A UTILIZAÇÃO DO QUARTZ NO AGENDAMENTO DE TAREFAS

Lierbet MEDEIROS(1); Rosemary BORGES(2); Diego SOUZA(3); Fellipe ALEIXO(4)

(1) CEFET RN, Av. Senador Salgado Filho 1559, Tirol, Natal-RN, CEP 59015-000, 4005-2637, e-mail: lierbetnietzsche@gmail.com
(2) CEFET RN, e-mail: maryborges1@yahoo.com.br
(3) CEFET RN, e-mail: diegouern@gmail.com
(4) CEFET RN, e-mail: fellipe@cefetrn.br

RESUMO

Os sistemas computacionais estão cada vez mais atrelados à vida das pessoas. Muitas atividades cotidianas já se encontram vinculadas à execução de softwares. Em alguns casos, é preciso que seja iniciada a execução de um determinado sistema sem a interferência direta de um ser humano. Para tal, se fazem necessários mecanismos que possibilitem o agendamento de execução desses sistemas. Este artigo apresenta uma alternativa de ferramenta para o auxílio no agendamento de tarefas, o Quartz. O Quartz é um framework de código aberto inteiramente escrito em Java e projetado para uso em aplicações JSE (Java Standard Edition) e JEE (Java Enterprise Edition). Esse framework define um mecanismo de agendamento de tarefas que tem como responsabilidade executar (ou notificar) outros componentes de software quando um horário pré-determinado ocorre. O artigo irá apresentar um estudo geral sobre a ferramenta e os benefícios gerados pela sua utilização, além de mostrar um estudo de caso sobre a sua utilização no desenvolvimento do Sistema de Informação de Educação Profissional e Tecnológica – SIEP Gerencial, onde a aplicação do Quartz se deu mais especificamente no agendamento de extração de dados no módulo Extrator. O projeto SIEP Gerencial é uma iniciativa da Secretaria de Educação Profissional, supervisionado pelo SETEC/MEC, que conta com a participação de 10 núcleos de pesquisa e desenvolvimento, envolvendo vários CEFETs distribuídos pelo território nacional.

Palavras-chave: Agendamento de tarefas, Quartz, SIEP Gerencial

1. INTRODUÇÃO

Os sistemas computacionais estão cada vez mais atrelados à vida das pessoas. Muitas atividades cotidianas já se encontram vinculadas à execução de softwares. Em alguns casos, é preciso que seja iniciada a execução de um determinado sistema sem a interferência direta de um ser humano. Para tal, se fazem necessários mecanismos que possibilitem o agendamento de execução desses sistemas. Se suas aplicações em Java precisam ser realizadas em um momento específico ou se seus sistemas têm de retornar manutenção de trabalhos que poderiam ser automatizados, então este artigo apresenta uma alternativa de ferramenta para o auxílio no agendamento de tarefas, o Quartz: um framework open source (código aberto) inteiramente escrito em Java e projetado para uso em aplicações JSE (Java Standard Edition) e JEE (Java Enterprise Edition). O O Quartz define um mecanismo de agendamento de tarefas que tem como responsabilidade executar (ou notificar) outros componentes de software quando um horário pré-determinado ocorre. Sendo totalmente flexível, o Quartz contém vários modelos que podem ser usado separadamente ou juntos, de acordo com as necessidades da aplicação, e permite ao desenvolvedor escrever seu código da maneira mais natural possível para o seu projeto. Necessita de pouquíssima configuração e pode ser usado para funcionar automaticamente se suas necessidades forem relativamente básicas. Apesar de ele ser extremante proveitoso para execução de processos simples em sistemas de agendamento de tarefas (job scheduler), esse framework apresenta todo o seu potencial no controle de fluxos dos processos de negócios de suas aplicações.

