Centro Federal de Educação Tecnológica Celso Suckow da Fonseca – *Campus* Maracanã

Danilo Paz Cardoso Lopes

Diana Rose Mesquita Lobo

Leonardo Izaú de Souza

Mariana Rocha Fortes da Silva

**Sistema de modelagem de diagrama de classes UML – Astah La Vista**

Rio de Janeiro

2019

Danilo Paz Cardoso Lopes

Diana Rose Mesquita Lobo

Leonardo Izaú de Souza

Mariana Rocha Fortes da Silva

**Sistema de modelagem de diagrama de classes UML – Astah La Vista**

Trabalho apresentado como forma de avaliação da disciplina de Arquitetura e Padrões de Software do curso Bacharelado em Ciência da Computação do Centro Federal de Educação Tecnológica Celso Suckow da Fonseca – Maracanã.

Professor: Diogo S. Mendonça

Rio de Janeiro

2019

**Sumário**

[**1.** **Introdução** 4](#_Toc25854350)

[**2.** **Estilos arquiteturais** 5](#_Toc25854351)

[**3.** **Usabilidade** 5](#_Toc25854352)

[**4.** **Design Patterns** 7](#_Toc25854353)

[**5.** **Referências Bibliográficas** 9](#_Toc25854354)

[**APÊNDICE A – Título do Apêndice** 10](#_Toc25854355)

# **Introdução**

Este trabalho trata-se da criação de uma arquitetura de sistema de modelagem de diagrama de classes UML. Nele é apresentado os diagramas de classe de projeto, diagrama de comunicação, como também quais estilos arquiteturais e *design patterns* utilizados. Este trabalho tem a finalidade de destacar as principais funcionalidades e mostrar o funcionamento do atributo de qualidade usabilidade (*undo* e *aggregate*).

# **Estilos arquiteturais**

Os estilos arquiteturais usados foram o de orientação a objetos e de camadas. O programa foi feito como conjunto de objetos, em que cada um tem seus atributos e métodos específicos. Pode-se observar no apêndice A, por meio dos diagramas de caso de uso, que foi usado o estilo MVC, que é *Model*, *View* e *Controller*, representando o estilo em camadas.

# **Usabilidade**

Foram usadas duas táticas de usabilidade: *undo* e *agreggate*. O *undo* foi feito de modo que sempre que o usuário for adicionar algum elemento ou relacionamento no diagrama, o programa primeiro atribui à “diagramaBackup” o diagrama chamando o método “atualizarDiagramaBackup()” e depois faz a adição no objeto “diagrama”, dessa forma, “diagramaBackup” sempre estará à um estado anterior do objeto “diagrama”. Quando se clica em desfazer, o programa agora atribui à “diagrama” o “diagramaBackup” chamando o método “atualizarDiagrama()”, carregando o estado anterior dele, como é mostrado no diagrama abaixo:

Diagrama de projeto:

