5
Estudos de Casos

Este capítulo detalha os estudos de caso realizados como parte da validação do framework proposto nesta dissertação de mestrado. O framework proposto foi instanciado para duas diferentes arquiteturas tradicionais usadas comumente em sistemas web.

5.1.
Aplicação Web Tradicional em Camadas

O primeiro estudo de caso envolveu a instanciação do framework para uma aplicação web tradicional estruturada em camadas, seguindo as diretrizes gerais do padrão arquitetural *Layer* [48]. A aplicação utilizada no nosso estudo de caso foi a aplicação exemplo desenvolvida como parte do curso de Projeto de Sistemas de Software do Departamento de Informática da PUC-Rio, de um sistema de matrícula escolar, denominado DAR (Departamento de Admissão e Registro).

O sistema DAR de matrícula escolar tem como objetivo principal possibilitar a matrícula de alunos universitários em disciplinas ofertadas em um dado semestre através da internet. O sistema é estruturado de acordo com o padrão arquitetural Layer [47] e refinado com diversos padrões de projeto [33, 47]. A Figura 19 apresenta um diagrama de classes parcial do sistema DAR. As classes são organizadas nas camadas de apresentação (GUI), negócio e dados. Interfaces são usadas para especificar os serviços que cada camada oferece.

A camada de apresentação agrega a classe `ServletWebMat`, cuja finalidade é receber requisições web e repassar tais requisições para classes estruturadas de acordo com os padrões de projeto *Front Controller* [47] e *Command* [33]. Cada classe do tipo comando da camada de apresentação realiza o tratamento de uma requisição específica do usuário, solicitando serviços da camada de negócio do sistema e retornando o resultado de tais requisições, por meio do uso da tecnologia Java Server Pages (JSP). A Figura 19 mostra três exemplos de classes

comando: `EfetivarMatriculaComando`, `ListarAlunosComando` e `ListarDepartamentosComando`. A finalidade da classe `ListarDepartamentosComando` é apresentar a lista de departamentos da universidade. A classe `EfetivarMatriculaComando` realiza a matrícula de um aluno em uma dada disciplina à sua escolha. A classe `ListarAlunosComando` apresenta a lista de alunos cadastrados no sistema WebMat.

A camada de negócio expõe seus serviços, por meio de duas interfaces, `ServicoAluno` e `ServicoCurso`. Essas interfaces são implementadas, respectivamente, pelas classes `ServicoAlunoImpl` e `ServicoCursoImpl`, representando instanciações concretas do padrão de projeto *Session Facade* [47]. Essas classes contêm as regras de negócio relacionadas às entidades Aluno e Curso, respectivamente. A classe ServicoAlunoImpl, por exemplo, é responsável por definir todas as regras de negócio relacionadas com a manipulação (inserção, atualização, remoção e consulta) de instâncias da classe `Aluno`. Finalmente, as classes `ServicoAlunoImpl` e `ServicoCursoImpl` também se responsabilizam pela recuperação e armazenamento das informações de instâncias de classes, usando os serviços da camada de dados.

A camada de dados expõe seus serviços usando diferentes interfaces. Cada uma dessas interfaces modulariza um conjunto de serviços de acesso aos dados de uma entidade do sistema. Por exemplo, a interface `AlunoDAO` especifica os serviços de acesso a dados relacionados a estudantes. Diferentes implementações podem ser oferecidas para tais interfaces de acordo com a tecnologia de acesso a dado que se deseja adotar. Na Figura 19 são apresentadas como implementações, as classes `AlunoDAOHibernate` e `CursoDAOHibernate`, que oferecem métodos concretos para os serviços de acesso a dados determinados por suas respectivas interfaces, usando o framework *Hibernate* [13]. A estruturação da camada de dados segue as diretrizes gerais do padrão de projeto *Data Access Object* (DAO) [47].

