МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ФГБОУ ВО «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

Подпись, дата Инициалы, фамилия

Члены комиссии

Подпись, дата Инициалы, фамилия

Подпись, дата Инициалы, фамилия

Нормоконтролер

Подпись, дата Инициалы, фамилия

Защищена \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата

2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ФГБОУ ВО «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, расчеты и прочее):

анализ предметной области и требований к программному обеспечению

(7 страниц); моделирование и разработка системы на основе принципов

ООП (19 страниц); реализация системы на общесистемном языке программирования (11 страниц); 13 рисунков, 1 приложение

Сроки выполнения этапов анализ предметной области и требований к программному обеспечению (01.03 – 25.03); моделирование и разработка системы на основе принципов ООП (26.03 – 19.04); реализация системы на общесистемном языке программирования (20.04 – 11.05); описание диалога

с пользователем (12.05 – 21.05); оформление пояснительной записки

(22.05 – 04.06)

Срок защиты курсового проекта 05.06.2022

Руководитель

Подпись, дата Инициалы, фамилия

Задание принял студент Д. В. Тюленев

Подпись, дата Инициалы, фамилия

**Содержание**

[Введение 4](#_Toc31754)

[1. Теоретический обзор 5](#_Toc32028)

[1.1. Службы и Service Control Manager в операционной системе Windows 5](#_Toc17283)

[1.2. Способы управления службами со стороны пользователей 10](#_Toc16296)

[2. Практическая часть 18](#_Toc6304)

[2.1. Постановка задачи 18](#_Toc13103)

[2.2. Обоснование средств разработки 19](#_Toc2568)

[2.3. Описание работы системы 22](#_Toc3359)

[Заключение 26](#_Toc15205)

[Список использованной литературы 27](#_Toc24219)

[Приложение 28](#_Toc11475)

**Введение**

1. **Теоретический обзор**
   1. **Службы и Service Control Manager в операционной системе Windows**

Службы Microsoft Windows (Windows Services), ранее известные как службы NT, позволяют создавать долговременные исполняемые приложения, которые запускаются в собственных сеансах Windows; процесс, который запускается на устройстве в фоновом режиме для выполнения определённых действий в ответ на запросы пользователей, приложений или системы; имеют общие черты с концепцией [демонов](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BC%D0%BE%D0%BD_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0)" \o "Демон (программа)) в [Unix](https://ru.wikipedia.org/wiki/Unix" \o "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. 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 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. 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. 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.

Диспетчер управления служб (Service Control Manager, SCM) является основным средством управления службами в Windows; в [Microsoft Windows](https://ru.wikipedia.org/wiki/Microsoft_Windows" \o "Microsoft Windows) (Windows\System32\Services.exe) особый системный [процесс](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)" \o "Процесс (информатика)), реализующий технологию [удалённого вызова процедур](https://ru.wikipedia.org/wiki/%D0%A3%D0%B4%D0%B0%D0%BB%D1%91%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B2%D1%8B%D0%B7%D0%BE%D0%B2_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80" \o "Удалённый вызов процедур) ([remote procedure call](https://ru.wikipedia.org/wiki/Remote_procedure_call" \o "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](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B1%D1%8B_Windows" \o "Службы Windows) [ОС](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0" \o "Операционная система). Стартует при [загрузке](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%87%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B0_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%B0" \o "Начальная загрузка компьютера) [системы](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0" \o "Операционная система), обеспечивает работу [журнала событий](https://ru.wikipedia.org/wiki/%D0%96%D1%83%D1%80%D0%BD%D0%B0%D0%BB_%D1%81%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D0%B9" \o "Журнал событий), а также позволяет выполнять манипуляцию процессами удалённой машины.

При [загрузке](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%87%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B0_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%B0" \o "Начальная загрузка компьютера) [операционной системы](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0" \o "Операционная система) SCM запускает все [службы](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B1%D1%8B_Windows" \o "Службы Windows), у которых указан тип запуска «Автоматически», а также все [службы](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B1%D1%8B_Windows" \o "Службы Windows), указанные в зависимостях автозапускаемых [служб](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B1%D1%8B_Windows" \o "Службы Windows). 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

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](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85" \o "База данных) 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.