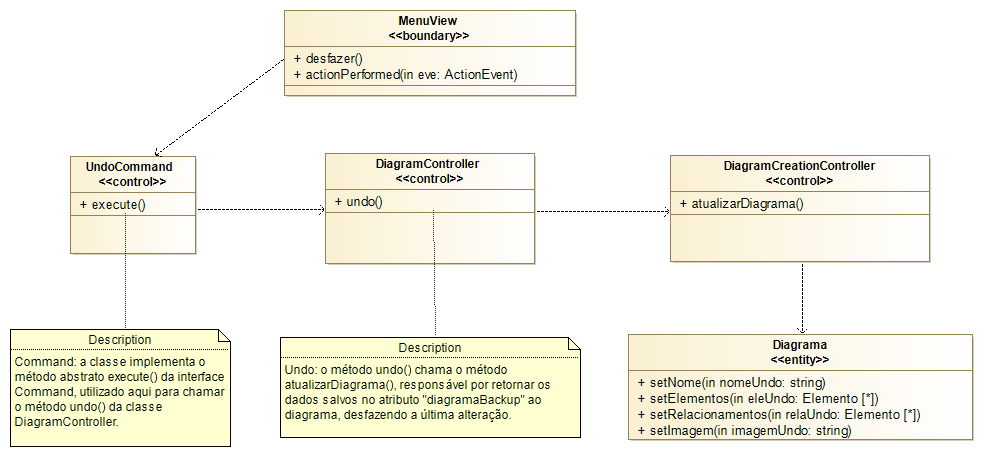
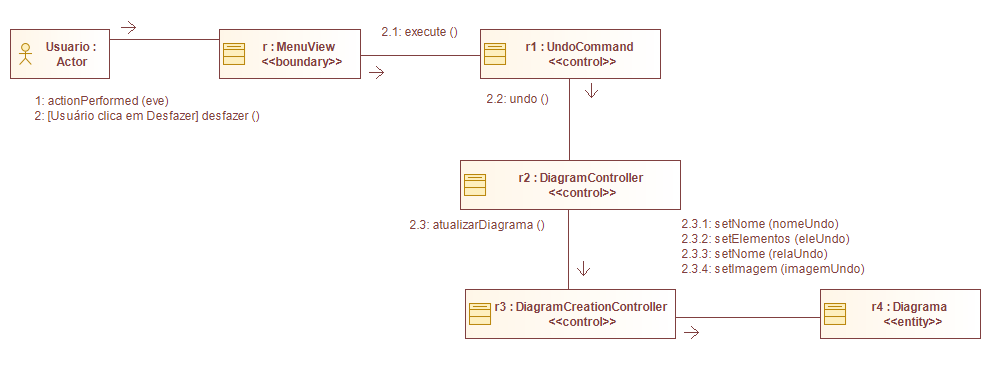


Figura 1. Diagrama de projeto do "Desfazer"

Diagrama de comunicação:

Figura 2. Diagrama de comunicação do "Desfazer"



A seguir pode-se observar a criação de um diagrama chamado “Diagrama 01” e duas classes (“classe” e “classe2”), num segundo momento é clicado no botão “Desfazer”, e logo em seguida a última classe criada é apagada:

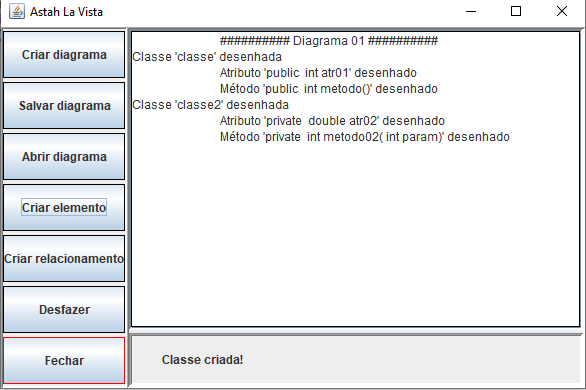


Figura 3.Duas classes foram criadas

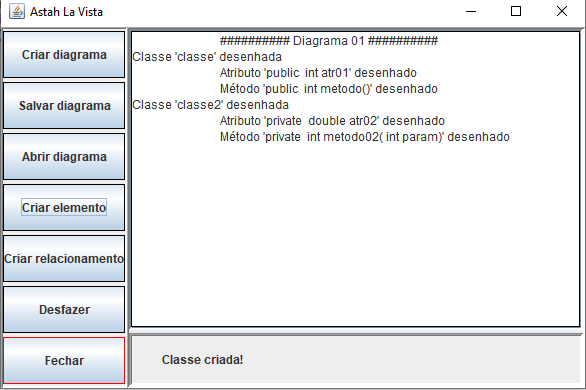
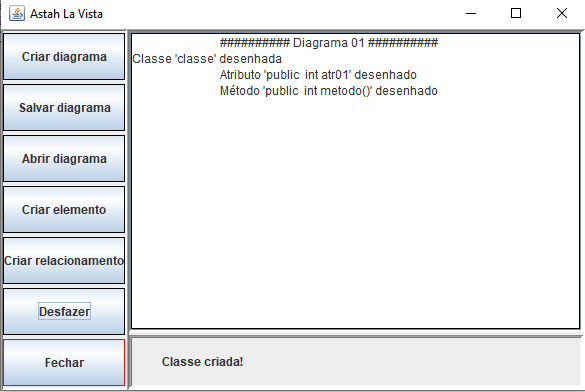


Figura 4. Opção de desfazer

****

Desfeito!