Neste trabalho, iremos apresentar um estudo geral sobre sistemas de agendamento de tarefas (Job Scheduler), sobre a ferramenta Quartz e os benefícios gerados pela sua utilização, que empresas e entidades estão utilizando ou contribuindo para o desenvolvimento deste framework, além de mostrar um estudo de caso sobre a sua utilização no desenvolvimento do Sistema de Informação de Educação Profissional e Tecnológica – SIEP Gerencial, onde a aplicação do Quartz se deu mais especificamente no agendamento de extração de dados no módulo Extrator. O projeto SIEP Gerencial é uma iniciativa da Secretaria de Educação Profissional, supervisionado pelo SETEC/MEC, que conta com a participação de 10 núcleos de pesquisa e desenvolvimento, envolvendo vários CEFETs distribuídos pelo território nacional.

2. JOB SCHEDULING COM QUARTZ

2.1. Job Scheduler

Job Scheduler (agendador de tarefa) é um sistema que tem como responsabilidade executar (ou notificar) outros componentes de software quando um pré-determinado horário ocorre e pode ser integrado com qualquer outro sistema. Hoje, os agendadores de tarefas são cada vez mais necessários para orquestrar a integração das atividades empresariais em tempo real, através de diferentes plataformas e sistemas operacionais aplicados a ambientes empresariais.

2.2. O que é Quartz?

Quartz é um framework de agendamento de tarefas (job scheduler) de código aberto (open source) inteiramente escrito em Java e projetado para uso em aplicações JSE (Java Standard Edition) e JEE (Java Enterprise Edition). Ele oferece uma grande flexibilidade e contém múltiplos usos de paradigmas que podem ser utilizados em conjunto ou separadamente para alcançar o comportamento desejado pela aplicação. E isso permite escrever o código no que parece ser o caminho mais natural para o desenvolvimento do projeto. Ele dispõe de recursos como: suporte de dados, clustering, plugins, pré postos de trabalho para EJB, JavaMail, além de muitos outros serviços.

Quartz é distribuído com uma pequena biblioteca Java (.jar) que contém todas as principais funcionalidades desse framework. A principal API para essas funcionalidades é a Scheduler. Ela prover operações simples como: scheduling/unscheduling de jobs, starting/stopping/pausing de scheduler.

Para agendar a execução dos próprios componentes, deve-se implementar a interface de um simples Job, o qual contém o método execute. Caso seja necessário ter componentes notificados quando um determinado agendamento ocorre, então os componentes devem implementar as interfaces TriggerListener ou JobListener.

O principal processo do Quartz pode ser iniciado e executado dentro da sua aplicação, como uma aplicação stand-alone (com uma user interface), ou dentro de servidor de aplicações J2EE para ser utilizado como recurso para os componentes de sua aplicação.

3. PROGRAMANDO COM QUARTZ

Antes de utilizar o scheduler é preciso instancia-lo. Para fazer isso você utiliza a Factory SchedulerFactory. Alguns usuários do Quartz gostam de manter a instancia da factory serializada no armazenamento JNDI, mas outros preferem (por ser mais fácil) instanciar e usar a factory diretamente (como o exemplo abaixo). Uma vez que o scheduler é instanciado, ele pode ser iniciado, pausado ou terminado. Note que uma vez que um scheduler é terminado, ele não pode ser reiniciado sem ser instanciado novamente. As Triggers não são disparadas (jobs não são executados) até que o scheduler tenha sido iniciado.

Na figura 1 temos o pequeno trecho de código, que instancia e inicia um scheduler, e agenda um job para ser executado:

```
SchedulerFactory schedFact = new org.quartz.impl.StdSchedulerFactory();

Scheduler sched = schedFact.getScheduler();

sched.start();

JobDetail jobDetail = new JobDetail("myJob", sched.DEFAULT_GROUP, DumbJob.class);

SimpleTrigger trigger = new SimpleTrigger("myTrigger",sched.DEFAULT_GROUP,new Date(),null,0,0L);

sched.scheduleJob(jobDetail, trigger);
```

Figura 1 –Trecho de código de um Scheduler

3.1. Jobs and triggers

As duas unidades de quartzo fundamentais da programação de pacotes são jobs e triggers. Um trabalho é uma tarefa executável que pode ser programada, ao mesmo tempo que proporciona uma desencadear um calendário para um emprego. Embora estas duas entidades poderia facilmente ter sido combinada, a sua separação em ambas as Quartz é intencional e benéfico.

