



PROJETO A3 - MODELOS, MÉTODOS E TÉCNICAS DA ENGENHARIA DE SOFTWARE

PRODUTO DE SOFTWARE - OiBot

ORIENTADOR: Professora Milkes Yone Alvarenga

Nome dos Alunos:

Gabriel Rodrigues da Silva Costa - 822137024

Lucas Ribeiro Pedroso - 823149292

Thiago Duarte Reis - 822141527

**São Paulo
2025**

ÍNDICE DETALHADO

1. INTRODUÇÃO

1. Introdução.....	3
1.1. Tema.....	3
1.2. Objetivos a serem alcançados.....	3
1.3. Escopo principal.....	4
2. Definição do Modelo de Processo.....	6
3. Requisitos do Sistema de Software.....	7
3.1. Requisitos Funcionais.....	7
3.2. Requisitos Não-Funcionais.....	9
Catálogo de Atores.....	12
Casos de Uso (UC01 a UC15).....	12
UC01 – Criar Bot.....	12
UC02 – Configurar Personalidade.....	13
UC03 – Integrar com Discord.....	13
UC04 – Processar Mensagens.....	14
UC05 – Visualizar Atendimentos (Dashboard).....	14
UC06 – Gerar Métricas.....	15
UC07 – Escalonar Atendimento para Humano.....	15
UC08 – Solicitar Consentimento LGPD.....	16
UC09 – Editar Bot.....	16
UC10 – Excluir Bot.....	16
UC11 – Monitorar Conexões (APIs / Discord).....	17
UC12 – Gerenciar Usuários do Painel.....	17
UC13 – Histórico do Bot.....	17
UC14 – Regras de Atendimento.....	17
UC15 – Logs e Auditoria.....	18
Diagrama de classe.....	18
4. Projeto.....	18
5. Protótipo de Interface.....	24
6. Critérios de Qualidade de Software.....	28
Ambiente.....	30
Critérios de Aceitação.....	30
Matriz de Testes (amostra).....	31
Exemplo: Roteiro UC03 — Integrar com Discord.....	31
Exemplo: Roteiro UC04 — Processar Mensagens (fluxo feliz).....	31
Exemplo: Roteiro UC07 — Escalonamento.....	31

1. Introdução

1.1. Tema

O projeto tem como tema o desenvolvimento de um bot de atendimento automatizado ao cliente baseado em Inteligência Artificial, capaz de oferecer suporte dinâmico, personalizado e com personalidade própria. O objetivo é criar uma solução inovadora que vá além dos tradicionais chatbots, tornando a comunicação mais natural, humanizada e eficiente. O projeto será desenvolvido seguindo boas práticas de engenharia de software, utilizando métodos ágeis e técnicas que assegurem a qualidade, escalabilidade e confiabilidade do sistema.

1.2. Objetivos a serem alcançados

Objetivo principal: Criar uma plataforma de bot inteligente que ofereça atendimento automatizado e personalizado, com traços de personalidade próprios para melhorar a experiência do cliente.

Objetivos específicos:

- Garantir respostas rápidas e assertivas, reduzindo o tempo de espera no atendimento.
- Automatizar processos repetitivos, liberando a equipe humana para atendimentos mais complexos.
- Humanizar o atendimento por meio de um bot que transmita empatia, carisma e adaptação ao estilo de comunicação do usuário.
- Aplicar práticas de gestão da qualidade de software, com foco em confiabilidade, desempenho e usabilidade.
- Implantar uma plataforma escalável que permita integrar o bot no canal Discord.
- Utilizar técnicas de inovação em IA generativa e processamento de linguagem natural para aprimorar a experiência do usuário.

Público-alvo: Empresas de pequeno, médio e grande porte que necessitam de atendimento automatizado de qualidade para melhorar a experiência de seus clientes.

Plataforma de desenvolvimento: Aplicação web com APIs para integração em múltiplos canais de atendimento, desenvolvida com frameworks modernos, banco de dados relacional e suporte a serviços em nuvem.

