in db.Processors on main.Id equals processor.MainId
                            select new
                            {
                                processor.Manufacturer,
                                processor.Name,
                                processor.Cores,
                                processor.EnabledCore,
                                processor.LogicalProcessors,
                                processor.Socket,
                                Virtualization = processor.VirtualizationEnabled,
                            };

            var BaseBoard = from main in db.MainInfos
                            where main.Id == Windows.CommonInfoWindow.ComputerName
                            join baseboard in db.BaseBoards on main.Id equals baseboard.MainId
                            select new
                            {
                                baseboard.Manufacturer,
                                baseboard.Product,
                            };

            ListBoxProcessor.Items.Add("ИНФОРМАЦИЯ О ПРОЦЕССОРЕ\n");
            foreach (var p in Processor)
            {
                ListBoxProcessor.Items.Add(string.Format("Производитель: {0}",
p.Manufacturer.Replace("Genuine", "")));
                ListBoxProcessor.Items.Add(string.Format("Модель: {0}", p.Name));
                ListBoxProcessor.Items.Add(string.Format("Количество ядер: {0}", p.Cores));
                ListBoxProcessor.Items.Add(string.Format("Количество задействованных ядер: {0}",
p.EnabledCore));
                ListBoxProcessor.Items.Add(string.Format("Количество логических процессоров: {0}",
p.LogicalProcessors));
                ListBoxProcessor.Items.Add(string.Format("Сокет процессора: {0}", p.Socket));
                if ((bool)p.Virtualization)
                    ListBoxProcessor.Items.Add("Виртуализация включена: да\n");
                else
                    ListBoxProcessor.Items.Add("Виртуализация включена: нет\n");
            }

            ListBoxBaseBoard.Items.Add("ИНФОРМАЦИЯ О МАТЕРИНСКОЙ ПЛАТЕ\n");
            foreach (var p in BaseBoard)
            {
                ListBoxBaseBoard.Items.Add(string.Format("Производитель: {0}", p.Manufacturer));
                ListBoxBaseBoard.Items.Add(string.Format("Модель: {0}\n", p.Product));
            }
        }
        //=============================================================================================
        private static void GetOSNetwork(ListBox ListBoxOperatingSystem, ListBox ListBoxUserProfile,
ListBox ListBoxNetworkAdapter, ComputersContext db)
        {
            var OperatingSystem = from main in db.MainInfos
                                  where main.Id == Windows.CommonInfoWindow.ComputerName
                                  join operatingsystem in db.OperatingSystems on main.Id equals
                                  operatingsystem.MainId
                                  select new
                                  {
                                      operatingsystem.Caption,
                                      operatingsystem.Version,
                                  };

            var UserProfile = from main in db.MainInfos
                              where main.Id == Windows.CommonInfoWindow.ComputerName
                              join userprofile in db.UserProfiles on main.Id equals userprofile.MainId
                              where !userprofile.LocalPath.StartsWith("C:\\Windows")
                              select new
                              {
                                  userprofile.LocalPath,
                                  SID = userprofile.Sid,
                              };
```

```
            var NetworkAdapter = from main in db.MainInfos
                                 where main.Id == Windows.CommonInfoWindow.ComputerName
                                 join network in db.NetworkAdapters on main.Id equals network.MainId
                                 select new
                                 {
                                     IP = network.Ipaddress,
                                     MAC = network.Macaddress
                                 };

            ListBoxOperatingSystem.Items.Add("ИНФОРМАЦИЯ ОБ ОПЕРАЦИОННОЙ СИСТЕМЕ\n");
            foreach (var p in OperatingSystem)
            {
                ListBoxOperatingSystem.Items.Add(string.Format("Операционная система:\n{0}", p.Caption));
                ListBoxOperatingSystem.Items.Add(string.Format("Версия (сборка): {0}\n",
Convert.ConvertWinVer(p.Version)));

            }

            ListBoxUserProfile.Items.Add("ИНФОРМАЦИЯ О ПРОФИЛЯХ ПОЛЬЗОВАТЕЛЕЙ\n");
            foreach (var p in UserProfile)
            {
                ListBoxUserProfile.Items.Add(string.Format("Путь к профилю: {0}", p.LocalPath));
                ListBoxUserProfile.Items.Add(string.Format("Идентификатор: {0}\n", p.SID));
            }

            ListBoxNetworkAdapter.Items.Add("ИНФОРМАЦИЯ О СЕТЕВЫХ АДАПТЕРАХ\n");
            foreach (var p in NetworkAdapter)
            {
                ListBoxNetworkAdapter.Items.Add(string.Format("IP адрес: {0}", p.IP));
                ListBoxNetworkAdapter.Items.Add(string.Format("MAC адрес: {0}\n", p.MAC));
            }
        }
        //=============================================================================================
        private static void GetDisks(ListBox ListBoxLogicalDisks, ListBox ListBoxPhysicalDisks,
ComputersContext db)
        {
            var DiskDrive = from main in db.MainInfos
                            where main.Id == Windows.CommonInfoWindow.ComputerName
                            join diskdrive in db.DiskDrives on main.Id equals diskdrive.MainId
                            select new
                            {
                                Interface = diskdrive.InterfaceType,
                                diskdrive.MediaType,
                                diskdrive.Model,
                                diskdrive.Size
                            };
            var LogicalDisk = from main in db.MainInfos
                              where main.Id == Windows.CommonInfoWindow.ComputerName
                              join logicaldisk in db.LogicalDisks on main.Id equals logicaldisk.MainId
                              select new
                              {
                                  DriveType = Convert.ConvertMediaType(logicaldisk.DriveType),
                                  logicaldisk.FileSystem,
                                  logicaldisk.FreeSpace,
                                  logicaldisk.Name,
                                  logicaldisk.Size
                              };