Ao manter o trabalho a ser realizado separado de sua programação, Quartz permite alterar o programado para acionar um posto de trabalho sem perder o trabalho em si, ou o contexto em torno dele. Além disso, nenhum singular trabalho pode ter muitas desencadeia a ele associados.

Podemos fazer componentes Java serem executados por um scheduler, simplesmente implementando a interface Job. Podemos conferir a interface na figura 2:

```
package org.quartz;

public interface Job {

  public void execute(JobExecutionContext context) throws JobExecutionException{
}
```

Figura 2 – Trecho de código de um Job

Nesse caso, não poderíamos adivinhar, quando a trigger do Job é disparada (mais do que em um momento), o método "execute" é invocado pelo scheduler. O objeto JobExecutionContext que é passado para este método, prove a instancia do job com informações sobre seu ambiente "run-time" - um handle para o Scheduler que é executado, um handle para a Trigger que dispara a execução, o JobDetail e alguns outros itens.

O objeto JobDetail é criado pela sua aplicação e no momento do Job é adicionado para o scheduler. Ele contém várias configurações para o Job, como o JobDataMap, o qual pode ser usado como um armazenador de informações de estado para uma dada instancia da sua classe de job.

Objetos Triggers são usados para disparar a execução de jobs. Quando se deseja agendar um job, instancia uma trigger e configura suas propriedades para prover um agendamento que se deseja. Existe atualmente dois tipos de triggers: SimpleTrigger e CronTrigger.

SimpleTrigger é um gerenciador. Se precisar de uma execução (uma simples execução de seu job em um determinado momento), ou se você precisar disparar um job em um determinado momento e tem que repetir N vezes, com um delay de T entre cada execução. CronTrigger é usado se deseja ter lançamentos baseados em agendamento "calendar-like" - como toda quarta-feira, ao dia ou as 12:00 do dia 15 de todo mês.

3.1.1. Por que Jobs e Triggers?

Muitos jobs agendados não tem noções de separação entre jobs e triggers. Alguns definem um job como uma simples execução em um determinado momento junto com alguns pequenos identificadores. Enquanto desenvolvendo Quartz, foi decidido que faria sentido criar uma separação entre o schedule e o trabalho para executar o schedule. Isto tem muitos benefícios.

Jobs e Triggers são dados nomes identificadores como eles são registrados no scheduler do Quartz. Jobs e triggers podem também estar juntos dentro de grupos os quais podem ser proveitosos para organização se seus jobs e triggers dentro de categorias para manutenções futuras. O nome de um job ou trigger devem ser únicos dentro do grupo - em outras palavras, o verdadeiro identificador de um job ou trigger é o "nome+grupo".

3.2. Jobs e JobsDetails

Enquanto a classe que está sendo implementada é a atual "job", Quartz precisa ser informado sobre vários atributos que desejamos que o job tenha. Isto é feito através da classe JobDetail.

```
JobDetail jobDetail = new JobDetail("myJob",                    // nome do job
                                    sched.DEFAULT_GROUP,        // grupo do job
                                    DumbJob.class);             // a classe Java que será executada

 SimpleTrigger trigger = new SimpleTrigger("myTrigger",
                                           sched.DEFAULT_GROUP,
                                           new Date(),
                                           null,
                                           0,
                                           0L);

 sched.scheduleJob(jobDetail, trigger);

agora considere a classe "DumJob" definida como:



 public class DumbJob implements Job {

   public DumbJob() {
   }

   public void execute(JobExecutionContext context)
     throws JobExecutionException
   {
     System.err.println("DumbJob esta sendo executado.");
   }
 }
```

Figura 3 – Trecho de código de um JobDetail

De acordo com a figura 3, foi informado ao scheduler uma instancia de JobDetail, e que esse refere ao job que será executado simplesmente provendo a classe do job. Cada tempo que o scheduler executa o job, ele cria uma nova instancia da classe antes de chamar o método execute. Uma das ramificações desta ação é o fato de que jobs devem ter um construtor sem parâmetros. Uma outra ramificação é que não faz sentido ter definido "data-members" na classe do job - como seus valores podem ser limpos a cada execução.

