4
Implementação do Framework em Java e AspectJ

Este capítulo apresenta a arquitetura detalhada dos componentes que compõem o framework orientado a aspectos para monitoramento e análise de processo de negócio proposto nesta dissertação. Além disso, serão abordadas as tecnologias utilizadas para o desenvolvimento do framework.

4.1.
Arquitetura Detalhada do Framework

O framework para monitoramento e análise de processos de negócio, foi desenvolvido usando as tecnologias Java e AspectJ [12]. Uma aplicação web baseada no framework Spring [10] foi utilizada para validar o projeto e implementação de tais componentes. O projeto do framework definiu a implementação dos seguintes componentes: Register, Monitor, Analyzer e Reporter, DatabaseAccess e GUI. Nas próximas seções, os componentes que constituem o framework são detalhados.

4.1.1.
Componente Register

O componente Register define classes para o cadastro e manipulação de processos e operações de negócio. A Figura 10 mostra o seu projeto detalhado, o qual contém os seguintes elementos: (i) interface `RegisterServices` – expõe os serviços disponibilizados pelo componente Register; (ii) classe `BusinessProcessServices` – essa classe implementa os serviços de cadastro de operações e processos de negócio (com respectivas validações de negócio necessárias), realizando acesso ao banco de dados através de chamada aos serviços do componente DatabaseAccess. Observe na Figura 10 que o componente Register utiliza a interface `BusinessProcessDAO` e a classe `BusinessProcessDAOHibernate` do componente DatabaseAccess para realizar esse acesso ao banco de dados.

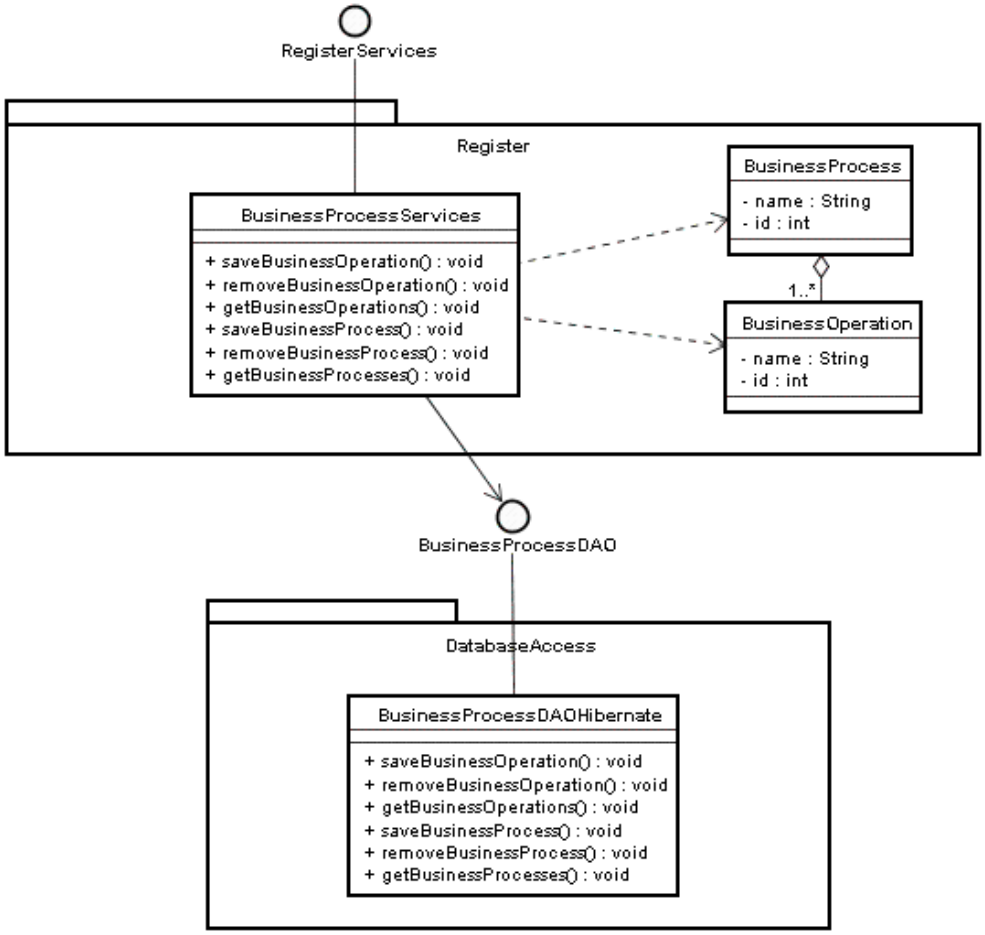

Figura 10. Componente Register.

