RIP: Requerimentos Ipanguaçu – Automatizando Resolução de Chamados Acadêmicos com Inteligência Artificial

José Cláudio Medeiros de Lima
Instituto Federal do Rio Grande do Norte - IFRN
claudio.medeiros@ifrn.edu.br

Gelson Iezzi de Medeiros Garcia
Universidade Federal Rural do Semi-Árido - UFERSA
gelson.garcia@ufersa.edu.br

RESUMO

A gestão de requerimentos estudantis no Instituto Federal do Rio Grande do Norte (IFRN) é um processo burocrático que demanda tempo e recursos humanos. O projeto RIP: Requerimentos Ipanguaçu busca automatizar a análise e classificação desses requerimentos utilizando modelos de linguagem natural (LLMs). A solução proposta integra-se ao Sistema Unificado de Administração Pública (SUAP), permitindo que servidores aprovem solicitações de forma rápida e eficiente. O aplicativo é desenvolvido inteiramente em Flutter, utilizando Dart tanto para a interface quanto para o processamento interno. Este artigo apresenta o processo de desenvolvimento, os métodos utilizados, a implementação e a validação do sistema.

Palavras-chave

Automatização; Processamento de Linguagem Natural; IFRN; Requerimentos Estudantis; LLM; SUAP.

Keywords

Automation; Natural Language Processing; IFRN; Student Requests; LLM.

1. INTRODUÇÃO

A análise manual de requerimentos estudantis pode gerar lentidão no atendimento das solicitações, impactando negativamente a experiência dos alunos e a eficiência dos servidores acadêmicos. Segundo Chiavenato [5], a automatização de processos administrativos melhora a eficiência operacional e reduz a carga de trabalho repetitiva, permitindo maior foco em atividades analíticas e estratégicas.

No contexto acadêmico, a agilidade na resposta a solicitações estudantis é essencial para garantir uma gestão educacional eficiente e alinhada com os princípios da administração pública. O Instituto Federal do Rio Grande do Norte (IFRN), conforme sua Organização Didática [16], estrutura-se de forma a garantir a transparência e a eficiência no atendimento aos discentes. Essa perspectiva dialoga com os princípios educacionais de Paulo Freire [8], que enfatiza a necessidade de uma educação libertadora e acessível para todos, promovendo autonomia e participação ativa dos estudantes no processo educativo. Dentre seus princípios, destacam-se a gestão democrática, a inclusão social e a integração entre ensino, pesquisa e extensão, aspectos fundamentais para a modernização de seus processos administrativos.

O IFRN tem como objetivo proporcionar uma educação profissional e tecnológica que atenda às demandas da sociedade, promovendo a formação de cidadãos qualificados para atuação no mercado de trabalho. Dentro dessa perspectiva, a informatização e a automação dos procedimentos acadêmicos, como a gestão de requerimentos, tornam-se fundamentais para assegurar maior eficiência e acessibilidade aos serviços prestados pela instituição.

A proposta do RIP: Requerimentos Ipanguaçu é utilizar Large Language Models (LLMs) para classificar e extrair informações dos chamados abertos pelos estudantes no SUAP. Dessa forma, os servidores podem revisar e aprovar os requerimentos com um clique, otimizando o fluxo de trabalho e garantindo maior precisão na classificação das solicitações.

2. TRABALHOS RELACIONADOS

Para fundamentar o desenvolvimento do RIP: Requerimentos Ipanguaçu, realizamos uma pesquisa bibliográfica para identificar trabalhos acadêmicos que abordam soluções similares de automação na análise de requerimentos estudantis e o uso de modelos de linguagem natural no contexto educacional. O processo de busca seguiu os seguintes passos:

- **Definição de palavras-chave**: Utilizamos termos como "automatização de análise de requerimentos estudantis", "processamento de linguagem natural na educação" e "inteligência artificial na gestão acadêmica".
- **Seleção de bases de dados**: Pesquisamos nas plataformas Google Scholar e Portal de Periódicos CAPES.
- **Critérios de inclusão e exclusão**: Consideramos artigos publicados nos últimos cinco anos, escritos em português ou inglês, que apresentassem soluções aplicáveis ao contexto acadêmico brasileiro.
- **Análise dos trabalhos selecionados**: Realizamos uma leitura crítica para identificar abordagens e tecnologias semelhantes às utilizadas no RIP.

Não foram encontrados artigos com a mesma proposta de automação de requerimentos estudantis do RIP. No entanto, para aumentar o embasamento teórico, dividimos O RIP em ideias menores e encontramos os seguintes artigos relevantes:

2.1 Aplicações do processamento de linguagem natural no ambiente educacional: uma revisão sistemática da literatura [10]

Os autores dissertam sobre os recentes avanços da inteligência artificial generativa no contexto escolar, seja gerando textos e agilizando as publicações, seja usando-a para analisar vastos conteúdos e resumi-los de forma rápida e eficiente, visando agilidade no consumo de grandes quantidades de informação.

Este trabalho dá base ao RIP no uso da IA generativa no contexto dos requerimentos estudantis, considerando que o contexto de RIP

é o de análise de grandes quantidades de textos em linguagem natural existentes nos chamados da Central de Serviços do SUAP e a consequente automatização das tarefas com base neles.

2.2 A inteligência artificial na automação dos processos negociais e os limites éticos de sua utilização [11]

Os autores investigam a utilização de inteligência artificial no contexto da tomada de decisões gerenciais, além dos limites éticos que envolvem o tratamento de dados sensíveis e transparência.

Este trabalho fornece uma base teórica que ajudará o RIP na análise de conteúdo privativo de alunos no contexto dos requerimentos escolares que, por vezes, possuem informações pessoais.

2.3 Chatbot para apoio à Divisão Académica [7]

O autor detalha a criação de um chatbot alimentado por inteligência artificial para retirada de dúvidas e prestação de suporte aos alunos do Instituto de Engenharia do Porto. Seu trabalho aborda a dificuldade enfrentada pelos setores acadêmicos frente à demanda gerada pela quantidade de alunos em contraste aos poucos funcionários daquela escola, e como a implementação de uma IA que tira dúvidas sobre procedimentos e regras acadêmicas poderia melhorar o relacionamento escola-aluno.