3.2.1. Atributos de Job

Sumario de propriedades que podem ser definidas para instancias do job através do objeto JobDetail:

- Durability - se um job é non-durable, ele é automaticamente apagado do scheduler uma vez que não tenha mais qualquer trigger ativa associada a ele.
- Volatility - se um job é volatile, ele não é persistido entre "restarts" do scheduler do Quartz.
- RequestsRecovery - se um job é "requests recovery" e durante a sua execução ocorre algum problema de "crashe" ou a maquina é desligada, então o job é re-executado quando o scheduler for iniciado novamente. Neste caso, o método JobExecutionContext.isRecovering() irá retornar true.
- JobListeners - um job pode ter um conjunto de zero ou mais JobListeners associado a ele. Quando o job é executado, os listeners são notificados.

3.2.2. O método Job.execute

Finalmente, é preciso informar alguns detalhes sobre o método Job.execute. O único tipo de exception que esta permitido lançar a partir do método execute é a JobExecutionException (incluindo RuntimeExceptions). Isto porque deve geralmente o conteúdo do método execute estar em um bloco "try-catch".

3.3. Triggers

Como jobs, triggers são relativamente fáceis de trabalhar, mas contém uma variedade de opções para customizar que você precisa estar atento e entender antes de você fazer uso total do Quartz. Existem diferentes tipos de triggers que podemos escolher de acordo com as necessidades de scheduling.

3.3.1. Calendar

Os objetos Calendar do Quartz (não são java.util.Calendar) podem ser associados com triggers no momento que a trigger é armazenada no scheduler. Calendars são úteis para controlar intervalos de tempo no disparo de triggers. Para uma instancia de um objeto de negócio, você pode criar uma trigger que dispara um job todo dia da semana as 13:30 am, então adicione um Calendar que exclui todos os feriados que estão em dias comerciais. Qualquer objeto serializável que implemente a interface Calendar pode ser um Calendar para as triggers, veja na figura 4:

```
package org.quartz;

public interface Calendar {

  public boolean isTimeIncluded(long timeStamp);

  public long getNextIncludedTime(long timeStamp);

}
```

Figura 4 – Trecho de código de Calendar

O Calendar deve ser instanciado e registrado no scheduler através do método addCalendar(). Se você usa o HolidayCalendar, após a instancia-lo, você deve usar o método addExcludedDate(Date date) para adicionar os dias que deverão ser excluídos do scheduler. A mesma instancia do Calendar pode ser usada com várias triggers, veja na figura 5:

```
HolidayCalendar cal = new HolidayCalendar();
cal.addExcludedDate( someDate );

sched.addCalendar("myHolidays", cal, false);

SimpleTrigger trigger = new SimpleTrigger("myTrigger",
                                          sched.DEFAULT_GROUP,
                                          new Date(),
                                          null,
                                          SimpleTrigger.REPEAT_INDEFINITELY,
                                          60L * 1000L);
trigger.setCalendarName("myHolidays");

// .. agendando o job com trigger

SimpleTrigger trigger2 = new SimpleTrigger("myTrigger",
                                           sched.DEFAULT_GROUP,
                                           new Date(),
                                           null,
                                           5,
                                           5L * 24L * 60L * 60L * 1000L);

trigger2.setCalendarName("myHolidays");

// .. agendando o job com trigger2
```

Figura 5 – Trecho de código de SimpleTrigger

O código cria duas triggers: uma que irá repetir a cada 60 segundos para sempre, e a outra que irá repetir cinco vezes com intervalo de cinco dias.

