

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГТУ», ВГТУ)

Факультет экономики, менеджмента и информационных технологий
(факультет)

Кафедра Систем управления информационных технологий в строительстве

КУРСОВОЙ ПРОЕКТ

по дисциплине «Операционные системы»

тема «Разработка GUI-скрипта Powershell для управления службами»

Расчетно-пояснительная записка

Разработал студент

_____	Д. В. Тюленев
Подпись, дата	Инициалы, фамилия

Руководитель

_____	К.А. Маковий
Подпись, дата	Инициалы, фамилия

Члены комиссии

Подпись, дата	Инициалы, фамилия

Подпись, дата	Инициалы, фамилия

Нормоконтролер

_____	К.А. Маковий
Подпись, дата	Инициалы, фамилия

Защищена _____
дата

Оценка _____

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГТУ», ВГТУ)

Кафедра Систем управления информационных технологий в строительстве

ЗАДАНИЕ
на курсовой проект

по дисциплине «Операционные системы»

тема «Разработка GUI-скрипта Powershell для управления службами»

Студент группы ИСТ-214

Тюленев Данил Вячеславович
Фамилия, имя, отчество

Вариант №8

Технические условия процессор Intel® Core™ i3-8145U CPU @ 2,10 ГГц,
операционная система Windows 10, ОЗУ 8192 МБ

Содержание и объем проекта (графические работы, расчёты и прочее):
теоретический основы управления службами в ОС Windows (12 страниц);
Разработка GUI скрипта для управления службами SCM (13 страниц);
14 рисунков, 1 приложение

Сроки выполнения этапов: анализ и постановка задачи (01.03-15.03);
изучение теоретического обоснования работы (15.03-15.04); реализация
программного решения (15.04-15.05); оформление пояснительной записки
(15.05-05.06)

Срок защиты курсового проекта 06.06.2022

Руководитель

К.А. Маковий
Подпись, дата Инициалы, фамилия

Задание принял студент

Д. В. Тюленев
Подпись, дата Инициалы, фамилия

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	4
1 Теоретический основы управления службами в ОС Windows.....	5
1.1 Службы и Диспетчер управления службами SCM.....	5
1.2 Способы управления службами со стороны пользователей.....	10
2 Разработка GUI скрипта для управления службами SCM.....	17
2.1 Постановка задачи.....	17
2.2 Обоснование средств разработки.....	20
2.3 Описание работы системы.....	23
ЗАКЛЮЧЕНИЕ.....	30
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	31
ПРИЛОЖЕНИЕ А.....	34

ВВЕДЕНИЕ

В работающей операционной системе запущенно большое количество процессов. Эти процессы создаются как запущенными на компьютере программами, так и системными службами.

Службой Windows является процесс, способный запускаться на устройстве в фоновом режиме для выполнений определённых действий в ответ на запросы пользователей, приложений или системы в целом.

В настоящее время не возможно представить операционную систему без служб (или аналогичных инструментов).

Вы когда-нибудь задумывались, сколько функций Windows работает одновременно, даже если у них нет сложного интерфейса? Что ж, это всё службы Windows, которые работают в фоновом режиме, чтобы выполнять свою работу, не мешая вам.

Windows автоматически находит обновления, проверяет стабильность системы, подстраиваем время обновлений в зависимости от текущего часового пояса, записывает события и защищает систему от угроз. У всех этих функций нет работающего приложения на рабочем столе, тогда как ими управлять? Все эти простые операции ложатся на плечи системных служб.

С другой стороны, есть некоторые службы, которые отключены, но вы можете быть заинтересованы в их включении. Например, вы можете установить для службы «Удалённый реестр» значение Вручную, чтобы другие люди могли удаленно настраивать ваш реестр Windows.

Целью курсового проекта является разработка скрипта (сценария) оболочки платформы Powershell, для управления зарегистрированными в менеджере по управлению объектами служб в системе Windows 10.

1 Теоретический основы управления службами в ОС Windows

1.1 Службы и Диспетчер управления службами SCM

Службы Microsoft Windows (Windows Services), ранее известные как службы NT, позволяют создавать долговременные исполняемые приложения, которые запускаются в собственных сеансах Windows [1]; процесс, который запускается на устройстве в фоновом режиме для выполнения определённых действий в ответ на запросы пользователей, приложений или системы [2]; имеют общие черты с концепцией демонов в Unix. Для этих служб не предусмотрен пользовательский интерфейс. Они могут запускаться автоматически при загрузке компьютера, их также можно приостанавливать и перезапускать. Благодаря этому службы идеально подходят для использования на сервере, а также в ситуациях, когда необходимы долго выполняемые процессы, которые не мешают работе пользователей на том же компьютере [3]. Службы могут выполняться в контексте безопасности определённой учётной записи пользователя, которая отличается от учётной записи вошедшего в систему пользователя или учётной записи компьютера по умолчанию.

Можно легко создавать службы, создавая приложение, которое устанавливается как служба. Предположим, что нужно отслеживать данные счётчика производительности и реагировать на пороговые значения. Можно написать и развернуть приложение-службу Windows для прослушивания данных счётчиков, а затем начать сбор и анализ данных. Службу можно разработать как проект Windows Service в интегрированной среде разработки Microsoft Visual Studio с кодом, который определяет, какие команды могут отправляться службе и какие действия должны быть выполнены при получении этих команд [4]. Команды, которые могут быть отправлены в службу, выполняют запуск, приостановку, возобновление и остановку службы. Также можно выполнять пользовательские команды.

Созданное приложение можно установить, запустив служебную программу командной строки «InstallUtil.exe» и передав путь к исполняемому файлу службы [5]. Затем вы можете использовать диспетчер служб для запуска, остановки, приостановки, продолжения работы и настройки службы. Многие из этих задач также можно выполнить в узле службы в Обозреватель сервера или с помощью класса. Служба проходит через несколько внутренних состояний за время своего существования. Во-первых, служба устанавливается в системе, в которой она будет выполняться. Этот процесс выполняет установщики для проекта службы и загружает службу в диспетчер служб для этого компьютера. Служба Windows должна соответствовать правилам интерфейса и протоколам диспетчера управления службами, компонента, отвечающего за управление службами Windows.

До Windows Vista службы, установленные как «интерактивная служба», могли взаимодействовать с рабочим столом Windows и отображать графический пользовательский интерфейс. Однако в Windows Vista интерактивные службы устарели и могут работать неправильно в результате усиления безопасности службы Windows.

Основное различие между службами и обычными приложениями заключается в том, что службы управляются диспетчером управления службами (SCM) [6]. Службы реализуются с помощью API, который управляет взаимодействием между SCM и службами. SCM поддерживает базу данных установленных служб и предоставляет единый способ управления ими. Диспетчер управления служб (Service Control Manager, SCM) является основным средством управления службами в Windows; в Microsoft Windows (Windows\System32\Services.exe) особый системный процесс, реализующий технологию удалённого вызова процедур [7] (remote procedure call, RPC) — определяет мощный инструмент для создания распределенных клиентских и серверных программ.

Заглушки и библиотеки времени выполнения RPC управляют большинством процессов, связанных с сетевыми протоколами и обменом

данными. Это позволяет сосредоточиться на деталях приложения, а не на сведениях о сети. Обеспечивает создание, удаление, запуск и остановку служб ОС. SCM Стартует при загрузке системы, обеспечивает работу журнала событий, а также позволяет выполнять манипуляцию процессами удалённой машины [8].

При загрузке операционной системы SCM запускает все службы, у которых указан тип запуска «Автоматически», а также все службы, указанные в зависимостях автозапускаемых служб [9]. Таким образом, при запуске службы с типом запуска «Автоматически», у которой в зависимостях указаны службы с типом запуска «Вручную», последние также будут запущены, несмотря на свой тип запуска. При запуске службы диспетчер выполняет следующие действия: получение сохранённой в базе данных информации по учётной записи, с правами которой должна запускаться служба; авторизация под этой учётной записью; получение пользовательского профиля; подготовка процесса службы к выполнению; привязка доступов учётной записи к порождённому процессу; запуск процесса службы на выполнение [10]. После загрузки операционной системы пользователь может вручную запустить необходимые службы, воспользовавшись консолью управления службами. Пользователь также может указать параметры запуска службы, которые будут переданы как аргументы функции «StartService» при запуске.

Структура базы служб включает в себя следующие данные: тип службы; тип запуска; уровень контроля ошибок; полный путь к исполняемому файлу; информация о зависимостях службы; логин и пароль учётной записи, с правами которой нужно запускать службу [11]. Графическое представление диспетчера служб Windows продемонстрировано на рисунке 1.

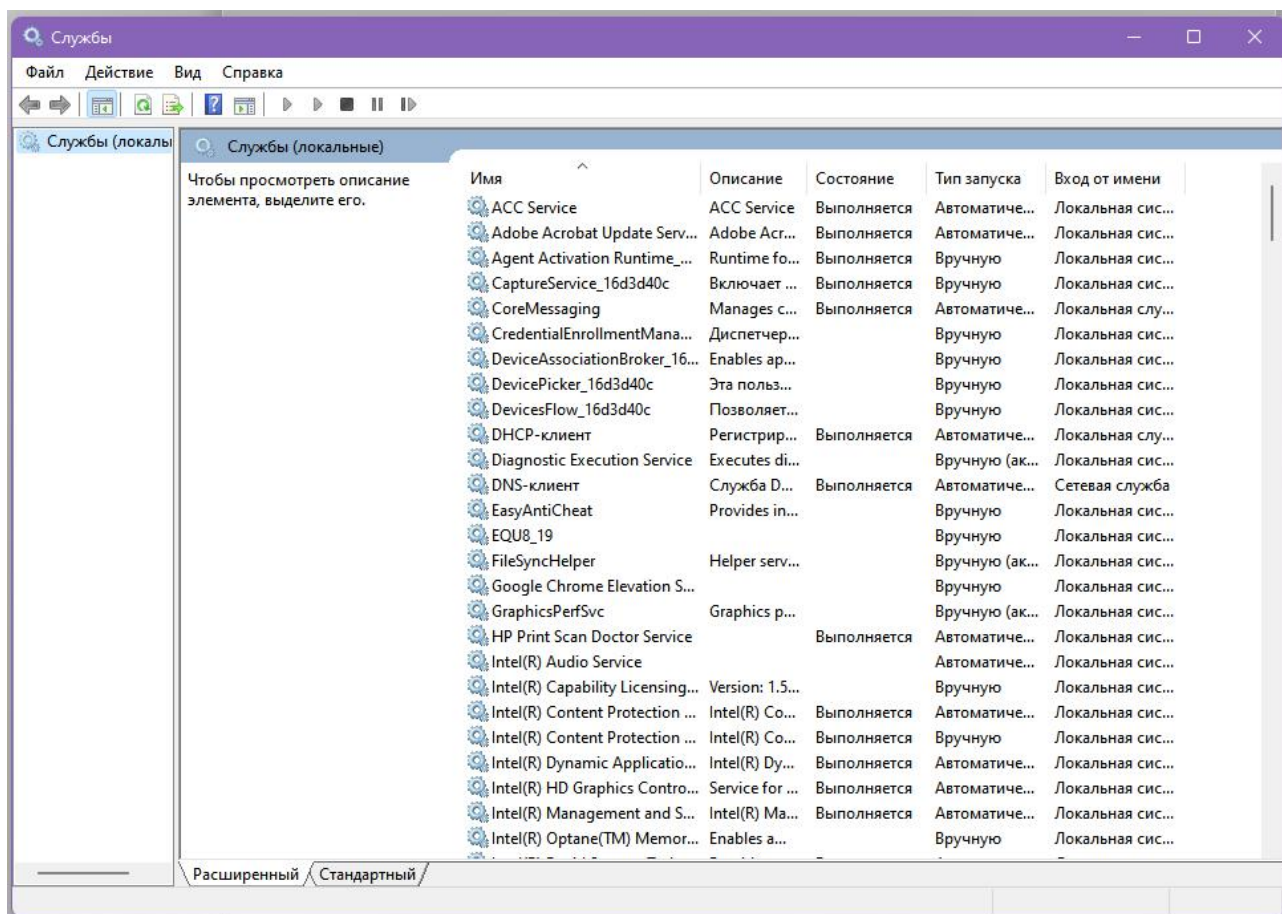


Рисунок 1 — Представление Менеджера служб (SCM).

Монтирование базы данных установленных служб операционной системы производится в системном реестре и располагается в ветке «HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services». Данная база используется SCM и другими программами для добавления, изменения или конфигурирования служб [12].

Функции менеджера представляют интерфейс для выполнения следующих задач: монтирование базы данных установленных служб; запуск служб при загрузке операционной системы (либо по требованию); получение количественной и качественной информации об установленных службах и системных драйверах; пересылка управляющих запросов запущенным службам; блокировка и разблокировка базы данных служб.

Службы Windows можно настроить на запуск при запуске операционной системы и работу в фоновом режиме до тех пор, пока работает

Windows. Кроме того, они могут быть запущены вручную или с помощью события. Операционные системы Windows NT включают в себя множество служб, которые работают в контексте трёх учётных записей пользователей: системной, сетевой службы и локальной службы. Эти компоненты Windows часто связаны с процессом узла для служб Windows. Поскольку службы Windows работают в контексте собственных выделенных учётных записей пользователей, они могут работать, когда пользователь не вошёл в систему. Службу можно запустить из диспетчера управления службами, из Обзорера сервера или из кода путём вызова метода. Метод «Start» передаёт обработку в метод «OnStart» приложения и обрабатывает любой код, определённый там [13].