Melhorias do processo atual: Substituição de atendimentos robotizados e engessados por uma abordagem interativa, dinâmica e personalizada.

Automatização e inovação: O sistema busca reduzir custos operacionais e, ao

mesmo tempo, entregar um atendimento mais eficiente e humano.

1.3. Escopo principal

O projeto propõe o desenvolvimento de um sistema de **Inteligência Artificial conversacional com personalidade própria**, voltado para a automatização e a melhoria da comunicação entre empresas e clientes. A solução será baseada em modelos de processamento de linguagem natural, consumindo APIs de IA avançadas (como o Kimi-k2), com foco na personalização do estilo de atendimento e na eficiência operacional.

Principais ações e implementações

- **Desenvolvimento do motor de IA conversacional**
 - Consumo da API do modelo Kimi-k2 para realizar processamento de linguagem natural.
 - Implementação de prompts customizados que definem a **personalidade e o estilo de comunicação** do bot
 - Utilização de técnicas de *prompt engineering* para guiar o comportamento e a coerência nas respostas.
- **Modelos de personalidade personalizáveis**
 - Criação de perfis de atendimento configuráveis (ex.: formal, simpático, técnico, descontraído).
 - Ajuste dinâmico da personalidade do bot de acordo com o público-alvo ou com a necessidade da empresa.
- **Integração com múltiplos canais de comunicação**
 - Conexão com a plataforma **Discord**.
 - Centralização do processamento da IA em um backend que garanta uniformidade nas respostas, independentemente do canal de contato.
- **Implementação de painel administrativo**
 - Desenvolvimento de um dashboard web para monitoramento das conversas
 - Funcionalidades principais:
 - Visualização em tempo real dos atendimentos.
 - Configuração de personalidades e regras de atendimento.
 - Geração de métricas de desempenho e relatórios.

- **Aplicação de métodos ágeis de desenvolvimento**

- Uso de **Scrum** para o planejamento e acompanhamento das sprints.
- Utilização de **Kanban** para controle do fluxo de tarefas.
- Entregas iterativas com foco em melhoria contínua e adaptação às necessidades do cliente.

- **Definição de indicadores de qualidade**

- Tempo médio de resposta inferior a 3 segundos.
- Redução significativa na demanda de atendimento humano para casos simples e repetitivos.
- Taxa de satisfação do cliente acima de 85% nas interações automatizadas.

- **Segurança e privacidade (conformidade com a LGPD)**

- Implementação de criptografia para dados sensíveis.
- Armazenamento apenas das informações estritamente necessárias para o atendimento.
- Solicitação explícita de consentimento dos usuários quanto ao tratamento de dados.

2. Definição do Modelo de Processo

Para o desenvolvimento do projeto será adotado o modelo de processo ágil, com ênfase no framework Scrum, complementado pelo uso do Kanban para acompanhamento visual das tarefas. Essa escolha se justifica pela necessidade de flexibilidade, adaptação contínua às mudanças e foco em entregas incrementais que agreguem valor ao cliente desde as primeiras etapas.

O Scrum será utilizado como base para organizar o fluxo de desenvolvimento em sprints de duas semanas, com reuniões de planejamento, acompanhamento diário (*daily meetings*) e retrospectivas ao final de cada ciclo. Esse modelo permite que o produto seja construído de forma iterativa e incremental, garantindo a evolução contínua das funcionalidades e possibilitando ajustes rápidos a partir de feedbacks.

O Kanban será aplicado em conjunto para proporcionar visibilidade e controle do fluxo de trabalho, possibilitando o acompanhamento das tarefas em tempo real, a identificação de gargalos e a melhoria da produtividade da equipe.

Essa abordagem ágil assegura maior qualidade no desenvolvimento do sistema de IA conversacional, permitindo entregas rápidas, adaptação às necessidades da empresa contratante e melhor gerenciamento dos riscos do projeto, em conformidade com as boas práticas de engenharia de software.

3. Requisitos do Sistema de Software

3.1. Requisitos Funcionais

Os requisitos funcionais descrevem as funções que o sistema deve executar para atender aos objetivos do projeto.