Figura 5. Após clicar em "Desfazer"

A outra tática de usabilidade é o *aggregate* usada para combinar elementos semelhantes. Ela foi realizada pegando vários elementos compostos em um elemento simples. Para facilitar sua utilização foi usado o *design pattern* “*composite*” e ela será explicada melhor no tópico seguinte.

1. **Design Patterns**

Um dos padrões utilizados foi o *command* que tem o método “*execute*” que isola os comandos da aplicação, ele tem a função de chamar os comandos de criar os elementos, salvar diagrama, fechar diagrama, desfazer e todos os demais comandos.

A seguir o diagrama que demostra isso:

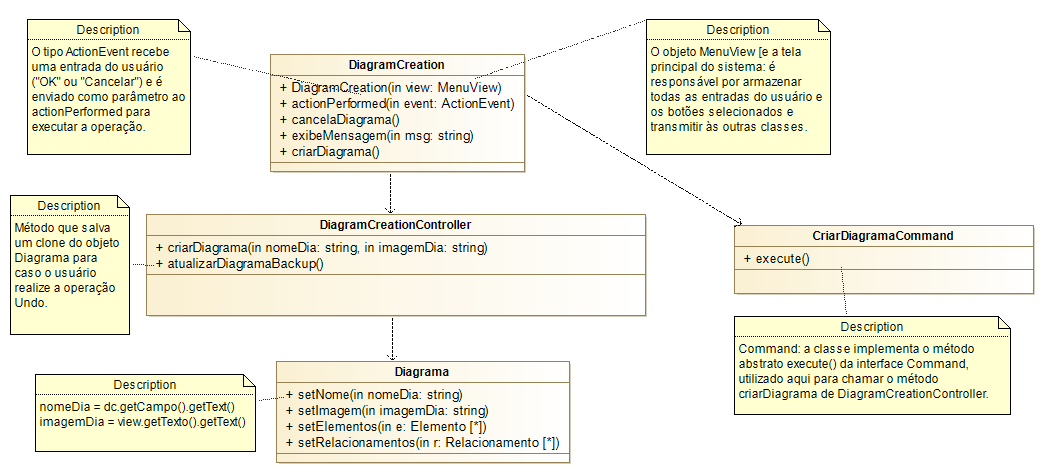


Figura 6. Diagrama de classe

O outro padrão usado foi o *composite* que pega um grupo de objetos que são tratados do mesmo modo, como se fosse uma instância do mesmo tipo de objeto. A ideia dele é “compor” objetos “primitivos” de maneira que represente hierarquias de partes inteiras, permitindo assim uma estrutura em árvore e para cada nó dela permita realizar uma tarefa. Abaixo uma demonstração de como foi implementado o *composite*:

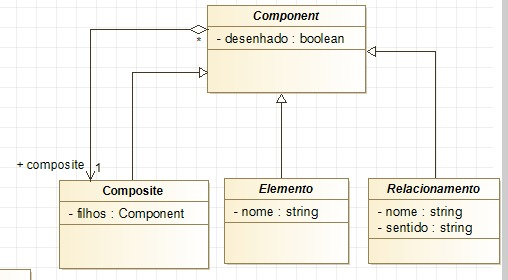


Figura 7. Composite

# **Referências Bibliográficas**

ROZANSKI, Nick; WOODS Eóin. **Software systems architecture: working with stakeholders using viewpoints and perspectives**. Ed 1. Addison-Wesley Professional, 2005

# **APÊNDICE A – Título do Apêndice**

Diagrama de classes de domínio

Caso de uso 1:

Diagrama de projeto

Diagrama de comunicação

Caso de uso 2:

Diagrama de projeto

Diagrama de comunicação

Caso de uso 3:

Diagrama de projeto

Diagrama de comunicação

Caso de uso 4:

Diagrama de projeto

Diagrama de comunicação