Оснастка «Службы», встроенная в консоль управления (MMC), может подключаться к локальному компьютеру или удалённому компьютеру в сети, позволяя пользователям: просмотр списка установленных служб, а также имени службы, описаний и конфигурации; запуск, остановка, приостановка или перезапуск служб; указание параметров службы, если это применимо; изменение типа запуска; изменение контекста учётной записи пользователя, в котором работает служба; настройка действий по восстановлению, которые необходимо выполнить в случае сбоя службы; проверка зависимостей служб; определение того, какие службы или драйверы устройств зависят от данной службы или от каких служб или драйверов устройств зависит данная служба; экспорт списка служб в виде текстового файла или «CSV-файла».

Существует несколько режимов для служб: ручной запуск по запросу (Manual), автоматический запуск при загрузке компьютера; автоматический, отложенный запуск (Automatic Delayed), обязательная служба/драйвер (автоматический запуск и невозможность для пользователя остановить службу), запрещён к запуску (Disabled) [14].

Запущенная служба может находиться в этом состоянии бесконечно, пока она не будет остановлена или приостановлена либо работа компьютера не будет завершена. Есть три основных состояния службы: «Running»,

«Paused» и «Stopped». Служба также может сообщать состояние ожидания выполнения команды: «ContinuePending», «PausePending», «StartPending» или «StopPending». Эти состояния указывают, что команда выдана (например, команда для приостановки службы или запуска службы), но ещё не выполнена. Вы можете запросить свойство «Status», чтобы определить, в каком состоянии находится служба, или использовать «WaitForStatus», чтобы выполнить действие при наступлении любого из этих состояний. Вы можете приостановить, остановить или возобновить работу службы из диспетчера служб или обозревателя сервера либо из кода, вызвав методы [15]. Каждое из этих действий вызывает соответствующую процедуру в службе («OnStop», «OnPause» или «OnContinue»), в которой можно определить дополнительную обработку на случай изменения состояния службы.

1.2 Способы управления службами со стороны пользователей

Есть много способов получить доступ к средству управления службами Windows. Вы можете получить к нему доступ из панели управления в разделе «Инструменты администратора» или введите «Службы Windows» в системном поиске Windows, чтобы получить к нему доступ. Другой простой способ — открыть «Выполнить» (сочетание клавиш WIN + R) ввести «services.msc» в диалоговом окне, чтобы открыть консоль управления службами Windows.

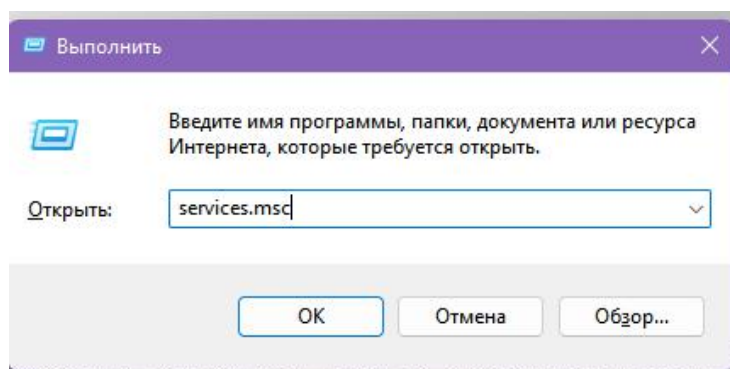


Рисунок 2 — Диалоговое окно «Выполнить» для запуска менеджера

Будет доступна Все службы Windows, перечисленные в правом столбце, и их описание в левом столбце. Нажмите на любую из служб, и вы увидите подробную информацию в левом столбце.

Также для отображения окна менеджера управления служб можно использовать утилиту «Управление Компьютером». Для запуска необходимо нажать правой кнопкой мыши по кнопке «Пуск» в панели задач и в открывшемся меню выберите пункт «Управление компьютером». Затем в открывшемся окне, в левой панели, в разделе «Службы и приложения» выберите «Службы». Список доступных служб откроется в этом же окне.

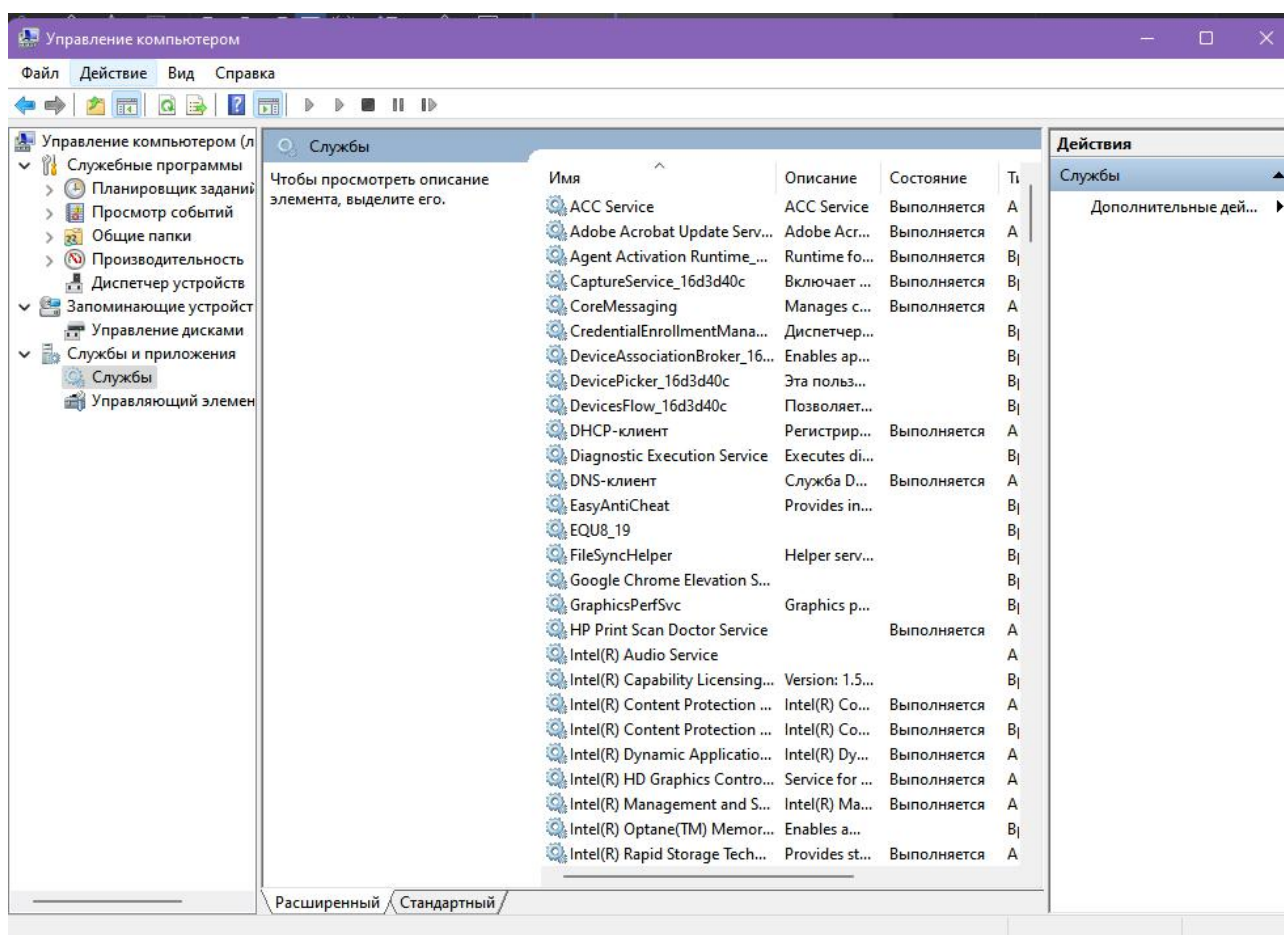


Рисунок 3 — Управление службами через «Управление компьютером»

Также получить доступ к службам можно при помощи консоли управления Microsoft «MMC» (Microsoft Management Console; используется

для создания, сохранения и открытия административных средств, называемых консоли, которые управляют аппаратными, программными и сетевыми компонентами операционной системы Майкрософт Windows). Для этого необходимо создать новую консоль «ММС»: Откроем окно «ММС», из консоли выберем пункт «Добавить или удалить оснастку». В появившемся диалоговом окне необходимо включить в список «Выбранные оснастки» элемент «Службы» и нажать «ОК», после чего в текущей редактируемой консоли будет доступен менеджер служб Windows внутри левой панели, где располагается «Корень консоли».

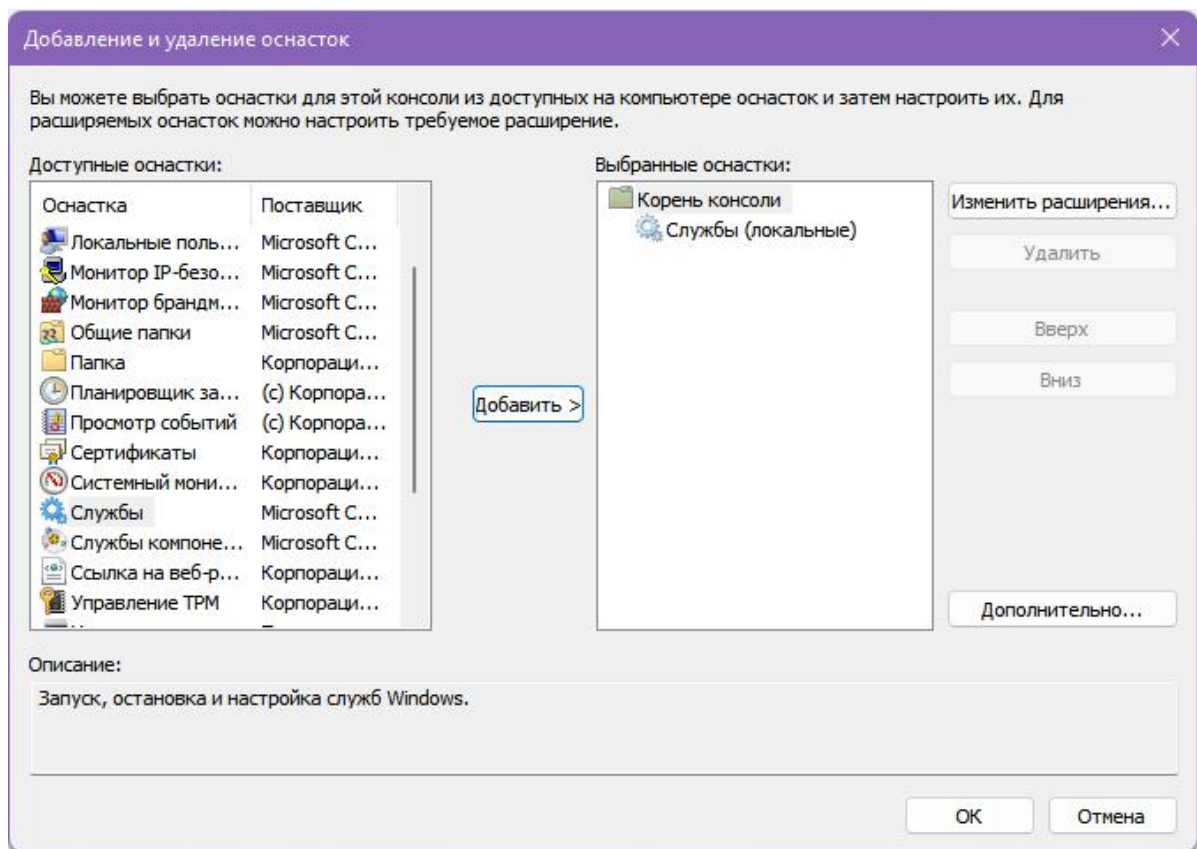


Рисунок 4 — Добавление оснастки диспетчера служб

Есть возможность принудительно запустить или остановить службу во время текущего сеанса Windows, для этого необходимо в открытом окне диспетчера служб щёлкнуть правой кнопкой мыши по необходимой службе в списке доступных сервисов, после чего появится контекстное меню с

пунктами «Запустить» или «Остановить». Если служба уже запущена, можно будет использовать кнопку «Остановить», в противном случае вы увидите «Запустить», если служба не отключена. Окно свойств выбранной службы показано на рисунке 5.