            ListBoxPhysicalDisks.Items.Add("ИНФОРМАЦИЯ О ФИЗИЧЕСКИХ ДИСКАХ\n");
            foreach (var p in DiskDrive)
            {
                ListBoxPhysicalDisks.Items.Add(string.Format("Интерфейс: {0}", p.Interface));
                ListBoxPhysicalDisks.Items.Add(string.Format("Тип: {0}", p.MediaType));
                ListBoxPhysicalDisks.Items.Add(string.Format("Модель: {0}", p.Model));
                ListBoxPhysicalDisks.Items.Add(string.Format("Размер: {0} Гб\n", p.Size));
            }

            ListBoxLogicalDisks.Items.Add("ИНФОРМАЦИЯ О ЛОГИЧЕСКИХ ДИСКАХ\n");
            foreach (var p in LogicalDisk)
            {
                ListBoxLogicalDisks.Items.Add(string.Format("Тип: {0}", p.DriveType));
                ListBoxLogicalDisks.Items.Add(string.Format("Файловая система: {0}", p.FileSystem));
                ListBoxLogicalDisks.Items.Add(string.Format("Свободное место: {0} Гб", p.FreeSpace));
                ListBoxLogicalDisks.Items.Add(string.Format("Имя: {0}", p.Name));
                ListBoxLogicalDisks.Items.Add(string.Format("Объем: {0} Гб\n", p.Size));
            }
        }
```

```
        //=============================================================================================
        private static void GetMemory(ListBox listBox, ComputersContext db)
        {
            var Memory = from main in db.MainInfos
                         where main.Id == Windows.CommonInfoWindow.ComputerName
                         join memory in db.PhysicalMemories on main.Id equals memory.MainId
                         select new
                         {
                             memory.BankLabel,
                             memory.Capacity,
                             memory.DeviceLocator,
                             FormFactor = Convert.ConvertFormFactor(memory.FormFactor),
                             memory.Manufacturer,
                             MemoryType = Convert.ConvertMemoryType(memory.MemoryType),
                             memory.Speed

                         };

            listBox.Items.Add("ИНФОРМАЦИЯ ОБ ОПЕРАТИВНОЙ ПАМЯТИ\n");
            foreach (var p in Memory)
            {
                listBox.Items.Add(string.Format("Слот: {0}", p.BankLabel));
                listBox.Items.Add(string.Format("Емкость: {0} Гб", p.Capacity));
                listBox.Items.Add(string.Format("Расположение: {0}", p.DeviceLocator));
                listBox.Items.Add(string.Format("Форм-фактор: {0}", p.FormFactor));
                listBox.Items.Add(string.Format("Производитель: {0}", p.Manufacturer));
                listBox.Items.Add(string.Format("Скорость: {0} МГц", p.Speed));
                listBox.Items.Add(string.Format("Тип: {0}\n", p.MemoryType));
            }
        }
    }
}
```

Convert.cs

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace DomainComputersInfo
{
    public static class Convert
    {
        //=============================================================================================
        public static long? ConvertToGb(object Bytes)
        {
            return (long?)((ulong)Bytes / (ulong)Math.Pow(2, 30));
        }
        //=============================================================================================
        public static String ConvertMemoryType(int? MemoryType)
        {
            List<String> Type = new()
            {
                "DDR",
                "DDR2",
                "DDR3",
                "DDR4",
                "Unknown"
            };
            string Value = MemoryType switch
            {
                20 => Type.ElementAt(0),
                21 => Type.ElementAt(1),
                24 => Type.ElementAt(2),
                26 => Type.ElementAt(3),
                _ => Type.ElementAt(4),
            };
            return Value;
        }
```

```
        //=============================================================================================
        public static String ConvertWinVer(String Version)
        {
            if (Version.StartsWith("10"))
            {
                Version = Version[5..];
                List<String> Codename = new() { "1507", "1511", "1607", "1703", "1709", "1803", "1809",
"1903", "1909", "2004", "20H2" };
                List<String> Build = new() { "10240", "10586", "14393", "15063", "16299", "17134",
"17763", "18362", "18363", "19041", "19042" };
                int i = Build.IndexOf(Version);
                Version = Codename[i];
            }
            else if (Version.StartsWith("6.1.7601"))
            {
                Version = "SP1";
            }
            return Version;
        }
        //=============================================================================================
        public static String ConvertFormFactor(int? FormFactor)
        {
            List<String> Type = new()
            {
                "DIMM",
                "SODIMM",
                "Unknown"
            };
            string Value = FormFactor switch
            {
                8 => Type.ElementAt(0),
                12 => Type.ElementAt(1),
                _ => Type.ElementAt(2),
            };
            return Value;
        }
        //=============================================================================================
        public static String ConvertMediaType(int? MediaType)
        {
            List<String> Type = new()
            {
                "Unknown",
                "No Root Directory",
                "Removable Disk",
                "Local Disk",
                "Network Drive",
                "Compact Disc",
                "RAM Disk"
            };
            string Value = MediaType switch
            {
                1 => Type.ElementAt(1),
                2 => Type.ElementAt(2),
                3 => Type.ElementAt(3),
                4 => Type.ElementAt(4),
                5 => Type.ElementAt(5),
                6 => Type.ElementAt(6),
                _ => Type.ElementAt(0),
            };
            return Value;
        }
        //=============================================================================================
    }
}
```

