Kroma: Sistema de Gestão para Barbearia com Interface em Linguagem Natural

Ariel M. Rodrigues[1], **Frank W. C. de Oliveira**[1]

[1]Instituto Federal do Paraná (IFPR) – Campus Paranavaí
Paranavaí – PR – Brasil

20230010210@estudantes.ifpr.edu.br, frank.willian@ifpr.edu.br

Pequenos estabelecimentos do setor de beleza podem enfrentar dificuldades na organização das informações quando atividades como agendamentos, cadastros e controle de atendimentos são realizadas manualmente. No estabelecimento utilizado como estudo de caso, os agendamentos eram registrados em anotações e as informações dos clientes ficavam principalmente nas conversas realizadas por aplicativos de mensagens. Além de dificultar a consulta, esse processo não mantinha as informações centralizadas e podia resultar em registros duplicados.

Nesse contexto, foi desenvolvido o Kroma[1], um sistema de gestão para barbearia que reúne o cadastro de clientes, serviços, agendamentos e informações gerenciais em uma única aplicação. O sistema também possui uma interface em linguagem natural, por meio da qual operações podem ser solicitadas utilizando comandos de texto ou voz. O código dessa interface está disponível em repositório próprio.[2]

A interface utiliza um modelo de linguagem integrado ao sistema por meio do *Model Context Protocol* (MCP). Dessa forma, uma solicitação do usuário pode ser convertida em uma operação estruturada no sistema. O objetivo deste trabalho foi desenvolver a solução e verificar seu comportamento por meio de testes técnicos e de avaliações realizadas durante o uso do sistema.

O trabalho caracteriza-se como uma pesquisa aplicada conduzida por meio de um estudo de caso. O levantamento das necessidades foi realizado em uma barbearia real, a partir de entrevista com o responsável, observação da rotina e identificação das principais dificuldades encontradas no processo de trabalho.

Com base nesse levantamento, foram definidos os requisitos e as regras de negócio da aplicação. O desenvolvimento seguiu os princípios da *Clean Architecture*, mantendo as regras de negócio independentes das tecnologias utilizadas na infraestrutura [Martin 2018]. O *backend* foi desenvolvido com NestJS e TypeScript, utilizando MySQL e Prisma para persistência dos dados. O aplicativo mobile foi desenvolvido em React Native. A interface MCP disponibiliza ferramentas para consultas, cadastros, agendamentos e relatórios, mantendo as validações na própria aplicação [Anthropic 2024].

A avaliação compreendeu a verificação dos requisitos e testes automatizados, a observação dos processos manual e informatizado e a análise de comandos em linguagem natural, tanto durante o uso quanto em ensaio controlado. Os 27 requisitos foram implementados, e os 127 testes automatizados (108 unitários e 19 de integração) foram

[1]https://github.com/ArielMcR/Kroma
[2]https://github.com/ArielMcR/API_Kroma

aprovados. A cobertura de instruções nas camadas instrumentadas de domínio e casos de uso foi de 52,8% (383/725). Nos 24 arquivos classificados como código com regras de negócio, alcançou 78,3%, contra 4,8% nos 43 arquivos predominantemente de delegação. Destes, 41 sem cobertura concentram 239 instruções, reduzindo a média global.

As regras de dias e horários de agendamento (RN01, RN02 e RN09), prevenção de conflitos (RN04) e proteção do último administrador (RN22) tiveram 100% de cobertura de instruções e ramos nos componentes examinados. Por outro lado, a troca de senha (RF22), a conclusão de atendimento (RF11), a emissão do *token* (RF18) e a validação do segredo JWT não foram cobertas. O processamento de comandos apresentou 54,2% de cobertura de ramos: a geração de relatórios foi exercitada no ensaio controlado, mas a execução por voz não foi avaliada por nenhuma das duas estratégias. Portanto, a cobertura agregada não substitui a análise das regras críticas sem testes.