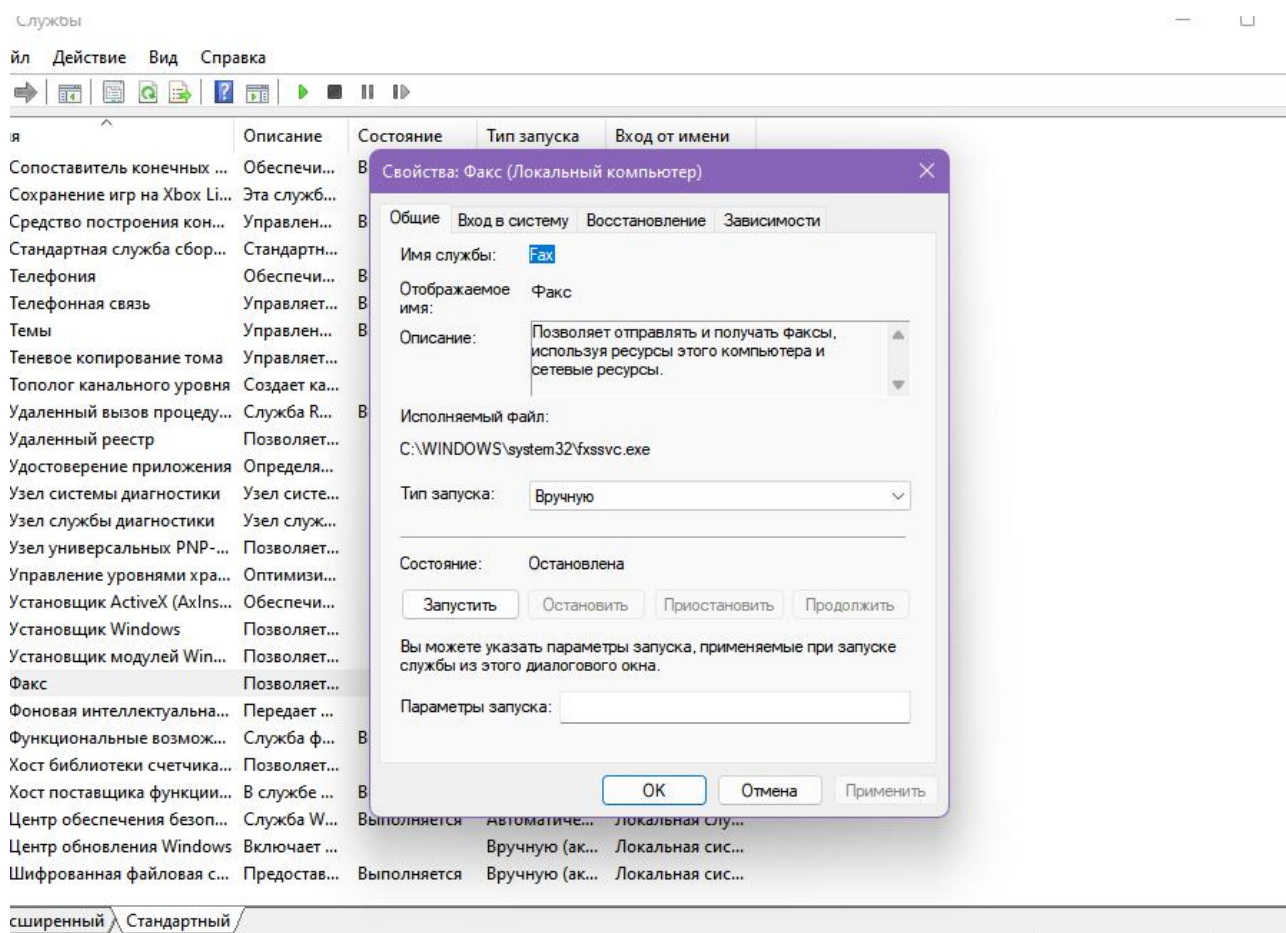


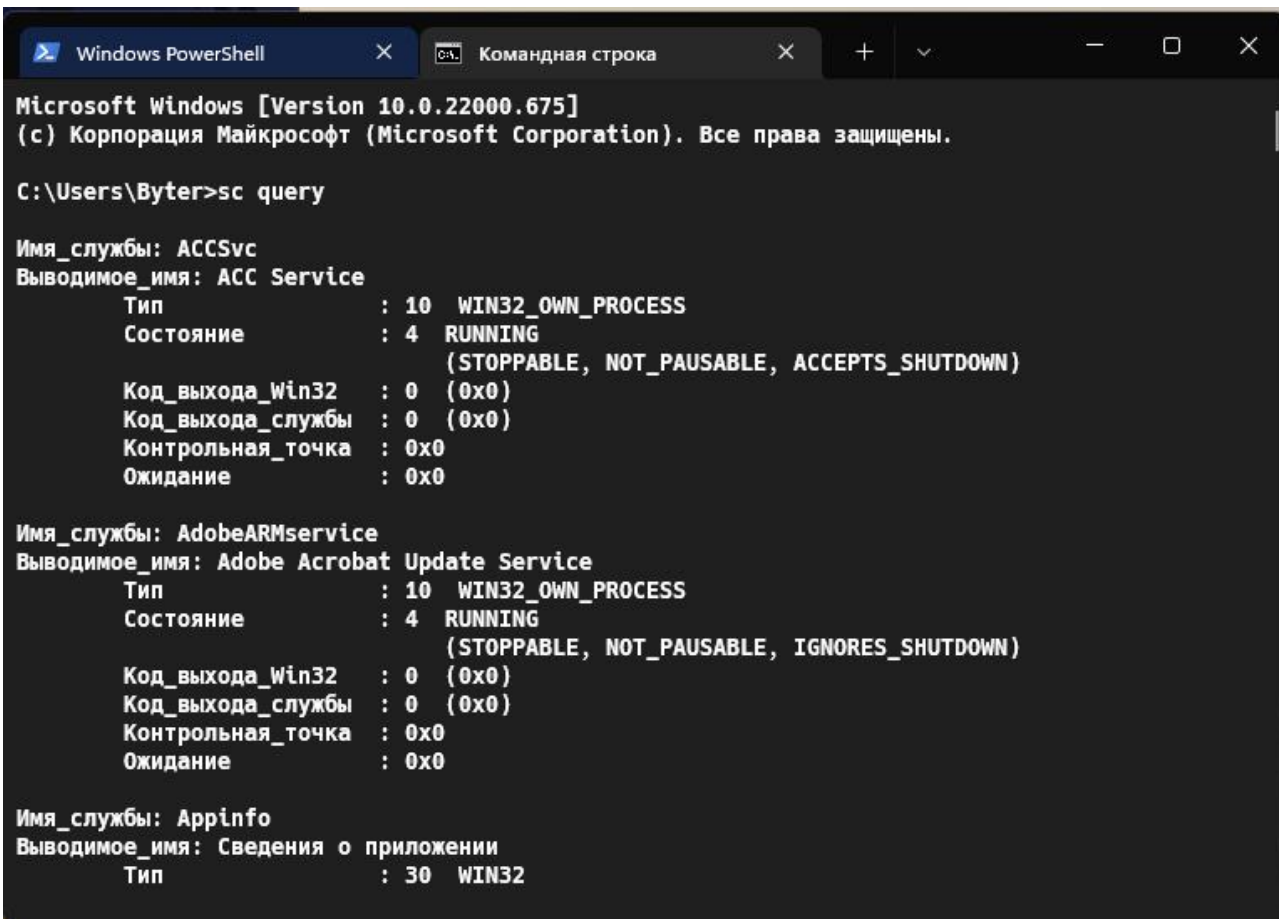
Рисунок 5 — Доступные свойства службы Windows

В диалоговом окне «Свойства» перечислены все элементы управления и информация о службе доступные в консоли управления службами Windows. Для отображения необходимо щёлкнуть правой кнопкой мыши службу и выберите «Свойства».

На вкладке «Общие» отображается путь к исполняемому файлу службы и описание выполняемой работы. Для выбранной службы есть возможность настроить тип запуска: Автоматически отложенный запуск, Автоматически, Вручную, Отключено. На вкладке «Зависимости» перечислены все

службы, зависящие от выбранной службы, а также службы, от которых зависит текущая выбранная служба.

Для управления подготовленными службами также подходит утилита «SC» (Windows Service Configuration Tool) — простая утилита командной строки «CMD», с помощью которой можно взаимодействовать, управлять или опрашивать диспетчер управления службами в Windows, может запрашивать подробную информацию о состоянии службы, запускать, останавливать или настраивать службы. В отличие от графического интерфейса Диспетчера управления службами (services.msc) данная утилита является идеальным средством для использования в скриптах [16]. Использование утилиты SC демонстрируется на рисунке 6.



```
Microsoft Windows [Version 10.0.22000.675]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\Byter>sc query

Имя_службы: ACCSvc
Выводимое_имя: ACC Service
        Тип               : 10  WIN32_OWN_PROCESS
        Состояние          : 4   RUNNING
                           (STOPPABLE, NOT_PAUSABLE, ACCEPTS_SHUTDOWN)
        Код_выхода_Win32    : 0   (0x0)
        Код_выхода_службы  : 0   (0x0)
        Контрольная_точка  : 0x0
        Ожидание           : 0x0

Имя_службы: AdobeARMservice
Выводимое_имя: Adobe Acrobat Update Service
        Тип               : 10  WIN32_OWN_PROCESS
        Состояние          : 4   RUNNING
                           (STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
        Код_выхода_Win32    : 0   (0x0)
        Код_выхода_службы  : 0   (0x0)
        Контрольная_точка  : 0x0
        Ожидание           : 0x0

Имя_службы: Appinfo
Выводимое_имя: Сведения о приложении
        Тип               : 30  WIN32
```

Рисунок 6 — Использование команды «sc query» для отображения списка служб Windows

При помощи команды «sc query» можно получить полный перечень зарегистрированных служб в сеансе операционной системе, а для того чтобы выбрать конкретную службу по имени, которое записано в базе данных реестра, используется данная команда с дополнительным параметром: «sc query имя_службы» [17]. Если вы знаете имя определённой службы, вы можете запросить информацию только по ней. Так же можно запустить или остановить службу: «sc stop | start имя_службы». Для получения полного списка доступных опций, можно использовать команду: «sc ?».

Чтобы управлять службами Windows также используются специальные командлеты оболочки «PowerShell» (платформа для автоматизации задач, которое включает оболочку командной строки, скриптовый язык и платформу управления конфигурацией ОС). Получение полного перечня служб ОС при помощи средств «Powershell» показано на рисунке 7.

```

Status      Name
-----
Running     Spooler
            Диспетчер печати

Byter@LAPTOP-G33P7QG5
$ Get-Service -Name *

Status      Name
-----
Running     AarSvc_16d3d40c  Agent Activation Runtime_16d3d40c
Running     ACCSvc           ACC Service
Running     AdobeARMSvc      Adobe Acrobat Update Service
Stopped     AJRouter         Служба маршрутизатора AllJoyn
Stopped     ALG              Служба шлюза уровня приложения
Stopped     AppIDSvc         Удостоверение приложения
Running     Appinfo          Сведения о приложении
Stopped     AppReadiness     Готовность приложений
Running     AppXSvc          Служба развертывания AppX (AppXSVC)
Running     AudioEndpointBu... Средство построения конечных точек ...
Running     Audiosrv         Windows Audio
Stopped     autotimesvc      Время в сети мобильной связи
Stopped     AxInstSV         Установщик ActiveX (AxInstSV)
Running     BcastDVRUserSer... Пользовательская служба DVR для игр...
Stopped     BDESVC           Служба шифрования дисков BitLocker
Running     BFE              Служба базовой фильтрации
  
```

Рисунок 7 — Использование командлета «Get-Service» для отображения списка зарегистрированных служб Windows.

Существует восемь основных командлетов Service (упрощенная команда, используемая в среде PowerShell), предназначенных для широкого спектра задач обслуживания, например, просмотра состояния и управления службами Windows: команда «Get-Service -Name имя_службы» позволяет получить службы на локальном или удалённом компьютере, как запущенные, так и остановленные; команда «Start-Service имя_службы» — запустить службу; «Stop-Service имя_службы» — остановить службу (отсылает сообщение об остановке диспетчеру служб Windows); команда «Set-Service -Name имя_службы {список необходимых атрибутов}» позволяет изменить параметры локальной или удалённой службы, включая состояние, описание, отображаемое имя и режим запуска (можно использовать для запуска, остановки или приостановки службы); «Resume-Service»; «Restart-Service».

[18]

Для работы с описанными выше командами оболочки командной строки Powershell необходимо запустить консоль Windows PowerShell с административными привилегиями, после чего выполнить команду, вписав её имя и установив необходимые атрибуты.

PowerShell поставляется с командлетами для работы с инструментарием WMI с момента выпуска. Мы можем также получить ссылку на определенный объект службы и затем непосредственно вызвать метод. Класс Win32_Service WMI представляет службу в компьютерной системе, работающей на ОС Windows.

2 Разработка GUI скрипта для управления службами SCM

2.1 Постановка задачи

Задача курсового проекта состоит в проектировании и реализации скрипта на языке сценариев Powershell, ориентированного на использование в небольших локальных сетях, для автоматизации процессов ОС Windows — управление зарегистрированными в менеджере (SCM) службами. Так же необходимо определить, как будет реализован диалог с пользователем при помощи использования пользовательского графического интерфейса (GUI): отображение доступный зарегистрированных служб, их свойств, система элементов для управления состояниями служб.

При проектировании решения были одобрены несколько основных идей. Необходимо, чтобы у пользователя перед глазами постоянно отображались доступные службы, с которыми можно будет взаимодействовать. Выбрав определённый элемент из данного списка, пользователю будет показана конфигурация службы, которую он сможет изменить путём редактирования значения текстового поля (или выбора значения из выпадающего списка). Также для каждой выбранного объекта службы необходимо отображать список зависимостей, к которым он имеет доступ благодаря SCM.

По нажатию кнопки, отвечающей за обновление свойств службы, будет загружена новая установленная пользователем конфигурация выбранной службы. Также необходимо добавить кнопку, которая будет предоставлять пользователю возможность изменять состояние службы: останавливать и запускать. Концепт представления визуальной составляющей программного решения представлен на рисунке 8.

