Cron NEST

- Creo el proyecto

> nest new cron-app

- Elimino los archivos que no necesito
- Instalo Swagger y lo configuro
- Creo la carpeta src/modules y genero el módulo de cron

> nest g res cron

- Instalo **@nestjs/schedule**, me va a permitir controlar los crons. Instalo los tipos con **@types/cron**
- En cron.module uso **ScheduleModule.forRoot**

```
import { ScheduleModule } from '@nestjs/schedule';

@Module({
  imports:[
    ScheduleModule.forRoot()  ],
  controllers: [CronController],
  providers: [CronService]
})
export class CronModule {}
```

- Evidentemente, **debo importar CronModule en el imports del app.module**
- En el cron.service cogemos su servicio **ScheduleRegistry** y lo inyectamos

```
import { SchedulerRegistry } from '@nestjs/schedule';

@Injectable()
export class CronService {

  constructor(private readonly scheduleRegistry: SchedulerRegistry){

  }
}
```

Crontab guru

- Creo tres métodos cron en el servicio. Puedo llamarlos cómo quiera
- Uso el decorador **@Cron**. Tenemos el tiempo y los parámetros. Cada asterisco significa cada minuto
- Si quieres añadir los segundos escribe */10
- Aquí podría colocarle la hora de la madrugarda que yo quiera, por ejemplo
- En el objeto le coloco la propiedad name que funciona como id. Tengo también el timezone

NOTA: con Ctrl + i veo las opciones disponibles del objeto (intellisense de Typescript)

```
import { Injectable } from '@nestjs/common';
import { Cron, SchedulerRegistry } from '@nestjs/schedule';

@Injectable()
export class CronService {

  constructor(private readonly scheduleRegistry: SchedulerRegistry){

  }

  @Cron(`*/10 * * * * *`, {
    name: 'cron1'

  })
  cron1(){
    console.log("cron1: Acción cada 10 segundos")
  }

  @Cron(`*/30 * * * * *`, {
    name: 'cron2'

  })
  cron2(){
    console.log("cron1: Acción cada 30 segundos")
  }

  @Cron(`* * * * *`, {
    name: 'cron3'

  })
  cron3(){
    console.log("cron1: Acción cada minuto")
  }
}
```

- No se necesita ejecutar ningún comando específico en la terminal, simplemente arrancar el servidor

Desactivando un cron

- Para desactivar un cron puedo usar PUT o POST
- Elegiré un PUT ya que realmente estoy haciendo algo parecido a modificar su estado

```
@Put('/desactivate/:name')
desactivateCron(@Param('name') name: string ){
```

```
return this.cronService.desactivateCron(name)
}
```

- En el service, primero debemos buscar si existe o no ese cron
- Si sitúo el cursor encima de job me dice que es de tipo CronJob. Le pongo tipado

```
desactivateCron(name:string){
    const job: CronJob = this.scheduleRegistry.getCronJob(name)

    if(!job){
      throw new NotFoundException("No se ha encontrado el cron")
    }else{
      job.stop()
      console.log(`El cron con el nombre: ${name} está desactivado`)
      return true
    }
  }
```

Activando un cron

- Practicamente hacemos un copy past del controlador anterior y cambiamos desactivate por activate
- controller

```
@Put('/activate/:name')
activateCron(@Param('name') name: string ){
  return this.cronService.activateCron(name)
}
```

- Y con el servicio igual, en lugar de poner .stop ponemos .start

```
activateCron(name:string){
    const job: CronJob = this.scheduleRegistry.getCronJob(name)

    if(!job){
      throw new NotFoundException("No se ha encontrado el cron")
    }else{
      job.start()
      console.log(`El cron con el nombre: ${name} está activado`)
      return true
    }
  }
```

Devolver todos los nombres de los crons

- Creo el endpoint en el controller

```
@Get()
getNameCrons(){
  return this.cronService.getNameCrons()
}
```

- En el servicio, obtengo los crons. .key son los nombres
- Uso un for of para llenar el array de cada uno de ellos

```
getNameCrons(){
    const names = []

    for(const name of this.scheduleRegistry.getCronJobs().keys()){
       names.push(name)
    }
    return names
  }
```

Desactivando todos los crons a la vez

- Usaremos un PUT

```
@Put('desactivate-all')
desactivateAllCrons(){
  return this.cronService.desactivateAllCrons()
}
```

- En el servicio, usaré el método anterior porque necesito los nombres
- Uso un ciclo for para recorrer el array y el metodo de desactivar anterior

```
desactivateAllCrons(){
    const names = this.getNameCrons()

    for(const name of names){
      this.desactivateCron(name)
    }
    return true
  }
```

Activando todos los crons

- Muy similar al anterior, en lugar de desactivar. Uso el método de activar
- En el controller

```
@Put('activate-all')
activateAllCrons(){
  return this.cronService.activateAllCrons()
}
```

- En el servicio

```
activateAllCrons(){
    const names = this.getNameCrons()

    for(const name of names){
      this.activateCron(name)
    }
    return true
  }
```