CONGRESSO CATARINENSE DE
CIÊNCIA DA COMPUTAÇÃO
2022. Ano 9, Vol. 9. ISSN: 2319-0655

RELATÓRIO TÉCNICO DESENVOLVIMENTO EM CAMADAS

Rafael José Camargo Bekhauser[1]
[1]Graduando Bacharelado em Ciência da Computação - IFC Rio do Sul
rafaelcamargo.inf@gmail.com

Abstract. *This project aims to present the implementation of a layered architecture of a simple system of student registration and grades, through the Java programming language. Through this approach, the application is divided into three layers: presentation, business and data access.*
Key-words: *Layered Architecture; Java; Programming.*

Resumo. *Este projeto tem como objetivo apresentar a implementação de uma arquitetura em camadas de um sistema simples de cadastro de alunos e notas, por meio da linguagem de programação Java. Através dessa abordagem, a aplicação é dividida em três camadas: apresentação, negócio e acesso a dados.*
Palavras-chave: *Arquitetura em Camadas; Java; Programação.*

1. Introdução

A arquitetura em camadas é um modelo de desenvolvimento de software que divide a aplicação em diferentes camadas, cada uma responsável por um conjunto específico de tarefas. É comumente utilizado em projetos de grande porte, onde a complexidade do sistema exige uma organização mais estruturada e modularizada. Essa abordagem oferece benefícios como a possibilidade de isolamento de funcionalidades, facilitando a manutenção e evolução do sistema, além de garantir a escalabilidade do sistema, sem afetar as outras camadas. Com isso, é possível ter um sistema mais robusto, flexível e adaptável às mudanças de demanda.

O presente projeto tem como objetivo a implementação de uma arquitetura em camadas em um sistema simples de cadastro de alunos e notas. A aplicação é dividida em três camadas: apresentação, negócio e acesso a dados. O sistema tem como finalidade cadastrar e listar os alunos de uma escola, permitindo a inclusão de nome, matrícula, curso, e-mail, telefone e bem como cadastrar e listar suas respectivas notas.

2. Modelagem do projeto

A fim de representar as classes, necessidades e relacionamentos do projeto de maneira visual, optamos pela utilização da modelagem por meio do diagrama de componentes e do diagrama de classes.

2.1. Diagrama de componentes

O diagrama de componentes auxilia na identificação das dependências entre os diferentes componentes e módulos, e na definição da arquitetura de software. Sendo especialmente aplicado em sistemas distribuídos, onde os componentes podem estar espalhados por diferentes servidores ou até mesmo em diferentes empresas.

No âmbito deste projeto, foi definido o seguinte contexto de implementação: uma aplicação de gestão de notas e alunos, com operações de inclusão, listagem e exclusão, que persiste as informações em um arquivo no formato textual. A modelagem da aplicação pode ser verificada na imagem abaixo.

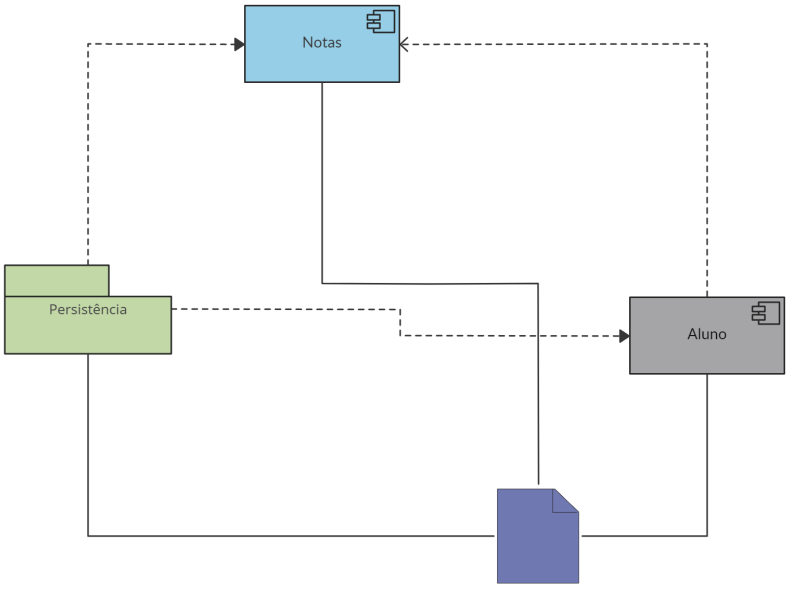

Figura 1 – Diagrama de componentes.

2.2. Diagrama de classe

O diagrama de classe é uma das principais ferramentas utilizadas na modelagem de sistemas orientados a objetos, permite representar de forma visual as classes que compõem um sistema, bem como seus atributos, métodos e relacionamentos. Essa representação ajuda a compreender melhor a estrutura do sistema, facilitando o processo de desenvolvimento e manutenção do código.