3.3.2. Erro de disparo

Uma outra propriedade importante da trigger é a "misfire instruction". Se ocorrer um erro de disparo com uma trigger persistente, perdendo o disparo por causa de um problema com o scheduler. Existem diferentes tipos de "misfire instructions" para os diferentes tipos de triggers. Pelo padrão é usado a instrução "smart policy" - a qual tem um comportamento dinâmico baseado no tipo de trigger e configurações. Quando o scheduler inicia, ele procura por qualquer tipo de trigger persistente que tem erro de disparo e então atualiza cada uma baseada nas suas configurações individuais de "misfire instructions". Os "misfire instructions" podem ser configurados para as instancias de triggers usando o método setMisfireInstruction(..).

3.3.3. CronTrigger

CronTriggers são mais úteis que as SimpleTrigger, pois apenas precisar agendar jobs que serão disparados utilizando informações de calendário, e não simplesmente intervalos. Com CronTrigger, pode-se especificar agendamento como toda segunda-feira ou todo dia as 8:30 am.

3.3.4. Expressões de Cron

Cron-Expressions são utilizadas para configurar instancias de CronTrigger. Cron-Expressions são strings que são divididas em sei partes que descrevem detalhes individuais do schedule. Essas strings (sub-expressions) são separadas com espaço em branco e representam:

- Segundos
- Minutos

- Horas
- Dias do mês
- Mês
- Dias da semana

Um exemplo de um cron-expression completa é a string "0 0 12 ? * WED" - a qual significa todas as quartas-feiras as 12:00 pm.

"Sub-expressions" podem conter intervalos e/ou listas. Por exemplo, o campo dia da semana pode conter "WED", "MON-FRI" ou "MON-WED, SAT".

Wild-cards (o caractere '*') são utilizados nos campos quando se quer utilizar todas as possibilidades. Um * no campo Dias da semana (Day-Of-Week) significa que estará sendo considerado todos os dias da semana.

Todos os campos tem um conjunto de valores válidos que podem ser especificados. Esses valores devem estar corretos - por exemplo números entre 0 e 59 para segundos e minutos, 0 e 23 para horas, 0 e 31 para dias do mês, 0 e 11 para meses do anos, mas também pode usar JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV e DEC, 1 e 7 para dias da semana (1=domingo) ou usar SUN, MON, TUE, WED, THU, FRI e SAT.

O caractere "/" pode ser utilizado para especificar incrementos para valores. Por exemplo se você utilizar "0/15" para o campo minutos, isso significa que a cada 15 minutos, iniciando no minuto 0. Se você usar "3/20" para o campo minutos isso significa a cada 20 minutos durante a hora, iniciando no minuto 3 - isso é o mesmo que especificar "3,23,43" para o campo minutos.

O caractere "?" é permitido para dias do mês e dias da semana. Ele é usado para especificar "valores sem especificação". Isto é muito útil quando você precisa especificar algo em um ou dois campos, mas não nos outros. Nos exemplos a seguir isso ficará mais claro.

O caractere "L" é permitido para dias do mês e dias da semana. Este caractere é usado por ultimo, mas ele faz diferença em cada um desses campos. Por exemplo: o valor "L" no campo dias do mês significa o ultimo dia do mês - dia 31 para Janeiro, 28/29 para Fevereiro e etc. Se for usado para dias da semana, isso significa 7 ou Sábado. Mas se for utilizado para dias da semana após outro valor, isso significa o ultimo dia "xxx" do mês - por exemplo: se usar "6L" ou "FRIL", significa a ultima sexta do mês.