4.1.2. Componente Monitor

O componente Monitor se responsabiliza pelo monitoramento de operações de negócio, ocorrendo num dado sistema web. A Figura 11 mostra o detalhamento desse componente. O aspecto abstrato `BusinessOperationMonitor` declara um ponto de corte (*pointcut*) e um método abstrato que serão especificados por seus subaspectos para, respectivamente, (i) concretizar os pontos de junção de aplicações web nos quais serão monitorados suas operações de negócio; e (ii) definir qual será o módulo de comunicação usado por cada subaspecto. Esse aspecto também agrega um objeto do tipo `CommunicationModule` que é usado para realizar a comunicação com o componente Analyzer. A Figura 11 também apresenta a instanciação desse componente, através da criação do subaspecto `MonitorBusinessOperationStruts` e da classe `RMICommunicationModule`, que representam instanciações concretas do componente Monitor para acompanhar

operações de negócio sendo executadas sobre uma aplicação estruturada usando o framework Struts [45] em sua camada de interface gráfica, e enviar tais informações para o componente Analyzer usando o mecanismo de invocação remota de Java RMI (*Remote Method Invocation*).

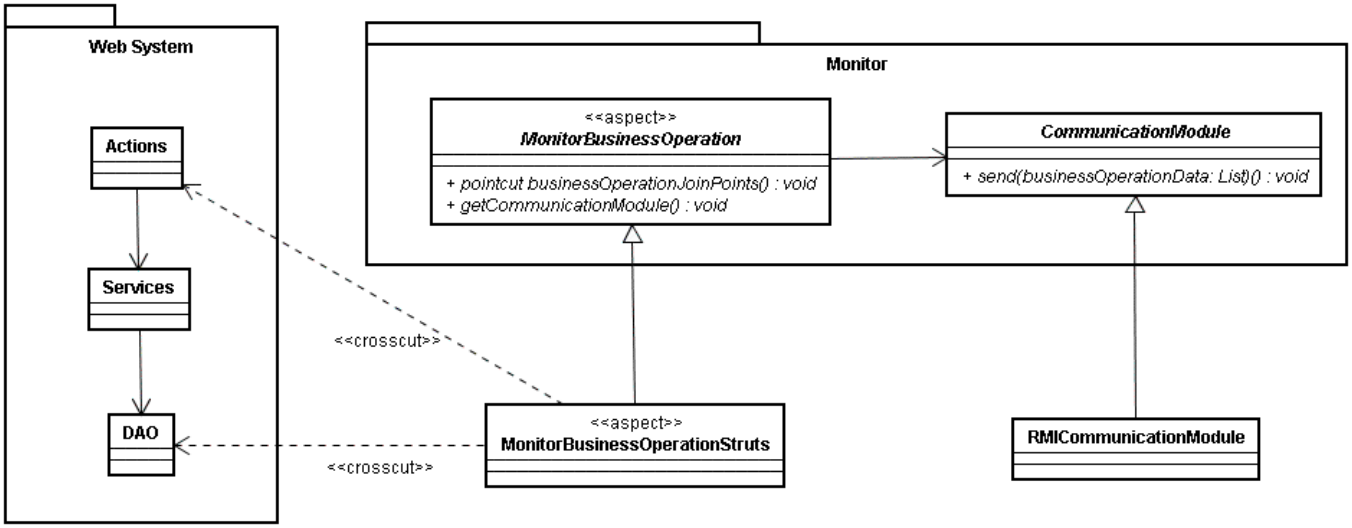

Figura 11. Componente Monitor.

