**ANÁLISE DE SISTEMAS ORIENTADA A OBJETOS**

Durante o desenvolvimento do sistema Clickdesk, usamos a análise orientada a objetos como base pra entender melhor as necessidades dos usuários e transformar isso em algo real e funcional. Esse tipo de abordagem ajuda bastante porque organiza tudo em torno de “objetos”, que representam elementos do sistema, como um chamado, um usuário, um técnico etc. Cada um desses objetos tem suas características (os famosos atributos) e comportamentos (os métodos), o que facilita tanto a visualização quanto a construção do sistema.

Pra colocar tudo isso no papel (ou na tela, né?), usamos os **diagramas UML**, que são ótimos pra visualizar a estrutura e o funcionamento do sistema. Eles ajudam a gente a comunicar melhor com a equipe e também a evitar confusões lá na frente. O primeiro que usamos foi o **Diagrama de Casos de Uso**, que basicamente mostra o que o usuário pode fazer no sistema. Por exemplo: abrir um chamado, responder, acompanhar o status ou encerrar o chamado. Isso já deu uma visão geral bem clara de como o sistema seria usado.

Depois, com o **Diagrama de Classe**, conseguimos estruturar melhor os dados e relações do sistema. Criamos classes como “Chamado”, “Usuário”, “Técnico”, “Categoria”, e mostramos como essas classes se conectam. Esse diagrama ajudou tanto na modelagem do banco de dados quanto na hora de codar.

Aí veio o **Diagrama de Sequência**, que foi importante pra entender a ordem das ações. Por exemplo: o cliente abre um chamado → a IA analisa → se resolver, manda a resposta; se não, encaminha pro técnico. Ver isso em sequência ajuda muito na hora de pensar em como programar isso tudo depois.

Pra fechar, também usamos o **Diagrama de Implantação**, que mostrou onde o sistema ia rodar. Tipo, em qual servidor, como os componentes se comunicam (frontend, backend, banco de dados) e como tudo estaria organizado na parte mais técnica mesmo. Isso é importante principalmente pra quem vai cuidar da infraestrutura.

Além dos diagramas, também levantamos todos os **requisitos do sistema**, pra garantir que o Clickdesk fosse útil de verdade. A gente dividiu isso em três partes:

Requisitos dos usuários: São as necessidades do dia a dia, como agilidade no atendimento, histórico de chamados, integração com chatbot, etc. É o que o usuário espera do sistema, de forma prática.

Requisitos funcionais: Aqui entra o que o sistema realmente faz. Tipo: cadastrar e responder chamados, categorizar com IA, consultar a base de conhecimento (FAQ), encaminhar pro técnico, gerar relatórios, etc.

Requisitos não-funcionais: São mais técnicos, mas essenciais. Por exemplo: o sistema precisa ser fácil de usar (usabilidade), responder rápido (desempenho), aguentar muitos acessos (capacidade), manter os dados seguros, e assim por diante.

No fim das contas, usar essa análise orientada a objetos junto com os diagramas e o levantamento de requisitos deixou o desenvolvimento mais organizado, e com menos chance de dar problema.

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. BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. *UML: Guia do Usuário*. Rio de Janeiro: Elsevier, 2006.
4. LARMAN, Craig. *Utilizando UML e Padrões: uma introdução à análise e ao projeto orientado a objetos*. 3. ed. Porto Alegre: Bookman, 2007.
5. ALENCAR, Felipe; BARBOSA, Bruno. *Requisitos de Software: uma abordagem prática*. São Paulo: Novatec, 2018.
6. IEEE. *IEEE Std 830-1998 - IEEE Recommended Practice for Software Requirements Specifications*. IEEE, 1998.