Este artigo segue o mesmo objetivo do RIP, onde automatizar a geração de conteúdo auxilia o sistema a suprir a falta de profissionais dedicados à retirada de dúvidas acadêmicas dos alunos do Instituto Federal do RN.

3. PROCEDIMENTOS E MÉTODOS

O desenvolvimento do RIP: Requerimentos Ipanguaçu seguiu um planejamento estruturado, fundamentado em princípios de metodologias ágeis para garantir flexibilidade e eficiência ao longo do processo. Foram estabelecidas três fases principais: concepção, implementação e validação.

Após a análise do cenário, decidiu-se utilizar princípios de métodos ágeis, que consistem em envolver o cliente no processo de desenvolvimento, permitindo que ele forneça e priorize novos requisitos e avalie as iterações do sistema (SOMMERVILLE, 2018) [14]

Além disso, metodologias ágeis incentivam entregas incrementais, onde os usuários podem especificar os requisitos a serem incluídos em cada incremento, desenvolver com foco nas pessoas, reconhecendo suas habilidades e incentivando uma maneira própria de trabalho, bem como aceitar mudanças nos requisitos ao longo do processo de desenvolvimento. Dessa forma, investiu-se na criação de um sistema que pudesse acomodar mudanças, manter a simplicidade no processo de desenvolvimento e eliminar complexidades desnecessárias.

As justificativas para a escolha de princípios ágeis incluíram o pequeno porte do sistema e o alto nível de engajamento dos servidores da diretoria acadêmica, que contribuíram ativamente com o desenvolvimento. Assim, cada iteração do projeto envolveu entrevistas com os usuários finais, a criação de protótipos para embasar as implementações e a validação contínua dos requisitos levantados.

O processo de desenvolvimento teve início com uma avaliação de sistemas similares existentes no mercado, bem como das tecnologias e ferramentas disponíveis. Em seguida, foram definidos ciclos de atividades que incluíram entrevistas com *stakeholders* para delimitar funcionalidades prioritárias por meio de um protótipo, que pode ser uma implementação ou uma representação concreta, ainda que não contemple todas as particularidades do design de um sistema (BENYON, 2013) [2]. Dessa forma, foram desenvolvidos protótipos de baixa fidelidade baseados em documentos e processos previamente utilizados, permitindo que as funcionalidades fossem gradualmente aprimoradas e validadas pelos usuários.

Durante a fase de implementação, cada funcionalidade foi desenvolvida e submetida a testes pelos próprios usuários. Caso a implementação atendesse aos requisitos levantados, um novo ciclo era iniciado; caso contrário, as etapas anteriores eram revisadas para corrigir inconsistências e melhorar a experiência do usuário.

Para o gerenciamento do projeto, utilizou-se o Github Issues, permitindo a implementação da metodologia Scrum, que fornece um processo flexível para o desenvolvimento de sistemas, dividindo o progresso em sprints, ou seja, ciclos de desenvolvimento curtos onde cada iteração resulta em uma nova funcionalidade testada (SCHWABER, 2004) [13]. Esse processo garantiu que cada parte do sistema fosse validada antes de avançar para a próxima fase, promovendo um desenvolvimento mais ágil e eficiente.

A fase final do projeto consistiu na validação do sistema, realizada por meio de testes em ambiente real com servidores acadêmicos. Foram aplicados questionários de usabilidade e conduzidos testes de desempenho para avaliar a eficácia da ferramenta. Caso as implementações não atendessem às expectativas dos usuários, ajustes eram realizados antes da liberação final. Esse processo iterativo garantiu que o RIP estivesse alinhado com as necessidades institucionais, oferecendo uma solução eficiente para a gestão de requerimentos acadêmicos no IFRN.

3.1 Concepção

O projeto iniciou-se com a análise do fluxo atual de processamento de requerimentos estudantis no IFRN. Foram realizadas entrevistas com cinco servidores de secretarias acadêmicas para entender os desafios enfrentados na gestão manual dos pedidos. Com base nessas informações, foram definidos os requisitos do sistema e elaborado um protótipo inicial da interface utilizando a ferramenta Figma [17].

3.2 Implementação

- **A implementação do RIP**: Requerimentos Ipanguaçu foi realizada em ciclos iterativos, permitindo ajustes contínuos conforme o feedback dos usuários e garantindo maior eficiência no desenvolvimento. A seguir, detalhamos cada uma das etapas do processo:
- **Desenvolvimento do Aplicativo**: O aplicativo foi desenvolvido inteiramente em Flutter [18], utilizando Dart [19] para gerenciar tanto a interface gráfica quanto o processamento interno dos dados. Foram criados componentes reutilizáveis para garantir modularidade, além de adotar boas práticas de design responsivo, proporcionando uma experiência fluida para os servidores acadêmicos.
- **Integração com LLM**: A comunicação entre o aplicativo e o modelo de linguagem natural foi implementada utilizando APIs especializadas em Processamento de Linguagem Natural (PLN).

Essa integração permite que o sistema analise o conteúdo dos requerimentos e os categorize de forma automática, reduzindo significativamente o tempo necessário para a triagem manual dos chamados.

- **Testes Internos:** Para garantir a estabilidade do sistema, foram conduzidos testes internos em cada iteração do desenvolvimento. Esses testes incluíram:
  - **Testes unitários**: Para verificar o correto funcionamento dos componentes individuais do aplicativo e dos objetos de domínio.
  - **Testes de integração**: Para validar a comunicação entre os diferentes módulos do sistema com as interfaces de integração com o SUAP.
  - **Testes de usabilidade**: Para avaliar a experiência do usuário e identificar melhorias na interface e no fluxo de uso.

A implementação seguiu um modelo iterativo baseado em metodologias ágeis e validações diárias, conforme novas situações eram apresentadas no trabalho.