A Figura 12 apresenta o código do aspecto abstrato `MonitorBusinessOperation`. Como pode ser visto, ele mantém referência para dois atributos (linhas 2-3): (i) um objeto do tipo `CommunicationModule` que é usado para transmitir uma dada operação de negócio capturada; e (ii) um objeto do tipo `Map` que armazena as requisições web realizadas por um dado usuário numa dada sessão (*thread*). O aspecto declara o *pointcut* abstrato `webActionExecution()` que é usado para capturar o `HttpServletRequest` ocorrendo num dado contexto de usuário (linhas 5-6). Um *advice* do tipo `before` é associado a tal ponto de corte, com o objetivo de armazenar no objeto `Map` anteriormente discutido, a sessão de execução daquela requisição, assim como a própria requisição (linhas 7-12). O ponto de corte abstrato `businessOperationCalls()` é usado para capturar a execução de operações de negócio no sistema web sendo monitorado (linha 13) . Ele possui um *advice* `after finally` associado, cujo objetivo é obter uma série de informações relacionadas a operação de negócio sendo executada (nome do usuário, nome da

operação de negócio, sessão de execução, e data da ocorrência) para permitir a criação de um objeto do tipo `UserBusinessOperation` (linhas 15-33). Tal objeto é enviado para o componente Analyzer do framework, usando um objeto do tipo `CommunicationModule` (linha 32). Durante a instanciação do framework, o aspecto abstrato é especializado para concretizar os pontos de corte `webActionExecution()` e `businessOperationCalls()` de forma a monitorar as operações de negócio de uma determinada aplicação web.

```
1. public abstract aspect MonitorBusinessOperation {
2.   private CommunicationModule communication;
3.   private Map hashRequestPerUserThread = new Hashtable();
4.
5.   public abstract pointcut webActionExecution
6.                           (HttpServletRequest request);
7.   before(HttpServletRequest request):
8.                    webActionExecution(request){
9.     String threadId = Long.toString
10.            (Thread.currentThread().getId());
11.    hashRequestPerUserThread.put(threadId, request);
12.  }
13.  public abstract pointcut businessOperationCalls();
14.
15.  after(): businessOperationCalls(){
16.     String threadId = Long.toString
17.              (Thread.currentThread().getId());
18.     HttpServletRequest request = (HttpServletRequest)
19.                  hashRequestPerUserThread.get(threadId);
20.
21.     String userBP = request.getUserPrincipal().getName();
22.     String nameBP = thisJoinPoint.getSignature().toString();
23.     String sessionBP = request.getSession().getId();
24.     Date occurrenceDate = new Date();
25.
26.     BusinessOperation bo = new BusinessOperation();
27.     bo.setUser(userBP);
28.     bo.setSession(sessionBP);
29.     bo.setName(nameBP);
30.     bo.setOccurrenceDate(occurrenceDate);
31.
32.     this.communication.registerBusinessOperation(bo);
33.  }
34.  ...
35. }
```

Figura 12. Aspecto Abstrato `MonitorBusinessOperation`

4.1.3. Componente Analyzer

O componente Analyzer tem como objetivo analisar as operações de negócio executadas e verificar se ocorreu um dado processo de negócio. A Figura 13 apresenta o componente Analyzer. Ele oferece um serviço para cadastro de operações de negócio executadas por usuários em um sistema web, através da interface `AnalyzerServices` e da classe `FacadeBusinessProcessFramework`.

Esse serviço é invocado pelo módulo de comunicação do componente Monitor. Cada operação de negócio do usuário (classe `UserBusinessOperation`) capturada e enviada pelo componente Monitor, é armazenada pelo componente Analyzer. Além disso, sempre que uma nova operação de negócio chega ao componente Analyzer, é executado um processo de caracterização de processo de negócio (classe `BusinessProcessAnalyzer`), a partir do histórico de operações de negócio executadas recentemente. Durante este processo, o conjunto de operações de negócio já executadas por um dado usuário é analisado, para verificar se composta com a nova operação de negócio coletada, foi caracterizado a execução de um dado processo de negócio de interesse. Caso seja identificada a execução de um processo de negócio do usuário, esse é cadastrado (classe `UserBusinessProcess`) no sistema juntamente com suas respectivas operações de negócio. O armazenamento dos dados é feito através da interface `UserBusinessProcessDAO` oferecida pelo componente DatabaseAccess.

