```
//NODE-SCHEDULE:

//O node-schedule (traduzido também por "cronograma do node"), é uma biblioteca node que possuí métodos que podem executar nossas funções de forma agendada, onde é possível programar funções para serem executadas de um determinado tempo até um determinado tempo, ou até mesmo de acordo com um determinado dia e hora específicos.

//Vejamos como utilizá-la, e quais são alguns dos seus métodos mais usuais...
//OBS: o node-schedule não é uma biblioteca interna do node, ela precisa ser baixada.

//PRIMEIRO IMPORTAMOS O NODE-SCHEDULE...
const schedule = require('node-schedule')


//CRIAMOS ABAIXO UMA FUNÇÃO QUE DEVERÁ SER EXECUTADA DE 5 EM 5 SEGUNDOS...
const tarefa1 = schedule.scheduleJob('*/5 * 16 * * 1', function() { //Veja que a biblioteca schedule possuí um método chamado "scheduleJob()" esse método irá executar uma função de acordo com parâmetro numérico passado onde cada número irá indicar "(segundo minuto hora dia mes dia-da-semana)", lembrando que esse conjunto de números deverá ser colocado dentro de uma única string, podemos ignorar um parâmetro destes colocando "*" ou podemos determinar que a função será executada de tanto em tanto tempo usando a notação "*/numero", ela indica uma execução de tanto em tanto tempo de acordo com o número equivalente a uma medida de tempo.
    console.log('Executando Tarefa 1', new Date().getSeconds()) //Em conjunto, estamos instanciamos um objeto de Date para mostrar em qual segundo atual do minuto presente a tarefa está sendo executada...
})

setTimeout(function() {//Usando setTimeout() nós iremos chamar o método "cancel()" outro método da biblioteca "node-schedule" que serve para cancelar uma execução recursiva...
    tarefa1.cancel() //Usamos o método cancel para cancelar a recursão de tarefa1...
    console.log('Cancelando a Tarefa 1!')
}, 20000) //Após 20 segundos, a recursão de tarefa1 é cancelado...


//USANDO A REGRA DE RECURSÃO DO NODE-SCHEDULE:
//Por atravé da biblioteca "node-schedule" é possível criar uma regra de recursividade de tempo para ser utilizada como parâmetro no método "scheduleJob()"...
const regra = new schedule.RecurrenceRule() //Perceba que temos que gerar um objeto dentro da função construtora RecurrenceRule() da biblioteca "node-schedule"
```

```
regra.dayOfWeek = [new schedule.Range(1, 5)] //Esse objeto tem chaves como "dayOfWeek" que pode guardar um dia da semana específico, ou podemos gerar um conjunto de dias usando a função construtora Range da biblioteca "node-schedule" que capturará um dia inicial e um final, lembrando que 0 equivale ao domingo e 6 ao sábado...
regra.hour = 16 //A chave hour define um horário específico...
regra.second = 30 //A chave second guarda um segundo específico para que uma função seja executada, como colocamos 30, isso quer dizer que a função só será executada quando cada minuto equivaler ao instante de 30 segundos daquele minuto...

const tarefa2 = schedule.scheduleJob(regra, function () { //Perceba que usamos a regra recorrente como parâmetro para executar a tarefa2...
    console.log('Executando Tarefa 2', new Date().getSeconds()) //Por através do método "getSeconds()" deja o momento exato em que a tarefa 2 captura o segundo de número 30 do minuto atual...
})
```

RESULTADO NO CONSOLE...

```
"c:\Users\GaGra\Documents\javascript\arquivos_das_aulas\Projetos_Back_End\Temporizador_no_Node\01-node_schedule__Trabalhando_com_Tarefas_Agendadas.js"
Executando Tarefa 1 40
Executando Tarefa 1 45
Executando Tarefa 1 50
Executando Tarefa 1 55
Cancelando a Tarefa 1!
Executando Tarefa 2 30
Executando Tarefa 2 30
Executando Tarefa 2 30

[Done] exited with code=1 in 226.251 seconds
```