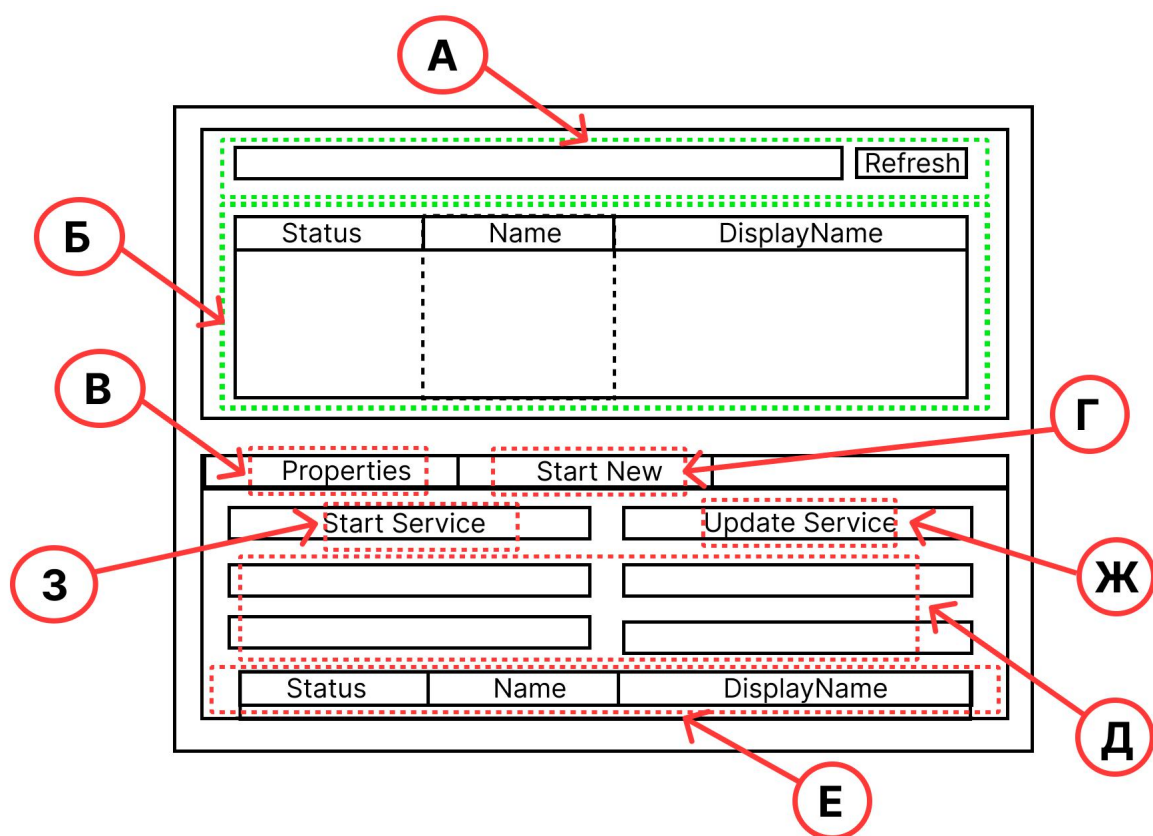


Рисунок 8 — Макет представления графического интерфейса

Как видно по макету, окно приложения поделено на две части: первая (верхняя) отвечает за поиск и выбор необходимой службы, а вторая (нижняя) отвечает за управление выбранной службой и создание новой. Элемент «А» служит средством поиска сервисов по их установленному имени, используя для этого текстовое поле, а также содержит кнопку сброса и очистки поиска. Элемент «Б» отвечает за отображение списка доступных установленных в системе служб, содержит три колонки для описания: текущего состояния, названия и отображаемого имени служб.

В нижней части приложения располагается панель элемента контроля вкладок. Внутри определены две страницы: «Элемент В» — вкладка, отвечающая за настройку конфигурации службы, «Элемент Г» — вкладка для создания и удаления служб.

«Элемент Ж» и «Элемент З» представляют собой кнопки для управления состоянием и обновления настроек свойств выбранной службы

соответственно. «Элемент Д», содержащий группу объектов управления (текстовые поля и выпадающие списки), используется для редактирования и отображения конфигурации службы.

Описание основных доступных свойств конфигурации служб Windows используемых при проектировании решения представлено в таблице 1.

Таблица 1 — Описание основных свойств экземпляра службы.

Имя свойства	Описание атрибута
Name	Уникальное имя службы зарегистрированной менеджером в базе данных
RequiredServices	Требуемые службы для работы, запрашиваемые от менеджера SCM
DisplayName	Отображаемое в диспетчере службы название службы
Status	Текущее состояние службы (Running, Stopped)
StartType	Тип запуска службы

Группа «Элементов Д-Ж» является дочерними (вложенными) объектами «Элемента В». Основные задачи, которые необходимо решить в рамках проектирования:

- Разработка удобного и понятного графического интерфейса для работы с пользователем;
- Отображение всех доступных сервисов установленных в базе данных ОС в виде таблицы: имя службы, видимое имя службы, статус (running, stopped);
- Предоставление пользователю возможности настраивать свойства выбранного из списка сервиса (отображаемое имя, тип запуска);

- Проектирование функции для создания и подключения новой службы SCM Windows;
- Предоставить пользователю возможность переключать состояние выбранной службы (включение, отключение по нажатию кнопки).

2.2 Обоснование средств разработки

При проектировании готового программного решения было принято решение использовать оболочку командной строки Windows Powershell (версии №5.1.22000), так как данная оболочка установлена по умолчанию в каждой ОС Windows, начиная с версии Windows 7 с пакетом обновления 1 (SP1) [19]. В качестве среды разработки для проектирования скриптовой части курсовой работы была взята интегрированная среда разработки сценариев Windows Powershell ISE — является ведущим приложением для Windows PowerShell, можно запускать команды, записывать и тестировать скрипты, а также выполнять их отладку в едином интерфейсе [20]. ISE поддерживает редактирование нескольких строк, заполнение нажатием клавиши TAB, раскраску синтаксических конструкций, выборочное выполнение и контекстную справку (с описанием всех предустановленных в оболочку командлетов). Элементы меню и сочетания клавиш подходят для выполнения большинства тех же задач, которые выполняются в консоли Windows PowerShell. Например, при отладке скрипта в ISE можно щёлкнуть строку кода на панели редактирования правой кнопкой мыши, чтобы задать точку останова. Область консоли отображает результаты выполненных команд и сценариев; команды можно выполнять в области консоли, также можно копировать и очищать её содержимое. Есть функция IntelliSense, которая позволяет просматривать перечень командлетов и их параметров, переменных, утилит и прочего. Поддерживаются сниппеты, есть возможность расширения набора функций за счёт различных аддонов. Процесс разработки скрипта Powershell показан на рисунке 9.

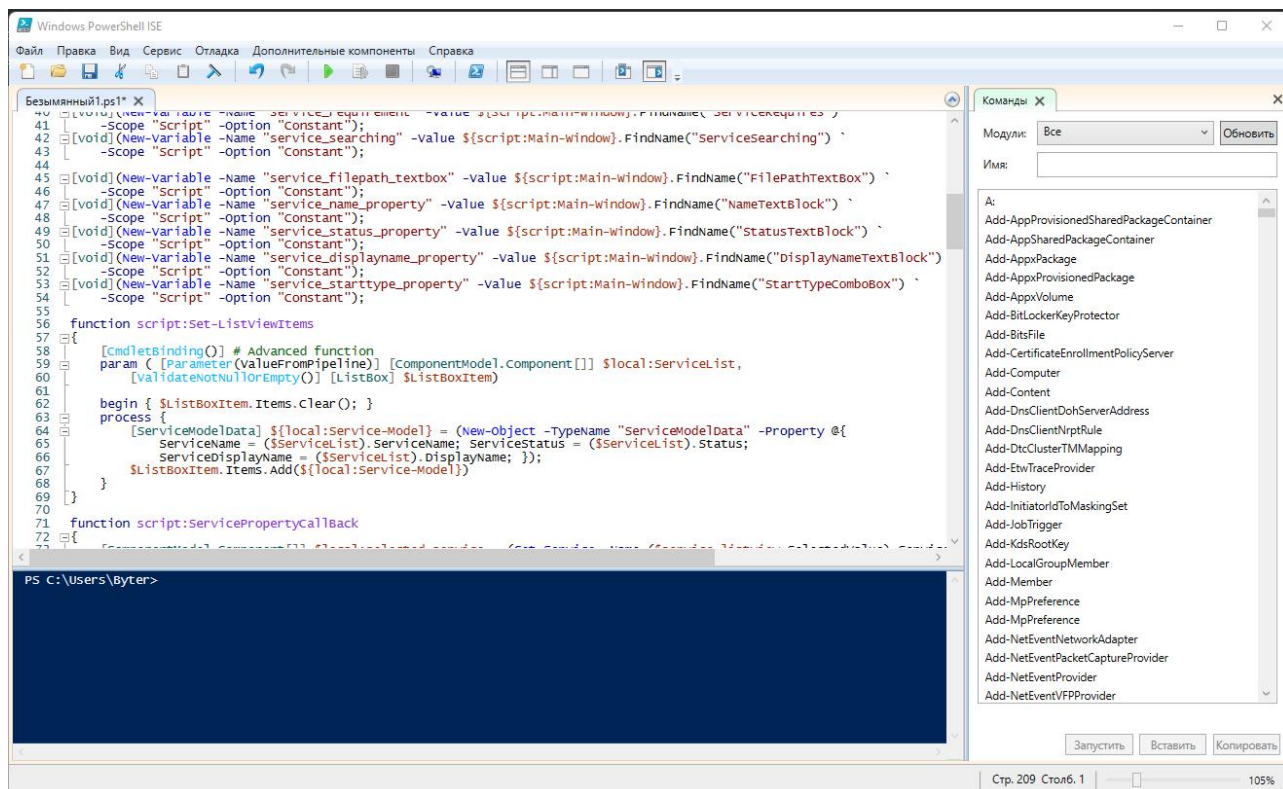


Рисунок 9 — Графическое представление скриптовой среды PS ISE

Для разработки графического оформления решения была выбрана интегрированные среды разработки приложений Microsoft Visual Studio 2022. Причина, по которой среда была выбрана в качестве основного инструмента для проектирования GUI, заключается в возможности создания проекта .NET WPF (платформа пользовательского интерфейса для создания клиентских приложений для настольных систем; платформа разработки WPF поддерживает широкий набор компонентов для разработки приложений) [21], функционал которого предоставляет возможность создания «окна WPF». Данная модель позволяет разместить компоненты управления (Control Elements) при помощи специального конструктора, который предоставляет визуальный интерфейс, помогающий проектировать приложения на основе XAML, такие как WPF и UWP, появляется возможность создавать пользовательские интерфейсы для приложений, перетаскивая элементы управления из окна «Панель элементов» и задавая свойства в окне «Свойства», или также можно

изменить код разметки XAML непосредственно в самом представлении. Демонстрация процесса разработки GUI на основе платформы WPF показана на рисунке 10.

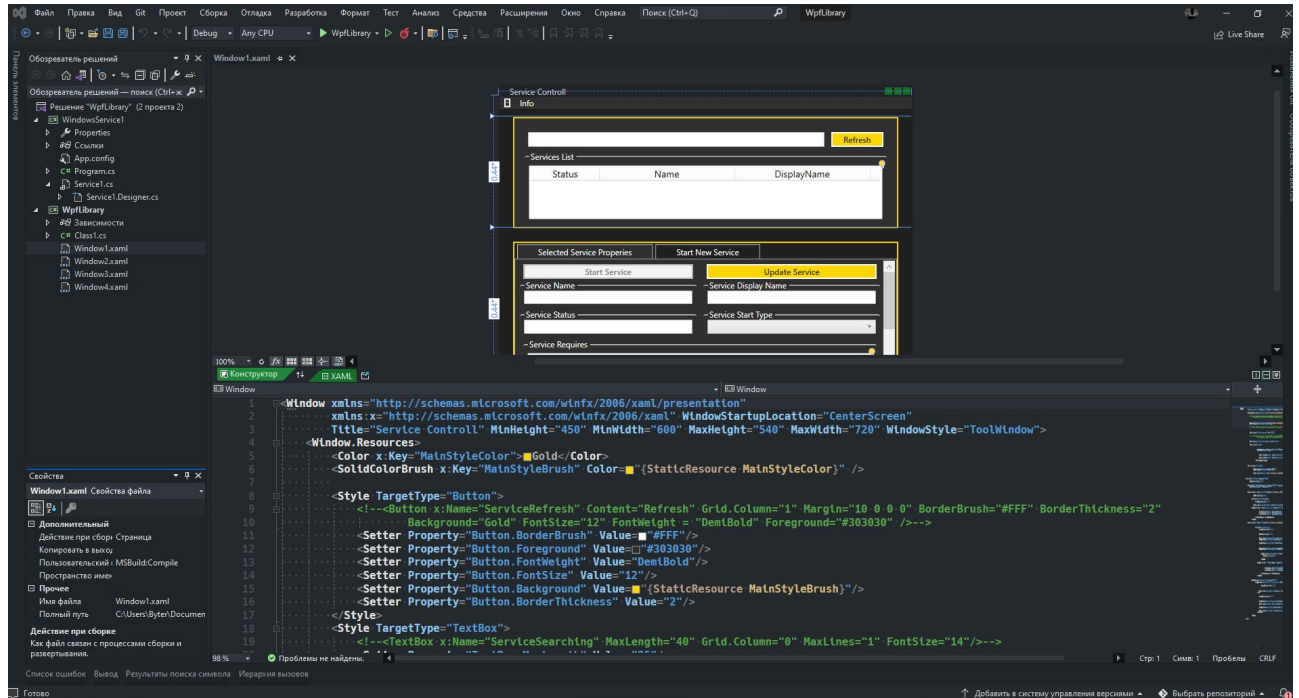


Рисунок 10 — Разработка пользовательского графического интерфейса при помощи конструктора «XAML».