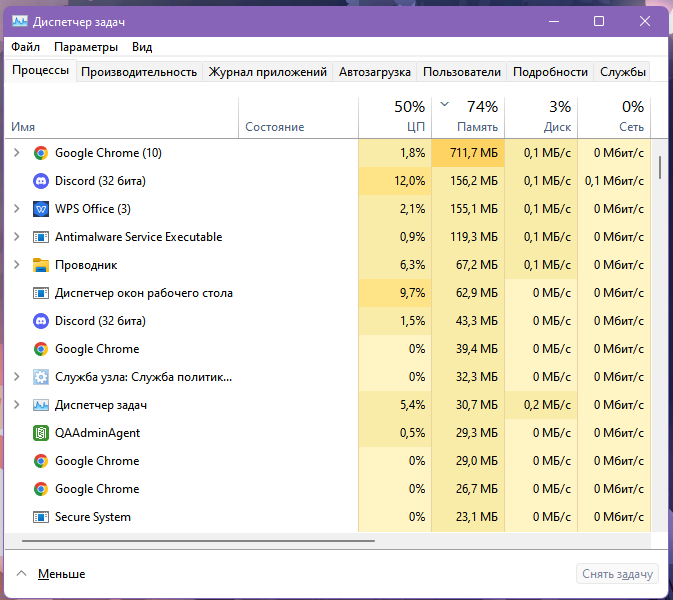


Рисунок 1 － Представление Менеджера служб (SCM).

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.

Функции менеджера представляют [интерфейс](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81" \o "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.

Загруженную службу необходимо запустить. Запущенная служба может выполнять свои задачи. Службу можно запустить из диспетчера управления службами, из Обозреватель сервера или из кода путём вызова метода. Метод «Start» передаёт обработку в метод «OnStart» приложения и обрабатывает любой код, определённый там.

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).

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. 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. **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.

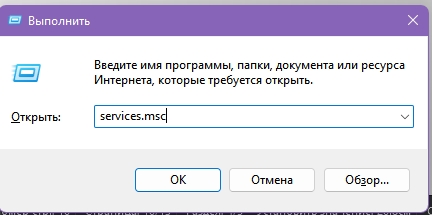


Рисунок 2 － Диалоговое окно «Выполнить» для запуска менеджера

**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.

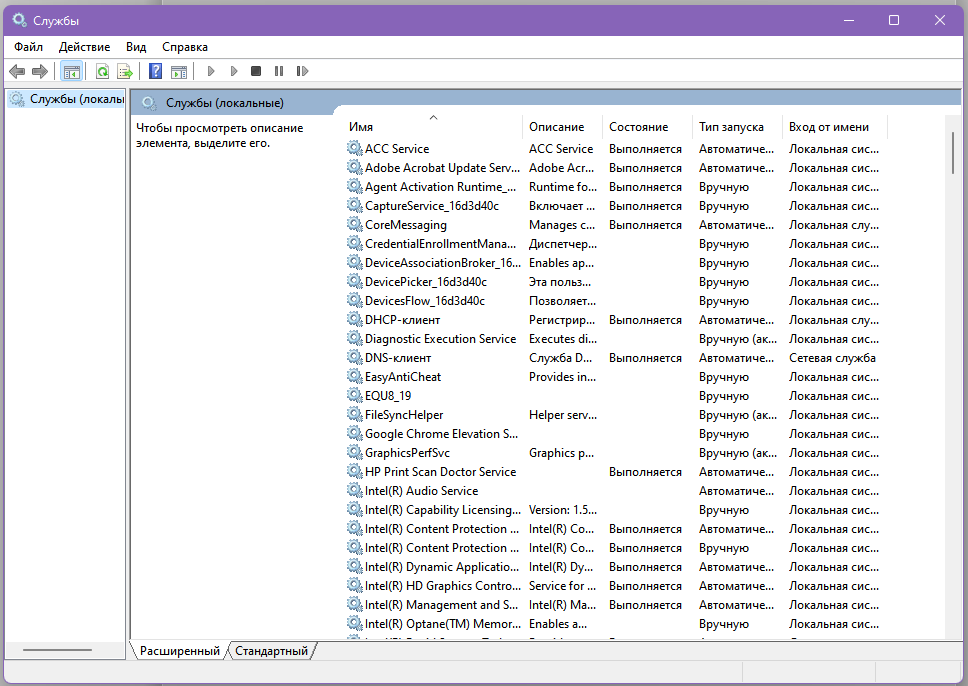


Рисунок 3 － Графический интерфейс окна Менеджера служб

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.