3.3 Validação

A validação do sistema foi realizada com a participação de cinco servidores acadêmicos do IFRN, que utilizaram o RIP em ambiente de testes. O processo de validação foi dividido em três etapas principais:

3.3.1 Aplicação de Questionário TAM

Para avaliar a aceitação e usabilidade do sistema, foi aplicado um questionário baseado no modelo TAM (Technology Acceptance Model). Este modelo objetiva medir a percepção de utilidade e a facilidade uso, os quais influenciam diretamente a intenção de uso e a aceitação final de um sistema (CRUZ, MOURÃO, BERNARDINI, VITERBO, & TREVISAN, 2022) [6].

O questionário continha dez perguntas, listadas a seguir:

P1: O sistema facilita a análise de requerimentos acadêmicos?

P2: A interface do sistema é intuitiva e fácil de usar?

P3: O tempo de resposta para análise dos requerimentos foi reduzido?

P4: Você se sente confortável em utilizar o sistema regularmente?

P5: O sistema melhora a organização e o acompanhamento dos requerimentos?

P6: O aplicativo apresenta desempenho adequado sem lentidões?

P7: A categorização automática dos requerimentos foi precisa?

P8: O sistema atende às principais necessidades dos servidores acadêmicos?

P9: O design visual e a disposição das informações são satisfatórios?

P10: Você recomendaria o uso do sistema para outros servidores?

Cada pergunta foi avaliada pelos participantes em uma escala de 1 (Muito Ruim) a 5 (Muito Bom). A tabela abaixo apresenta as respostas individuais e a média geral:

Tabela 1 - Notas

Participante	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	Média
Servidor 1	4	5	4	4	5	4	5	4	4	5	**4,4**
Servidor 2	5	4	5	4	4	5	4	5	4	4	**4,4**
Servidor 3	4	5	4	5	5	4	5	4	5	4	**4,6**
Servidor 4	5	4	5	5	4	5	4	5	4	5	**4,7**
Servidor 5	4	5	4	4	5	4	5	4	5	4	**4,6**
Média Geral	**4,4**	**4,6**	**4,4**	**4,4**	**4,6**	**4,4**	**4,8**	**4,4**	**4,6**	**4,6**	**4,5**

Os resultados indicam que a maioria dos participantes avaliou o sistema de maneira positiva, com uma média geral de 4.5, sugerindo que o RIP é bem aceito e facilita a gestão dos requerimentos acadêmicos.

3.3.2 Testes de Eficiência

Além do questionário, foram realizados testes para medir o impacto da automação no tempo de análise dos requerimentos. Os servidores foram cronometrados ao processar um conjunto de 10 requerimentos manualmente e, posteriormente, utilizando o RIP.

- **Tempo médio manual**: 2 minutos por requerimento.
- **Tempo médio com o RIP**: 45 segundos por requerimento.
- **Ganho de eficiência**: Redução no tempo de análise, possibilitando maior produtividade dos servidores.

Os dados demonstram uma melhora expressiva na rapidez do processamento dos requerimentos, reduzindo a carga de trabalho dos servidores e permitindo um atendimento mais ágil aos estudantes.

3.3.3 Coleta de Feedback Qualitativo

Após os testes, os cinco servidores acadêmicos que participaram da validação foram convidados a relatar suas impressões sobre o sistema, destacando pontos positivos e possíveis melhorias. A tabela abaixo apresenta as sugestões conforme foram descritas pelos participantes:

Tabela 2 - Feedbacks

Servidor	Feedback
Servidor 1	"O sistema é rápido e facilita bastante o nosso trabalho, mas seria bom se tivesse uma opção de relatório para acompanhar a quantidade de requerimentos finalizados."
Servidor 2	"Gostei muito da ferramenta! Está muito eficiente, mas um modo escuro ajudaria a reduzir o cansaço visual, principalmente quando usamos por períodos mais longos."
Servidor 3	"O tempo de resposta do sistema é ótimo, mas uma opção para exportar os dados dos requerimentos para Excel seria muito útil."
Servidor 4	"Achei a interface intuitiva e fácil de usar. No entanto, um relatório automático com estatísticas sobre os requerimentos mais frequentes ajudaria na gestão do setor."
Servidor 5	"O sistema está muito bom, mas seria interessante ter um modo escuro para facilitar a leitura à noite e uma busca mais avançada por

	palavras-chave nos requerimentos."

Com base nessas respostas, sintetizamos os principais pontos de melhoria identificados:

- **Geração automática de relatórios**: funcionalidade solicitada para facilitar o acompanhamento da quantidade e tipos de requerimentos processados.
- **Implementação de um modo escuro**: sugerido por dois servidores para melhorar o conforto visual durante o uso prolongado.
- **Exportação de dados para planilhas**: permitir que os servidores façam download dos requerimentos em formato de planilha.
- **Busca avançada por palavras-chave**: aprimorar a localização de requerimentos específicos dentro do sistema.

Esses feedbacks serão considerados na próxima fase de aprimoramento do sistema, garantindo uma melhor adaptação às necessidades dos usuários.

4. DESENVOLVIMENTO

4.1 Requisitos

Os requisitos do RIP: Requerimentos Ipanguaçu foram definidos com base nas necessidades dos servidores acadêmicos e nas funcionalidades essenciais para a automatização da análise de requerimentos estudantis. Eles estão divididos em requisitos funcionais e requisitos não funcionais.

4.1.1 Requisitos Funcionais

Os requisitos funcionais descrevem as funcionalidades que o sistema deve oferecer para atender às demandas dos usuários.

