**Engenharia de Software II**

Nesta fase do projeto, colocamos em prática os conceitos aprendidos na disciplina de Engenharia de Software II — é onde a teoria se transforma em soluções reais. Nosso foco esteve na modelagem detalhada dos casos de uso, no planejamento de testes e na definição clara dos comportamentos esperados para cada funcionalidade do sistema. Essa etapa foi fundamental para tornar o Clickdesk mais robusto, confiável e pronto para uso, tanto do ponto de vista técnico quanto da experiência do usuário.

Iniciamos com a construção dos modelos de casos de uso, representações gráficas que ilustram como os usuários interagem com as funcionalidades do sistema. Criamos um diagrama geral para oferecer uma visão macro da solução e, em seguida, detalhamos os processos por meio de diagramas específicos para cada cenário — como abrir um chamado, consultar a FAQ, responder ou encerrar um chamado, entre outros.

Cada caso de uso foi descrito de forma objetiva, incluindo os fluxos principais (o caminho padrão da ação), os fluxos alternativos (possíveis variações) e os fluxos de exceção (erros ou situações imprevistas, como uma falha na IA ou problemas de conexão). Também foram definidas as pré-condições (por exemplo, o usuário precisa estar logado para registrar um chamado) e as pós-condições (como o registro automático do chamado com seu respectivo número e status).

Utilizamos os relacionamentos da UML para estruturar os diagramas de forma mais clara e expressiva, seguindo orientações de autores como Larman (2007) e Booch et al. (2006). O relacionamento «include» foi aplicado para indicar ações obrigatórias vinculadas a outras, como no caso de “Atualizar Status” dentro de “Encerrar Chamado”. Já o «extend» foi útil para representar comportamentos opcionais, como o “Autoatendimento com IA”, que só é ativado em contextos específicos. Também empregamos generalizações para agrupar ações semelhantes, como “Responder Chamado por Técnico” e “Responder Chamado por IA”, que compartilham o comportamento de um caso de uso mais genérico, “Responder Chamado”.

Com a modelagem concluída, avançamos para a elaboração de uma planilha de testes, onde cada caso de teste foi documentado de forma estruturada: com um identificador, a funcionalidade avaliada, as entradas necessárias, os passos a seguir, o resultado esperado e o resultado obtido. Essa documentação foi essencial para validar as funcionalidades projetadas e garantir que os fluxos estivessem coerentes.

A realização dos testes permitiu identificar e corrigir falhas antes da implantação definitiva do sistema. Muitos ajustes só foram percebidos durante essa etapa, o que comprova sua importância. Além de garantir a estabilidade da aplicação, os testes contribuíram significativamente para melhorar a experiência do usuário final — alinhando-se com os princípios defendidos por Pressman (2016) e Sommerville (2019) sobre a importância da verificação e validação no ciclo de vida do software.

**Referências**

1. PRESSMAN, Roger S. *Engenharia de Software: uma abordagem profissional*. 7. ed. Porto Alegre: AMGH, 2016.
2. SOMMERVILLE, Ian. *Engenharia de Software*. 10. ed. São Paulo: Pearson, 2019.
3. LARMAN, Craig. *Utilizando UML e Padrões: uma introdução à análise e ao projeto orientado a objetos*. 3. ed. Porto Alegre: Bookman, 2007.
4. BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. *UML: Guia do Usuário*. Rio de Janeiro: Elsevier, 2006.
5. ALENCAR, Felipe; BARBOSA, Bruno. *Requisitos de Software: uma abordagem prática*. São Paulo: Novatec, 2018.