Cada requisição de cadastro de uma operação de negócio é manipulada por uma diferente linha de controle de execução (*thread*), dentro do componente Analyzer. O objetivo principal é permitir que o núcleo do framework funcione como um servidor *multi-threaded*. A implementação de tal funcionalidade no componente Analyzer foi realizada através da definição de um *Active Object* [46]. A funcionalidade de tal serviço consiste em basicamente interceptar, através de POA, execuções do método `registerBusinessOperations()` da classe `FacadeBusinessProcessFramework`. Após a interceptação de tal método, um *advice* do tipo *around* é executado para solicitar a continuação da invocação do método em uma nova *thread* gerenciada por um *Active Object*. Duas diferentes políticas de gerenciamento são oferecidas pelo *Active Object* para serem escolhidas durante a instanciação do framework: (i) *ThreadPerRequest* – que determina a criação de um thread para cada nova requisição de cadastro de operação de negócio; e (ii) *ThreadPool* – contempla a criação de um número específico e limitado de *threads* para executar as requisições do usuário. Por *default*, a política *ThreadPerRequest* é instanciada automaticamente pelo framework.

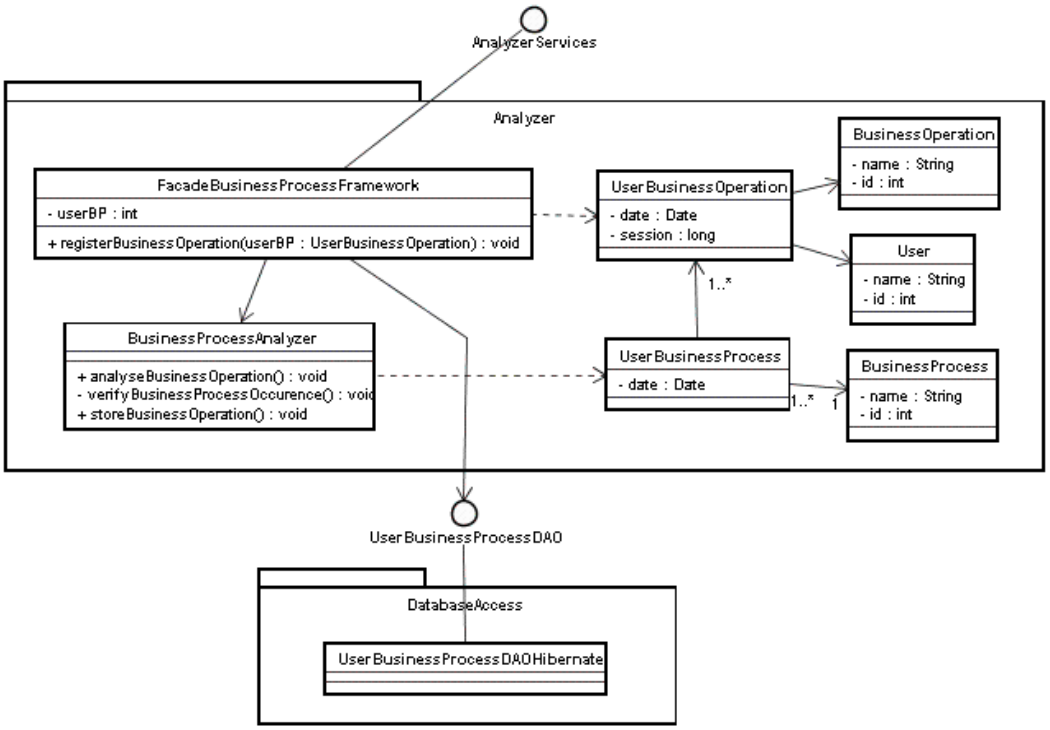

Figura 13. Componente Analyzer.