- **RF01**: O sistema deve permitir que os servidores acessem e visualizem a lista de requerimentos estudantis.
- **RF02**: O sistema deve integrar-se ao SUAP para importar automaticamente os requerimentos cadastrados.
- **F03**: O sistema deve utilizar uma LLM (Large Language Model) para analisar e classificar os requerimentos automaticamente.
- **RF04**: O usuário deve poder revisar e editar a categorização sugerida pela IA antes de aprovar ou recusar um requerimento.
- **RF05**: O sistema deve permitir a adição de interessados nos chamados para acompanhamento e notificações.
- **RF06**: Deve ser possível configurar respostas rápidas para resolução, cancelamento e comentários nos chamados.
- **RF07**: O sistema deve oferecer um histórico detalhado de cada requerimento, exibindo todas as interações em uma linha do tempo.
- **RF08**: O usuário deve poder configurar o modelo de IA utilizado e ajustar seu token de acesso.
- **RF09**: O sistema deve permitir a personalização do prompt da IA para melhorar a classificação dos requerimentos.
- **RF10**: Deve ser possível buscar chamados por palavras-chave, aluno ou categoria.
- **RF11**: O sistema deve oferecer estatísticas sobre os requerimentos, incluindo tipos mais frequentes e tempo médio de resposta.
- **RF12**: Deve haver um modo escuro para facilitar a leitura em ambientes de baixa luminosidade.
- **RF13**: O usuário deve poder exportar os dados dos requerimentos para planilhas.

4.1.2 Requisitos não funcionais

Os requisitos não funcionais estabelecem critérios de desempenho, segurança e usabilidade do sistema.

- **RNF01**: O sistema deve ser desenvolvido em Flutter, garantindo compatibilidade com Windows.
- **RNF02**: O tempo de resposta para carregamento de um requerimento não deve ultrapassar 2 segundos.
- **RNF03**: O armazenamento dos dados deve ser seguro, garantindo confidencialidade e integridade.
- **RNF04**: O sistema deve seguir os princípios de Clean Code para facilitar a manutenção do código-fonte.
- **RNF05**: A arquitetura do sistema deve seguir o padrão Clean Architecture, garantindo modularidade e escalabilidade.
- **RNF06**: O sistema deve ser intuitivo e fácil de usar, minimizando a necessidade de treinamentos extensivos.
- **RNF07**: A integração com o SUAP deve ocorrer de forma automatizada e segura.
- **RNF08**: O sistema deve permitir atualizações frequentes sem comprometer a estabilidade das funcionalidades existentes.

Esses requisitos garantem que o RIP seja eficiente, confiável e alinhado com as necessidades institucionais do IFRN.

4.2 SUAP

O Sistema Unificado de Administração Pública (SUAP) é a plataforma oficial de gestão acadêmica e administrativa do Instituto Federal do Rio Grande do Norte (IFRN). Ele centraliza informações institucionais e permite que alunos, professores e servidores realizem diversas atividades burocráticas de forma digital. O SUAP integra vários módulos, abrangendo desde matrículas e notas até a gestão de documentos e requerimentos. (Casadei., 2018) [4]

4.2.1 Central de Serviços

A Central de Serviços é o módulo do SUAP utilizado para a realização de requerimentos por alunos e servidores do IFRN. Antes da pandemia, essas solicitações eram feitas por meio de formulários em papel, o que demandava tempo e gerava dificuldades na gestão e acompanhamento dos processos. Com a digitalização forçada pelo contexto pandêmico, a Central de Serviços passou a ser amplamente utilizada para receber os requerimentos, suprindo a falta de um módulo especializado para isso.

Na Central de Serviços, os demandantes podem inserir arquivos e comentários em seus chamados. Os chamados podem ser marcados como resolvidos ou cancelados, permitindo um melhor controle sobre as solicitações. Além disso, é possível incluir outras pessoas como interessadas no chamado, para que acompanhem seu andamento e recebam notificações sobre atualizações no processo.

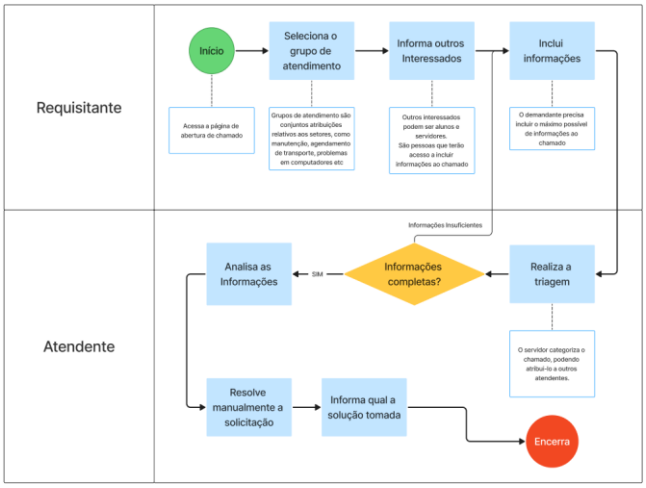

Figura 1 - Fluxo de um chamado na Central de Serviços

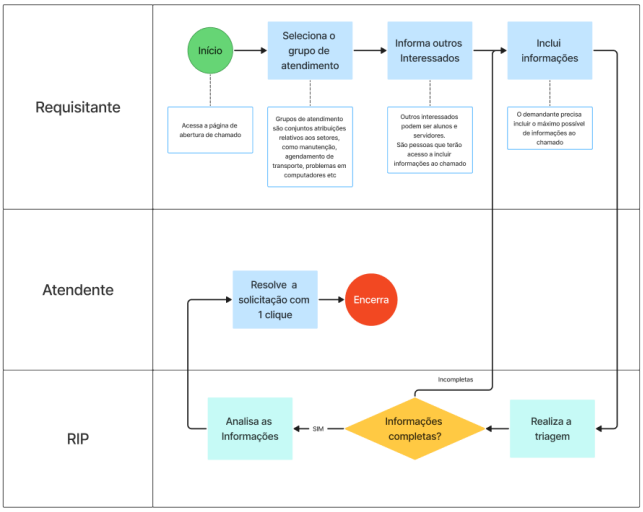

Figura 2 - Fluxo de resolução com Inteligência Artificial

Fluxo de abertura do chamado: A Central de Serviços do SUAP é o módulo mais democrático do Sistema. Nele, todos os usuários, sejam servidores ou alunos, podem fazer solicitações.

Estes pedidos variam entre simples conserto de goteiras a emissões de diplomas. Cada setor possui seus próprios grupos de atendimento e recebem somente os chamados a si direcionados.

