Case de Estudo e Portfólio:

Automação de Comunicação Sonora em Ambiente Hospitalar

Título

Sistema Automatizado de Execução Sonora em Ambiente Hospitalar: Otimização de Processos de Acreditação via PowerShell

Perfil do Desenvolvedor

Detalhe	Informação
Autor e Criador	Henrique
Formação Técnica	Desenvolvedor Back-End com foco em Análise de Dados
LinkedIn	https://www.linkedin.com/in/henriquesilvatech/
GitHub	https://github.com/86HenriqueSilva
E-mail	edson.henriquetech@outlook.com.br
Contato (Tel.)	081 9 8860-8666
Data de Criação	26/09/2025
Última Revisão	11/11/2025 (Refatoração para Portfólio)

1. Resumo Executivo (Foco na Solução e Impacto)

Este projeto, concebido e desenvolvido por **Henrique**, implementa um sistema robusto e auditável para a automação da comunicação sonora em áreas hospitalares. A solução utiliza a linguagem **PowerShell** e recursos nativos do ambiente Windows para eliminar a dependência de intervenção humana, garantindo o cumprimento de rotinas de agendamento complexas, balanceamento de informações e a **geração de evidências detalhadas (logs) cruciais para processos de qualidade e acreditação hospitalar**.

Destaque: Este sistema transforma um processo manual e falível em um processo autônomo, rastreável e em total conformidade, refletindo a aplicação prática da lógica de programação Back-End e gestão de dados (Logs/Auditoria) em um ambiente de missão crítica.

2. Introdução e Justificativa Técnica (O PORQUÊ)

A comunicação contínua e precisa é um pilar da segurança e da qualidade assistencial. Este desenvolvimento justifica-se pela necessidade de:

1. **Garantir a Oportunidade:** As mensagens críticas são veiculadas no momento exato, sem falhas, conforme o agendamento.
2. **Fornecer Evidência:** A geração de logs detalhados atende diretamente aos requisitos de auditoria e acreditação.
3. **Otimizar Recursos:** Libera o tempo da equipe para tarefas assistenciais diretas.

Dedicação e Processo de Desenvolvimento

O projeto exigiu dedicação e pesquisa aprofundada. Desde a primeira tentativa, foram investidas horas no estudo da melhor forma de gerenciar o *scheduling* e o estado do sistema (contagem diária, áudio em execução), garantindo que o *script* fosse resiliente e funcional. A complexidade do agendamento (rotinas fixas, alternância, balanceamento) reflete um rigoroso processo de **análise de requisitos** traduzido em código.

2.1 Justificativa da Escolha Tecnológica (PowerShell)

A escolha do PowerShell (em um ambiente Windows) foi estratégica e deliberada, em contraste com a ideia inicial de utilizar Python/Linux, pelos seguintes motivos técnicos e operacionais:

- **Nativo e Sem Dependências:** O PowerShell e a biblioteca .NET PresentationCore são nativos do ambiente Windows (plataforma padrão do servidor de áudio), eliminando a necessidade de instalar e manter *runtime environments* ou bibliotecas de terceiros (como Pygame ou Schedule).
- **Facilidade de Agendamento:** A execução contínua (while ($true)) e a persistência do script são gerenciadas de forma robusta e simples através do **Agendador de Tarefas (Scheduled Task)** do Windows, que é a ferramenta nativa de orquestração do ambiente.
- **Redução da Complexidade de Suporte:** A manutenção e a *troubleshooting* são simplificadas, pois toda a infraestrutura de execução (linguagem, *scheduler* e *media player*) faz parte do ecossistema padrão do sistema operacional.

3. Objetivos do Sistema

- **Implementar Agendamento:** Execução de áudios fixos e programados em horários exatos.
- **Garantir Balanceamento:** Distribuir áudios aleatórios de forma equilibrada ao longo do dia, respeitando um limite máximo.
- **Gerar Auditoria:** Registrar logs de todas as tentativas e execuções com carimbo de data/hora e usuário.

- **Executar Rotinas Complexas:** Suportar repetições e alternâncias sequenciais de múltiplos áudios.

4. Arquitetura e Requisitos Operacionais

O sistema é um script PowerShell de execução contínua.

4.1 Requisitos de Hardware e Software

- **Sistema Operacional:** Windows com suporte ao PowerShell 5.0 ou superior.
- **Biblioteca Crítica:** O script exige o .NET Framework para a utilização da classe System.Windows.Media.MediaPlayer através do Add-Type -AssemblyName presentationCore.
- **Execução:** O script deve ser executado através de uma **Tarefa Agendada (Scheduled Task)** configurada para rodar continuamente e reiniciar em caso de falha do processo.
- **Privilégios:** A conta de usuário que executa a Tarefa Agendada deve ter privilégios para criar diretórios e escrever arquivos na pasta C:\execucao de audio.

4.2 Estrutura de Arquivos e Armazenamento

Local	Variável PowerShell	Conteúdo
C:\execucao de audio	$pastaAudios	Contém todos os arquivos MP3 (Fixo, A e B). **(Ponto de Backup Crítico)**
C:\execucao de audio\logs	$logDir	Pasta criada automaticamente para armazenar logs diários.