4. SIEP GERENCIAL

O SIEP Gerencial se propõe a centralizar os dados da Educação Profissional e Tecnológica no Brasil na SETEC/MEC. A arquitetura proposta para o sistema foi baseada na plataforma Java Enterprise Edition, a qual é voltada para o desenvolvimento de sistemas corporativos. O SIEP Gerencial é um dos projetos desenvolvidos para essa secretaria, a qual também apóia a Biblioteca Digital, o Portal Nacional de EPT e o SIGA-EPT.

4.1. Módulo extrator

O projeto SIEP Gerencial é composto de vários módulos, dentre eles o módulo Extrator, a cargo do núcleo CEFET-RN. O módulo Extrator é formado por diversos componentes que realizam atividades de extração, visualização e coleta das informações da instituição. Quando o módulo está em funcionamento em um IFET, o componente de Extração deverá capturar os dados utilizando a técnica de Reflection, garantindo robustez e dinamicidade ao sistema, que pode ter o seu modelo de dados alterado sem comprometer o funcionamento do componente de Extração. Em seguida, os dados extraídos devem ser aprovados e assinados digitalmente pelo gestor responsável pela instituição através do componente Visualizador. Por fim, o componente Coletor compacta e criptografa os dados extraídos e disponibiliza-os via Web Service para que o módulo Atualizador realize o depósito na base de dados central do projeto SIEP Gerencial.

4.2. Quartz no agendamento da extração de dados

Este caso de uso é responsável pelo agendamento das Extrações realizadas pelo Subsistema Extrator.O Agendamento é composto por informações como o dia e horário. Ele é executado em segundo plano e é iniciado junto ao Sistema Operacional.