O processo inicia-se quando um requisitante precisa fazer algum requerimento a algum dos setores. No contexto das secretarias acadêmicas, o pedido mais comum é o de justificativa de falta. Para isso, o aluno informa uma data no conteúdo do chamado e anexa um documento comprobatório. Este documento pode ser um atestado médico, declaração de falta de ônibus da prefeitura do município onde o aluno reside e uma convocação para o serviço militar (Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, 2012) [16].

Fluxo de resolução: O atendente da Secretaria Acadêmica necessita ler o conteúdo, analisar o arquivo anexado, definir qual a atividade será realizada e efetivamente realizá-la. Esta tarefa pode repetir-se centenas de vezes em um dia, basta, para isso, a quebra de um ônibus escolar ou um feriado municipal, por exemplo.

A efetivação do requerimento exige a digitação manual dos dados de cada um dos requerimentos nos mais diversos módulos estudantis, como criação de processos, solicitações de diploma, justificativas de falta, inclusão e remoção de disciplinas, trancamentos de semestre etc.

Neste contexto de dispersão de tarefas, surge a necessidade de automatizar o processo de análise e resolução, concentrando o trabalho em uma única tela com análise realizada com inteligência artificial resolvendo 90% do trabalho, deixando apenas a confirmação para o servidor atendente.

Fluxo de resolução com IA: Segundo o fluxo de chamados com o RIP, o atendente apenas confere a análise realizada pela inteligência artificial e aplica a solução indicada.

Todos os outros passos comuns aos atendentes são feitos automaticamente, deixando mais tempo para se concentrar em tarefas que exigem concentração, como fiscalização de contratos, pagamentos de bolsas acadêmicas, atendimento presencial etc.

4.3 Telas do Sistema

O RIP: Requerimentos Ipanguaçu possui diversas funcionalidades projetadas para otimizar o fluxo de análise de requerimentos estudantis.

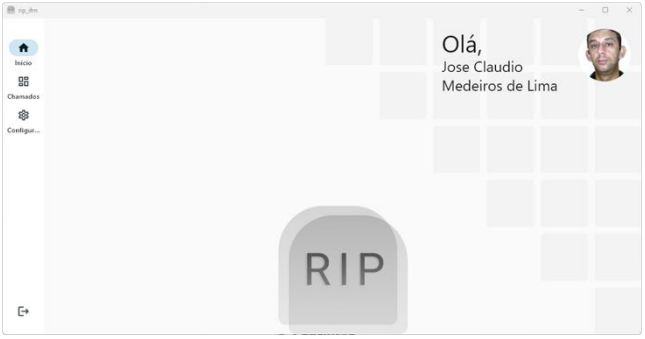

Figura 3 - Tela inicial

A seguir, descrevemos as principais telas do sistema e suas respectivas funções.

4.3.1 Tela de Configurações

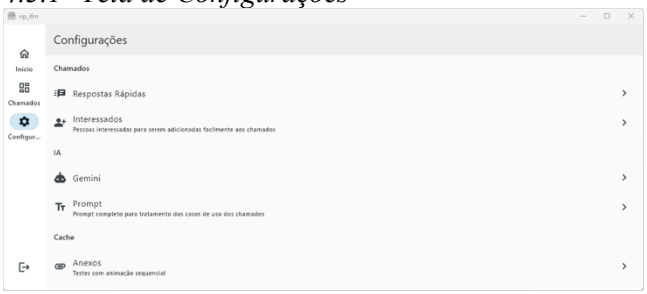

Figura 4 - Tela de configurações

A tela de configurações permite a personalização do sistema, oferecendo os seguintes recursos:

Respostas rápidas: permite criar respostas rápidas para resolução, cancelamento e comentários nos chamados, além de tipos de justificativa, como atestado e declaração de comparecimento. Cada resposta rápida pode ser vinculada a um dos módulos: comentário no chamado, texto de resolução, texto de cancelamento ou tipo de justificativa.

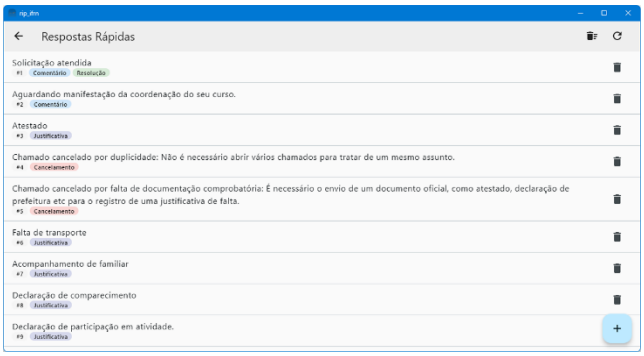

Figura 5 - Tela de respostas rápidas

Interessados: permite criar uma lista de pessoas previamente selecionadas para serem adicionadas rapidamente aos chamados. Os interessados são usuários que têm acesso ao chamado, podem comentar e recebem notificações. Normalmente, adicionamos interessados quando o assunto do chamado tem relevância para essas pessoas, seja para que tenham ciência, seja para que possam dar um parecer.

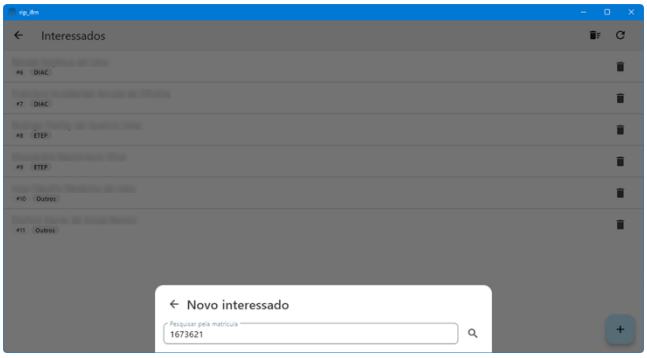

Figura 6 - Tela de interessados

Gemini: permite configurar o modelo de linguagem utilizado e inserir o token de acesso para conexão com a API da LLM.