RF01 - Cadastro e autenticação de usuários

O sistema deve permitir que administradores façam login e logout de forma segura.

Deve haver controle básico de acesso às funcionalidades administrativas.

RF02 - Criação e configuração do bot

O sistema deve permitir a criação e edição do bot de atendimento automatizado.

O administrador deve poder definir o nome, avatar e descrição do bot.

Cada bot deve poder ser associado a um perfil de personalidade predefinido (ex.: formal, simpático, técnico, descontraído).

RF03 - Integração com canais de comunicação

O sistema deve permitir a integração dos bots com o Discord.

As mensagens enviadas pelos usuários devem ser processadas e respondidas através desses canais.

RF04 - Processamento de mensagens

O sistema deve enviar as mensagens dos usuários para a API de Inteligência Artificial configurada (ex.:Kimi-k2).

As respostas devem ser retornadas e exibidas em tempo real para o usuário.

As conversas são temporárias e não são armazenadas permanentemente, sendo removidas após o atendimento.

RF05 - Painel administrativo (Dashboard)

O sistema deve oferecer um painel web para:

Visualizar os atendimentos em andamento.
Configurar bots e canais de integração.

Monitorar o status das conexões com APIs e serviços externos.

RF06 - Métricas e relatórios

O sistema deve gerar relatórios de:
Tempo médio de resposta do bot.
Número total de atendimentos realizados.
Taxa de disponibilidade e erros de conexão.

RF07 - Escalonamento de atendimento

O sistema deve permitir o encaminhamento manual ou automático de uma conversa para um atendente humano, quando o bot não compreender a solicitação.

RF08 - Gestão de dados e consentimento

O sistema deve solicitar o consentimento do usuário para o tratamento de dados conforme a LGPD.

Apenas informações temporárias e necessárias ao atendimento devem ser utilizadas.

3.2. Requisitos Não-Funcionais

Os requisitos não funcionais descrevem as qualidades e restrições que o sistema deve atender.

RNF01 - Desempenho

O sistema deve responder às solicitações do usuário em até 3 segundos em condições normais de operação.

RNF02 - Escalabilidade

O sistema deve permitir a adição de novos bots e canais de comunicação sem necessidade de grandes alterações na arquitetura.

RNF03 - Segurança

Todas as comunicações devem ocorrer por conexões seguras (HTTPS).

As chaves de API e dados sensíveis devem ser armazenados de forma protegida.

RNF04 - Usabilidade

A interface deve ser simples, intuitiva e responsiva, adaptando-se a diferentes dispositivos.

RNF05 - Confiabilidade

O sistema deve garantir alta disponibilidade e recuperação automática em caso de falhas temporárias nas APIs.

RNF06 - Manutenibilidade

O código deve ser modular, versionado e documentado, facilitando atualizações e correções.

