Autor: Felipe Monteiro

Base de Conhecimento: Criando um agendador de tarefas com Worker Service

- **Objetivo:** Criar uma rotina programada para ser executada de acordo com um horário definido em .net core

1. **O que é um Job Scheduler?**
   Um Job Scheduler pode ser definido como um agendador de tarefas, que permite a execução de tarefas automatizadas, poupando trabalho manual de iniciar tarefas e além de várias atividades repetitivas que podem ser feitas por rotinas, de acordo com um horário específico.

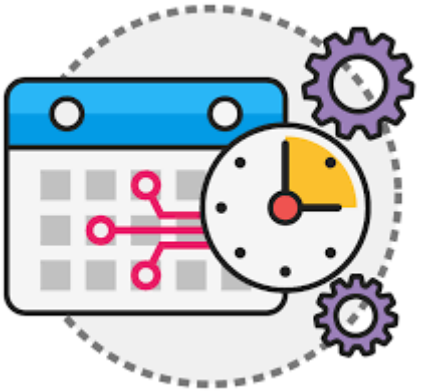


Imagem ilustrativa link: **https://aggregate.digital/technology/management/job-scheduler.html**

Dessa forma, possuem recursos avançados, como o monitoramento do status das tarefas, distribuição de tarefas e entre outras. Para monitorar o job é bom utilizar logs para acompanhamento do processo de execução das tarefas.

2. **O que é um Worker Service?**

È um tipo de aplicativo que permite criar serviços de execução continuada ou tarefas agendadas, podem ser executadas em segundo plano, por meio de Threads, com um agendamento programado, no Visual Studio pode ser criado um projeto com a estrutura já pronta para a configuração de acordo com a necessidade. Ao criar um Worker Service, você pode implementar a lógica de negócio na classe de serviço, que é responsável por realizar as tarefas necessárias. Essa classe pode ser configurada para ser executada continuamente ou agendada com base em um intervalo de tempo ou em eventos específicos. Para mais informações segue o link da documentação: https://learn.microsoft.com/pt-br/dotnet/core/extensions/workers?pivots=dotnet-7-0.
Ela também pode ser utilizada em uma Web Api, sendo configurado no arquivo Program da aplicação semelhante ao projeto Worker Service.

3. **O que é e como podemos utilizar a biblioteca Quartz.net?**

É uma biblioteca de job scheduling com implementação Job e Triggers utilizado para agendamento de tarefas, executando assim de acordo com que o desenvolvedor programe um horário pelas expressões Cron, podendo ser gerenciado essas tarefas. Temos que implementar o interface IJob que tem o método Execute, sendo que podemos implementar a lógica nessa chamada quando estiver no momento em que foi configurado o horário de execução. Mais informações sobre pode ser consultado a documentação: **https://www.quartz-scheduler.net/**.

4. **Como funcionam as Expressões Cron?**

As expressões Cron representam qual momento em que será executada a tarefa no método que define o horário/dia de execução da tarefa. podemos definir como:

- Execute a cada minuto a cada uma hora.
- Execute a cada hora, a partir da marca de 15 minutos da hora.
- Executar a cada hora, exceto no horário entre 02h00. e 05:00

Composta por cinco campos separados por espaços, que indicam o horário e a frequência de execução da tarefa.

```
*     *     *     *     *

-     -     -     -     -

|     |     |     |     |

|     |     |     |     +----- dia da semana (0 - 6) (Domingo=0)

|     |     |     +------- mês (1 - 12)

|     |     +--------- dia do mês (1 - 31)

|     +------------ hora (0 - 23)

   +-------------- minuto (0 - 59)
```

E assim cada campo pode ser preenchido com um valor numérico e assim podemos realizar o agendamento. Segue abaixo links para configurar o horário de execução.

Site para configurar tempo de execução de um cron job:

https://www.vivaolinux.com.br/artigo/Como-executar-tarefas-a-cada-5-10-ou-15-minutos
https://crontab.guru/#5_3_*_*_*

Exemplo:

```
"Quartz": {
        "JobImplementacion": "0 10 11 * * ?",
        "JobImplementacion2": "0 11 11 * * ?"
}
```

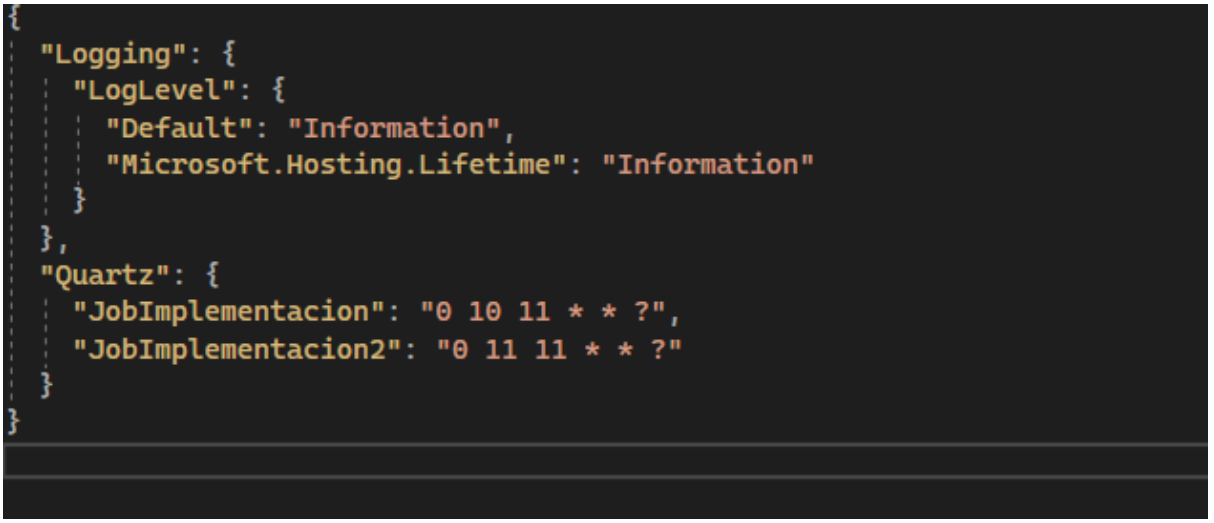
Nesse exemplo, temos dois jobs que serão executados em momentos diferentes, um todo dia às 10:11 e outro às 11:11 da manhã.