Partindo do contexto antes apresentado para a implentação, juntamente com a adição da modelagem em camadas, se faz necessário a definição de classes DTO, DAO e BO.

A classe DTO (Data Transfer Object) é responsável por transferir os dados entre as camadas da aplicação, contendo apenas atributos e métodos de acesso, sem lógica de negócio. É utilizada para reduzir o acoplamento entre as camadas da aplicação.

A classe DAO (Data Access Object) é responsável por realizar a comunicação com a camada de persistência de dados. Encapsulando a lógica de acesso aos dados e oferecendo métodos para realizar operações de listagem, adição e exclusão.

A classe BO (Business Object) é responsável pela lógica de negócio da aplicação. Implementando as regras de negócio da aplicação e utilizando as classes DAO para acessar e manipular os dados. A classe BO contém a lógica que implementa as funcionalidades da aplicação, como validação de dados, cálculos, etc.

Dessa forma, pode-se definir no diagrama de classes uma classe DTO, uma classe DAO e uma classe BO para cada objeto presente, ou seja, para os alunos e para as notas. Essas classes adicionais são necessárias para implementar a modelagem em camadas e garantir a separação de responsabilidades no sistema. Ademais, para fins de unificação e gestão das demais classes, inclui-se a classe Main, que atua como ponto de controle central da aplicação.

Figura 2 – Diagrama de classe.

3. Implementação

Conforme contexto apresentado anteriormente a aplicação é dividida em três camadas: apresentação, negócio e acesso a dados.

A camada de apresentação é a responsável por interagir com o usuário, exibindo as informações de forma amigável e coletando os dados inseridos. Para isso, foi criada uma interface simples em console, utilizando a linguagem de programação Java. O usuário pode cadastrar novos alunos, listar os alunos cadastrados e fazer a exclusão dos mesmos. Ao cadastrar um novo aluno, o sistema solicita o preenchimento do nome, matrícula, curso, e-mail e telefone. De forma análoga ao apresentado o mesmo ocorre no cadastro de notas.

A camada de negócio é responsável por realizar as regras de negócio, validar os dados recebidos, e chamar os serviços da camada de acesso a dados. Para isto foram criadas as classes AlunoBO e NotasBO, conforme exemplificação abaixo.

```java
public class NotasBO {

    public void inserir(NotasDTO nota) {
        NotasDAO.insereNotas(nota);
    }

    public void remover(String matricula) {
        NotasDAO.deletarNotas(matricula);
    }

    public NotasDTO buscar(String matricula) {
        return NotasDAO.buscarNotas(matricula);
    }

    public double media(String matricula) {
        NotasDTO notasDTO = NotasDAO.buscarNotas(matricula);
        return (notasDTO.getNota1() + notasDTO.getNota2() +
notasDTO.getNota3()) / 3;
    }
}
```

Figura 3 – Exemplo de código de classe BO.

A camada de acesso a dados é responsável por realizar as operações de persistência no banco de dados. Para isso, foram criadas classes DAO, que são responsáveis por realizar as operações de inclusão e consulta dos alunos e notas. A classe DAO utiliza a biblioteca *java.io*, mais especificamente as classes *FileWriter* e *PrintWriter*, para realizar a persistência em um arquivo de texto. Na classe DAO estão presentes métodos que realizam inserção, busca e exclusão dos dados, conforme exemplo abaixo.

```java
public static void insereAluno(AlunoDTO alunoDTO) {
    if (alunoDTO != null) {
        try {
            arquivo = new FileWriter(NOME_ARQUIVO, true);
            gravarArquivo = new PrintWriter(arquivo);
            gravarArquivo.println(
                    alunoDTO.getMatricula() + ";"
                    + alunoDTO.getNome() + ";"
                    + alunoDTO.getCurso() + ";"
                    + alunoDTO.getEmail() + ";"
                    + alunoDTO.getTelefone()
            );
            arquivo.close();
        } catch (Exception e) {
            System.out.println("Erro ao gravar no arquivo: " +
e.getMessage());
        }
    }
}
```

Figura 4 – Exemplo de método de inserção.

4. Conclusão

A implementação da arquitetura em camadas trouxe diversos benefícios para o sistema de cadastro de alunos e notas. A separação das camadas permitiu uma melhor organização e modularização do código, facilitando a manutenção e evolução do sistema. Além disso, a implementação da arquitetura em camadas tornou o sistema mais escalável, uma vez que as mudanças em uma camada não afetam as outras.

O sistema de cadastro de alunos e notas implementado é um exemplo simples de como a arquitetura em camadas pode ser aplicada em projetos de desenvolvimento de software.