O processo manual foi observado de 7 a 11 de setembro de 2026, e o uso do aplicativo, de 14 a 18 de setembro de 2026, em intervalos de cinco dias cada. O tempo de realização de um agendamento passou de 18,87 para 16,62 segundos (redução de 11,9%). Como houve uma única rodada de cronometragem, a comparação é exploratória [Wohlin et al. 2012]. No período manual ocorreram dois retrabalhos por alteração do serviço de agendamentos; no período com aplicativo não foram observados casos semelhantes, conflitos de horário ou perda de dados. A diferença não permite concluir eliminação de retrabalho, pois o volume de atendimentos não foi controlado.

No uso da interface em linguagem natural, 36 de 50 comandos tiveram sucesso (72,0%). Dez falharam por ausência de resposta do serviço externo; entre as 40 execuções que obtiveram resposta, 36 tiveram sucesso (90,0%). Esses percentuais descrevem condições distintas, não medidas independentes de acurácia do modelo.

A avaliação controlada incluiu 43 cenários repetidos três vezes (129 execuções iniciais). As 126 execuções de 42 cenários compuseram a base de análise; as três restantes examinaram uma limitação das funções disponíveis, e não a interpretação dos comandos. A Tabela 1 resume os desfechos, em categorias mutuamente exclusivas.

Tabela 1. Desfechos da avaliação controlada (126 execuções)

Resultado	Qtd.	%
Comportamento funcional esperado	101	80,2
Erro na identificação da operação	3	2,4
Divergência na interpretação de parâmetros	3	2,4
Argumento malformado rejeitado	2	1,6
Falha externa antes da chamada de função	17	13,5
Total	126	100,0

As 17 falhas externas (13,5%) ocorreram antes de qualquer chamada de função. Entre as 109 execuções restantes, 101 apresentaram comportamento funcional esperado (92,7%). Os três erros de identificação corresponderam à criação de um agendamento sem que o serviço obrigatório fosse informado: o modelo presumiu esse dado em vez de solicitá-lo. As três divergências de parâmetros envolveram a expressão "semana passada", cuja interpretação foi plausível, mas diferente dos critérios de aceitação predefinidos. Em outros dois casos, datas com pontuação indevida foram corretamente rejeitadas pela aplicação. Nas 21 execuções envolvendo violações de regras de negócio, houve 19 rejeições esperadas e duas falhas externas, sem persistência de registros. Portanto, rejeitar corretamente um horário ocupado não foi considerado erro de interpretação. Essa distinção separa falhas de comando, validações corretas e indisponibilidade da infraestrutura.

Em 13 execuções (10,3%), a segunda chamada ao modelo, destinada a redigir a resposta, falhou por *timeout* (dez casos) ou HTTP 503 (três casos). Essa degradação é transversal: ocorreu em 12 execuções funcionalmente corretas e em uma com identificação incorreta da operação. Em sete casos houve persistência de dados sem confirmação adequada em linguagem natural, revelando cumprimento parcial do RF29. Assim, apenas 89 das 126 execuções (70,6%) combinaram comportamento esperado e resposta completa. Foram feitas ainda 17 reexecuções para investigar falhas (14 sucessos e três novas falhas), excluídas das taxas; com elas, foram gerados 146 registros.

O Kroma centralizou informações antes dispersas e permitiu executar operações por linguagem natural, mas o usuário considerou os formulários mais práticos para agendamentos. Os resultados, obtidos com um usuário, um modelo gratuito e observação limitada, não permitem generalizações. Como continuidade, propõem-se testes das regras ainda descobertas, avaliação do canal de voz, períodos maiores de uso e comparação entre modelos, com atenção à confiabilidade da confirmação das operações.

Referências

Anthropic (2024). Introducing the Model Context Protocol. Disponivel em: `https://www.anthropic.com/news/model-context-protocol`. Publicado em: 25 nov. 2024. Acesso em: 20 out. 2026.

Martin, R. C. (2018). *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Prentice Hall.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in Software Engineering*. Springer, Berlin, Heidelberg.