4.1.4. Componente DatabaseAccess

O componente DatabaseAccess define um conjunto de classes para acesso as informações mantidas pelo framework. Seus serviços são expostos pela interface `UserBusinessProcessDao` e `BusinessProcessDAO`.

Este componente utiliza o framework de persistência Hibernate [13] para persistir a informação de objetos em tabelas no banco de dados. Uma vantagem da utilização do framework Hibernate para fazer a persistência dos dados, deve-se ao fato de que ele provê uma forma transparente de acesso a diversos tipos de sistemas de gerenciamento de banco de dados. Assim, a mudança no tipo de banco de dados utilizado não acarretará em grandes mudanças no framework. Ele representa a implementação do padrão de projeto *Data Access Object* (DAO) [47]

4.1.5. Componente GUI

O componente GUI é responsável por apresentar uma interface gráfica para que o administrador possa realizar as seguintes tarefas : (i) cadastrar operações de

negócio; (ii) cadastrar processo de negócio; e (iii) visualização e modificação de processos e operações de negócio já cadastrados.

Inicialmente uma tela para o acesso ao sistema é apresentada. A Figura 14 apresenta a tela de acesso para que o administrador se conecte ao sistema.

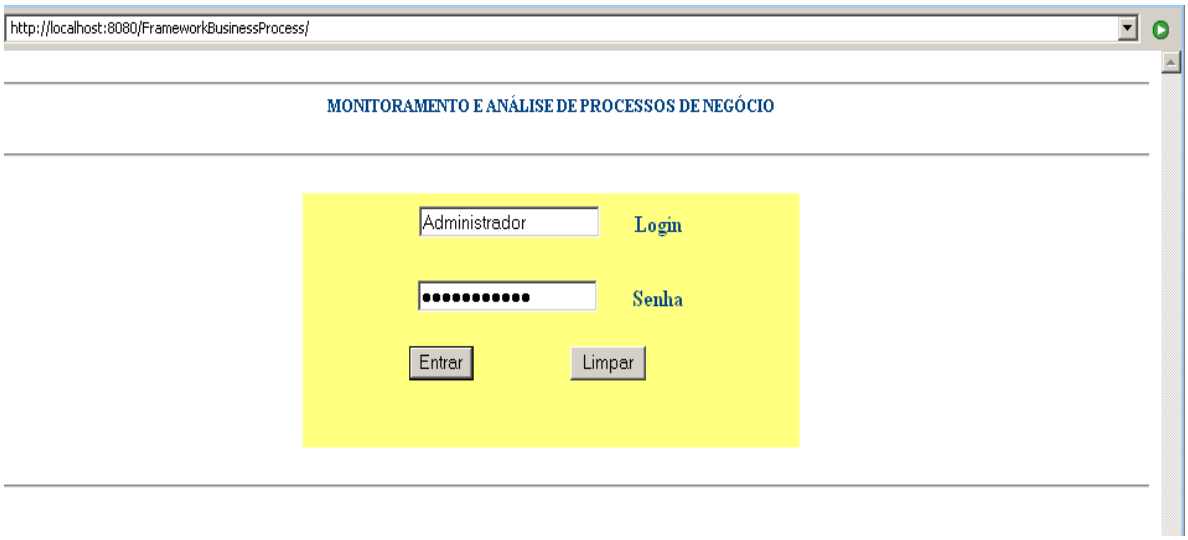

Figura 14. Tela de acesso ao sistema.

Após conectar-se ao sistema, uma tela contendo operações de negócio cadastradas é apresentada para o administrador para que possa efetuar as tarefas (Figura 15): (i) cadastro de operação de negócio; (ii) cadastro de processo de negócio; (iii) visualizar quais operações de negócio que foram executadas e (iv) visualizar os processos de negócio que foram executados.