Statistics.cs

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace DomainComputersInfo
{
    class Statistics
    {
        //=============================================================================================
        public static void GetFromDB()
        {
            using ComputersContext db = new();
            var Statistics = from main in db.MainInfos
                             join processor in db.Processors on main.Id equals processor.MainId
                             join memory in db.TotalMemories on main.Id equals memory.MainId
                             join os in db.OperatingSystems on main.Id equals os.MainId
                             select new
                             {
                                 proc = processor.Name,
                                 socket = processor.Socket,
                                 virt = processor.VirtualizationEnabled,
                                 memory = memory.Capacity,
                                 memtype = memory.MemoryType,
                                 osname = os.Caption,
                             };
            List<String> procname = new();
            List<String> socket = new();
            List<bool> virtualization = new();
            List<long> memsize = new();
            List<int> memtype = new();
            List<String> osName = new();
            foreach (var p in Statistics)
            {
                procname.Add(p.proc);
                socket.Add(p.socket);
                virtualization.Add((bool)p.virt);
                memsize.Add((long)(p.memory + 1));
                memtype.Add((int)p.memtype);
                osName.Add(p.osname);
            }
            Logics.StatValues.Processor = GetProcessorStatistics(procname);
            Logics.StatValues.Socket = GetSocketStatistics(socket);
            Logics.StatValues.Virtualization = GetVirtualizationStatistics(virtualization);
            Logics.StatValues.MemorySize = GetMemorySizeStatistics(memsize);
            Logics.StatValues.MemoryType = GetMemoryTypeStatistics(memtype);
            Logics.StatValues.OSName = GetOSNameStatistics(osName);
            Logics.StatValues.OSType = GetOSTypeStatistics(osName);
        }
        //=============================================================================================
        private static List<int> GetProcessorStatistics(List<String> items)
        {
            var (i3, i5, i7, i9, other) = (0, 0, 0, 0, 0);
            foreach (var p in items)
            {
                if (p.Contains("i3")) i3++;
                else if (p.Contains("i5")) i5++;
                else if (p.Contains("i7")) i7++;
                else if (p.Contains("i9")) i9++;
                else other++;
            }
            List<int> result = new() { i3, i5, i7, i9, other };
            return result;
        }
        //=============================================================================================
        private static List<int> GetSocketStatistics(List<String> items)
        {
            var (lga1156, lga1155, lga1150, lga1151, lgaOther) = (0, 0, 0, 0, 0);
            foreach (var p in items)
            {
                if (p.Contains("1156")) lga1156++;
                else if (p.Contains("1155")) lga1155++;
                else if (p.Contains("1150")) lga1150++;
                else if (p.Contains("1151")) lga1151++;
```

```
                else lgaOther++;
            }
            List<int> result = new() { lga1156, lga1155, lga1150, lga1151, lgaOther };
            return result;
        }
        //=============================================================================================
        private static List<int> GetVirtualizationStatistics(List<bool> items)
        {
            var (vTrue, vFalse) = (0, 0);
            foreach (var p in items)
            {
                if (p) vTrue++;
                else vFalse++;
            }
            List<int> result = new() { vTrue, vFalse };
            return result;
        }
        //=============================================================================================
        private static List<int> GetMemorySizeStatistics(List<long> items)
        {
            var (gb4, gb8, gb16, gb32, gb64, other) = (0, 0, 0, 0, 0, 0);
            foreach (var p in items)
            {
                if (p == 4) gb4++;
                else if (p == 8) gb8++;
                else if (p == 16) gb16++;
                else if (p == 32) gb32++;
                else if (p == 64) gb64++;
                else other++;
            }
            List<int> result = new() { gb4, gb8, gb16, gb32, gb64, other };
            return result;
        }
        //=============================================================================================
        private static List<int> GetMemoryTypeStatistics(List<int> items)
        {
            var (ddr3, ddr4, other) = (0, 0, 0);
            foreach (var p in items)
            {
                if (p == 24) ddr3++;
                else if (p == 26) ddr4++;
                else other++;
            }
            List<int> result = new() { ddr3, ddr4, other };
            return result;
        }
        //=============================================================================================
        private static List<int> GetOSNameStatistics(List<String> items)
        {
            var (w7, w10, other) = (0, 0, 0);
            foreach (var p in items)
            {
                if (p.Contains("Windows 7")) w7++;
                else if (p.Contains("Windows 10")) w10++;
                else other++;
            }
            List<int> result = new() { w7, w10, other };
            return result;
        }
        //=============================================================================================
        private static List<int> GetOSTypeStatistics(List<String> items)
        {
            var (pro, ent, other) = (0, 0, 0);
            foreach (var p in items)
            {
                if (p.Contains("Pro") || p.Contains("Про")) pro++;
                else if (p.Contains("Ent") || p.Contains("Корп")) ent++;
                else other++;
            }
            List<int> result = new() { pro, ent, other };
            return result;
        }
        //=============================================================================================
    }
}
```

Values.cs

```
using System;
using System.Collections.Generic;

namespace DomainComputersInfo.Logics
{
    //=============================================================================================
    public class DisplayData
    {
        public String PCname { get; set; }
        public String OSname { get; set; }
        public String OSversion { get; set; }
        public String ProcName { get; set; }
        public String Socket { get; set; }
        public String VirtEn { get; set; }
        public String BBname { get; set; }
        public String Ipaddr { get; set; }
        public String Macaddr { get; set; }
        public String Memory { get; set; }
        public String Memtype { get; set; }
    }
    //=============================================================================================
    public static class Credentials
    {
        public static String Login { get; set; }
        public static String Password { get; set; }
        public static String Domain { get; set; }
    }
    //=============================================================================================
    public static class WMI
    {
        public static readonly List<String> Path = new()
        {
            "Win32_BaseBoard",
            "Win32_DiskDrive",
            "Win32_LogicalDisk",
            "Win32_NetworkAdapterConfiguration",
            "Win32_OperatingSystem",
            "Win32_PhysicalMemory",
            "Win32_Processor",
            "Win32_UserProfile",
            "Win32_ComputerSystem"
        };
    }
    //=============================================================================================
    public static class StatValues
    {
        public static List<int> Processor { get; set; }
        public static List<int> Socket { get; set; }
        public static List<int> Virtualization { get; set; }
        public static List<int> MemorySize { get; set; }
        public static List<int> MemoryType { get; set; }
        public static List<int> OSName { get; set; }
        public static List<int> OSType { get; set; }
    }
    //=============================================================================================
}
```

ПРИЛОЖЕНИЕ 5 – ИСХОДНЫЙ КОД КЛАССОВ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА

MainWindow