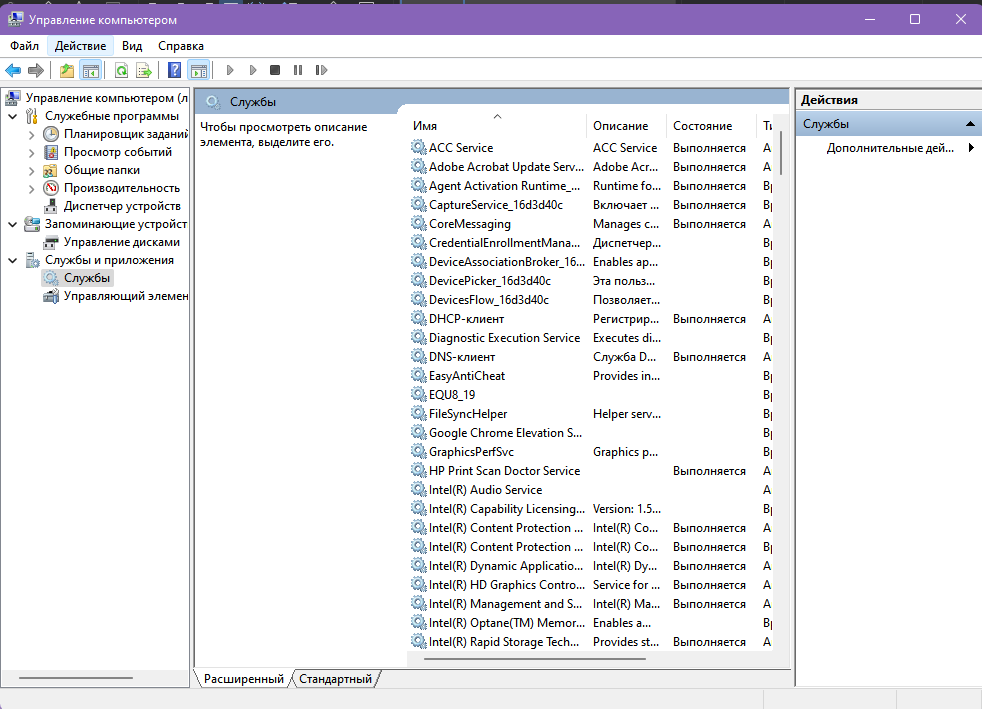


Рисунок 4 － Управление службами через «Управление компьютером»

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 «MMC»: Отроем окно «MMC», 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».

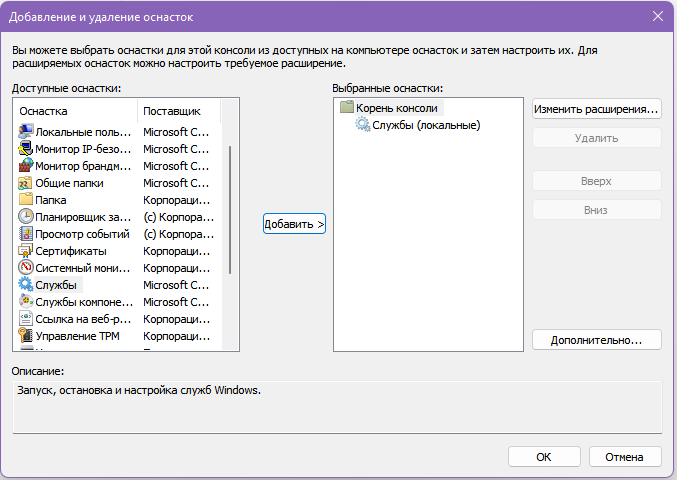
\

Рисунок 5 － 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.

