**4. TÓPICOS ESPECIAIS DE PROGRAMAÇÃO ORIENTADA A OBJETOS**

A disciplina de Tópicos Especiais de Programação Orientada a Objetos (POO) tem como foco o estudo e a aplicação de conceitos avançados de orientação a objetos, com ênfase em técnicas e padrões que aprimoram a arquitetura e a manutenibilidade de sistemas. No contexto do projeto “Desenvolvimento de um Sistema Integrado para Gestão de Chamados e Suporte Técnico com Apoio de Inteligência Artificial (IA)”, esses conceitos foram aplicados para estruturar um sistema robusto, escalável e modular.

Segundo Wazlawick (2014), a orientação a objetos proporciona um modelo de desenvolvimento que se aproxima da forma como o ser humano percebe o mundo, permitindo representar entidades por meio de classes, atributos e métodos. Essa abordagem foi essencial na implementação de herança, polimorfismo e abstração no sistema proposto, garantindo melhor reutilização de código e menor acoplamento entre os módulos de software.

A herança foi empregada para estruturar classes genéricas, como “Usuário” e “Chamado”, que servem de base para subclasses especializadas, como “Técnico” e “Cliente”. Isso permitiu a criação de hierarquias lógicas e o reaproveitamento de atributos e métodos comuns, conforme recomendado por Bezerra (2014). O polimorfismo, por sua vez, foi utilizado para redefinir comportamentos em métodos herdados, possibilitando que diferentes tipos de usuários executassem ações específicas sobre os chamados, como registro, resposta e fechamento de tickets.

A abstração foi aplicada na criação de classes e interfaces que representam funcionalidades essenciais, mas ocultam a implementação detalhada. Essa técnica facilitou o desenvolvimento colaborativo, permitindo que diferentes membros da equipe trabalhassem de forma independente nos módulos do sistema. De acordo com Dennis, Wixom e Roth (2014), a abstração é fundamental para reduzir a complexidade e melhorar a clareza estrutural de sistemas orientados a objetos.

Foram implementados testes automatizados de unidade para validar a consistência das classes e métodos críticos, utilizando frameworks como MSTest e NUnit. Esses testes garantiram a confiabilidade das operações de abertura, atualização e encerramento de chamados, além de apoiar a integração contínua durante o ciclo de desenvolvimento. Sommerville (2019) destaca que os testes automatizados são uma prática essencial no contexto da engenharia de software moderna, pois aumentam a estabilidade do produto e reduzem o retrabalho.

Além disso, foram aplicados padrões de projeto (design patterns), como Singleton, para o controle centralizado de acesso à base de dados, e Factory Method, para a criação flexível de objetos relacionados aos chamados e notificações. Conforme Gamma et al. (1995), o uso de padrões de projeto padroniza soluções recorrentes e promove a reutilização de componentes, contribuindo para a qualidade e escalabilidade do sistema.

A aplicação dos conceitos abordados em Tópicos Especiais de POO foi decisiva para garantir a integridade e a flexibilidade do sistema de gestão de chamados. O uso combinado de herança, polimorfismo, abstração, testes automatizados e padrões de projeto resultou em uma arquitetura sólida, permitindo fácil manutenção, evolução do código e integração eficiente entre as plataformas Desktop, Web e Mobile.

**Referências**

BEZERRA, Eduardo. Princípios de Análise e Projeto de Sistemas com UML. 3. ed. Rio de Janeiro: LTC, 2014.

DENNIS, Alan; WIXOM, Barbara Haley; ROTH, Roberta M. Análise e Projeto de Sistemas. 5. ed. Rio de Janeiro: LTC, 2014.

GAMMA, Erich et al. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.

SOMMERVILLE, Ian. Engenharia de Software. 10. ed. São Paulo: Pearson, 2019.

WAZLAWICK, Raul Sidnei. Análise e Design Orientados a Objetos para Sistemas de Informação: Modelagem com UML, OCL e IFML. Rio de Janeiro: Elsevier, 2014.