XAML (Extensible Application Markup Language) является декларативным языком разметки, упрощает создание пользовательского интерфейса для приложения .NET Core; используемый для инициализации объектов в технологиях на платформе .NET [22]. Можно создать видимые элементы пользовательского интерфейса в декларативной XAML-разметке, а затем отделить определение пользовательского интерфейса от логики времени выполнения, используя файлы кода программной части, присоединенные к разметке с помощью определений разделяемых классов.

Язык XAML напрямую представляет создание экземпляров объектов в конкретном наборе резервных типов, определённых в сборках; предлагает очень простую и ясную схему определения различных элементов и их

свойств. Каждый элемент, как и любой элемент XML, должен иметь открытый и закрытый тег, либо элемент может иметь сокращённую форму с закрывающим слешем в конце.

В отличие от элементов XML каждый элемент в XAML соответствует определённому классу C# .NET из сборки «PresentationFramework.dll». Например, элемент Button соответствует классу «Controls.Button». А свойства этого класса соответствуют атрибутам элемента Button.

2.3 Описание работы системы

Главной задачей при работе со скриптовой частью работы является отображение графического интерфейса GUI. Для решения данной задачи используется функция «LoadXamlFormFile», которая принимает в качестве входного параметра строку абсолютного пути до файла с компоновкой элементов интерфейса (powershell-view.xaml) и возвращает собранный экземпляр класса «System.Windows.Window», отвечающий за суть окна подготовленного приложения.

Внутри функции используется встроенный в Powershell командлет «Get-Content» для чтения содержимого файла разметки. После чего данная строковый объект с кодом XAML используется в вызове конструктора класса «Xml.XmlNodeReader» для создания объекта, обеспечивающий средство чтения прямого доступа к данным XML формата. Далее путём вызова статического метода «Load» типа «Windows.Markup.XamlReader» и передачи в него предыдущего результата в качестве фактического параметра создаётся экземпляр окна (для этого используется кодогенерация, которая транслирует записи тегов объектов компоновки XAML в класс платформы .NET).

Данный объект типа «Window» инкапсулирует внутри себя объекты пользовательского управления, которые были добавлены в момент разметки в качестве декларативного элемента XAML. Все элементы управления наследуются от общего класса «System.Windows.Controls.Control» и имеют

ряд общих свойств. Класс «Window» предоставляет интерфейс «FindName» для доступа к всем установленным элементам управления, используя для этого значения атрибута «x:Name». Метод возвращает ссылку на сокрытый элемент коллекции, предоставляя полный доступ к его конфигурации и состоянию.

Чтобы интерпретатор PS мог определить объекты каких классов используются при выполнении сценария используется командлет «Add-Type», который в качестве аргумента принимает название встраиваемой сборки «DLL», в данном случае «PresentationFramework» (WPF).

При помощи данной функции внутри скрипта происходит обращение ко всем созданным элементам. Для объектов класса «Controls.Button» в теле сценария определены и реализованы уникальные функции-обработчики, которые отвечают за взаимодействие пользователя и окна приложения.

Для главного списка, отвечающего за отображение коллекции с записями о зарегистрированных службах, определён специальный обработчик событий, который определяет какой из элементов списка «Controls.ListView» (элемент управления отображает информацию на множестве строк и столбцов) [23] был выбран, после чего производит с полученными данными необходимые операции. Чтобы указать элементы для элемента управления «Controls.ListView» используется свойство «ListView.Items», которое возвращает коллекцию, используемую для создания содержимого «ItemsControl» (представляет элемента управления, который может быть использован для представления коллекции элементов).

Пользователь может выбрать из списка служб необходимый элемент, конфигурация которого отобразиться в нижней панели, в вкладке «Свойства выбранной службы». В этом разделе пользователь может просмотреть свойства службы и список служб, которые она требует для работы. Также раздел предоставляет возможность обновить список значений свойств службы или изменить её состояние. Чтение параметров службы продемонстрировано на рисунке 11.

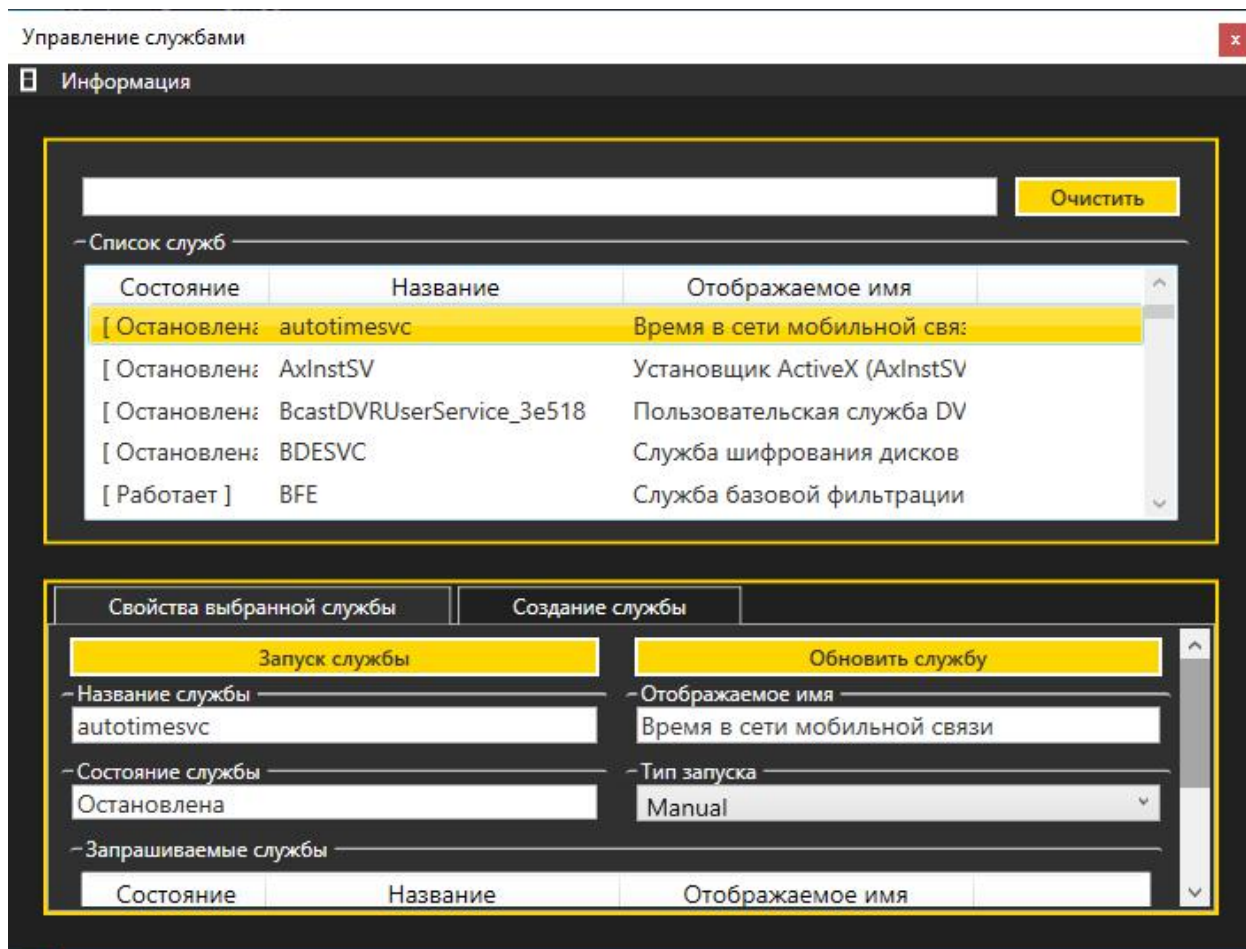


Рисунок 11 — Окно списка служб, их свойств с возможностью изменить конфигурацию

Чтобы отделить операцию заполнения элемента управления списками разработана расширенная функция «Set-ListViewItems» [24], которая в качестве параметров принимает указатель на элемент класса «Controls.ListView» (список) и добавляемый элемент службы. Она содержит реализацию механизма, позволяющий на основе объекта службы, создать элемент списка на основе класса «ServiceModelData», описывающий модель данных о службе. В основном используется в процессе конвейеризации после вызова командлета «Get-Service» (поставляет результат поиска информации о службе передаётся в функцию).

У пользователя решения есть возможность использования поисковой строки. Для этого ему необходимо установить в качестве значения

текстового поля часть названия (или полное название) искомой службы, после чего список служб обновиться в соответствии с результатом поиска.

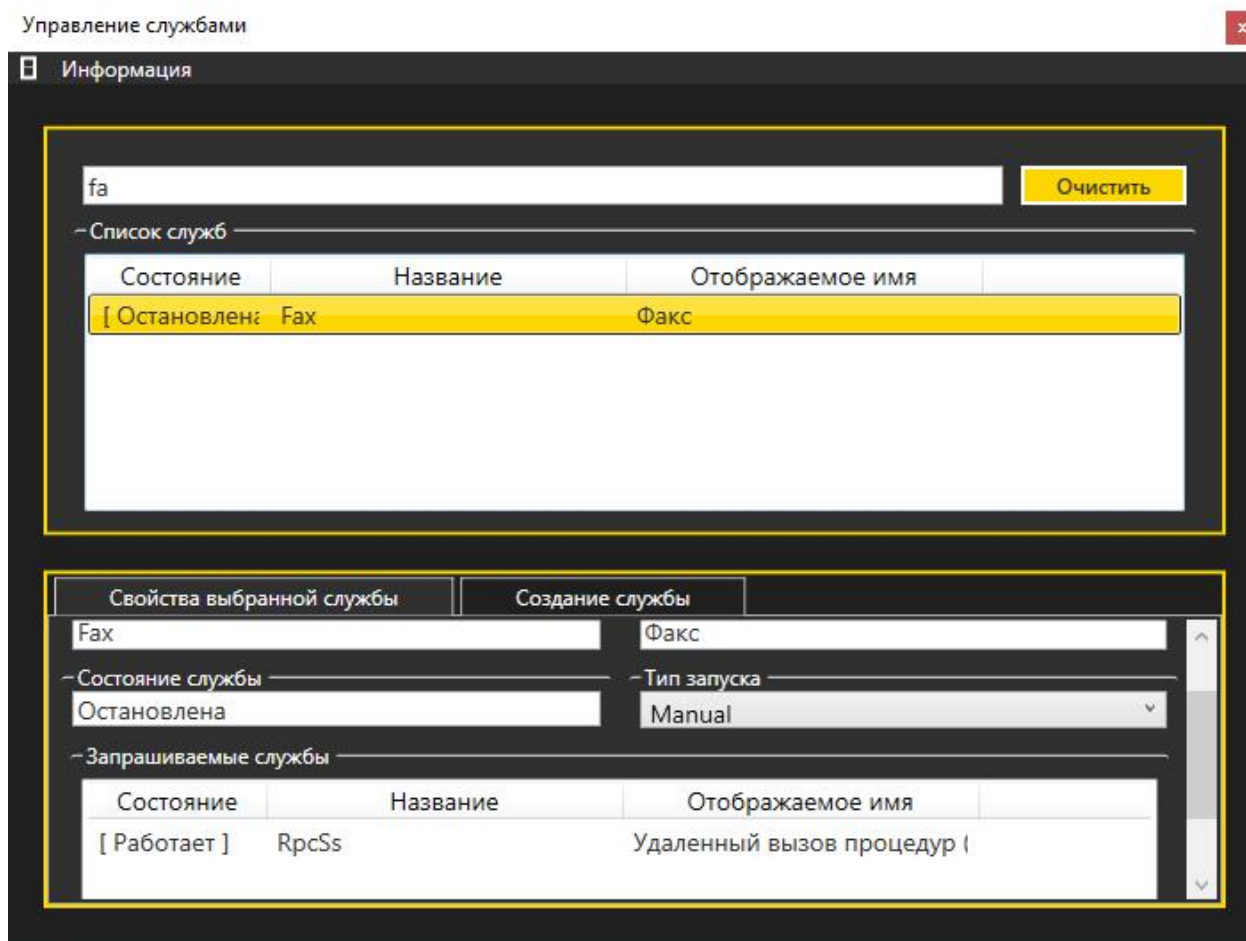


Рисунок 12 — Демонстрация процесса поиска необходимой службы

Для упрощения поиска необходимой службы используется объект текстового поля класса «Controls.TextBox», которое с помощью события «TextChanged» отслеживает изменение количества установленных внутри символов — при изменении текста срабатывает триггер (привязанная функция), который пытается на основе введённого значения строки найти службы или группы служб, используя для этого механизм регулярных выражений. Рядом расположена кнопка для сброса все установленных значений внутри формы.

После того как найден необходимый сервис Windows для пользователя откроется доступ для получения данных свойств конфигурации установленные для выбранной службы. Получение этих данных осуществляется путём вызова командлета «Get-Service -Name имя_службы», который собирает объект класса «ServiceController», который содержит свойства для описания службы: подключённые зависимости, отображаемое имя, состояние, тип запуска.