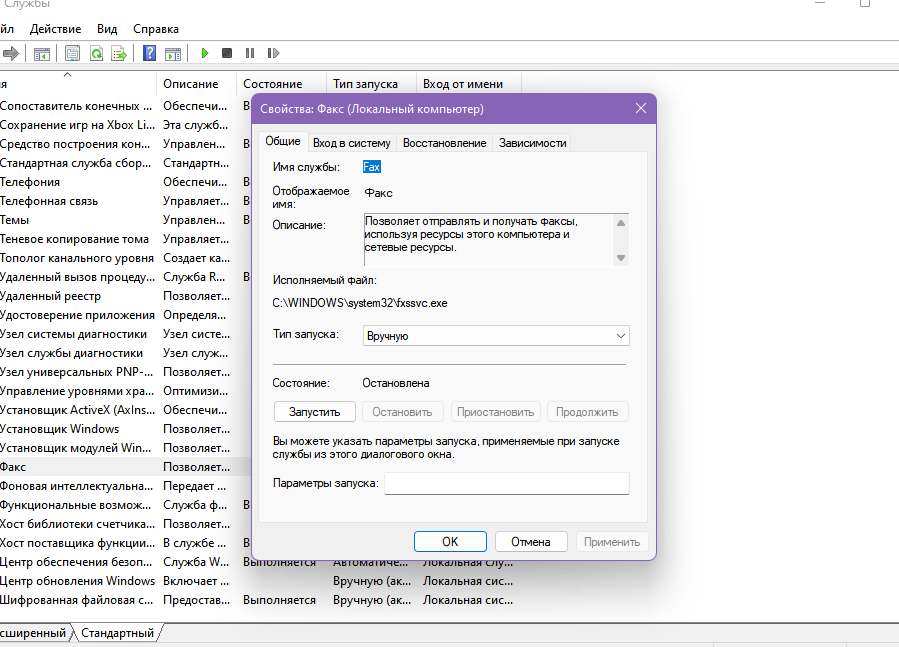


Рисунок 6 － Доступные свойства службы Windows

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.

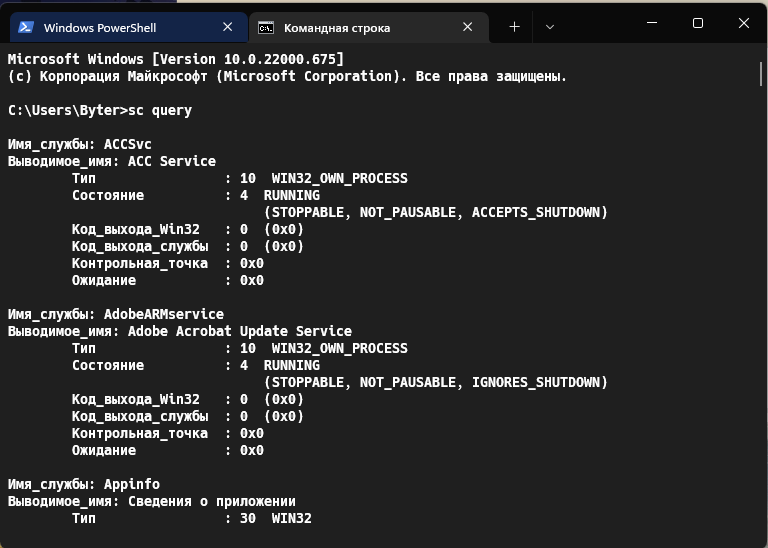


Рисунок 7 － Использование команды «sc query» для отображения списка служб Windows

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». 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).

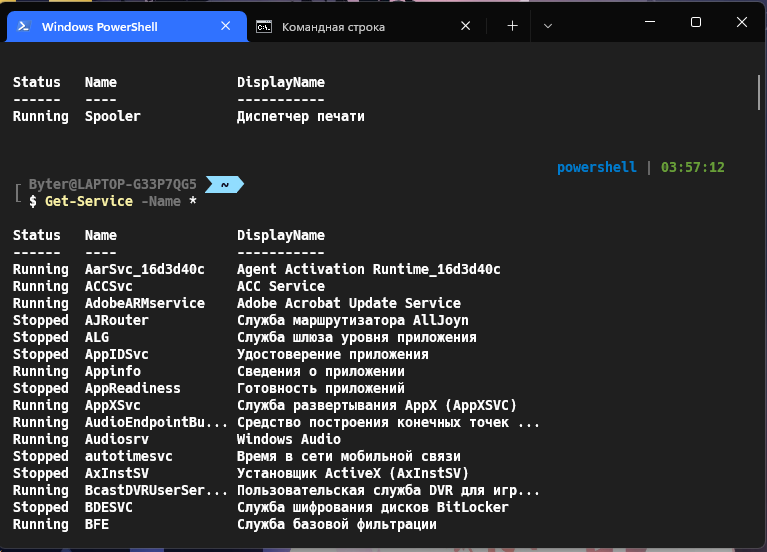


Рисунок 8 － Использование командлета «Get-Service» для отображения списка зарегистрированных служб Windows.