```
<Window x:Class="DomainComputersInfo.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        mc:Ignorable="d"
        Title="Информация о компьютерах" Width="599" ResizeMode="NoResize" SizeToContent="WidthAndHeight"
WindowStartupLocation="CenterScreen">
    <Grid>
        <Button x:Name="Database" ToolTip="Работа с базой данных" Click="Database_Click"
                HorizontalAlignment="Left" VerticalAlignment="Top" Width="200" Height="200"
                BorderBrush="#8FC858" Background="#8FC858" Foreground="White">
            <StackPanel>
                <Image Source="/Icons/database.png" Height="100" Width="100"/>
            </StackPanel>
        </Button>
        <Button x:Name="CommonInfo" ToolTip="Общая информация о компьютерах" Click="CommonInfo_Click"
                HorizontalAlignment="Left" Margin="200,0,0,0" VerticalAlignment="Top" Width="200"
Height="200"
                Background="#DF7162" BorderBrush="#DF7162" Foreground="White">
            <StackPanel>
                <Image Source="/Icons/spreadsheet.png" Height="100" Width="100"/>
            </StackPanel>
        </Button>
        <Button x:Name="Statistics" ToolTip="Вывод статистики" Click="Statistics_Click"
                HorizontalAlignment="Left" Margin="400,0,0,0" VerticalAlignment="Top" Width="200"
Height="200"
                Foreground="White" BorderBrush="#3D8B82" Background="#3D8B82">
            <StackPanel>
                <Image Source="/Icons/chart.png" Height="100" Width="100"/>
            </StackPanel>
        </Button>
    </Grid>
</Window>
```

```
using System.Windows;

namespace DomainComputersInfo
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }

//=============================================================================================
        private void Database_Click(object sender, RoutedEventArgs e)
        {
            Windows.DatabaseWindow databaseWindow = new();
            databaseWindow.Show();
        }

//=============================================================================================
        private void CommonInfo_Click(object sender, RoutedEventArgs e)
        {
            Windows.CommonInfoWindow commonInfoWindow = new();
            commonInfoWindow.Show();
        }
//=============================================================================================
        private void Statistics_Click(object sender, RoutedEventArgs e)
        {
            Windows.StatisticsWindow statisticsWindow = new();
            statisticsWindow.Show();
        }
```

```
//=============================================================================================
        private void DetailedInfo_Click(object sender, RoutedEventArgs e)
        {
            Windows.DetailedInfoWindow detailedInfoWindow = new();
            detailedInfoWindow.Show();
        }
    }
}
```

DatabaseWindow

```
<Window x:Class="DomainComputersInfo.Windows.DatabaseWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        mc:Ignorable="d"
        Title="Работа с базой данных" ResizeMode="NoResize" SizeToContent="WidthAndHeight"
WindowStartupLocation="CenterScreen" Width="400">
    <Grid>
        <Button x:Name="Create" ToolTip="Создать базу данных" Click="Create_Click"
                HorizontalAlignment="Left" VerticalAlignment="Top" Width="200" Height="200"
                BorderBrush="#8FC858" Background="#8FC858" Foreground="White">
            <StackPanel>
                <Image Source="/Icons/create.png" Height="100" Width="100"/>
            </StackPanel>
        </Button>
        <Button x:Name="Fill" ToolTip="Заполнить базу данных" Click="Fill_Click"
                HorizontalAlignment="Left" Margin="200,0,0,0" VerticalAlignment="Top" Width="200"
Height="200"
                Background="#DF7162" BorderBrush="#DF7162" Foreground="White">
            <StackPanel>
                <Image Source="/Icons/fill.png" Height="100" Width="100"/>
            </StackPanel>
        </Button>
    </Grid>
</Window>
```

```
using System.Windows;

namespace DomainComputersInfo.Windows
{
    public partial class DatabaseWindow : Window
    {
        public DatabaseWindow()
        {
            InitializeComponent();
        }
//=============================================================================================
        private void Create_Click(object sender, RoutedEventArgs e)
        {
            Windows.AcceptDeletion acceptDeletion = new();
            if (acceptDeletion.ShowDialog() == true)
            {
                DomainComputersInfo.WorkWithDB.RecreateDB();
            }
            else MessageBox.Show("Действие отменено");
        }
//=============================================================================================
        private void Fill_Click(object sender, RoutedEventArgs e)
        {
            SelectionWindow selectionWindow = new();
            selectionWindow.Show();
        }
    }
}
```

AcceptDeletion

```
<Window x:Class="DomainComputersInfo.Windows.AcceptDeletion"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        mc:Ignorable="d"
        Title="Подтверждение" Width="400" ResizeMode="NoResize" SizeToContent="WidthAndHeight"
WindowStartupLocation="CenterScreen">
    <Grid>
        <Button x:Name="Yes" ToolTip="Удалить всю текущую информацию из базы и создать ее заново"
Click="Yes_Click" IsDefault="True"
                HorizontalAlignment="Left" VerticalAlignment="Top" Width="200" Height="200"
                BorderBrush="#E78383" Background="#E78383" Foreground="White">
            <StackPanel>
                <Image Source="/Icons/delete.png" Height="100" Width="100"/>
            </StackPanel>
        </Button>
        <Button x:Name="No" ToolTip="Отменить действие" IsCancel="True"
                HorizontalAlignment="Left" Margin="200,0,0,0" VerticalAlignment="Top" Width="200"
Height="200"
                Background="#6B80B0" BorderBrush="#6B80B0" Foreground="White" Click="No_Click">
            <StackPanel>
                <Image Source="/Icons/cancel.png" Height="100" Width="100"/>
            </StackPanel>
        </Button>
    </Grid>
</Window>
```

```
using System.Windows;

namespace DomainComputersInfo.Windows
{
    public partial class AcceptDeletion : Window
    {
        public AcceptDeletion()
        {
            InitializeComponent();
        }

//=============================================================================================
        private void Yes_Click(object sender, RoutedEventArgs e)
        {
            MessageBox.Show("Действие выполнено");
            this.DialogResult = true;
        }
//=============================================================================================
        private void No_Click(object sender, RoutedEventArgs e)
        {
            this.DialogResult = false;
        }
    }
}
```

SelectionWindow