5. **Como criar um projeto Worker Service com horário programado?**

   Passo 1: criar projeto no visual studio, como projeto Worker Service em .net core na versão mais recente.

   Passo 2: Instalar a biblioteca de cron job Quartz no projeto pelo gerenciador de pacotes Nuget

   Passo 3: Definir expressão cron no appsetting.json neste exemplo está definido dois jobs cada um em um horário diferente o primeiro todo dia às 10:11 e o outro às 11:11 da manhã.



   Passo 4: Criar uma classe que irá implementar a interface IJob, que tem o método **Execute** que é nele que será executado a lógica a ser programada naquele momento.

```
4 referências
public class JobImplementacion : IJob
{
    private readonly ILogger<JobImplementacion> _logger;

    0 referências
    public JobImplementacion(ILogger<JobImplementacion> logger)
    {
        _logger = logger;
    }
    0 referências
    public Task Execute(IJobExecutionContext context)
    {
        _logger.LogInformation("Rodando: job 1 {time}",DateTime.Now);
        return Task.CompletedTask;
    }
}
```

Passo 5: Foi criada uma classe estática genérica com metodo estatico AddJobAndTrigger, responsável pela configuração das classes job com horário que estará definido pela expressão cron adicionado no arquivo appsettings.json, nessa classe que foi chamada de JobExtension, com isso adicionamos o Job e trigger cada um com um identificador único tendo como base a configuração IServiceCollectionQuartzConfigurator e IJob para a configuração no program.cs das classes criadas com a lógica por exemplo a jobImplementacion:

```
public static void AddJobAndTrigger<T>(
    this IServiceCollectionQuartzConfigurator quartz,
    IConfiguration config)
    where T : IJob
{
    //Pegar nome da classe - mesmo nome da chave do config
    string nomeJob = typeof(T).Name;

    var configKey = $"Quartz:{nomeJob}";
    //Site para configurar tempo de execucao de um cron job https://www.vivaolinux.com.br/artigo/Como-executar-tarefas-a-cada-5-10-ou-15-minutos
    //https://crontab.guru/#5_3_*_*_*
    var cronHorarioExecucao = config[configKey]; //5seg

    if (string.IsNullOrEmpty(cronHorarioExecucao))
    {
        throw new Exception($"No Quartz.NET Cron schedule found for job in configuration at {configKey}");
    }

    //registrando o job
    var jobKey = new JobKey(nomeJob);
    quartz.AddJob<T>(opts => opts.WithIdentity(jobKey));

    quartz.AddTrigger(opts => opts
        .ForJob(jobKey)
        .WithIdentity(nomeJob + "-trigger")
        .WithCronSchedule(cronHorarioExecucao));
}
```

Passo 6: Configurar o arquivo Program.cs adicionando o serviço da biblioteca Quartz com as configurações necessárias adicionando o método estático defenido na classe job Extension, nesse exemplo temos duas classes jobimplementacion e jobimplementacion2 que são duas lógicas que serão executadas em dois momentos distintos definidos pela expression cron, e com isso temos o nosso Cron job para ser executado.

```
IHost host = Host.CreateDefaultBuilder(args)
    .ConfigureServices((hostContext, services) =>
    {
        XmlConfigurator.Configure(new FileInfo("log4net.config"));
        //provedor
        services.AddLogging(loggingBuilder =>
        {
            loggingBuilder.AddLog4Net(); // Adiciona o provedor de logging do Log4Net
        });

        services.AddQuartz(q =>
        {
            q.UseMicrosoftDependencyInjectionJobFactory();
            q.AddJobAndTrigger<JobImplementacion>(hostContext.Configuration);
            q.AddJobAndTrigger<JobImplementacion2>(hostContext.Configuration);
        });
        services.AddQuartzHostedService(q => q.WaitForJobsToComplete = true);
    })
    .Build();

await host.RunAsync();
```

Concluindo : Podemos também adicionar essa biblioteca a uma Web Api também, configurando no mesmo arquivo Program.cs o Quartz.

Referências:

https://www.youtube.com/watch?v=EO_cg41frAM
https://www.quartz-scheduler.net/
https://www.hostinger.com.br/tutoriais/cron-job-guia
https://www.vivaolinux.com.br/artigo/Como-executar-tarefas-a-cada-5-10-ou-15-minutos
https://crontab.guru/#5_3_*_*_*
Cron Job: Guia completo para automatizar tarefas (hostgator.com.br)
https://learn.microsoft.com/pt-br/dotnet/core/extensions/workers?pivots=dotnet-7-0