## Exercício de Projeto: Escola

Arquitetura e Programação de Software

Juno Takano

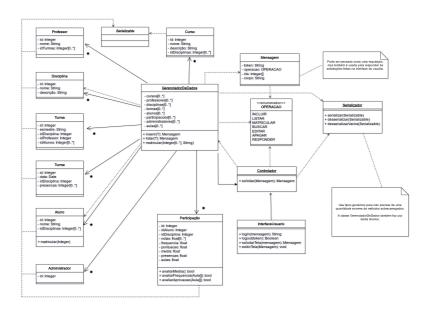


Figura 1: Diagrama de classes da arquitetura final.

A implementação final do trabalho seguiu o projeto inicial, que aproveitou o fato das estruturas originais já possuírem IDs numéricos. A escolha arquitetural buscou desacoplar as entidades e centralizar sua persistência na classe estática GerenciadorDeDados, concentrando assim as dependências em um lugar onde podem ser melhor aproveitadas.

Com exceção de apenas duas entidades, Aluno e Participação, todo o comportamento foi extraído para fora das camadas mais internas da aplicação. Um controlador foi adicionado, que através de técnicas de serialização e des-

Trabalho realizado para a disciplina de Arquitetura e Programação de Software (APSI5) do curso de *Análise e Desenvolvimento de Sistemas* do Instituto Federal de São Paulo, campus Jacareí.

serialização, providas por uma classe utilitária Serializador, permitem receber e enviar objetos para serem manipulados externamente à aplicação.

Estas solicitações são encapsuladas em instâncias da classe Mensagem, que possui atributos para transportar entidades serializadas como *strings* (corpo) e um atributo tipado através do *enum* OPERACAO, que permite ao controlador identificar se trata-se de uma operação de inclusão ou de listagem.

Quando o controlador recebe uma nova solicitação, ele utiliza esta enumeração para saber qual método estático do gerenciador de dados chamar: listar ou incluir.

O Gerenciador DeDados, assim como o Serializador, utiliza tipos genéricos para evitar a criação de muitos métodos sobrecarregados, cada um específico para cada classe.

Através desta técnica, foi possível realizar todas as operações de escrita e leitura, para todas as entidades, a partir de um único método. Isto foi especialmente interessante dado que a lógica aplicada é praticamente idêntica para todos eles.

Cabe ressaltar que o uso de tipos genéricos na classe Serializador não é totalmente seguro em termos de segurança de tipos, apesar de serem feitas três verificações aninhadas antes de realizar a conversão final, do tipo desconhecido para o HashMap correspondente ao tipo recebido como parâmetro.

O modelo buscou minimizar as dependências entre as classes e otimizar o fluxo de dados, com especial atenção aos pontos de contato cada vez mais externos até a interface de usuário, levando para fora da camada interna da aplicação a manipulação das informações.

O código acompanha um conjunto de testes automatizados, configurados para serem executados em sequência. Através do envio de mensagens pelo controlador, eles criam entidades no GerenciadorDeDados e verificam se as entidades retornadas ao listar o que foi armazenado correspondem à informação esperada.

Anexos a este trabalho estão os arquivos contendo o código fonte, também disponível <u>online</u>, e um relatório em HTML da execução do conjunto de testes.