A instanciação do framework de análise e monitoramento de processos de negócios envolveu basicamente os seguintes passos: (i) a identificação e criação dos processos de negócio a serem monitorados pelo framework; (ii) a criação de um aspecto concreto para monitoramento das operações de negócio do sistema web em questão; (iii) a implementação (ou escolha) da política de comunicação

entre os componentes Monitor e Analyzer; e (iv) a escolha de umas das políticas de processamento concorrente do framework (*thread per request* e *thread pool*).

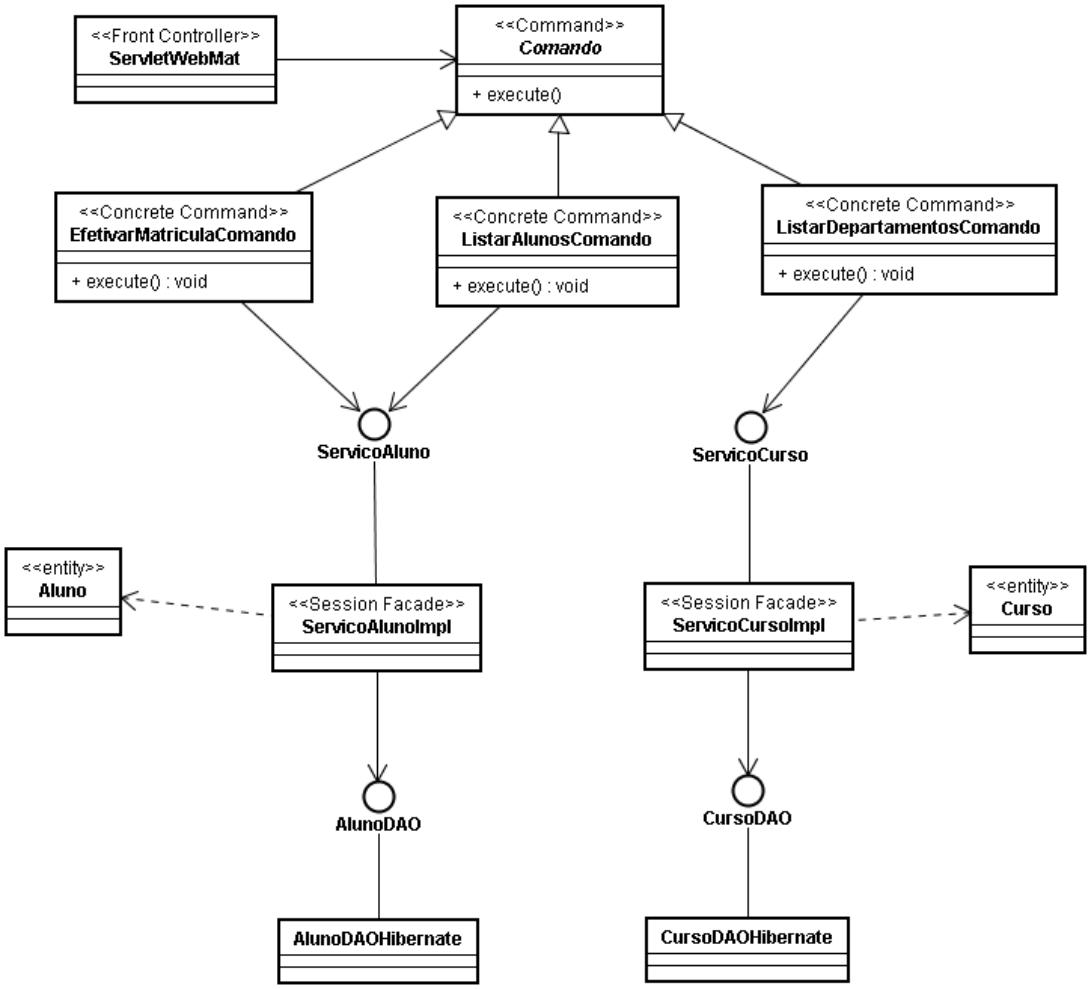

Figura 19. Diagrama de Classes parcial do Sistema DAR.