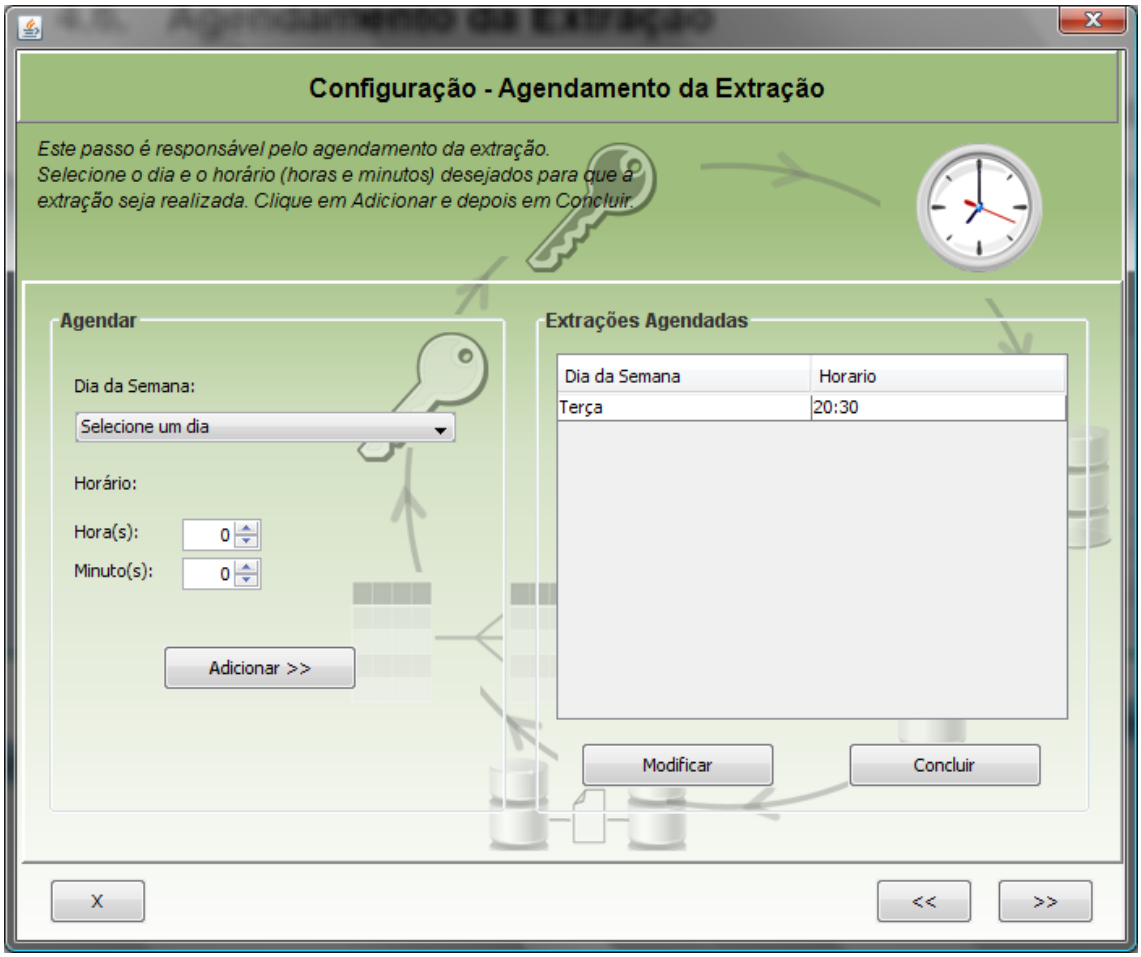

Figura 4 – Imagem da tela de configuração do agendamento da extração dados

Após serem realizadas todas as devidas configurações inerentes ao processo de extração de dados, o sistema exibe a tela para configuração do agendamento da extração. Nessa tela, o operador IFET irá selecionar o dia da semana, hora e o minuto, ao qual deseja que seja realizada a extração. As informações aparecem na tabela “Extrações Agendadas” e em seguida o operador clica em “concluir” e clica em “>>” para avançar para a próxima sessão.

5. CONCLUSÂO

Com o mundo cada vez mais globalizado e a evolução das tecnologias e da Internet, percebemos muitas atividades cotidianas vinculadas ao uso de softwares. Em alguns casos, se faz necessário que a execução de um determinado sistema seja realizado sem a interferência direta de um ser humano, e para isso, vimos que é necessário o uso de um agendador de tarefas (Job Scheduler) para se responsabilizar em executar outros componentes de software quando um determinado horário ocorrer. E como agendador de tarefas, vimos o Quartz, que é um framework que se mostra cada vez mais eficaz para ser usado no agendamento de tarefas de alguma aplicação vinculada a sistemas de grande e/ou pequeno porte, onde ele consegue manter toda a flexibilade requerida pela a aplicação sem prejudicar na codificação por parte do desenvolvedor.

E foi através de uma necessidade do SIEP-Gerencial, mais especificamente no módulo extrator, que utilizamos o Quartz para o Operador IFET ter condições em agendar as extrações de dados da instituição no dia da semana, hora(s) e minuto(s) desejados. A sua utilização no sistema responsável pela extração dos dados do projeto SIEP foi de grande importância, já que resultou na diminuição do tempo de desenvolvimento, numa melhor reusabilidade do sistema e também na diminuição do código.

REFERÊNCIAS

C**A**VANESS, C. **What is quartz.** Disponível em:

< http://www.onjava.com/pub/a/onjava/2005/09/28/what-is-quartz.html?page=1> Acesso em: 03 ago 2008.

LIPTON, M.; JANG, S. **Job scheduling with Quartz.** Disponível em:
< http://www-128.ibm.com/developerworks/java/library/j-quartz/> Acesso em: 05 ago 2008.

Quartz user guide. Disponível em:
< http://www.roseindia.net/quartz/index.shtml> Acesso em: 02 ago 2008.

Quartz tutorial. Disponível em:
< http://www.roseindia.net/quartz/index.shtml> Acesso em: 02 ago 2008.

The official quartz tutorial. Disponível em:
< http://www.opensymphony.com/quartz/wikidocs/Tutorial.html > Acesso em: 02 ago 2008.

AGRADECIMENTOS

Agradecemos ao professor Fellipe Aleixo e ao colega de desenvolvimento Bruno Gomes pelo o auxílio nos estudos de Quartz.