RNF07 - Compatibilidade

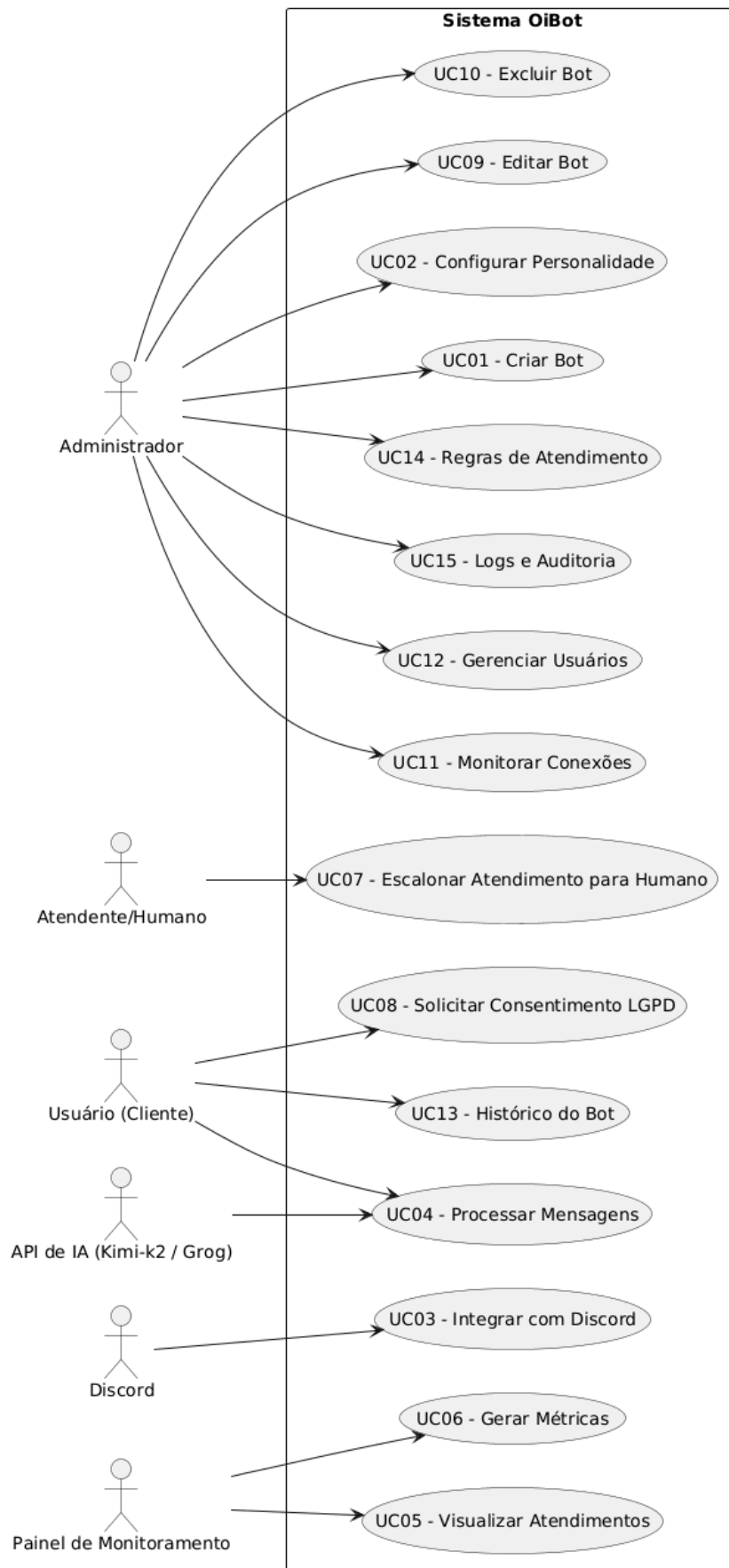
O sistema deve funcionar corretamente nos navegadores Google Chrome, Microsoft Edge e Mozilla Firefox.

As integrações devem seguir o padrão RESTful.

RNF08 - Conformidade Legal

O sistema deve estar em conformidade com a Lei Geral de Proteção de Dados (LGPD).

Deve excluir automaticamente os dados temporários após o encerramento da conversa



Catálogo de Atores

- **Administrador:** usuário com permissões de configuração, criação/exclusão de bots e gestão do painel.
- **Atendente / Humano:** operador que recebe conversas escaladas.
- **Usuário (Cliente):** pessoa que interage com o bot via Discord ou outro canal.
- **Discord (Canal):** sistema externo que envia/recebe mensagens.
- **API de IA (Kimi-k2 / Grog / outro):** serviço de inferência que gera respostas.
- **Painel de Monitoramento:** subsistema que exibe métricas, logs e permite intervenções.

Casos de Uso (UC01 a UC15)

Para cada UC abaixo: Breve descrição; Atores; Pré-condições; Fluxo básico; Fluxos alternativos; Mensagens do sistema; Pós-condição.

UC01 – Criar Bot

Descrição: Permitir ao Administrador criar uma nova instância de bot com nome, avatar e perfil inicial.

Atores: Administrador.

Pré-condições: Administrador autenticado.