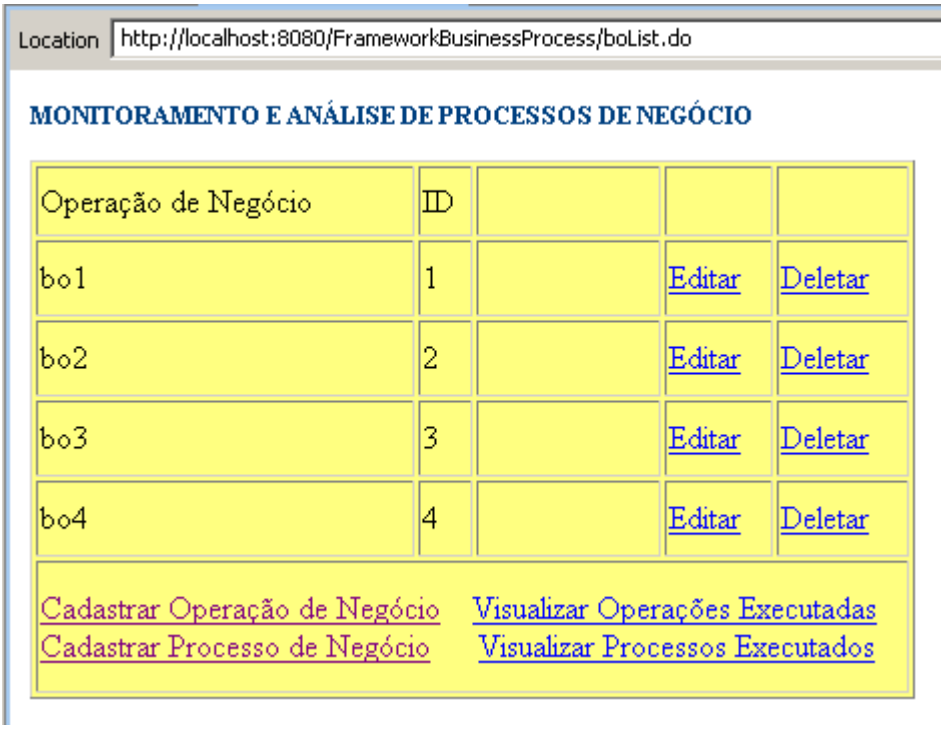

Figura 15. Tela após o *login* ao sistema.

A Figura 16 apresenta o formulário web de cadastro de uma operação de negócio. Para cadastrar um processo de negócio, o administrador tem que definir quais operações de negócio caracterizam um processo de negócio. A caracterização de um processo de negócio ocorre quando o administrador seleciona um conjunto de operações de negócio, agrupando-as para indicar que elas definem um processo de negócio. A Figura 17 apresenta o cadastro de um processo de negócio.

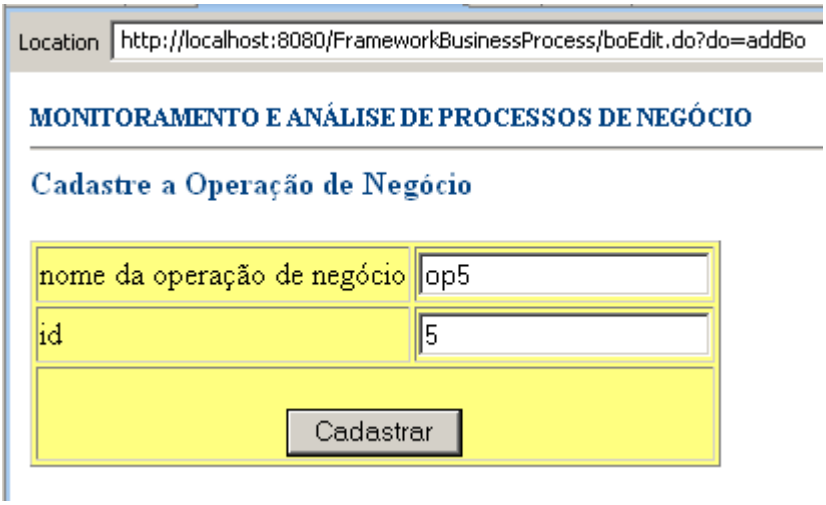

Figura 16. Tela de cadastro de operação de negócio.