```
<Window x:Class="DomainComputersInfo.Windows.SelectionWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        mc:Ignorable="d"
        Title="Выбор способа работы с базой" ResizeMode="NoResize" SizeToContent="WidthAndHeight"
WindowStartupLocation="CenterScreen" Width="400">
    <Grid>
        <Button x:Name="Automatic" ToolTip="Заполнить базу из домена" Click="Automatic_Click"
                HorizontalAlignment="Left" VerticalAlignment="Top" Width="200" Height="200"
                BorderBrush="#9D66AF" Background="#9D66AF" Foreground="White">
            <StackPanel>
                <Image Source="/Icons/ad.png" Height="100" Width="100"/>
            </StackPanel>
        </Button>
        <Button x:Name="Manual" ToolTip="Заполнить базу по имени компьютеров" Click="Manual_Click"
                HorizontalAlignment="Left" Margin="200,0,0,0" VerticalAlignment="Top" Width="200"
Height="200"
                Background="#DDE581" BorderBrush="#DDE581" Foreground="White">
            <StackPanel>
                <Image Source="/Icons/hand.png" Height="100" Width="100"/>
            </StackPanel>
        </Button>
    </Grid>
</Window>
```

```
using System.Windows;

namespace DomainComputersInfo.Windows
{
    public partial class SelectionWindow : Window
    {
        public SelectionWindow()
        {
            InitializeComponent();
        }
//============================================================================================
        private void Automatic_Click(object sender, RoutedEventArgs e)
        {
            Windows.Credentials credentials = new();
            if (credentials.ShowDialog() == true)
            {

DomainComputersInfo.WorkWithDB.GetAllInfo(GetComputers.GetFromAD(Logics.Credentials.Domain),
Logics.Credentials.Login, Logics.Credentials.Password);
                MessageBox.Show("База заполнена");

            }
            else MessageBox.Show("Введите учетные данные!");
        }
//============================================================================================
        private void Manual_Click(object sender, RoutedEventArgs e)
        {
            ManualWindow manualWindow = new();
            manualWindow.Show();
        }
    }
}
```

ManualWindow

```
<Window x:Class="DomainComputersInfo.Windows.ManualWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        mc:Ignorable="d"
        Title="Список компьютеров" Height="220" Width="380" ResizeMode="NoResize"
SizeToContent="WidthAndHeight">
    <Grid Margin="0,0,20,20">
        <TextBox x:Name="pcname" HorizontalAlignment="Left" Margin="20,39,0,0" TextWrapping="Wrap"
VerticalAlignment="Top" Width="160" Height="25"/>
        <TextBox x:Name="domain" HorizontalAlignment="Left" Margin="200,39,0,0" TextWrapping="Wrap"
VerticalAlignment="Top" Width="162" Height="25"/>
        <TextBlock HorizontalAlignment="Left" Margin="20,10,0,0" Text="Имя компьютера" TextWrapping="Wrap"
VerticalAlignment="Top" FontSize="14" Height="20" Width="160"/>
        <TextBlock HorizontalAlignment="Left" Margin="200,10,0,0" Text="Домен" TextWrapping="Wrap"
VerticalAlignment="Top" FontSize="14" Width="160" Height="20"/>
        <Button x:Name="add" ToolTip="Добавить компьютер в список" Click="add_Click"
                HorizontalAlignment="Left" VerticalAlignment="Top" Width="100" Height="100"
                BorderBrush="White" Background="White" Foreground="White" Margin="20,84,0,0">
            <StackPanel>
                <Image Source="/Icons/add.png" Height="95" Width="95"/>
            </StackPanel>
        </Button>
        <Button x:Name="ok" ToolTip="Внести компьютеры в базу" Click="ok_Click"
                HorizontalAlignment="Left" Margin="140,84,0,0" VerticalAlignment="Top" Width="100"
Height="100"
                Background="White" BorderBrush="White" Foreground="White">
            <StackPanel>
                <Image Source="/Icons/ok.png" Height="95" Width="95"/>
            </StackPanel>
        </Button>
        <Button x:Name="cancel" ToolTip="Отмена" Click="cancel_Click"
                HorizontalAlignment="Left" Margin="260,84,0,0" VerticalAlignment="Top" Width="100"
Height="100"
                Foreground="White" BorderBrush="White" Background="White">
            <StackPanel>
                <Image Source="/Icons/cancel2.png" Height="95" Width="95"/>
            </StackPanel>
        </Button>
    </Grid>
</Window>
```

```
using System;
using System.Collections.Generic;
using System.Windows;

namespace DomainComputersInfo.Windows
{
    public partial class ManualWindow : Window
    {
        public ManualWindow()
        {
            InitializeComponent();
            PCNames.Names = new();
        }
//============================================================================================
        private void add_Click(object sender, RoutedEventArgs e)
        {

            if (pcname.Text != "" && domain.Text != "")
            {
              PCNames.Names = GetComputers.GetManual(domain.Text, pcname.Text, PCNames.Names);
            }
        }
//============================================================================================
        private void ok_Click(object sender, RoutedEventArgs e)
        {
            Credentials credentials = new();
            if (credentials.ShowDialog() == true)
            {
                if (PCNames.Names.Count > 0)
```

```
                {
                    DomainComputersInfo.WorkWithDB.GetAllInfo(PCNames.Names, Logics.Credentials.Login,
Logics.Credentials.Password);
                    MessageBox.Show("База заполнена");
                }
                else MessageBox.Show("Список компьютеров пуст");
            }
            else MessageBox.Show("Введите учетные данные!");
        }
//=============================================================================================
        private static class PCNames
        {
            public static List<String> Names { get; set; }
        }
//=============================================================================================
        private void cancel_Click(object sender, RoutedEventArgs e)
        {
            this.Close();
        }
    }
}
```

Credentials