С помощью верхних кнопок «Start | Stop Service» и «Update Service» внутри вкладки, можно остановить или запустить остановленную службу или обновить конфигурацию службы, полагаясь на значения текстовых полей, расположенных ниже. Также вкладка демонстрирует список служб, которые требуются для работы текущей выбранной службы (от которых она зависит).

Вкладка «Start New Service» позволяет запустить службу на основе исполняемого файла (с расширением .exe), заполнив для этого поля с названием и отображаемым именем, а также удалить выбранную службу из списка.

Для регистрации новой службы используется командлет «New-Service», который в качестве аргумента «BinaryPathName», принимает полный путь до необходимого файла, «Name» — название службы, «DisplayName» — отображаемое в диспетчере служб имя. Чтобы пользователь не прописывал весь путь до файла в ручную, с помощью кнопки «Открыть файл» открывается файловое диалоговое окно, по результату которого формируется строка полного пути до исполняемого файла.

Чтобы зарегистрировать новую службу пользователь должен установить значения для полей: «Полный путь до файла», «Отображаемое имя» и «Название службы». Далее нажать на кнопку «Создание службы», после чего, в случае успеха, отобразиться сообщение «Служба была создана» (в противном случае создастся сообщение об ошибке). Процесса создания службы продемонстрирован на рисунке 13.

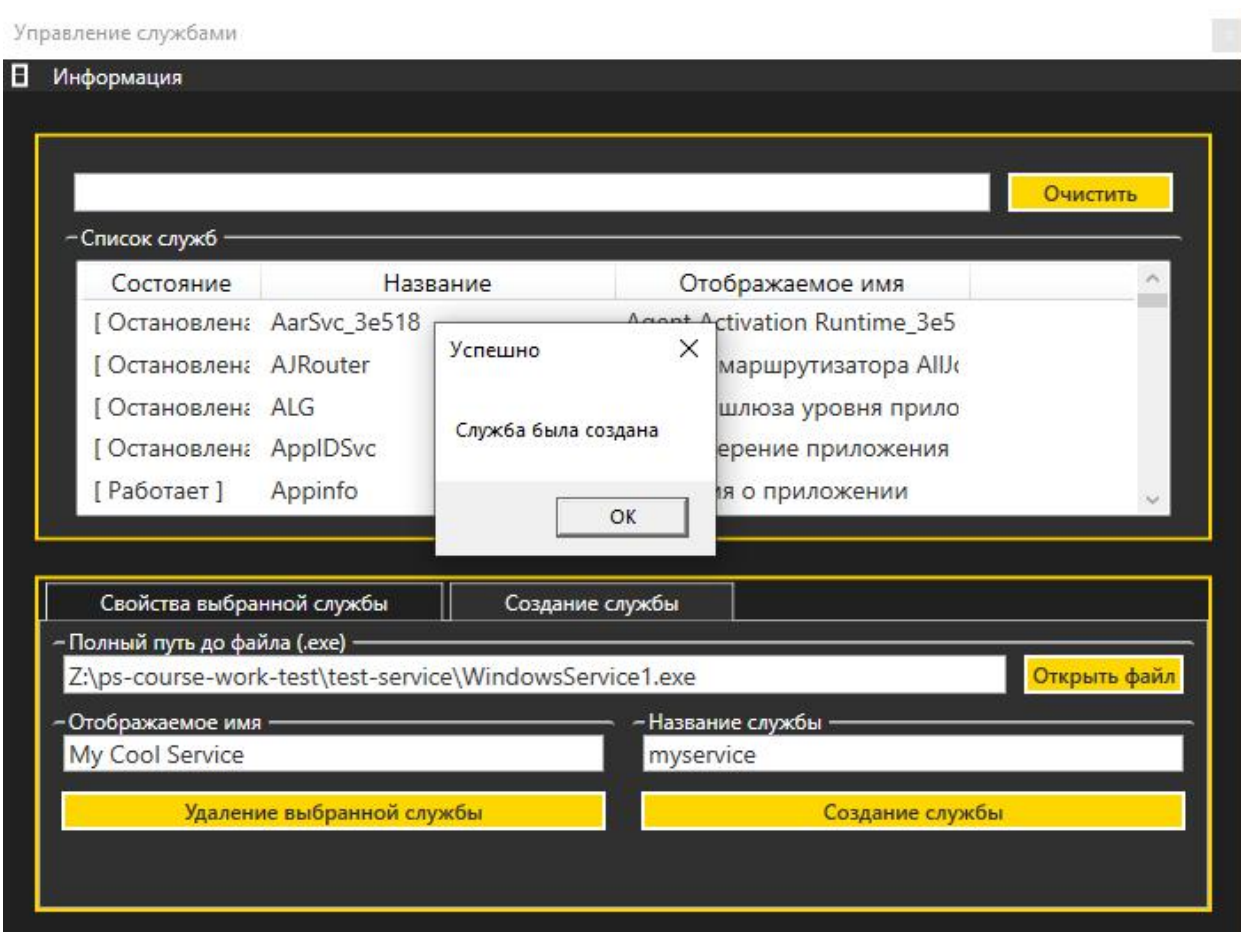
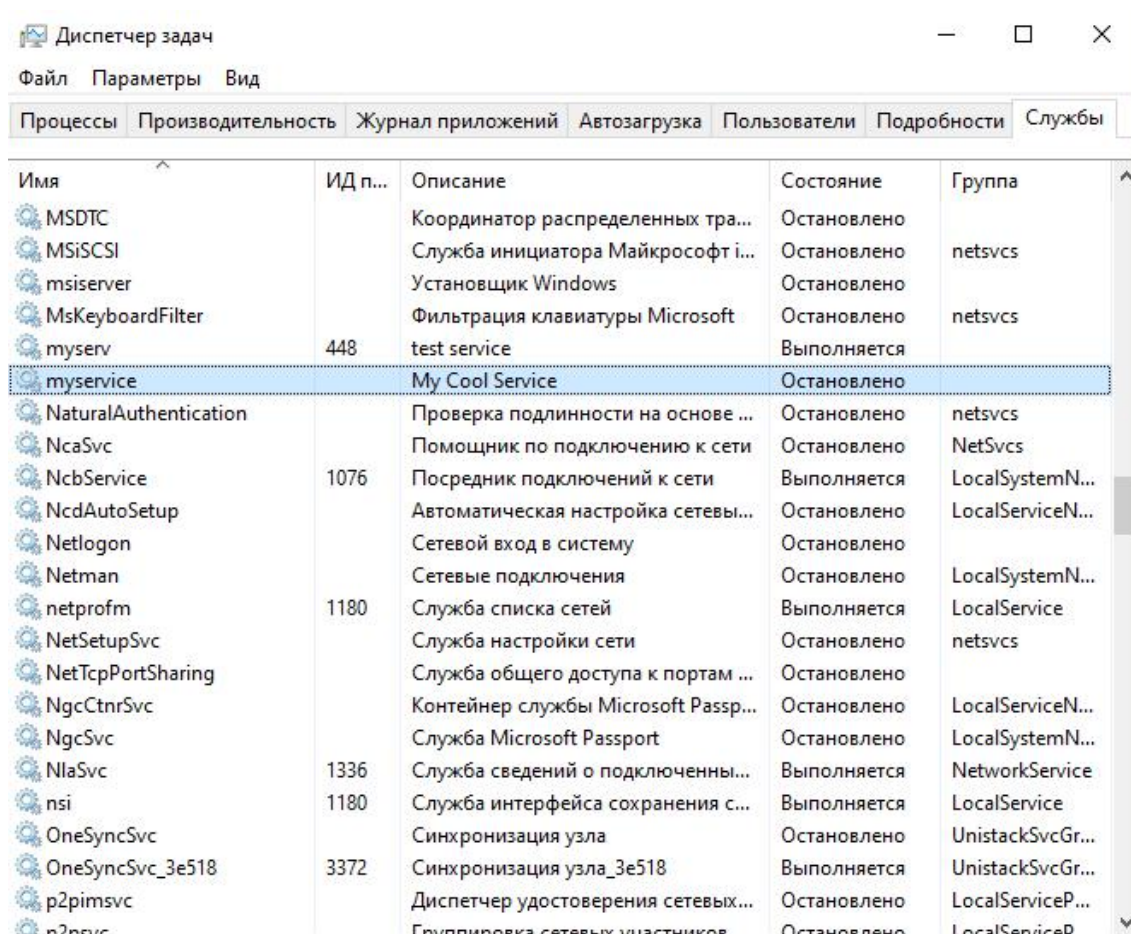


Рисунок 13 — Процесс создания новой службы Windows

Чтобы удалить службу используется пользователю необходимо выбрать службу из списка, после чего во вкладке «Создание службы» нажать кнопку «Удаление выбранной службы». Для этого используется вызов метода «Delete» у объекта класса «Win32-Service» [25] (выбранной службы), который возможно получить путём вызова к командлету «Get-WmiObject» (WMI-запрос). Данная команда получает экземпляры классов WMI (технологий для централизованного управления и слежения за работой различных частей компьютерной инфраструктуры под управлением платформы Windows) или сведения о доступных классах WMI, используя для поиска имя и тип объекта. Демонстрирование созданной службы в SCM показано на рисунке 14.



Имя	ИД п...	Описание	Состояние	Группа
MSDTC		Координатор распределенных тра...	Остановлено	
MSiSCSI		Служба инициатора Майкрософт i...	Остановлено	netsvcs
msiserver		Установщик Windows	Остановлено	
MsKeyboardFilter		Фильтрация клавиатуры Microsoft	Остановлено	netsvcs
myserv	448	test service	Выполняется	
myservice		My Cool Service	Остановлено	
NaturalAuthentication		Проверка подлинности на основе ...	Остановлено	netsvcs
NcaSvc		Помощник по подключению к сети	Остановлено	NetSvc
NcbService	1076	Посредник подключений к сети	Выполняется	LocalSystemN...
NcdAutoSetup		Автоматическая настройка сетевы...	Остановлено	LocalServiceN...
Netlogon		Сетевой вход в систему	Остановлено	
Netman		Сетевые подключения	Остановлено	LocalSystemN...
netprofm	1180	Служба списка сетей	Выполняется	LocalService
NetSetupSvc		Служба настройки сети	Остановлено	netsvcs
NetTcpPortSharing		Служба общего доступа к портам ...	Остановлено	
NgcCtnrSvc		Контейнер службы Microsoft Passp...	Остановлено	LocalServiceN...
NgcSvc		Служба Microsoft Passport	Остановлено	LocalSystemN...
NlaSvc	1336	Служба сведений о подключенны...	Выполняется	NetworkService
nsi	1180	Служба интерфейса сохранения с...	Выполняется	LocalService
OneSyncSvc		Синхронизация узла	Остановлено	UnistackSvcGr...
OneSyncSvc_3e518	3372	Синхронизация узла_3e518	Выполняется	UnistackSvcGr...
p2pimsvc		Диспетчер удостоверения сетевых...	Остановлено	LocalServiceP...
p2psvc		Группировка сетевых участников	Остановлено	LocalServiceP...

Рисунок 14 — Результат добавления новой службы в SCM

Внутри каждого блока обработки действий с элементами управлений используется конструкция «try-catch-finally» [26] для обработки исключений, которые могут быть сгенерированы в момент запроса к SCM для работы с службами. В случае возникновения ошибки создаётся диалоговое окно, путём вызова статического метода «Show» у класса «Windows.MessageBox» из сборки .NET, которое демонстрирует сообщение об исключении (принимает строку заголовка и строку тела окна).

ЗАКЛЮЧЕНИЕ

В ходе курсового проектирования автономного сценария на скриптовом языке PowerShell с использованием платформы WPF (XAML), был реализован скрипт для отображения данных о установленных менеджером SCM службах.

Разработанный скрипт предоставляет пользователю систему с удобным интерфейсом для просмотра списка доступных служб Windows, настройки конфигураций для отдельного выбранного экземпляра, изменение состояния путём старта и остановки выполнения службы, и запуск новой службы.

Организован контроль ошибок при их возникновении, печать сообщения пользователю на экран для информативного выявления исключений. Все требования к программе выполнены успешно, реализована поставленная задача.