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».**

**Для работы с описанными выше командами оболочки командной строки Powershell необходимо** [запустить консоль Windows PoweShell](https://winnote.ru/main/285-kak-zapustit-windows-powershell.html) с административными привилегиями, после чего выполнить команду, вписав её имя и установив необходимые атрибуты.

1. **Практическая часть**
   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).

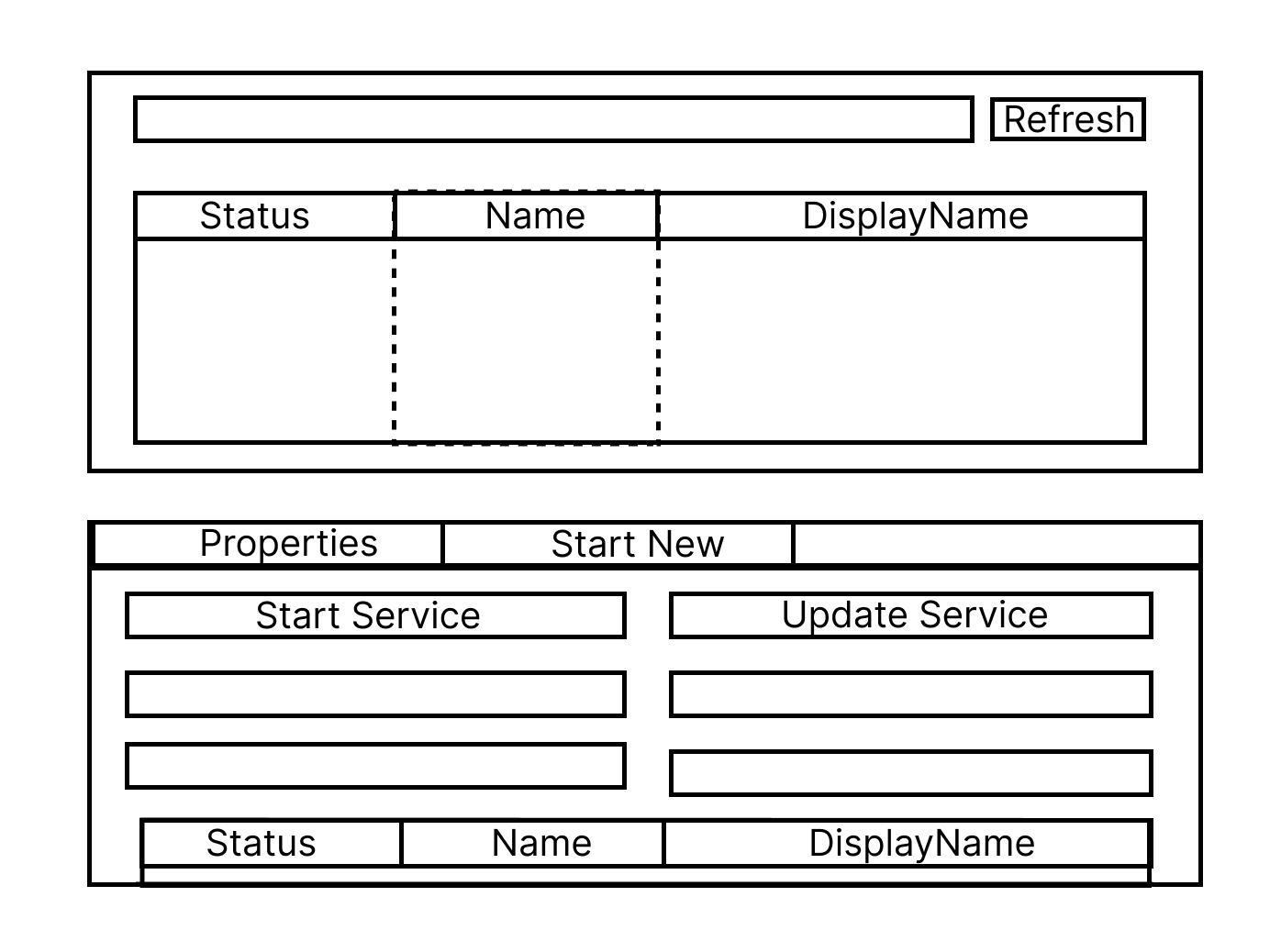


Рисунок 9 － Макет представления графического интерфейса

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 Service Control Manager.
* Предоставить пользователю возможность переключать состояние выбранной службы (включение, отключение по нажатии кнопки)

Таблица 1 － Описание основных атрибутов служб.

|  |  |
| --- | --- |