```
<Window x:Class="DomainComputersInfo.Windows.Credentials"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        mc:Ignorable="d"
        Title="Учетные данные" SizeToContent="WidthAndHeight" ResizeMode="NoResize"
WindowStartupLocation="CenterScreen">
    <Grid Margin="0,0,10,10">
        <TextBox x:Name="login" HorizontalAlignment="Left" Text="" TextWrapping="Wrap"
VerticalAlignment="Top" FontSize="16" Width="150" Margin="90,9,0,0"/>
        <PasswordBox x:Name="password" HorizontalAlignment="Left" VerticalAlignment="Top" Width="150"
FontSize="16" Margin="90,37,0,0"/>
        <TextBox x:Name="domain" HorizontalAlignment="Left" Text="" TextWrapping="Wrap"
VerticalAlignment="Top" Width="150" FontSize="16" Margin="90,65,0,0"/>
        <TextBlock HorizontalAlignment="Left" Margin="10,11,0,0" Text="Логин" TextWrapping="Wrap"
VerticalAlignment="Top" FontSize="16" Width="80"/>
        <TextBlock HorizontalAlignment="Left" Text="Пароль" TextWrapping="Wrap" VerticalAlignment="Top"
FontSize="16" Width="80" Margin="10,39,0,0"/>
        <TextBlock HorizontalAlignment="Left" Text="Домен" TextWrapping="Wrap" VerticalAlignment="Top"
FontSize="16" Width="80" Margin="10,67,0,0"/>
        <Button x:Name="ok" ToolTip="OK" HorizontalAlignment="Left" VerticalAlignment="Top"
Click="ok_Click" Height="50" Width="50" Margin="65,95,0,0" Background="White" BorderBrush="White">
            <StackPanel>
                <Image Source="/Icons/ok.png" Height="45"/>
            </StackPanel>
        </Button>
        <Button x:Name="cancel" ToolTip="Отмена" HorizontalAlignment="Left" Margin="135,95,0,0"
VerticalAlignment="Top" Height="50" Width="50" Click="cancel_Click" BorderBrush="White"
Background="White">
            <StackPanel>
                <Image Source="/Icons/cancel2.png" Height="45"/>
            </StackPanel>
        </Button>
    </Grid>
</Window>
```

```
using System.Windows;

namespace DomainComputersInfo.Windows
{
    public partial class Credentials : Window
    {
        public Credentials()
        {
            InitializeComponent();
        }
```

```
//=============================================================================================
        private void ok_Click(object sender, RoutedEventArgs e)
        {
            Logics.Credentials.Login = login.Text;
            Logics.Credentials.Password = password.Password;
            Logics.Credentials.Domain = domain.Text;
            if (Logics.Credentials.Login != "" && Logics.Credentials.Password != "" &&
Logics.Credentials.Domain != "")
                this.DialogResult = true;
            else
                this.DialogResult = false;
        }
//=============================================================================================
        private void cancel_Click(object sender, RoutedEventArgs e)
        {
            this.DialogResult = false;
        }
    }
}
```

CommonInfoWindow

```
<Window x:Class="DomainComputersInfo.Windows.CommonInfoWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        mc:Ignorable="d"
        Title="Основная информация о компьютерах" ResizeMode="NoResize" SizeToContent="WidthAndHeight"
WindowStartupLocation="CenterScreen" Width="1280" Height="768">
    <Grid>
        <ListView x:Name="MainInfoListView" Height="668" Margin="22,66,20,0" VerticalAlignment="Top"
SelectionMode="Single">
            <ListView.ContextMenu>
                <ContextMenu>
                    <MenuItem Header="Копировать имя компьютера" x:Name="Copy" Click="Copy_Click"/>
                    <MenuItem Header="Удалить" x:Name="Delete" Click="Delete_Click"/>
                    <Separator/>
                    <MenuItem Header="Подробная информация" x:Name="Details" Click="Details_Click"/>
                </ContextMenu>
            </ListView.ContextMenu>
            <ListView.View>
                <GridView>
                    <GridViewColumn Header="Имя компьютера" DisplayMemberBinding="{Binding PCname}"/>
                    <GridViewColumn Header="Операционная система" DisplayMemberBinding="{Binding
OSname}"/>
                    <GridViewColumn Header="Версия ОС" DisplayMemberBinding="{Binding OSversion}"/>
                    <GridViewColumn Header="Процессор" DisplayMemberBinding="{Binding ProcName}"/>
                    <GridViewColumn Header="Сокет" DisplayMemberBinding="{Binding Socket}"/>
                    <GridViewColumn Header="Виртуализация" DisplayMemberBinding="{Binding VirtEn}"/>
                    <GridViewColumn Header="Материнская плата" DisplayMemberBinding="{Binding BBname}"/>
                    <GridViewColumn Header="IP адрес" DisplayMemberBinding="{Binding Ipaddr}"/>
                    <GridViewColumn Header="MAC адрес" DisplayMemberBinding="{Binding Macaddr}"/>
                    <GridViewColumn Header="Размер памяти" DisplayMemberBinding="{Binding Memory}"/>
                    <GridViewColumn Header="Тип памяти" DisplayMemberBinding="{Binding Memtype}"/>
                </GridView>
            </ListView.View>
        </ListView>
        <TextBlock Text="Фильтр по имени компьютера" TextWrapping="Wrap" Margin="20,20,1050,708"
FontSize="14" Height="25" Width="205"/>
        <TextBox x:Name="PCName_Filter" TextChanged="PCName_Filter_TextChanged" TextWrapping="Wrap"
Margin="240,20,835,707" FontSize="14" Height="25" Width="205"/>
    </Grid>
</Window>


using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
namespace DomainComputersInfo.Windows
{
    public partial class CommonInfoWindow : Window
    {
```