Fluxo básico:

1. Admin abre o painel → Módulo “Bots”.
2. Clica em “Criar Bot”.
3. Preenche nome, avatar, descrição, perfil de personalidade padrão.
4. Clica em “Salvar”.
5. Sistema valida dados e cria registro.

Fluxo alternativo:

- FA01: Campos obrigatórios vazios → exibe erro e impede salvar.
Mensagens: “Preencha os campos obrigatórios.” / “Bot criado com sucesso.”
Pós-condição: Bot salvo no banco e listado no painel.

UC02 – Configurar Personalidade

Descrição: Configurar/editar perfil de personalidade (formal, simpático, técnico, etc.) e salvar prompts.

Atores: Administrador.

Pré-condições: Bot existente; Admin autenticado.

Fluxo básico:

1. Admin abre a página do bot → “Personalidade”.
2. Seleciona perfil ou edita prompts e parâmetros (temperatura, tom).
3. Salva alterações.

Fluxo alternativo: FA01: Prompt inválido → validação.

Mensagens: “Configurações salvas.” / “Erro: prompt inválido.”

Pós-condição: Motor de prompts atualizado.

UC03 – Integrar com Discord

Descrição: Ligar o bot a um servidor/chanel do Discord via token/URL webhook.

Atores: Administrador, Discord.

Pré-condições: Token do Discord disponível; permissão no servidor.

Fluxo básico:

1. Admin vai em “Integrações” → escolhe Discord.
2. Insere token/ID do servidor e configura mapeamentos de canais.
3. Testa conexão.
4. Confirma e salva.

Fluxo alternativo: FA01: Token inválido → falha no teste.

Mensagens: “Conexão bem-sucedida.” / “Falha na autenticação do Discord.”

Pós-condição: Conector Discord ativo; mensagens serão recebidas/enviadas.

UC04 – Processar Mensagens

Descrição: Receber mensagem do usuário, enviar para API de IA e retornar resposta.

Atores: Usuário, API de IA, Discord.

Pré-condições: Integração com Discord ativa; bot ativo.

Fluxo básico:

1. Usuário envia mensagem no Discord.
2. Sistema captura e normaliza texto.
3. Verifica consentimento LGPD (UC08).
4. Envia prompt + contexto para a API de IA.
5. Recebe resposta, aplica pós-processamento (moderação, placeholders).
6. Envia resposta ao Discord.

Fluxos alternativos: FA01: API de IA sem resposta → mensagem padrão / encaminha para humano.

Mensagens: “Desculpe, não entendi.” / “Conectando com suporte humano...”

7. **Pós-condição:** Conversa respondida; logs temporários gravados.

UC05 – Visualizar Atendimentos (Dashboard)

Descrição: Painel com atendimentos em andamento, filtros e ações (ver detalhes, encaminhar).

Atores: Painel/Administrador/Atendente.

Pré-condições: Usuário autenticado com permissão.

Fluxo básico:

1. Acessa Dashboard → lista de conversas em tempo real.
2. Aplica filtros (status, bot, canal).
3. Clica em atendimento para visualizar histórico e métricas.

Fluxo alternativo: FA01: Sem dados → exibe “Nenhum atendimento no momento”.

Mensagens: “Conversa encaminhada para [nome].”

Pós-condição: Operador pode tomar ação (escalonar, encerrar).

UC06 – Gerar Métricas

Descrição: Gerar relatórios de tempo médio de resposta, número de atendimentos, satisfação etc.

Atores: Administrador.

Pré-condições: Histórico mínimo de logs.

Fluxo básico:

1. Acessa Relatórios → seleciona período/filtro.
2. Clica “Gerar”.
3. Sistema calcula métricas e apresenta gráfico/tabela.
4. Pode exportar CSV/PDF.

Fluxo alternativo: FA01: Período sem dados → mensagem informativa.

Mensagens: “Relatório gerado com sucesso.”

Pós-condição: Relatório disponível para download.

UC07 – Escalonar Atendimento para Humano

Descrição: Encaminhar conversa automaticamente (por regra) ou manualmente para atendente humano.

Atores: Bot, Atendente, Administrador.