Os seguintes processos de negócio foram identificados baseado nas funcionalidades do sistema DAR: (i) Matricular Aluno – que envolve as operações de negócio "Selecionar Aluno", "Selecionar Disciplinas" e "Confirmar Matrícula"; (ii) Consultar Comprovante de Matrícula do Aluno – que envolve as seguintes operações de negócio "Selecionar Aluno" e "Relatório de Matrícula de Aluno"; (iii) Consultar Detalhes de Disciplina – que envolve as operações de negócio "Selecionar Disciplina" e "Relatório Detalhado de Disciplina".

O subaspecto `MonitorBusinessOperationDAR` foi criado como uma especialização do aspecto `MonitorBusinessOperation`. Ele é responsável por: (i) interceptar as classes *Command* da camada GUI, de forma a capturar a identificação do usuário responsável pela execução de uma dada operação de negócio, a partir do *HttpRequest*; e (ii) interceptar as chamadas aos métodos das classes *Service* da camada de negócio, de forma a identificar a operação de negócio sendo executada num dado. Essa identificação é realizada a partir da captura do nome do método de negócio sendo executado. Cada um dos processos de negócio cadastrados deve necessariamente ser associado a esses nomes de métodos de negócio que são capturados pelos aspectos

Para esse sistema em específico, foram escolhidos: (i) uma política de comunicação local, pois ambas as aplicações executam sobre o mesmo *container*; e (ii) uma política de tratamento concorrente que cria um novo *thread* para cada requisição.

5.2. Aplicação J2EE PetStore

O segundo estudo de caso que foi usado para instanciação do framework, foi a aplicação Java PetStore [50], desenvolvida pela Sun Microsystems, para ilustrar o uso das melhores práticas de desenvolvimento e projeto de aplicações J2EE. A aplicação PetStore implementa uma aplicação web padrão para compras online, especificamente de animais de estimação. Ela permite aos seus usuários: navegar no catálogo de animais disponível; selecionar os animais desejados para serem armazenados em um "carrinho" virtual de compras, criação e acesso autenticado de contas de usuários, assim como a compra de animais selecionados.

A Figura 20 apresenta a estrutura geral da aplicação Java PetStore. De forma idêntica a aplicação DAR apresentada anteriormente (Seção 5.1), ela também é estruturada seguindo uma estrutura em três camadas: interface gráfica, negócios e dados. Entretanto, diferentes tecnologias foram usadas ao longo de tais camadas para implementar seus serviços.

A camada de apresentação da aplicação PetStore agrega a classe `MainServlet`, cuja finalidade é receber requisições web e repassá-las para classes estruturadas responsáveis pelo seu tratamento. O framework WAF (Web

Application Framework) é responsável pela implementação de tal camada. Ele implementa uma série de serviços utilizados por aplicações Web, tais como, filtragem e repasse de requisições, geração de templates de visualização e controle do fluxo entre telas. Cada requisição web é transforma em um determinado evento pelo framework WAF e é, em seguida, repassado para ser tratado por determinado Enterprise Java Bean (EJB) da camada de negócio. A Figura 20 mostra a classe `ShoppingMainController` responsável pela manipulação de todas as requisições HTTP que chegam a aplicação PetStore. Ela se responsabiliza pelo: (i) mapeamento entre ações do usuário (exemplo: classe `CreateUserHTMLAction`) e eventos do sistema (exemplo: `CreateUserEvent`); e (ii) despacho de eventos requisitados para componentes EJB.

A camada de negócio é estruturada através de um conjunto de *Sessions Beans*, um tipo especial de EJB que permite pode ser acessado como um objeto distribuído e que também pode executar operações de negócio como transações. A Figura 20 apresenta dois *Sessions Beans* da aplicação PetStore, as classes `CatalogEJB` e `ShoppingCartEJB`, responsáveis, respectivamente, pela definição de serviços relacionados ao catálogo de animais e pela compra virtual. Como pode ser visto, na Figura 20, os Enterprise Sessions Beans estão conectados a camada de dados por meio do uso do padrão de projeto DAO (*Data Access Object*), de forma idêntica ao sistema DAR (Seção 5.1).