| Имя атрибута | 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 |

* 1. **Обоснование средств разработки**

При проектировании готового программного решения было принято решение использовать оболочку командной строки Windows Powershell (версии №5.1.22000). 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. ISE поддерживает редактирование нескольких строк, заполнение нажатием клавиши TAB, раскраску синтаксических конструкций, выборочное выполнение и контекстную справку (с описание 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.

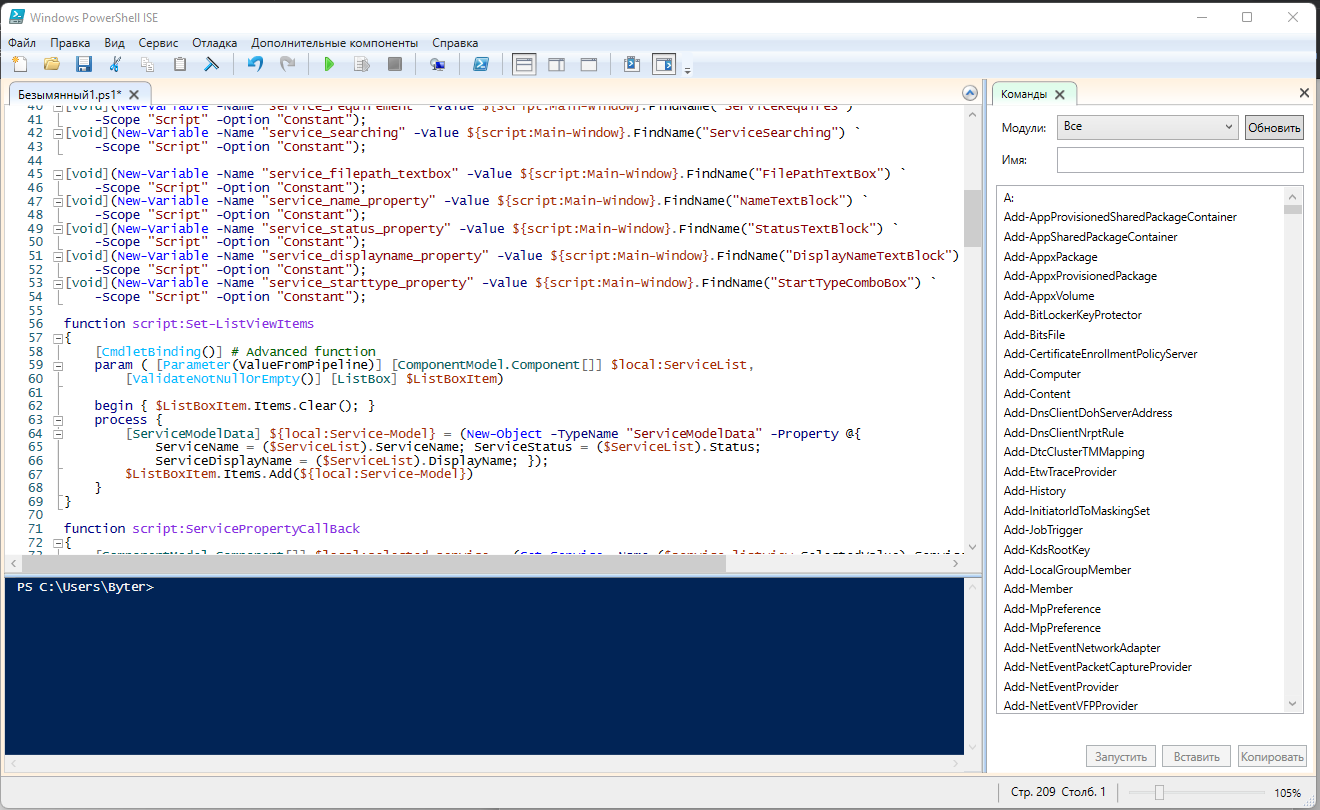


Рисунок 10 － Графическое представление скриптовой среды PS ISE

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), функционал которого предоставляет возможность создания «окна WPF». Данная модель позволяет размести 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.