Pré-condições: Atendentes online; canal de atendimento humano configurado.

Fluxo básico:

1. Regra detecta intenção de alta complexidade OU usuário solicita “Falar com humano”.
2. Sistema tenta encaminhar para um atendente disponível.
3. Se atendente aceita, conversa é transferida.

Fluxo alternativo: FA01: Nenhum atendente disponível → enfileira ou apresenta opção de agendamento.

Mensagens: “Encaminhando para um atendente...” / “Nenhum atendente disponível no momento.”

Pós-condição: Conversa passa a ser gerida por humano; histórico sincronizado.

UC08 – Solicitar Consentimento LGPD

Descrição: Apresentar termo de consentimento ao usuário e registrar decisão.

Atores: Usuário, Sistema.

Pré-condições: Primeira interação do usuário com o bot (ou quando necessário).

Fluxo básico:

1. Bot envia mensagem explicativa sobre tratamento de dados.
2. Usuário escolhe “Aceito” ou “Recuso”.
3. Sistema registra consentimento e prossegue ou encerra.

Fluxo alternativo: FA01: Usuário não responde → lembrar 1x e depois encerrar sessão temporária.

Mensagens: “Você aceita que usemos suas mensagens para melhor atendimento?”
[Aceito] [Recuso]”

Pós-condição: Consentimento registrado com timestamp; se recusado, limitações aplicadas.

UC09 – Editar Bot

Descrição: Alterar dados cadastrais e configurações de um bot existente.

Atores: Administrador.

Pré-condições: Bot existente; Admin autenticado.

Fluxo básico: Edita campos e salva; sistema valida e aplica.

Alternativo: Validação falha → mensagem de erro.

Mensagens: “Bot atualizado com sucesso.”

Pós-condição: Configurações aplicadas em tempo real.

UC10 – Excluir Bot

Descrição: Remover instância de bot (confirmação necessária).

Atores: Administrador.

Pré-condições: Permissão elevada; backup opcional.

Fluxo básico: Click “Excluir” → confirmação → exclusão (soft-delete preferível).

Fluxo alternativo: Cancelar → nada é alterado.

Mensagens: “Tem certeza? Esta ação não pode ser desfeita.” / “Bot excluído.”

Pós-condição: Bot removido (soft-delete com possibilidade de recuperação).

UC11 – Monitorar Conexões (APIs / Discord)

Descrição: Verificar saúde dos serviços externos (latência, erros).

Atores: Administrador.

Pré-condições: Monitoramento habilitado (CloudWatch / Datadog).

Fluxo básico: Dashboard mostra status; alerta em caso de falha.

Fluxo alternativo: FA01: Falha detectada → notificação por email/Slack.

Mensagens: “Conexão com API Kimi-k2 instável.”

Pós-condição: Alerta gerado; logs gravados.

UC12 – Gerenciar Usuários do Painel

Descrição: CRUD de usuários administrativos com RBAC.

Atores: Administrador.

Pré-condições: Admin master autenticado.

Fluxo básico: Cadastrar, editar, remover, atribuir roles.

Alternativo: Validação de e-mail duplicado.

Mensagens: “Usuário criado.” / “Permissão atualizada.”

Pós-condição: Controle de acesso atualizado.

UC13 – Histórico do Bot

Descrição: Visualizar histórico de conversas e interações para auditoria.

Atores: Administrador, Atendente.

Pré-condições: Logs armazenados (respeitando LGPD).

Fluxo básico: Pesquisar por usuário/periodo → visualizar conversas.

Fluxo alternativo: Dados apagados por política de retenção.

Mensagens: “Registro removido por política de retenção.”

Pós-condição: Histórico exibido ou mensagem de retenção aplicada.

UC14 – Regras de Atendimento

Descrição: Definir regras (p.ex. escalonamento por intenção, respostas padrão).

Atores: Administrador.

Pré-condições: Admin autenticado.

Fluxo básico: Criar/editar regra com condição e ação → salvar → aplicar.

Fluxo alternativo: Regra conflitua → alertar.