Durante a instanciação do framework de monitoramento e análise de processo de negócios para a aplicação PetStore, os seguintes processos de negócio foram definidos: (i) navegação por entre produtos – consiste no usuário executar diversas vezes operações de negócio relacionadas com busca e procura de produtos; (ii) compra de produto – consiste na seleção e compra de produtos (animais) dentro do sistema; (iii) compra invalidada – consiste na falha de um processo de compra, devido a algum problema (dados inválidos, impossibilidade de validação dos dados do cartão de credito), após a seleção correta de produtos.

O monitoramento das operações de negócio do sistema PetStore é realizado pelo aspecto `MonitorBusinessOperationPetStore` que foi codificado como uma especialização do aspecto `MonitorBusinessOperation`. Ele basicamente monitora a execução das seguintes classes: (i) subclasses da classe `HTMLAction` – para capturar as informações de usuário que estão submetendo requisições HTTP;

e (ii) subclasses da classe `EJBAction` – para identificar que uma dada operação de negócio foi executada.

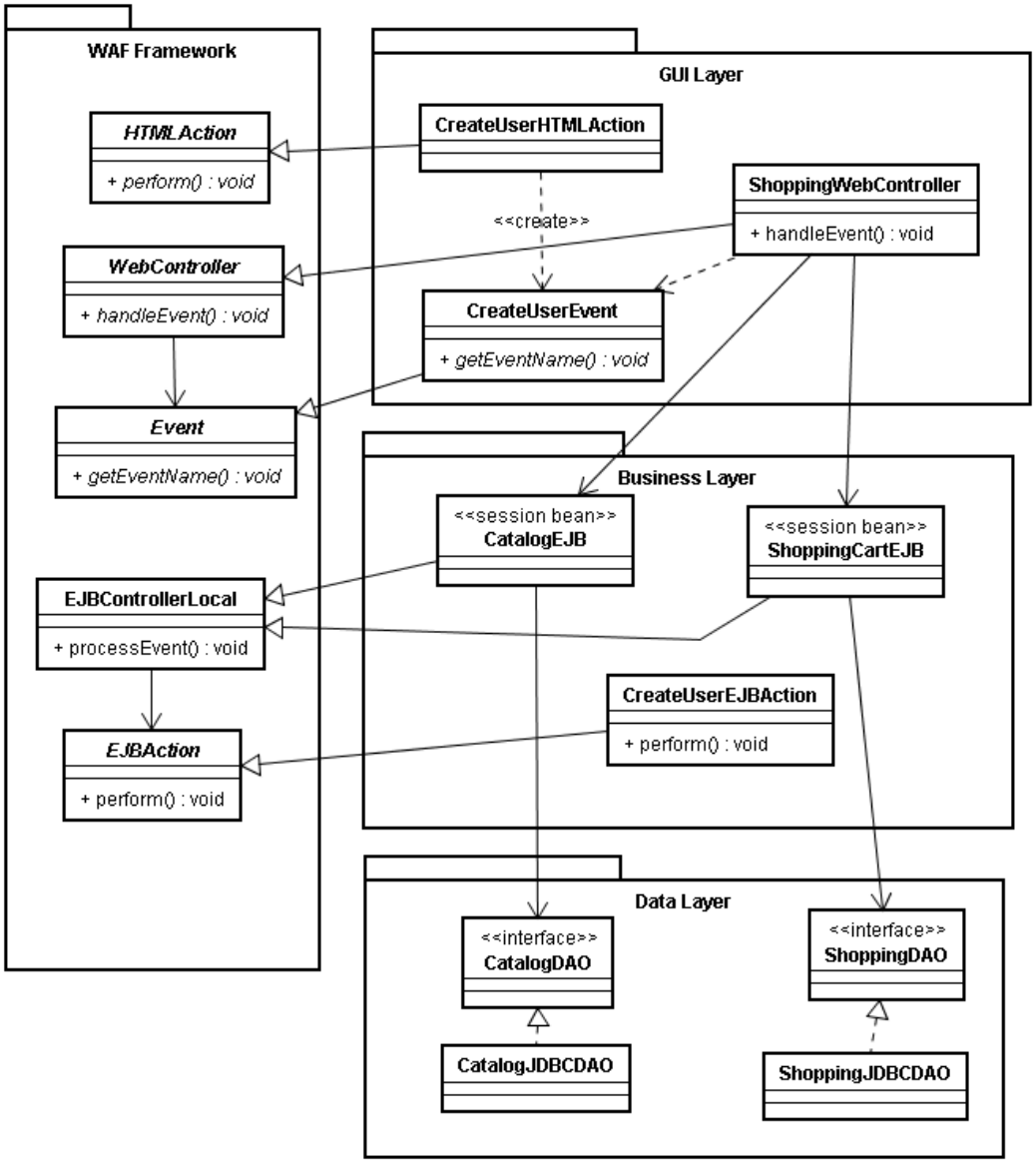

Figura 20. Diagrama de Classes parcial do Sistema PetStore.

5.3. Análise dos Estudos de Casos e Discussões

A instanciação do framework de monitoramento e análise de processos de negócio permitiu validar em aplicações web tradicionais, o correto projeto e funcionamento de seus pontos de extensão. A adoção do framework em duas

aplicações com processos de negócio bem distintos e implementados usando tecnologias Java completamente diferentes, permitiu avaliar a utilidade do framework em dois domínios de negócio e de tecnologia.

Para complementar os estudos de caso, o framework também foi instanciado para executar sobre diferentes aplicações geradas pela ferramenta AppFuse [51]. Essa ferramenta possibilita a geração de código de arquiteturas web padrão, oferecendo a opção de escolher entre diferentes tecnologias para a concretização de sua interface gráfica (Struts, Spring MVC, Tapestry, JSF). Durante esse experimento, diferentes aplicações foram geradas com a ferramenta AppFuse, cada um delas usando uma diferente tecnologia de interface gráfica. Em seguida, o framework foi instanciado para cada uma dessas aplicações através da definição de diferentes especializações do aspecto `MonitorBusinessOperation`, de forma a permitir a interceptação e captura dos dados de usuário de classes específicas da tecnologia sendo usada.

