RELATÓRIO DE ORIENTAÇÃO A OBJETOS
ENTREGA TP3
PADRÃO MODEL-VIEW-CONTROLLER (MVC)

Fabrício Macedo de Queiroz 202046087

Arthur de Melo Viana 211029147

Eduardo Ferreira de Aquino 211030710

Uires Carlos de Oliveira 202043307

1 OBJETIVO

Nesta pesquisa, toma-se por finalidade a compreensão do comportamento do padrão *Model-View-Controller* (MVC), ou por outra, entender como as classes responsáveis pelos três principais aspectos se relacionam e se organizam entre si, além de sua estrutura como um todo. Por fim, consciente desse modelo, implementar-se-á em Java um programa que caracteriza um aluno da UnB, por meio dos seguintes dados de entrada: nome completo, curso, matrícula e e-mail.

2 INTRODUÇÃO

Com o desenvolvimento de um software em mente, deve-se antes organizar tal projeto por meio de uma estrutura de modelagem a fim de facilitar a visualização da composição do código, a manutenção e até mesmo o processo de desenvolvimento do programa. Sendo assim, tem-se que um dos modelos já conhecidos é o padrão de desenvolvimento MVC, e esse será utilizado no código desenvolvido. No entanto, é necessário ter o discernimento acerca das três categorias de classes do MVC: de domínio, de interface e de controle.

Portanto, instancia-se que as classes de domínio são classes que fazem parte do universo do problema, ou seja, representam conceitos do mundo real, como um trabalhador ou uma mercadoria. Além disso, ao fazer tais classes, desconsidera-se aspectos intrínsecos à solução computacional. Para compreender as classes de interface, é necessário entender que as informações trafegam pelos limites do sistema e são inseridas ou examinadas por agentes externos. Logo, as classes de interface são instanciadas nos extremos entre o sistema e os agentes e também realizam a comunicação entre eles. Por fim, avalia-se que as classes de controle se responsabilizam pela interação entre os outros dois tipos de classe, dessa forma, devem discernir os passos que devem ser realizados para a execução de um serviço no sistema. Ademais, também são utilizadas na administração de eventos na interface gráfica do Java.

3 DESENVOLVIMENTO

Num primeiro momento, a partir do uso do software *Eclipse IDE 2022 – 03*, dividiu-se o projeto em Java em três pacotes: model, view e controller. Portanto, fez-se uma classe AlunoUnB dentro do pacote model, a qual instanciava quatro variáveis de visibilidade privada que seriam utilizadas para caracterizar o aluno que seria cadastrado: nome, matrícula, curso e e-mail. Dada a necessidade da instanciação deste objeto AlunoUnB, implementou-se os métodos construtores-padrão e alternativos.

Sendo assim, com o intuito de gerenciar a interface do usuário para a aplicação, criou-se um novo arquivo no pacote view nomeado AlunoUnBView, que estende a classe JFrame do Java: a qual realiza o gerenciamento de aplicações gráficas. Desse modo, com o intuito de confeccionar a interface gráfica, empregou-se a ferramenta *WindowBuilder*. Em tal, organizou-se a exibição dos elementos visuais na janela: título, campos de texto, rótulos e botões. Em seguida, associou-se as variáveis de AlunoUnB às *Strings* inseridas nos campos de texto. O botão "Cadastrar" possui a função de cadastrar, ou seja, instanciar o objeto de tipo AlunoUnB a partir de um método da classe *control*. Por fim, o botão "Dados" explicita os atributos do objeto instanciado, para tal, necessitou-se de um método implementado na classe AlunoUnB — o qual foi nomeado "relatorio" —, nessa tinha como função imprimir os elementos do objeto.

Deste modo, por meio da classe *control*, onde portava o método "EnviarAluno" — responsável por encaminhar os parâmetros informados nos campos de texto da interface ao construtor alternativo da classe *model* —, havia o tratamento dos dados.

4 CONCLUSÃO

Isso posto, compreendida a natureza do padrão MVC, entende-se que a classe *view* não interfere diretamente na classe *model*, nem vice-versa, tal interação é intermediada pela classe *control*. Ademais, tendo em vista o desenvolvimento de uma aplicação que suporta apenas uma casta de cliente, toma-se por favorável o entrelace dos dados com a interface destinada à apresentação e controle dos dados.

Além disso, uma vicissitude possível seria a dependência do código em relação à interface, visto que, caso haja a demanda de alguma alteração em algum destes, ocorreria uma duplicação dos esforços de implementação e manutenção. Outro entrave encontrado foi a delegação de métodos às classes do MVC, ou por outra, não se sabia ao certo em qual classe se definiria métodos adversos.

Em contrapartida, o padrão MVC auxiliou na simplificação e divisão das tarefas, dessarte, tem-se um código mais compreensível e organizado. Para mais, a interface pode ser alterada sem afetar a classe domínio, ou seja, a classe domínio é independente da interface do sistema.

BIBLIOGRAFIA

ALMEIDA, Rodrigo. Model-View-Controller (MVC). DSC UFCG. Disponível em: <http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/arqu/mvc/mvc.htm>. Acesso em: 20/09/2022.

TOLEDO, Gilberto. Java #14 - MVC - Model, View, Controller - Parte 1. YouTube, 2016. Disponível em: <https://www.youtube.com/watch?v=3Rlm70kz0Kc>. Acesso em: 20/09/2022.

GIGLIO, Giuliano. MVC: Abordagem Teórico-Prática. Facom UFU, 2019. Disponível em:<https://www.facom.ufu.br/~ronaldooliveira/PDS-2019-2/Aula12-MVC.pdf>. Acesso em: 20/09/2022.