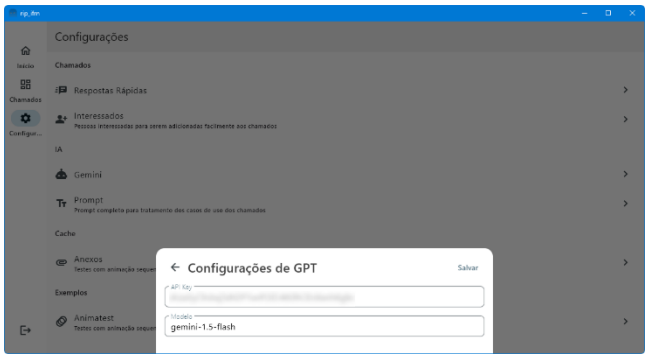

Figura 7 - Tela de configurações do GPT

Prompt: O prompt é o texto em linguagem natural que explica ao LLM como ele deve se comportar ao analisar os textos recebidos.

Esta tela possibilita ajustes nestas regras, permitindo que pequenas alterações sejam feitas rapidamente para classificar corretamente os chamados. Ou seja, caso a IA não identifique um chamado corretamente, o usuário pode modificar o prompt e solicitar uma nova análise.

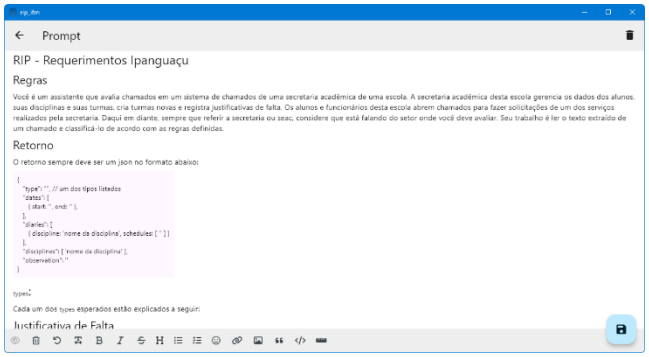

Figura 8 - Tela do prompt

4.3.2 Tela de Chamados

A tela de chamados divide-se em 2 seções principais:

À esquerda, uma listagem com os chamados disponíveis. À direita, o chamado completo exibido em formato de *timeline*.

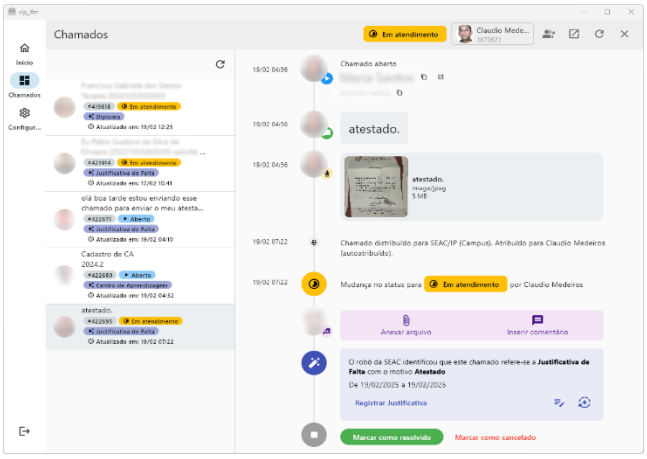

Figura 9 - Tela de chamados

4.4 Tecnologias e Ferramentas

O RIP: Requerimentos Ipanguaçu foi desenvolvido utilizando tecnologias modernas e boas práticas de desenvolvimento de software para garantir escalabilidade, eficiência e qualidade do código.

4.4.1 Flutter

Flutter [18] é um framework de código aberto desenvolvido pelo Google em 2017 para a criação de aplicativos multiplataforma. Ele permite que um único código seja utilizado para compilar aplicações para Windows, Android, iOS e Web, reduzindo o tempo e o custo de desenvolvimento.

Sua arquitetura se baseia no uso de widgets personalizáveis e um mecanismo de renderização próprio, garantindo interfaces fluidas e alto desempenho.

Segundo Arif et al. [12], o Flutter se destaca por sua capacidade de criar aplicações nativas responsivas e por seu ecossistema robusto, tornando-se uma das tecnologias mais adotadas para o desenvolvimento mobile moderno.

4.4.2 Dart

Dart [19] é a linguagem de programação utilizada no Flutter. Criada pelo Google em 2011, foi projetada para desenvolvimento rápido e eficiente de aplicações multiplataforma. A linguagem combina paradigmas de programação orientada a objetos e funcional, oferecendo recursos como tipagem opcional e execução Just-In-Time (JIT) e Ahead-Of-Time (AOT), otimizando o desempenho em diferentes contextos.

Sua sintaxe intuitiva e suporte à programação assíncrona são essenciais para aplicativos que interagem frequentemente com APIs e modelos de inteligência artificial.

Segundo Swathiga, (2021) [15], Dart se destaca por sua eficiência no desenvolvimento de interfaces dinâmicas e sua integração nativa com o Flutter, tornando-o uma escolha ideal para aplicativos modernos.

4.4.3 Gemini API

A API Gemini (Google, s.d.) [20], desenvolvida pelo Google, é utilizada para processar e classificar automaticamente os requerimentos estudantis. Esse modelo de inteligência artificial baseado em aprendizado profundo permite interpretar o conteúdo dos chamados, categorizá-los e sugerir ações rápidas para os servidores.

Para garantir flexibilidade e facilitar futuras substituições da LLM, a implementação da Gemini foi feita de forma desacoplada, seguindo os princípios SOLID. Especificamente, o sistema utiliza o padrão Abstract Factory Pattern, permitindo que uma nova IA possa ser integrada rapidamente sem alterar a estrutura principal do código. Isso garante maior adaptabilidade do sistema a possíveis mudanças tecnológicas no futuro.

4.4.4 SUAP (Sistema Unificado de Administração Pública)

Segundo Casadei (Casadei., 2018) [4], o SUAP é o sistema oficial utilizado pelo IFRN para gestão acadêmica e administrativa. Teve seu desenvolvimento iniciado em 2006 pela Coordenação de Sistemas de Informação (COSINF) do Instituto Federal do RN, e foi escrito em Python usando o framework Django.