Apesar da versão atual do framework ser relativamente fácil de instanciar, foi identificado que o uso de uma ferramenta automática de instanciação poderia auxiliar tanto no processo de definição dos processos de negócio a serem monitorados, que poderiam por exemplo serem modelados com um diagrama de atividade, por exemplo, assim como nas políticas de monitoramento, processamento concorrente e comunicação distribuída a ser adotado pela aplicação. Estas últimas opções poderiam ser selecionadas automaticamente usando uma ferramenta de instanciação baseada no modelo de *features*, tal como o GenArch [52]. Também o uso de tecnologias padrões de componentes, tais como, Spring e OSGi podem ao nosso apoiar uma melhor estruturação do framework, facilitando inclusive sua instanciação automática por ferramentas existentes.

Outro ponto que foi observado durante o desenvolvimento e instanciação do framework foi a dificuldade de relacionar os processos e operações de negócio com determinadas ações (métodos) de classes da arquitetura do sistema web. Dois mecanismos podem ser usados com tal finalidade: (i) codificar anotações no código que indicam explicitamente o nome das operações de negócio a serem executadas com respectivos processos de negócio do qual fazem parte: e (ii) prover uma funcionalidade para caracterização automática de processos e operações de negócio, a partir da coleta inicial de supostos processos e operações de negócio executados no sistema.

Finalmente, outro aspecto que merece ser explorado é a análise do impacto das interceptações sobre a execução do sistema. Embora a configuração do framework para executar em uma máquina distinta da aplicação web sendo monitorada, possa auxiliar na diminuição do impacto de mudança. Ainda assim o framework pode trazer algum impacto para a execução da aplicação web sendo monitorada, durante a interrupção da execução das operações de negócio para coletar informação relativa a monitoramento. Para evitar que as operações de monitoramento interrompam a execução de operações de negócio, pode-se criar *threads* internas nos aspectos monitores, garantindo a execução da aplicação e do gerenciamento dos processos de negócio em linhas de controle (*threads*) distintas.