```
        public static String ComputerName;
        public static CollectionView view;
//=============================================================================================
        public CommonInfoWindow()
        {
            InitializeComponent();
            this.Loaded += CommonInfoWindow_Loaded;
        }
//=============================================================================================
        private bool GetPCName()
        {
            if (MainInfoListView.SelectedItems.Count > 0)
            {
                var value = (Logics.DisplayData)MainInfoListView.SelectedItem;
                ComputerName = value.PCname;
                return true;
            }
            else return false;
        }
//=============================================================================================
        private void CommonInfoWindow_Loaded(object sender, RoutedEventArgs e)
        {
            Logics.GetFromDB.GetMainInfo(MainInfoListView);
            view = (CollectionView)CollectionViewSource.GetDefaultView(MainInfoListView.ItemsSource);
        }
//=============================================================================================
        private void Copy_Click(object sender, RoutedEventArgs e)
        {
            if (GetPCName())
            {
                Clipboard.SetText(ComputerName);
            }
        }
//=============================================================================================
        private void Details_Click(object sender, RoutedEventArgs e)
        {
            if (GetPCName())
            {
                Windows.DetailedInfoWindow detailedInfoWindow = new();
                detailedInfoWindow.Show();
            }
        }
//=============================================================================================
        private bool PCNameFilter(object item)
        {
            if (String.IsNullOrEmpty(PCName_Filter.Text))
                return true;
            else
                return (item as Logics.DisplayData).PCname.IndexOf(PCName_Filter.Text,
StringComparison.OrdinalIgnoreCase) >= 0;
        }
//=============================================================================================
        private void PCName_Filter_TextChanged(object sender, TextChangedEventArgs e)
        {
            view.Filter = PCNameFilter;
            CollectionViewSource.GetDefaultView(MainInfoListView.ItemsSource).Refresh();
        }
//=============================================================================================
        private void Delete_Click(object sender, RoutedEventArgs e)
        {
            if (GetPCName())
            {
                Windows.AcceptDeletion acceptDeletion = new();
                if (acceptDeletion.ShowDialog() == true)
                {
                    DomainComputersInfo.WorkWithDB.DeleteEntry();
                }
                else MessageBox.Show("Действие отменено");
            }
        }
    }
}
```

```
//=============================================================================================
        private void Details_Click(object sender, RoutedEventArgs e)
        {
            if (MainInfoListView.SelectedItems.Count > 0)
            {
                var value = (Logics.DisplayData)MainInfoListView.SelectedItem;
                ComputerName = value.PCname;
            }
            Windows.DetailedInfoWindow detailedInfoWindow = new();
            detailedInfoWindow.Show();
        }

//=============================================================================================
        private bool PCNameFilter(object item)
        {
            if (String.IsNullOrEmpty(PCName_Filter.Text))
                return true;
            else
                return (item as Logics.DisplayData).PCname.IndexOf(PCName_Filter.Text,
StringComparison.OrdinalIgnoreCase) >= 0;
        }
//=============================================================================================
        private void PCName_Filter_TextChanged(object sender, TextChangedEventArgs e)
        {
            view.Filter = PCNameFilter;
            CollectionViewSource.GetDefaultView(MainInfoListView.ItemsSource).Refresh();
        }
    }
}
```

DetailedInfoWindow

```
<Window x:Class="DomainComputersInfo.Windows.DetailedInfoWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        mc:Ignorable="d"
        Height="512" Width="1024">
    <Grid>
        <TabControl>
            <TabItem Header="Материнская плата и процессор">
                <Grid Background="#FFE5E5E5" Width="1000">
                    <StackPanel Margin="0,0,0,0" Orientation="Horizontal">
                        <ListBox x:Name="ListBoxBaseBoard" FontSize="14"  Width="500" Height="468"/>
                        <ListBox x:Name="ListBoxProcessor" FontSize="14" Width="500" Height="468"/>
                    </StackPanel>
                </Grid>
            </TabItem>
            <TabItem Header="Операционная система и сетевой адаптер">
                <Grid Background="#FFE5E5E5" Width="1000">
                    <StackPanel Margin="0,0,600,0" Orientation="Vertical">
                        <ListBox x:Name="ListBoxOperatingSystem" FontSize="14" Width="400" Height="234"/>
                        <ListBox x:Name="ListBoxNetworkAdapter" FontSize="14" Width="400" Height="234"/>
                    </StackPanel>
                    <ListBox x:Name="ListBoxUserProfile" FontSize="14" Margin="400,0,0,0"/>
                </Grid>
            </TabItem>
            <TabItem Header="Логические и физические диски">
                <Grid Background="#FFE5E5E5" Width="1000">
                    <StackPanel Margin="0,0,0,0" Orientation="Horizontal">
                        <ListBox x:Name="ListBoxLogicalDisks" FontSize="14"  Width="500" Height="468"/>
                        <ListBox x:Name="ListBoxPhysicalDisks" FontSize="14" Width="500" Height="468"/>
                    </StackPanel>
                </Grid>
            </TabItem>
            <TabItem Header="Оперативная память">
                <Grid Background="#FFE5E5E5">
                    <ListBox x:Name="ListBoxMemory" FontSize="14"/>
                </Grid>
            </TabItem>
        </TabControl>
    </Grid>
</Window>
```

```
using System.Windows;

namespace DomainComputersInfo.Windows
{
    public partial class DetailedInfoWindow : Window
    {
        public DetailedInfoWindow()
        {
            InitializeComponent();
            this.Title = CommonInfoWindow.ComputerName;
            Logics.GetAllFromDB.GetAllInfo(ListBoxProcessor, ListBoxBaseBoard, ListBoxOperatingSystem,
ListBoxUserProfile, ListBoxNetworkAdapter, ListBoxLogicalDisks, ListBoxPhysicalDisks, ListBoxMemory);
        }
    }
}
```

StatisticsWindow

```
<Window x:Class="DomainComputersInfo.Windows.StatisticsWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        mc:Ignorable="d"
        Title="Вывод статистики" Width="599" ResizeMode="NoResize" SizeToContent="WidthAndHeight"
WindowStartupLocation="CenterScreen">
    <Grid>
        <Button x:Name="OS" ToolTip="Статистика по операционным системам" Click="OS_Click"
                HorizontalAlignment="Left" VerticalAlignment="Top" Width="200" Height="200"
                BorderBrush="#8FC858" Background="#8FC858" Foreground="White">
            <StackPanel>
                <Image Source="/Icons/os.png" Height="100" Width="100"/>
            </StackPanel>
        </Button>
        <Button x:Name="Memory" ToolTip="Статистика по оперативной памяти" Click="Memory_Click"
                HorizontalAlignment="Left" Margin="200,0,0,0" VerticalAlignment="Top" Width="200"
Height="200"
                Background="#DF7162" BorderBrush="#DF7162" Foreground="White">
            <StackPanel>
                <Image Source="/Icons/ram.png" Height="100" Width="100"/>
            </StackPanel>
        </Button>
        <Button x:Name="Processor" ToolTip="Статистика по процессорам" Click="Processor_Click"
                HorizontalAlignment="Left" Margin="400,0,0,0" VerticalAlignment="Top" Width="200"
Height="200"
                Foreground="White" BorderBrush="#3D8B82" Background="#3D8B82">
            <StackPanel>
                <Image Source="/Icons/cpu.png" Height="100" Width="100"/>
            </StackPanel>
        </Button>
    </Grid>
</Window>
```