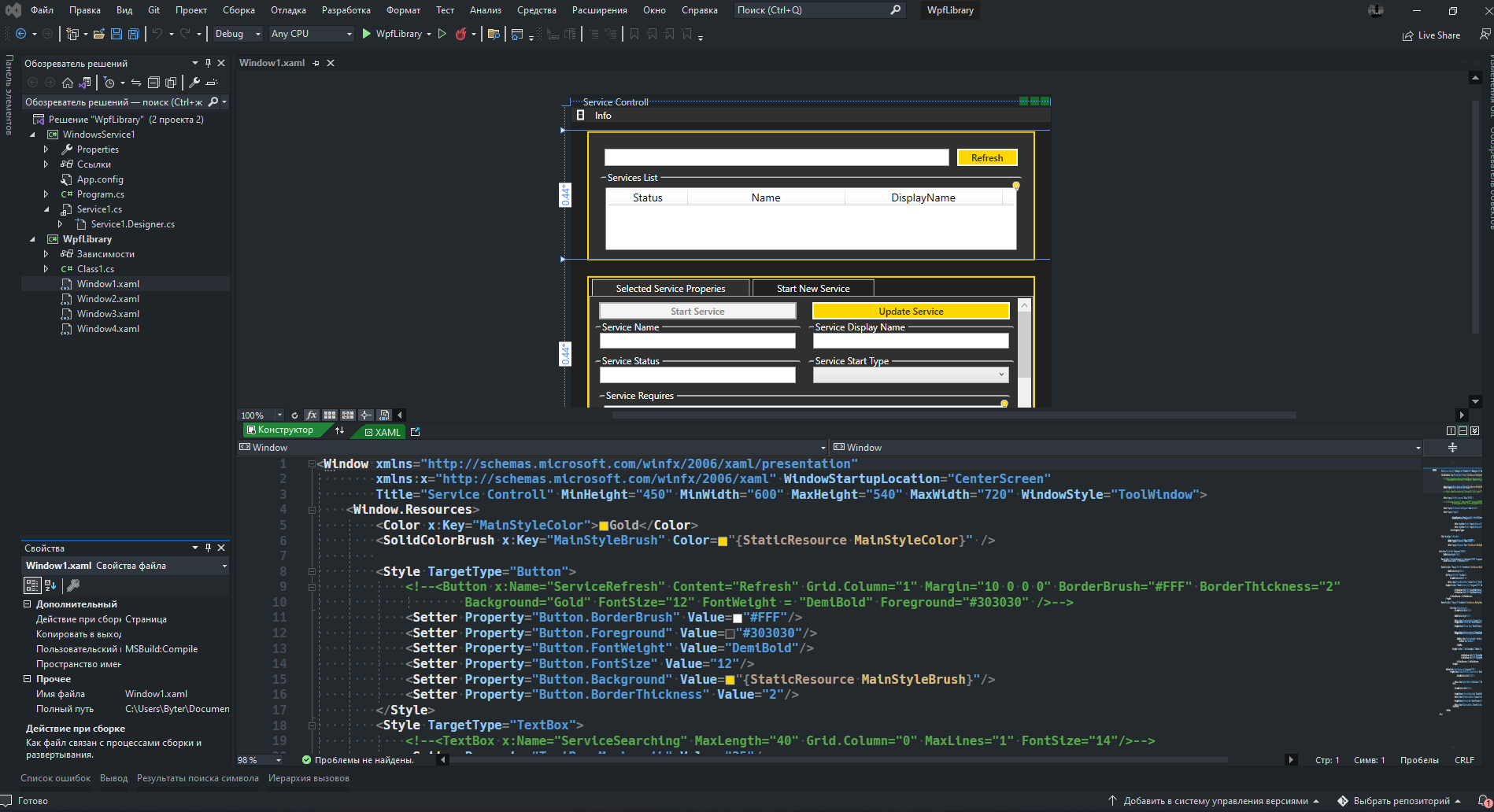


Рисунок 11 － Разработка пользовательского графического интерфейса при помощи конструктора «XAML».