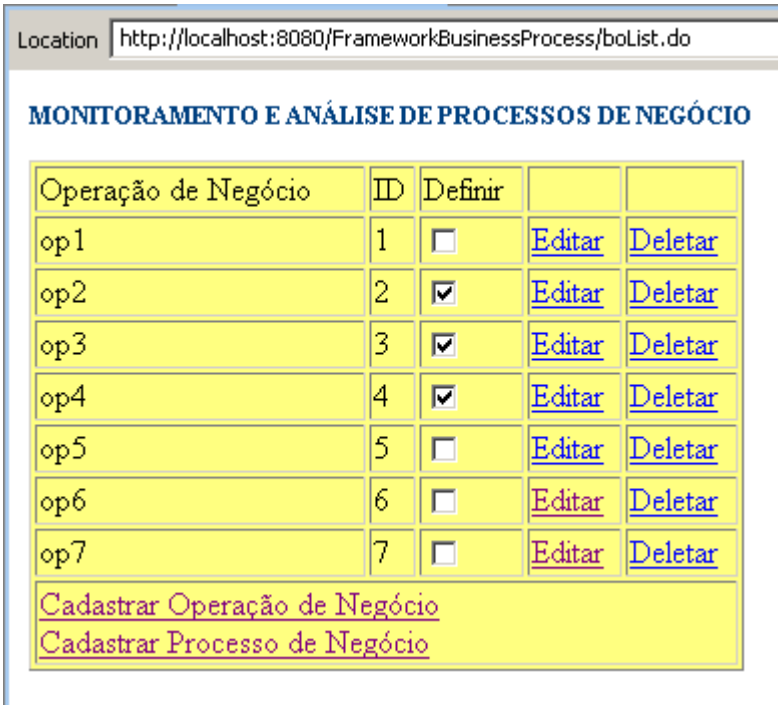

Figura 17. Tela de cadastro de processo de negócio.

4.1.6. Componente Reporter

O componente Reporter é responsável por apresentar relatório sobre quais processos e operações de negócio foram executados. Diferentes informações são apresentadas sobre os processos e operações de negócio executados, tais como, data e horário da execução, ordem de execução das operações de negócio, e nome do usuário responsável pela execução.

Além de apresentar quais operações de negócios foram executadas, é possível verificar quais processos de negócio foram executados. A Figura 18 apresenta a interface gráfica que apresenta quais processos de negócio foram executados.

Ambos componentes GUI e Reporter foram apenas prototipados neste trabalho. Uma implementação de referência apropriada deveria contemplar versões mais detalhadas de tais componentes.

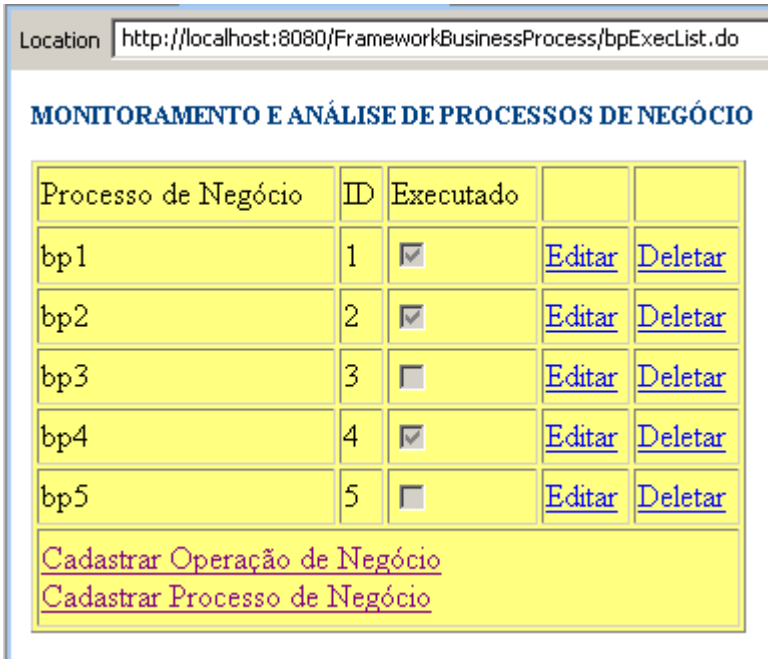

Figura 18. Tela de processos de negócio executados.