Mensagens: “Regra salva.” / “Conflito detectado.”

Pós-condição: Motor de orquestração aplica regra em tempo real.

UC15 – Logs e Auditoria

Descrição: Gerenciar logs, exportar e auditar ações sensíveis.

Atores: Administrador.

Pré-condições: Sistema de logging habilitado.

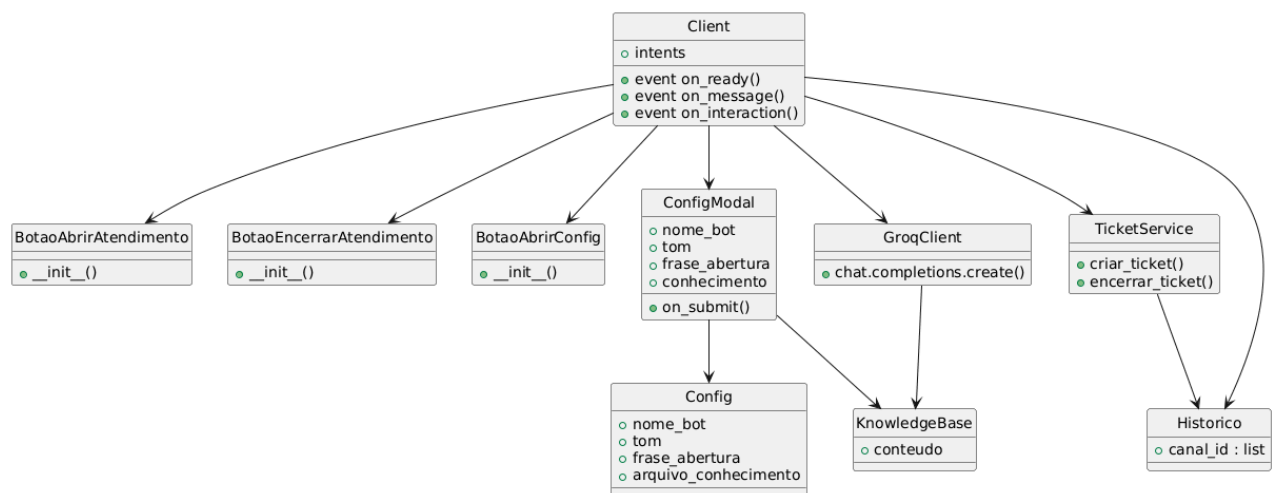
Fluxo básico: Filtrar logs por tipo/periodo → exportar → gerar relatório.

Fluxo alternativo: Falha na exportação → tentar novamente.

Mensagens: “Exportação concluída.”

Pós-condição: Relatório de auditoria disponível.

Diagrama de classe



4. Projeto

Arquitetura Lógica

O sistema é estruturado em três camadas principais, garantindo modularidade, escalabilidade e facilidade de manutenção:

Camada de Apresentação

Responsável pela interface com o usuário.

- Componentes: Painel administrativo web e interfaces de chat.
- Tecnologias: React.js e WebSocket.
- Função: Permite interação em tempo real entre usuário e sistema.

Camada de Aplicação

Responsável pela lógica de negócio e orquestração das conversas.

- Linguagem e Framework: Python com Flask ou FastAPI.
- Principais Módulos:
 - Gestor de Bots: Gerencia as instâncias e personalidades dos bots.
 - Conectores de Canais: Integra com Discord.
 - Prompt Engine: Controla os prompts e interações com o modelo de IA.
 - Escalonamento Humano: Direciona atendimentos para humanos quando necessário.
 - Métricas: Registra e monitora o desempenho das conversas e bots.
 - Integrações: APIs de IA (ex: Kimi-k2 e Grog).

Camada de Persistência

Gerencia o armazenamento e acesso aos dados.

- Banco de Dados: MongoDB (armazenamento não relacional e flexível).
- Cache e Sessões: Redis (armazenamento temporário).
- Padrão de Acesso: Repository Pattern, garantindo organização e manutenibilidade do código.