XAML (Extensible Application Markup Language) является декларативным языком разметки, упрощает создание пользовательского интерфейса для приложения .NET Core. 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.

* 1. **Описание работы системы**

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

Внутри функции используется встроенный в Powershell командлет «Get-Content» для чтения содержимого файла разметки. После чего данная строка 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.Window.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 состоянии.

Чтобы интерпретатор PS мог определить объекты каких классов используются при выполнения 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 «ListView» был выбран, после чего производит с полученными данными необходимые операции.

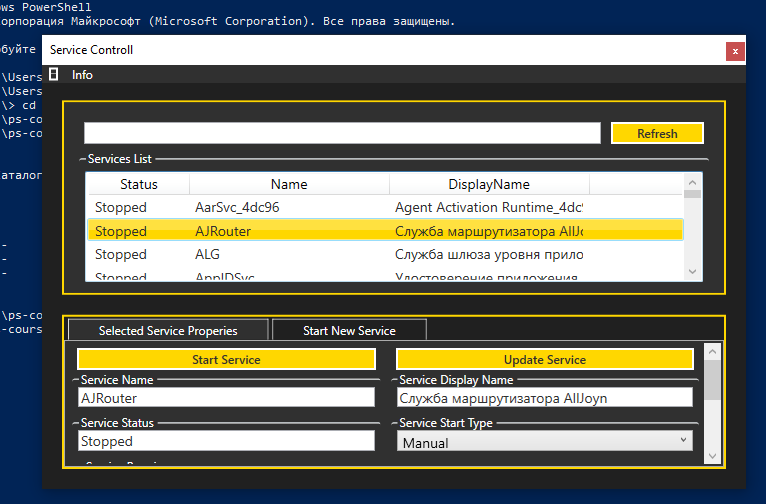


Рисунок 12 － Окно списка служб, их свойств с возможностью изменить конфигурацию

Для упрощения поиска необходимой службы используется текстовое поле «TextBox», которое отслеживает изменение количества установленных внутрь символов － при изменении текста срабатывает триггер, 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.

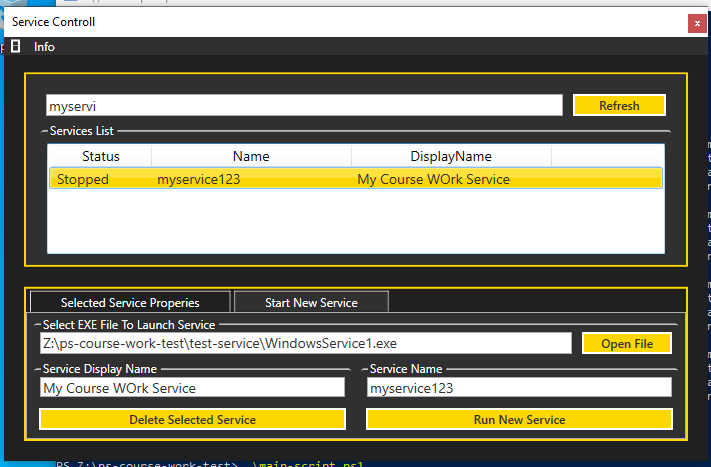


Рисунок 13 － Окно создания и удаления служб.

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, открывается файловое диалоговое окно.

Чтобы удалить службу используется вызов метода «Delete» у объекта «Win32-Service» (выбранной службы), который возможно получить путём обращения к командлету «Get-WmiObject». 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.

**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.

**Список использованной литературы**

**Приложение**

«Файл 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.SelectedValue).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] $local:new\_service\_diplayname = (${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\_diplayname `

        -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.SelectedValue).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);

«Файл 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 Properies" Foreground="#FFF">

                    <ScrollViewer>

                        <Grid x:Name="ServiceProperies">

                            <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>

                    </ScrollViewer>

                </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>