A integração entre o RIP e o SUAP é realizada por meio de web scraping e expressões regulares, permitindo extrair e interpretar as informações dos chamados de maneira estruturada, garantindo que os requerimentos sejam processados corretamente pela IA.

Esse método foi escolhido devido à ausência de uma API oficial para acesso automatizado aos chamados, tornando o scraping a solução mais viável para a obtenção de informações diretamente do sistema.

4.4.5 HiveDB

O HiveDB [21] é um banco de dados do tipo chave-valor, projetado especificamente para atender às necessidades multiplataformas do Dart e Flutter. Extremamente rápido, o Hive mantém uma cópia dos dados em memória enquanto realiza a gravação no disco em segundo plano, garantindo alto desempenho e eficiência.

Seu funcionamento baseia-se no conceito de boxes, que atuam como bancos de dados independentes, permitindo uma organização modular e flexível das informações.

No RIP, o Hive complementa perfeitamente o suporte ao funcionamento offline da maioria das funcionalidades, com exceção daquelas integradas ao SUAP, que opera exclusivamente em ambiente online.

4.4.6 GetX

O GetX [3] é um framework para Flutter criado por Jonata Borges em 2019, oferecendo soluções para gerenciamento de estado, injeção de dependências e roteamento de maneira leve e performática. Sua abordagem reativa reduz a necessidade de reconstrução excessiva da interface, tornando o desempenho da aplicação mais eficiente.

No RIP: Requerimentos Ipanguaçu, o GetX foi utilizado para gerenciar o estado da aplicação, garantindo que atualizações na interface ocorressem de forma dinâmica e sem complexidade excessiva. A injeção de dependências proporcionada pelo GetX permitiu um alto nível de desacoplamento entre os módulos do sistema, tornando possível a substituição de funcionalidades sem grandes impactos na arquitetura. Dessa forma, o SUAP poderia ser trocado pelo SIGAA, o Gemini pelo ChatGPT e o banco de

dados Hive pelo SQLite, caso necessário. Essa flexibilidade assegura que o sistema possa evoluir e se adaptar a novas tecnologias sem a necessidade de grandes reestruturações no código.

4.4.7 Figma

O Figma [17] é uma plataforma colaborativa de design de interfaces e prototipagem, acessível diretamente pelo navegador, sem necessidade de instalação de software.

Lançado em 2016 por Dylan Field e Evan Wallace, o Figma permite que designers e equipes trabalhem simultaneamente em projetos, facilitando a criação de interfaces para aplicativos, websites e outros produtos digitais.

A escolha do Figma para este projeto se deu por sua interface intuitiva, leveza e facilidade de uso, tornando a curva de aprendizado mais acessível, possibilitando uma rápida prototipação do aplicativo.

4.4.8 Git

O Git [22] é um sistema de controle de versão distribuído criado por Linus Torvalds em 2005 para gerenciar o desenvolvimento do kernel do Linux. Ele permite o rastreamento de alterações no código-fonte, facilitando o trabalho colaborativo entre desenvolvedores e garantindo a integridade do histórico de modificações. O Git possibilita a criação de múltiplas ramificações (branches), tornando o processo de desenvolvimento mais organizado e seguro.

No RIP: Requerimentos Ipanguaçu, o Git foi utilizado para versionamento do código-fonte, garantindo controle sobre as diferentes fases do desenvolvimento. O repositório do sistema está hospedado no GitHub, permitindo acesso remoto e colaboração eficiente. Além disso, o GitHub Issues foi empregado para gerenciar tarefas, acompanhar o progresso das implementações e registrar discussões técnicas. Esse fluxo de trabalho estruturado possibilitou a adoção de metodologias ágeis, assegurando a rastreabilidade das mudanças e a estabilidade do projeto.

4.4.9 Boas Práticas de Desenvolvimento

Além do uso de ferramentas modernas, o desenvolvimento do RIP segue princípios sólidos de engenharia de software, garantindo qualidade e organização no código:

SOLID: Conjunto de princípios para desenvolvimento orientado a objetos apresentados por Robert Martin em 2002 em seu livro Agile Software Development: Principles, Patterns, and Practices [9]. Os princípios SOLID ajudam os desenvolvedores a criar código modular e flexível, promovendo a injeção de dependências, a responsabilidade única das classes e a segregação de interfaces, entre outros conceitos essenciais.

No RIP, a comunicação com o SUAP segue rigorosamente esses princípios, possibilitando a substituição completa do banco de dados, da API de inteligência artificial e até mesmo do próprio SUAP, apenas com a injeção de novas instâncias.

Test-Driven Development (TDD): O desenvolvimento do RIP: Requerimentos Ipanguaçu seguiu os princípios do Test-Driven Development (TDD), uma metodologia que prioriza a escrita de testes antes da implementação do código. O TDD consiste em um ciclo iterativo de três etapas: escrever um teste falho, implementar o código mínimo necessário para passar no teste e refatorar garantindo que os testes continuem aprovados. Esse processo melhora a confiabilidade do software, reduz erros e facilita a manutenção do código (BECK, 2002) [1].

No RIP, o TDD foi aplicado principalmente nas classes responsáveis pela comunicação com o SUAP, garantindo 100% de cobertura de código nesses módulos. Testes automatizados verificam a correta extração e manipulação dos dados dos chamados acadêmicos, assegurando que mudanças na API do SUAP não comprometam o funcionamento do sistema. Essa abordagem fortalece a estabilidade da aplicação, permitindo uma integração segura e eficiente com o ambiente acadêmico digital do IFRN.

4.5 Arquitetura

O **RIP: Requerimentos Ipanguaçu** adota a arquitetura **MVVM (Model-View-ViewModel)** para garantir modularidade, separação de responsabilidades e melhor manutenção do código.

A camada **Model** gerencia os dados e regras de negócio, incluindo a integração com o **SUAP** e o armazenamento local via **Hive**. A **ViewModel** atua como intermediária, fornecendo os dados processados à interface e garantindo reatividade. A camada **View**, composta pelos widgets do **Flutter**, exibe as informações e interage com os usuários.