```
using System.Windows;

namespace DomainComputersInfo.Windows
{
    public partial class StatisticsWindow : Window
    {
        public StatisticsWindow()
        {
            InitializeComponent();
            Statistics.GetFromDB();
        }

//=============================================================================================
        private void Processor_Click(object sender, RoutedEventArgs e)
        {
            Charts.ProcessorChartWindow processorChartWindow = new();
            processorChartWindow.Show();
        }
```

```
//============================================================================================
        private void Memory_Click(object sender, RoutedEventArgs e)
        {
            Charts.MemoryChartWindow memoryChartWindow = new();
            memoryChartWindow.Show();
        }
//============================================================================================
        private void OS_Click(object sender, RoutedEventArgs e)
        {
            Charts.OSChartWindow oSChartWindow = new();
            oSChartWindow.Show();
        }
    }
}
```

MemoryChartWindow

```
<Window x:Class="DomainComputersInfo.Charts.MemoryChartWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:DomainComputersInfo.Charts"
        mc:Ignorable="d"
        Title="Статистика по оперативной памяти" Height="736" Width="1024">
    <Grid>
        <local:MemorySizeChart Margin="0,0,0,365"></local:MemorySizeChart>
        <local:MemoryTypeChart Margin="0,365,0,0"></local:MemoryTypeChart>
    </Grid>
</Window>
```

```
using System.Windows;

namespace DomainComputersInfo.Charts
{
    public partial class MemoryChartWindow : Window
    {
        public MemoryChartWindow()
        {
            InitializeComponent();
        }
    }
}
```

MemorySizeChart

```
<UserControl x:Class="DomainComputersInfo.Charts.MemorySizeChart"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:local="clr-namespace:DomainComputersInfo.Charts"
             xmlns:lvc="clr-namespace:LiveCharts.Wpf;assembly=LiveCharts.Wpf"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="800"
             d:DataContext="{d:DesignInstance local:MemorySizeChart}">
    <Grid>
        <lvc:CartesianChart Series="{Binding SeriesCollection}" LegendLocation="Left">
            <lvc:CartesianChart.AxisX>
                <lvc:Axis Title="Размер оперативной памяти" Labels="{Binding Labels}"></lvc:Axis>
            </lvc:CartesianChart.AxisX>
            <lvc:CartesianChart.AxisY>
                <lvc:Axis Title="Количество" LabelFormatter="{Binding Formatter}"></lvc:Axis>
            </lvc:CartesianChart.AxisY>
        </lvc:CartesianChart>
    </Grid>
</UserControl>
```

```
using LiveCharts;
using LiveCharts.Wpf;
using System;
using System.Windows.Controls;

namespace DomainComputersInfo.Charts
{
    public partial class MemorySizeChart : UserControl
    {
        public MemorySizeChart()
        {
            InitializeComponent();
            ChartValues<int> Quantity = new();

            foreach (int i in Statistics.StatValues.MemorySize)
            {
                Quantity.Add(i);
            }

            SeriesCollection = new SeriesCollection
            {
                new ColumnSeries
                {
                    Title = "",
                    Values = Quantity
                }
            };
            Labels = new[] { "4 Гб", "8 Гб", "16 Гб", "32 Гб", "64 Гб", "Другое" };
            Formatter = value => value.ToString("N");

            DataContext = this;
        }
//=============================================================================================
        public SeriesCollection SeriesCollection { get; set; }
        public string[] Labels { get; set; }
        public Func<double, string> Formatter { get; set; }
    }
}
```

MemoryTypeChart

```
<UserControl x:Class="DomainComputersInfo.Charts.MemoryTypeChart"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:local="clr-namespace:DomainComputersInfo.Charts"
             xmlns:lvc="clr-namespace:LiveCharts.Wpf;assembly=LiveCharts.Wpf"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="800"
             d:DataContext="{d:DesignInstance local:MemoryTypeChart}">
    <Grid>
        <lvc:CartesianChart Series="{Binding SeriesCollection}" LegendLocation="Left">
            <lvc:CartesianChart.AxisX>
                <lvc:Axis Title="Тип оперативной памяти" Labels="{Binding Labels}"></lvc:Axis>
            </lvc:CartesianChart.AxisX>
            <lvc:CartesianChart.AxisY>
                <lvc:Axis Title="Количество" LabelFormatter="{Binding Formatter}"></lvc:Axis>
            </lvc:CartesianChart.AxisY>
        </lvc:CartesianChart>
    </Grid>
</UserControl>
```

```
using LiveCharts;
using LiveCharts.Wpf;
using System;
using System.Windows.Controls;

namespace DomainComputersInfo.Charts
{
    public partial class MemoryTypeChart : UserControl
    {
        public MemoryTypeChart()
        {
            InitializeComponent();

            ChartValues<int> Quantity = new();

            foreach (int i in Statistics.StatValues.MemoryType)
            {
                Quantity.Add(i);
            }

            SeriesCollection = new SeriesCollection
            {
                new ColumnSeries
                {
                    Title = "",
                    Values = Quantity
                }
            };
            Labels = new[] { "DDR3", "DDR4", "Другие" };
            Formatter = value => value.ToString("N");

            DataContext = this;
        }
//=============================================================================================
        public SeriesCollection SeriesCollection { get; set; }
        public string[] Labels { get; set; }
        public Func<double, string> Formatter { get; set; }
    }
}
```