4.3 Regras de Agendamento (Lógica do switch e if)

Tipo de Áudio	Nome Exemplo	Regras de Execução	Lógica de Programação
Fixo (Ouvidoria)	Ouvidoria Alfa.mp3	Horários fixos: 09:00, 10:00, 11:00, 13:00, 14:00, 15:00, 16:00. **Apenas de segunda a sexta.**	Controle via $horariosOuvidoria e $executadosHojeOuvidoria.
Programado (B1-B6)	B1_ENFER_01_06.mp3	Execução em horários exatos (ex: 11:00, 15:00, 15:30, 16:30). **Repetição de 3 vezes por execução.**	Utiliza a função Tocar-Varias-Vezes.

Rotina Alternada (17:30)	B4 e B2	Alternância sequencial dos dois áudios, 3 vezes cada.	Utiliza a função Tocar-Alternado.
Rotina Noturna (20:30)	B6, B5, B3, B4, B2	**NOVA DEMANDA:** Execução sequencial alternada dos 5 áudios. **Ciclo completo repetido 3 vezes.**	Utiliza a função Tocar-Alternado-Multiplo.
Aleatórios (A1-A6)	A1 - Crachá...mp3	Distribuição balanceada no período de **07:30h às 22:00h**. **Limite de 20 execuções diárias (aprox. a cada 43 minutos).**	Controle por $contagemExecucoesAleatorias e verificação de intervalo de tempo.

5. Logs, Auditoria e Governança

O sistema foi desenhado com foco em auditoria para processos de Acreditação (Ex: ONA, JCI).

- **Log de Eventos (Evidência Auditável):**
    - **Localização:** C:\execucao de audio\logs\execucao_YYYY-MM-DD.log
    - **Dados Registrados:** Data/Hora, Usuário ($env:USERNAME), Mensagem do evento (Início/Fim/Erro da execução).
    - **Importância:** Prova objetiva de que o requisito de comunicação sonora foi atendido, detalhando o "o quê", "quando" e "quem" executou o aviso.

5.1 Monitoramento e Plano de Contingência (Adição Crítica de Governança)

Em um ambiente crítico, a ausência de execução precisa ser imediatamente detectada.

- **Monitoramento de Processo:** A equipe de TI deve monitorar ativamente o processo PowerShell executado pela Tarefa Agendada. A interrupção deste processo deve gerar um alerta via ferramenta de monitoramento (ex: Zabbix, Nagios ou Monitoramento de Event Viewer).
- **Contingência de Falha:** Em caso de falha ou erro na execução de áudio (catch no código), o script registra o erro no log e continua o *loop* de monitoramento, garantindo que o próximo horário programado seja atendido.

- **Backup:** É mandatório que o diretório $pastaAudios (C:\execucao de audio) seja incluído na rotina de backup de servidores da TI.

6. Código Base (PowerShell - Trecho Essencial)

A lógica central de agendamento se concentra no loop while ($true) e na estrutura switch que verifica a hora atual ($horaStr).

```
# Executa áudios programados B1 a B6 e novo bloco 20:30
$executouAudioFixo = $false
switch ($horaStr) {
    # ... Casos para 11:00, 15:00, 15:30, 16:30, 17:30
    "20:30" {
        $arquivos20h30 = @(
            Join-Path $pastaAudios "B6 - UTI_09.mp3",
            Join-Path $pastaAudios "B5 - UTI_08.mp3",
            Join-Path $pastaAudios "B3 - UTI_04.mp3",
            Join-Path $pastaAudios "B4 - UTI_05.mp3",
            Join-Path $pastaAudios "B2 - UTI_01.mp3"
        )
        Tocar-Alternado-Multiplo $arquivos20h30 3
        $executouAudioFixo = $true
    }
}

# Executa áudios aleatórios balanceados
if (-not $executouAudioFixo) {
    # Verifica horário (07:30-22:00) e intervalo ($intervaloMinutos)
    # Lógica de balanceamento por contagem ($contagemExecucoesAleatorias)
    # ...
}
```

7. Próximos Passos (Evolução e Visão de Análise de Dados)

O sistema atual é altamente funcional. Futuras evoluções devem focar em:

1. **Configuração Externa:** Mover horários e nomes de arquivos de dentro do script para um arquivo de configuração externo (JSON ou XML), facilitando a manutenção e a alteração das regras de negócio sem tocar no código fonte.
2. **Interface Gráfica (Futuro):** Desenvolvimento de uma GUI simples para visualização do status e contadores de áudios aleatórios, utilizando WPF ou C# para uma *interface* de monitoramento.
3. **Análise de Dados de Execução:** Transformar os logs de execução (evidências auditáveis) em *dashboards* para analisar a distribuição real dos áudios, a taxa de falhas (erros de execução) e a eficácia da comunicação em diferentes horários.

8. Resultados e Conclusão (O Orgulho do Desenvolvedor)

O sistema automatizado em PowerShell não apenas atingiu o objetivo de otimizar a rotina hospitalar, mas superou as expectativas ao incorporar o rigor de um sistema auditável. A capacidade de lidar com rotinas complexas e fornecer registros de evidência demonstra a aplicação direta da minha formação em desenvolvimento Back-End e análise de requisitos para resolver problemas de negócio no setor crítico de saúde.

É com grande orgulho que, ao ouvir a execução do áudio distribuída pelos alto-falantes do hospital, posso afirmar que esta solução é o resultado de um esforço dedicado e de um compromisso em transformar processos manuais em automação robusta e inteligente.