Essa abordagem possibilita testes isolados da lógica de negócio, facilita a substituição de tecnologias e melhora a escalabilidade do sistema. Com a reatividade da **ViewModel**, mudanças nos chamados ou na classificação automática pelos **LLMs** são refletidas instantaneamente na UI. Além disso, o uso de **Getx** permite um gerenciamento eficiente de estado, assegurando que o **RIP** opere de forma fluida e organizada, otimizando a experiência dos usuários.

5. CONCLUSÕES

Este trabalho teve como objetivo aprimorar o processo de análise e categorização de requerimentos estudantis no IFRN por meio da automatização com modelos de linguagem natural. Para isso, foi desenvolvido o RIP: Requerimentos Ipanguaçu, um sistema que se integra ao SUAP e permite que servidores aprovem ou classifiquem solicitações de forma rápida e eficiente. O aplicativo foi desenvolvido utilizando Flutter e Dart, garantindo compatibilidade e eficiência na execução.

A implementação do sistema possibilitou uma redução significativa no tempo de processamento dos requerimentos, conforme indicado pelos testes realizados com servidores acadêmicos. Além disso, a validação por meio do questionário TAM demonstrou uma boa aceitação da ferramenta, embora tenham sido identificadas oportunidades de melhorias, como ajustes na interface e a inclusão de funcionalidades adicionais.

Dado o escopo do projeto e o cronograma de desenvolvimento, algumas funcionalidades foram priorizadas, enquanto outras foram deixadas para futuras implementações. Como trabalhos futuros, sugere-se a inclusão de novas ferramentas de análise estatística, aprimoramento da personalização do modelo de IA e expansão da integração com outros sistemas acadêmicos. Dessa forma, o RIP poderá continuar evoluindo para atender de maneira ainda mais eficiente às necessidades da Secretaria Acadêmica do IFRN.

O projeto RIP: Requerimentos Ipanguaçu representa um avanço significativo na gestão de requerimentos estudantis, trazendo agilidade e eficiência para a Secretaria Acadêmica do IFRN. A integração com LLMs permite uma classificação precisa das

solicitações, reduzindo a carga de trabalho dos servidores e melhorando a experiência dos estudantes.

6. REFERÊNCIAS BIBLIOGRÁFICAS

[1] Kent Beck. 2022. *Test driven development: By example*. Addison-Wesley Professional.

[2] David Benyon. 2014. *Designing interactive systems: A comprehensive guide to HCI, UX and interaction design*. Pearson Harlow.

[3] Jonatas Borges. GetX. Retrieved from https://github.com/jonataslaw/getx

[4] Tarso Latorraca Casadei. 2018. Avaliação arquitetural do Sistema SUAP: uma análise sistematizada sobre desempenho. (December 2018). Retrieved September 26, 2025 from https://repositorio.ufrn.br/jspui/handle/123456789/26904

[5] Idalberto Chiavenato. 2002. *Teoria geral da administração*. Elsevier Brasil.

[6] Matheus Cruz, Érica Mourão, Flávia Bernardini, José Viterbo, and Daniela Trevisan. 2022. Uso do TAM—Technology Acceptance model—no ciclo de design de aplicações computacionais. In *Simpósio Brasileiro sobre Fatores Humanos em Sistemas Computacionais (IHC)*, 2022. SBC, 3–4. Retrieved from https://sol.sbc.org.br/index.php/ihc_estendido/article/view/22027

[7] João Miguel Resende da Fonseca. 2022. Chatbot para apoio à Divisão Académica. (2022). Retrieved September 26, 2025 from http://hdl.handle.net/10400.22/20825

[8] Paulo FREIRE. 2021. *Pedagogia do oprimido*. Editora Paz e Terra.

[9] Robert C. Martin. 2017. *Clean Architecture: A Craftsman's Guide to Software Structure and Design* (1st ed.). Prentice Hall Press, USA.

[10] Fernanda Pereira Santana and Leandro Corrêa Magalhães. 2024. APLICAÇÕES DO PROCESSAMENTO DE LINGUAGEM NATURAL NO AMBIENTE EDUCACIONAL: UMA REVISÃO SISTEMÁTICA DA LITERATURA. *Rev. FOCO* 17, 1 (January 2024), e3921. https://doi.org/10.54751/revistafoco.v17n1-032

[11] Elaine Cristina dos Santos, Wildes Luz Lima, and Luciano Bergamo. 2024. A INTELIGÊNCIA ARTIFICIAL NA AUTOMAÇÃO DOS PROCESSOS NEGOCIAIS E OS LIMITES ÉTICOS DE SUA UTILIZAÇÃO. *Rev. Acadêmica Online* 10, 51 (June 2024), 1–11. https://doi.org/10.36238/2359-5787.2024.v10n51.180

[12] Arif Md. Sattar, Preetam Soni, Mritunjay Kumar Ranjan, Amar Kumar, Chandrahas Sahu, Shilpi Saxena, and Prafulla Chaudhari. 2023. Accelerating Cross-platform Development with Flutter Framework. *J. OPEN SOURCE Dev.* (2023). https://doi.org/10.37591/joosd.v10i2.580

[13] Ken Schwaber. 2004. *Agile Project Management with Scrum*. Microsoft Press.

[14] Ian Sommerville. 2016. *Software engineering* (Tenth edition ed.). Pearson, Boston Columbus Indianapolis New York San Francisco Hoboken Amsterdam Cape Town Dubai London.

[15] UUAS Swathiga, P. Vinodhini, and V. Sasikala. 2021. An interpretation of dart programming Language. *DRSR J.* 11, 3 (2021), 144–149.

[16] 2012. Organização Didática do IFRN. Retrieved from https://portal.ifrn.edu.br/documents/2438/OrganizacaoDidatica_2012_versaoFINAL_20mai2012.pdf

[17] Figma. Retrieved from https://www.figma.com/

[18] Flutter. Retrieved from https://flutter.dev/

[19] Dart. Retrieved from https://dart.dev/

[20] Gemini. Retrieved from https://ai.google

[21] HiveDB. Retrieved from https://github.com/hivedb

[22] Git. Retrieved from https://git-scm.com/