В ходе разработки программного решения для выбранной темы были выполнены следующие задачи:

- Разработка удобного и понятного графического интерфейса для работы с пользователем;
- Отображение всех доступных сервисов установленных в реестре ОС в виде таблицы: имя службы, видимое имя службы, статус (running, stopped);
- Предоставление пользователю решения возможности настраивать свойства выбранного из списка сервиса (отображаемое имя, тип запуска);
- Разработана функции для создания и подключения новой службы Service Control Manager;
- Предоставление пользователю возможность переключать состояние выбранной службы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Знакомство с приложениями служб Microsoft Windows / [Электронный ресурс] — Режим доступа: <https://docs.microsoft.com/ru-ru/dotnet/framework/windows-services/introduction-to-windows-service-applications>;
2. Русинович М., Соломон Д., Алекс И. Внутреннее устройство Windows. 7-е изд. — "Издательский дом" Питер""", 2018. — С. 29-30;
3. Введение в приложения-службы Windows. Типы сервисов / [Электронный ресурс] — Режим доступа: <https://docs.microsoft.com/en-us/dotnet/framework/windows-services/introduction-to-windows-service-applications>;
4. Учебник. Создание приложения службы Windows / [Электронный ресурс] — Режим доступа: <https://docs.microsoft.com/ru-ru/dotnet/framework/windows-services/walkthrough-creating-a-windows-service-application-in-the-component-designer>;
5. Практическое руководство. Установка и удаление служб Windows. Установка с помощью программы InstallUtil.exe / [Электронный ресурс] — Режим доступа: <https://docs.microsoft.com/ru-ru/dotnet/framework/windows-services/how-to-install-and-uninstall-services>;
6. "Службы в операционной системе Windows". MSDN. Microsoft / [Электронный ресурс] — Режим доступа: [https://docs.microsoft.com/en-us/previous-versions/windows/hardware/design/dn642110\(v=vs.85\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/windows/hardware/design/dn642110(v=vs.85)?redirectedfrom=MSDN). — С. 3-4;
7. Удаленный вызов процедур. Модель RPC, принцип работы / [Электронный ресурс] — Режим доступа: <https://docs.microsoft.com/ru-ru/windows/win32/rpc/microsoft-rpc-model>;
8. Сведения о службах. Диспетчер служб Windows / [Электронный ресурс] — Режим доступа: <https://docs.microsoft.com/ru-ru/windows/win32/services/service-control-manager>;

9. Службы Windows. Автоматический запуск служб / [Электронный ресурс] — Режим доступа: <https://docs.microsoft.com/ru-ru/windows/win32/services/automatically-starting-services>;

10. Системные службы. Запуск служб по запросу / [Электронный ресурс] — Режим доступа: <https://docs.microsoft.com/ru-ru/windows/win32/services/starting-services-on-demand>;

11. Технологии для настольных систем. База данных установленных служб / [Электронный ресурс] — Режим доступа: <https://docs.microsoft.com/ru-ru/windows/win32/services/database-of-installed-services>;

12. Службы для отдельных пользователей в Windows 10 / [Электронный ресурс] — Режим доступа: <https://docs.microsoft.com/ru-ru/windows/application-management/per-user-services-in-windows>;

13. NET Framework. Разработка приложений служб Windows. Время существования службы / [Электронный ресурс] — Режим доступа: <https://docs.microsoft.com/ru-ru/dotnet/framework/windows-services/introduction-to-windows-service-applications?redirectedfrom=MSDN>;

14. API .NET. Режим запуска службы. Перечисление «ServiceStartMode». / [Электронный ресурс] — Режим доступа: <https://docs.microsoft.com/ru-ru/dotnet/api/system.serviceprocess.servicestartmode?view=dotnet-plat-ext-6.0>;

15. Разработка приложений служб Windows. Программная архитектура приложений служб / [Электронный ресурс] — Режим доступа: <https://docs.microsoft.com/ru-ru/dotnet/framework/windows-services/service-application-programming-architecture>;

16. Системные службы. Настройка службы с помощью SC / [Электронный ресурс] — Режим доступа: <https://docs.microsoft.com/ru-ru/windows/win32/services/configuring-a-service-using-sc>;

17. Управление службами с помощью Windows SC / [Электронный ресурс] — Режим доступа: <https://www.poftut.com/service-management-windows-sc-command-line/>;
18. Управление службами Windows с помощью PowerShell / [Электронный ресурс] — Режим доступа: <https://winitpro.ru/index.php/2019/09/05/upravlenie-sluzhbami-windows-powershell/>;
19. Установка Windows Powershell / [Электронный ресурс] — Режим доступа: [https://docs.microsoft.com/ru-ru/powershell/scripting/windows-powershell/install/installing-windows-powershell?view=powershell-7.2](https://docs.microsoft.com/ru-ru/powershell/scripting/windows-powershell/install/installing-windows-powershell?view=powershell-7.2;);
20. Коробко И. PowerShell как средство автоматического администрирования. — Litres, 2022;
21. Начало работы с WPF. Начало работы с WPF / [Электронный ресурс] — Режим доступа: [https://docs.microsoft.com/ru-ru/visualstudio/designers/getting-started-with-wpf?view=vs-2022](https://docs.microsoft.com/ru-ru/visualstudio/designers/getting-started-with-wpf?view=vs-2022;);
22. Обзор XAML. Объектные элементы XAML / [Электронный ресурс] — Режим доступа: [https://docs.microsoft.com/ru-ru/dotnet/desktop/wpf/xaml/?view=netdesktop-6.0](https://docs.microsoft.com/ru-ru/dotnet/desktop/wpf/xaml/?view=netdesktop-6.0;);
23. Введение в WPF. Элементы управления. ListView / [Электронный ресурс] — Режим доступа: <https://metanit.com/sharp/wpf/5.9.php>;
24. Станек, У. И. Windows PowerShell 2 : 0 / У. И. Станек. — Москва : Издательство «Русская редакция», 2010. — 416 с. — ISBN 978-5-7502-0396-3. — EDN SDQVSX. — С. 159-161;
25. Инструментарий управления Windows (WMI). Класс Win32_Service / [Электронный ресурс] — Режим доступа: <https://docs.microsoft.com/ru-ru/windows/win32/cimwin32prov/win32-service>;
26. Обработка ошибок в PowerShell / [Электронный ресурс] — Режим доступа: <https://windowsnotes.ru/powershell-2/obrabotka-oshibok-v-powershell-chast-2/>.

ПРИЛОЖЕНИЕ А

Листинг скрипта powershell-script.ps1

```
using namespace System;
using namespace System.ComponentModel;
using namespace System.Windows.Controls;
using namespace System.Windows;
using namespace Microsoft;

[void] (Add-Type -AssemblyName PresentationFramework);

class ServiceModelData : System.Object
{
    [ValidateNotNullOrEmpty()] [System.String] $script:ServiceStatus;
    [ValidateNotNullOrEmpty()] [System.String] $script:ServiceName;
    [ValidateNotNullOrEmpty()] [System.String] $script:ServiceDisplayName;

    static [ServiceModelData] CreateFromService([System.String] $service_name)
    {
        [ComponentModel.Component] $local:service_item = (Get-Service -Name
$service_name);
        return ([ServiceModelData] @ { ServiceDisplayName = ($service_item).DisplayName;
            ServiceStatus = ($service_item).Status; ServiceName =
($service_item).ServiceName;
        });
    }
}

function local:LoadXamlFormFile([string] ${local:File-Path})
{
    [Xml.XmlDocument] ${local:Xaml-File} = (Get-Content -Path ${local:File-Path});
    [Xml.XmlNodeReader] ${local:Xaml-Reader} = (New-Object System.Xml.XmlNodeReader
${Xaml-File});
    return ([Windows.Markup.XamlReader]::Load(${Xaml-Reader}));
}

Write-Host("$PSScriptRoot\powershell-view.xaml");

try { ${script:Main-Window} = local:LoadXamlFormFile("$PSScriptRoot\powershell-
view.xaml"); }
catch { Write-Host("Can't load Windows.Markup.XamlReader; ($PSItem.Exteption.Message)");
Exit; }

[void](New-Variable -Name "service_status_button" -Value ${script:Main-
Window}.FindName("ChangeStatusButton") `
    -Scope "Script" -Option "Constant");
[void](New-Variable -Name "service_listview" -Value ${script:Main-
Window}.FindName("ServiceList") `
```

```

        -Scope "Script" -Option "Constant");
[void](New-Variable -Name "service_requirement" -Value ${script:Main-
Window}.FindName("ServiceRequires") `
        -Scope "Script" -Option "Constant");
[void](New-Variable -Name "service_searching" -Value ${script:Main-
Window}.FindName("ServiceSearching") `
        -Scope "Script" -Option "Constant");

[void](New-Variable -Name "service_filepath_textbox" -Value ${script:Main-
Window}.FindName("FilePathTextBox") `
        -Scope "Script" -Option "Constant");
[void](New-Variable -Name "service_name_property" -Value ${script:Main-
Window}.FindName("NameTextBlock") `
        -Scope "Script" -Option "Constant");
[void](New-Variable -Name "service_status_property" -Value ${script:Main-
Window}.FindName("StatusTextBlock") `
        -Scope "Script" -Option "Constant");
[void](New-Variable -Name "service_displayname_property" -Value ${script:Main-
Window}.FindName("DisplayNameTextBlock") `
        -Scope "Script" -Option "Constant");
[void](New-Variable -Name "service_starttype_property" -Value ${script:Main-
Window}.FindName("StartTypeComboBox") `
        -Scope "Script" -Option "Constant");

```

```

function script:Set-ListViewItems
{
    [CmdletBinding()] # Advanced function
    param ( [Parameter(ValueFromPipeline)] [ComponentModel.Component[]]
$local:ServiceList,
           [ValidateNotNullOrEmpty()] [ListBox] $ListBoxItem)

```

```

    begin { $ListBoxItem.Items.Clear(); }
    process {
        [ServiceModelData] ${local:Service-Model} = (New-Object -TypeName
"ServiceModelData" -Property @{
            ServiceName = ($ServiceList).ServiceName; ServiceStatus =
($ServiceList).Status;
            ServiceDisplayName = ($ServiceList).DisplayName; });
        $ListBoxItem.Items.Add(${local:Service-Model})
    }
}

```

```

function script:ServicePropertyCallBack
{
    [ComponentModel.Component[]] $local:selected_service = (Get-Service -Name
($service_listview.Selected.Value).ServiceName);
    [void] ($script:service_status_button.IsEnabled = $global:true);

    switch($local:selected_service.Status)
    {
        "Running" { $script:service_status_button.Content = "Stop Service"; }
        "Stopped" { $script:service_status_button.Content = "Start Service"; }
    }
}

```

```

    }
    [void] ($script:service_name_property.Text = ($local:selected_service).ServiceName);
    [void] ($script:service_status_property.Text = ($local:selected_service).Status);

    [void] ($script:service_displayname_property.Text
= ($local:selected_service).DisplayName);
    [void] ($script:service_starttype_property.Text
= ($local:selected_service).StartType);

    if($local:selected_service.RequiredServices -eq $global:null)
{ $script:service_requirement.Items.Clear(); return [void]; }
    $local:selected_service.RequiredServices | ForEach-Object -Process {
        [void]((Get-Service -Name $PSItem.ServiceName) | Set-ListViewItems -ListBoxItem
$script:service_requirement) };
}

function script:WindowsItemsClear
{
    [void] ($script:service_status_button.IsEnabled = $global:false);
    [void] ($script:service_status_button.Content = "Start Service");

    [void] ($script:service_name_property.Text = $global:null);
    [void] ($script:service_status_property.Text = $global:null);

    [void] ($script:service_displayname_property.Text = $global:null);
    [void] ($script:service_starttype_property.Text = $global:null);

    $script:service_searching.Text = $global:null;
    $script:service_requirement.Items.Clear();
}

function script:ButtonRefreshCallback()
{
    [System.ComponentModel.Component[]] ${local:Service-List} = (Get-Service -Name "*");
    [void] (${local:Service-List} | Set-ListViewItems -ListBoxItem
$script:service_listview);
    [void] (script:WindowsItemsClear);
}

function script:ButtonUpdateCallback()
{
    [System.String] $local:service_display_name =
$script:service_displayname_property.Text;
    [System.String] $local:service_start_type = $script:service_starttype_property.Text;
    if(($local:service_display_name).Length -lt 5) { [MessageBox]::Show("Short
Name","Error"); return [void]; }

    try {
        [void] (Set-Service -Name
([ServiceModelData]$service_listview.SelectedItems[0]).ServiceName `
        -DisplayName $local:service_display_name -StartupType
$local:service_start_type);
    }
}

```

```

        [void] (($script:service_listview.SelectedItems[0]).ServiceDisplayName =
$local:service_display_name);
    }
    catch{ [MessageBox]::Show("Cannot set service properties","Error"); }

    [void] ($script:service_listview.Items.Refresh());
    [void] (script:ServicePropertyCallback);
}

```

```

function script:ButtonStatusCallback()
{
    [ServiceModelData] $local:selected_service = $service_listview.SelectedItems[0];
    try {
        switch((Get-Service -Name $local:selected_service.ServiceName).Status)
        {
            "Running" { [void] (Get-Service -Name $local:selected_service.ServiceName |
Stop-Service -Force -PassThru); }
            "Stopped" { [void] (Get-Service -Name $local:selected_service.ServiceName |
Start-Service -PassThru); }
        }
        [void] ($local:selected_service.ServiceStatus = (Get-Service -Name
$local:selected_service.ServiceName).Status);
    } catch { [MessageBox]::Show("Cannot change service status","Error"); }

    [void] ($script:service_listview.Items.Refresh());
    [void] (script:ServicePropertyCallback);
}

```

```

function script:ServiceSearchingCallback()
{
    for([int]$local:i = 0; $local:i -lt (($script:service_searching.Text).Length); $i++)
    {
        if(($script:service_searching.Text)[$i] -notlike "[A-Za-z0-1 ]") { return
[void]; }
    }
    [System.ComponentModel.Component[]]$local:Service-List = (Get-Service -Name
($script:service_searching.Text+ "*"));

```

```

    if($local:Service-List -eq $global:null) { return [void]; }
    [void]($local:Service-List | Set-ListViewItems -ListBoxItem
$script:service_listview);
}

```

```

function script:FilePathServiceCallback()
{
    [Win32.OpenFileDialog] $local:filepath_dialog = [Win32.OpenFileDialog]::new();
    [void] (($local:filepath_dialog).FileName = "Windows Service");
    [void] (($local:filepath_dialog).Filter = "Executable File (.exe)|*.exe");
    [void] (($local:filepath_dialog).DefaultExt = ".exe");

    if(($local:filepath_dialog).ShowDialog() -ne $global:true) { return [void]; }
    [void] (($script:service_filepath_textbox).Text = $local:filepath_dialog.FileName);
}

```

```

}

function script:NewServiceCallback()
{
    [System.String] $local:new_service_name = (${script:Main-
Window}.FindName("NewServiceName")).Text;
    [System.String] $local:new_service_displayname = (${script:Main-
Window}.FindName("NewServiceDisplayName")).Text;

    try {
        [Collections.ArrayList] $local:cmdler_error = $global:null;

        [void] (New-Service -Name $local:new_service_name -DisplayName
$local:new_service_displayname `
        -BinaryPathName (${script:service_filepath_textbox}).Text -ErrorVariable
cmdler_error `
        -ErrorAction SilentlyContinue);

        if($local:cmdler_error -ne $global:null) { throw [System.Exception]::new("Cmdlet
Error"); }
        [MessageBox]::Show("New Service was Created","Success");
    } catch { [MessageBox]::Show("Cannot create new service;
${$PSItem.Exception.Message)","Error"); }
}

function script>DeleteServiceCallback()
{
    try { [System.String] $local:selected_service =
($service_listview.Selected.Value).ServiceName;
        (Get-WmiObject Win32_Service -Filter
("name='$local:selected_service')).Delete();
        [MessageBox]::Show("Selected Service was Deleted","Success");
    } catch { [MessageBox]::Show("Cannot delete service;
${$PSItem.Exception.Message)","Error"); }
}

${script:Main-
Window}.FindName("ServiceRefreshButton").Add_Click($function:ButtonRefreshCallback)
${script:Main-
Window}.FindName("UpdateServiceButton").Add_Click($function:ButtonUpdateCallback);
$script:service_status_button.Add_Click($function:ButtonStatusCallback);

$script:service_listview.Add_MouseUp($function:ServicePropertyCallBack);
$script:service_searching.Add_TextChanged($function:ServiceSearchingCallback);

${script:Main-
Window}.FindName("NewServiceButton").Add_Click($function:NewServiceCallback);
${script:Main-
Window}.FindName("DeleteServiceButton").Add_Click($function>DeleteServiceCallback);
${script:Main-
Window}.FindName("OpenFileButton").Add_Click($function:FilePathServiceCallback);

```

```

${script:Main-Window}.FindName("MenuItemInfoButton").Add_Click({
    [MessageBox]::Show("Service Control Course-Work [BIST-214]", "Info");});
[void]($script:Main-Window).ShowDialog() | Out-Null);

```

Листинг разметки powershell-view.xaml

```

<Window xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
WindowStartupLocation="CenterScreen"
        Title="Service Controll" MinHeight="450" MinWidth="600" MaxHeight="540"
MaxWidth="720" WindowStyle="ToolWindow">
    <Window.Resources>
        <Color x:Key="MainStyleColor">Gold</Color>
        <SolidColorBrush x:Key="MainStyleBrush" Color="{StaticResource MainStyleColor}"
/>

        <Style TargetType="Button">
            <!--<Button x:Name="ServiceRefresh" Content="Refresh" Grid.Column="1"
Margin="10 0 0 0" BorderBrush="#FFF" BorderThickness="2"
                Background="Gold" FontSize="12" FontWeight = "DemiBold"
Foreground="#303030" />-->
            <Setter Property="Button.BorderBrush" Value="#FFF"/>
            <Setter Property="Button.Foreground" Value="#303030"/>
            <Setter Property="Button.FontWeight" Value="DemiBold"/>
            <Setter Property="Button.FontSize" Value="12"/>
            <Setter Property="Button.Background" Value="{StaticResource
MainStyleBrush}"/>
            <Setter Property="Button.BorderThickness" Value="2"/>
        </Style>
        <Style TargetType="TextBox">
            <!--<TextBox x:Name="ServiceSearching" MaxLength="40" Grid.Column="0"
MaxLines="1" FontSize="14"/>-->
            <Setter Property="TextBox.MaxLength" Value="25"/>
            <Setter Property="TextBox.MaxLines" Value="1"/>
            <Setter Property="TextBox.FontSize" Value="14"/>
            <Setter Property="TextBox.Background" Value="#FFF"/>
            <Setter Property="TextBox.Foreground" Value="#303030"/>
        </Style>
        <Style TargetType="ListView">
            <!--<ListView Name="ServiceList" Background="#FFF" Foreground="#303030"
FontSize="14"
                VerticalAlignment="Stretch" HorizontalAlignment="Stretch">-->
            <Setter Property="ListView.Background" Value="#FFF"/>
            <Setter Property="ListView.Foreground" Value="#303030"/>
            <Setter Property="ListView.FontSize" Value="14"/>
            <Setter Property="ListView.VerticalAlignment" Value="Stretch"/>
        </Style>
        <Style TargetType="TabItem">

```

```

        <Setter Property="Template">
            <Setter.Value>
                <ControlTemplate TargetType="TabItem">
                    <Border Name="Border" BorderThickness="1,1,1,0"
BorderBrush="Gainsboro" Margin="2,0">
                        <ContentPresenter x:Name="ContentSite"
VerticalAlignment="Center" HorizontalAlignment="Center"
                        ContentSource="Header" Margin="30 2"/>
                    </Border>
                    <ControlTemplate.Triggers>
                        <Trigger Property="IsSelected" Value="True">
                            <Setter TargetName="Border" Property="Background"
Value="#303030" />
                        </Trigger>
                        <Trigger Property="IsSelected" Value="False">
                            <Setter TargetName="Border" Property="Background"
Value="#202020" />
                        </Trigger>
                    </ControlTemplate.Triggers>
                </ControlTemplate>
            </Setter.Value>
        </Setter>
    </Style>
    <Style TargetType="ListViewItem">
        <Style.Triggers>
            <Trigger Property="IsMouseOver" Value="True">
                <Setter Property="Background" Value="#22DDDDDD" />
                <Setter Property="BorderBrush" Value="#202020" />
            </Trigger>
            <Trigger Property="IsSelected" Value="True">
                <Setter Property="Background" Value="{StaticResource
MainStyleBrush}" />
            </Trigger>
        </Style.Triggers>
    </Style>
</Window.Resources>
<Grid x:Name="MainGridNode" Background="#202020">
    <Grid.RowDefinitions>
        <RowDefinition Height="0.08*" />
        <RowDefinition Height="0.44*" />
        <RowDefinition Height="0.44*" />
    </Grid.RowDefinitions>
    <Menu Grid.Row="0" VerticalAlignment="Top" Background="#303030"
Foreground="#FFF">
        <Border BorderThickness="2" BorderBrush="#FFF">
            <Separator Width="5" Height="5"/>
        </Border>
        <MenuItem x:Name="MenuItemInfoButton" Header="Info"
HorizontalAlignment="Left" />
    </Menu>

```



```

        <Border Grid.Row="1" Margin="20 0 20 0" BorderBrush="{StaticResource
MainStyleBrush}" BorderThickness="2" Background="#303030">
            <Grid x:Name="ServiceStackList">
                <Grid.RowDefinitions>
                    <RowDefinition Height="Auto"/>
                    <RowDefinition Height="*/>
                </Grid.RowDefinitions>
                <Grid Margin="20 20 20 5" Grid.Row="0">
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="*/>
                        <ColumnDefinition Width="0.2*/>
                    </Grid.ColumnDefinitions>
                    <TextBox x:Name="ServiceSearching" MaxLength="40" Grid.Column="0"/>
                    <Button x:Name="ServiceRefreshButton" Content="Refresh"
Grid.Column="1" Margin="10 0 0 0"/>
                </Grid>
                <GroupBox Grid.Row="2" Header="Services List" Foreground="#FFF"
Padding="0 5 0 0" BorderThickness="0 0.5 0 0" Margin="15 0 15 5">
                    <ListView x:Name="ServiceList" HorizontalAlignment="Stretch"
SelectionMode="Single">
                        <ListView.View>
                            <GridView>
                                <GridViewColumn Width="100"
DisplayMemberBinding="{Binding Path=ServiceStatus}">Status</GridViewColumn>
                                <GridViewColumn Width="200"
DisplayMemberBinding="{Binding Path=ServiceName}">Name</GridViewColumn>
                                <GridViewColumn Width="200"
DisplayMemberBinding="{Binding Path=ServiceDisplayName}">DisplayName</GridViewColumn>
                            </GridView>
                        </ListView.View>
                    <ListView.Resources></ListView.Resources>
                </ListView>
            </GroupBox>
        </Grid>
    </Border>
    <Border Grid.Row="2" Margin="20" BorderBrush="{StaticResource MainStyleBrush}"
BorderThickness="2">
        <TabControl Background="#303030">
            <TabItem Header="Selected Service Properties" Foreground="#FFF">
                <ScrollViewer>
                    <Grid x:Name="ServiceProperties">
                        <Grid.ColumnDefinitions>
                            <ColumnDefinition Width="0.5*/>
                            <ColumnDefinition Width="0.5*/>
                        </Grid.ColumnDefinitions>
                        <Grid.RowDefinitions>
                            <RowDefinition Height="0.5*/>
                            <RowDefinition Height="0.5*/>
                        </Grid.RowDefinitions>
                        <StackPanel Grid.Row="0" Grid.Column="0" Margin="5 0">

```

```

        <Button x:Name="ChangeStatusButton" Content="Start
Service" IsEnabled="False" Margin="5 5 5 0"/>
        <GroupBox Header="Service Name" BorderThickness="0 0.5 0
0">
            <TextBox x:Name="NameTextBlock" IsReadOnly="True"/>
        </GroupBox>
        <GroupBox Header="Service Status" BorderThickness="0 0.5
0 0">
            <TextBox x:Name="StatusTextBlock"
IsReadOnly="True"/>
        </GroupBox>
    </StackPanel>
    <StackPanel Grid.Row="0" Grid.Column="1" Margin="5 0">
        <Button x:Name="UpdateServiceButton" Content="Update
Service" Margin="5 5 5 0"/>
        <GroupBox Header="Service Display Name"
BorderThickness="0 0.5 0 0">
            <TextBox x:Name="DisplayNameTextBlock"/>
        </GroupBox>
        <GroupBox Header="Service Start Type" BorderThickness="0
0.5 0 0">
            <ComboBox x:Name="StartTypeComboBox" FontSize="14"
VerticalAlignment="Top" IsReadOnly="True"
                Height="21" Background="#FFF">
                <TextBlock Text="Automatic"/>
                <TextBlock Text="Manual"/>
                <TextBlock Text="Disabled"/>
            </ComboBox>
        </GroupBox>
    </StackPanel>
    <GroupBox Grid.Row="1" Grid.ColumnSpan="2" Header="Service
Requires" Margin="10 0" BorderThickness="0 0.5 0 0"
        Height="150">
        <ListView x:Name="ServiceRequires" Margin="0 5 0 0">
            <ListView.View>
                <GridView>
                    <GridViewColumn Width="100"
DisplayMemberBinding="{Binding Path=ServiceStatus}">Status</GridViewColumn>
                    <GridViewColumn Width="200"
DisplayMemberBinding="{Binding Path=ServiceName}">Name</GridViewColumn>
                    <GridViewColumn Width="200"
DisplayMemberBinding="{Binding Path=ServiceDisplayName}">DisplayName</GridViewColumn>
                </GridView>
            </ListView.View>
            <ListView.Resources></ListView.Resources>
        </ListView>
    </GroupBox>
</Grid>
</ScrollView>
</TabItem>

```

```

        <TabItem Header="Start New Service" Foreground="#FFF" >
            <StackPanel Margin="5 0">
                <GroupBox Header="Select EXE File To Launch Service"
BorderThickness="0 0.5 0 0" Grid.ColumnSpan="2">
                    <Grid x:Name="StartService">
                        <Grid.ColumnDefinitions>
                            <ColumnDefinition Width="0.85*"/>
                            <ColumnDefinition Width="0.15*"/>
                        </Grid.ColumnDefinitions>

                        <TextBox x:Name="FilePathTextBox" Grid.Column="0"
Margin="0 0 5 0"/>

                        <Button x:Name="OpenFileButton" Grid.Column="1"
Margin="5 0 0 0" Content="Open File"/>
                    </Grid>
                </GroupBox>
                <Grid>
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="0.5*"/>
                        <ColumnDefinition Width="0.5*"/>
                    </Grid.ColumnDefinitions>
                    <StackPanel Grid.Column="0" Margin="0 0 5 0">
                        <GroupBox Header="Service Display Name"
BorderThickness="0 0.5 0 0">
                            <TextBox x:Name="NewServiceDisplayName"/>
                        </GroupBox>
                        <Button x:Name="DeleteServiceButton" Content="Delete
Selected Service" Margin="5 5 5 0"/>
                    </StackPanel>
                    <StackPanel Grid.Column="1" Margin="5 0 0 0">
                        <GroupBox Header="Service Name" BorderThickness="0 0.5 0
0">
                            <TextBox x:Name="NewServiceName"/>
                        </GroupBox>
                        <Button x:Name="NewServiceButton" Content="Run New
Service" Margin="5"/>
                    </StackPanel>
                </Grid>
            </StackPanel>
        </TabItem>
    </TabControl>
</Border>
</Grid>
</Window>

```