Padrões de Projeto Adotados

- Clean Architecture: separação clara entre camadas.
- MVC (Model-View-Controller): organização da aplicação.
- Strategy Pattern: configuração dinâmica de personalidades dos bots.
- Circuit Breaker: tratamento de falhas em chamadas externas.

Componentes Externos

- APIs de Mensagens: Discord.
- APIs de IA: Kimi-k2 e Grog.
- Serviços de Autenticação e Monitoramento.

Esses componentes funcionam de forma integrada, garantindo escalabilidade, segurança e confiabilidade do sistema.

4.2. Arquitetura Física

A aplicação será implantada em uma infraestrutura de nuvem escalável e segura, utilizando serviços gerenciados da AWS.

Infraestrutura

- Balanceador de Carga (AWS ALB): distribui o tráfego entre instâncias do backend (ECS/EKS).
- Servidor de Aplicação: contêineres executando Flask/FastAPI.
- Banco de Dados: MongoDB hospedado em instâncias gerenciadas.
- Redis (ElastiCache): cache e gerenciamento de sessões.
- Serviço de Inferência: modelos de IA (Kimi-k2) hospedados em instâncias GPU ou serviço externo.
- Sistema de Filas (SQS): processamento assíncrono e geração de relatórios.
- Monitoramento: CloudWatch, Datadog e ELK Stack.

Segurança e Conformidade

- Comunicação 100% via HTTPS/TLS.
- Credenciais e chaves protegidas no AWS Secrets Manager.
- Conformidade com a LGPD, incluindo:
 - Coleta de consentimento.
 - Criptografia de dados sensíveis.

- Exclusão automática de dados temporários após o atendimento.

Alta Disponibilidade e Escalabilidade

- Autoescalonamento de contêineres conforme demanda.
- Backups automáticos e replicação de dados.
- Recuperação automática em caso de falhas.
- Tempo médio de resposta: inferior a 3 segundos.

Resumo do Diagrama de Implantação

Usuário



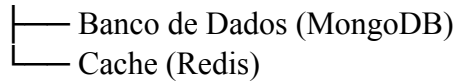
Internet



Balanceador de Carga (AWS ALB)



Servidor de Aplicação (Flask / FastAPI)

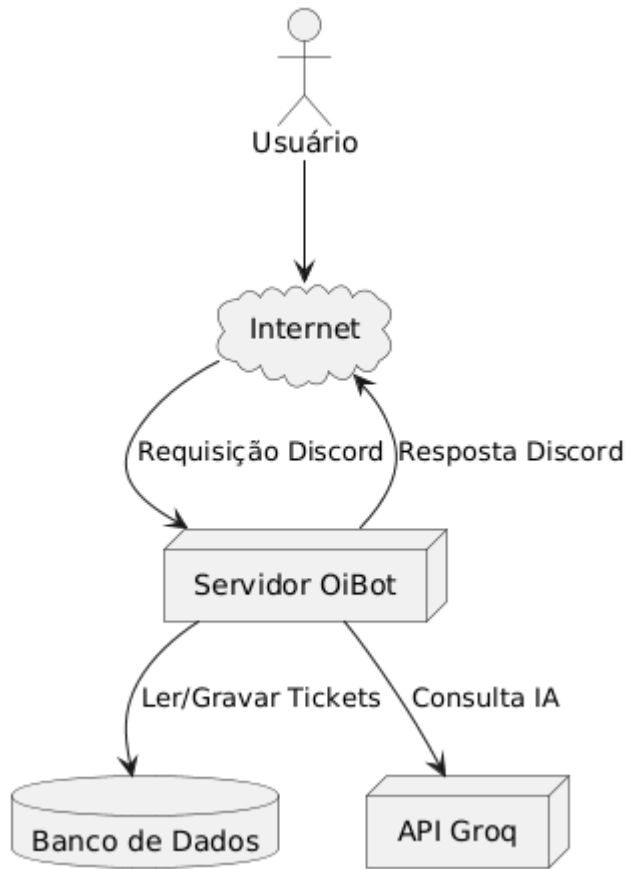


— Banco de Dados (MongoDB)

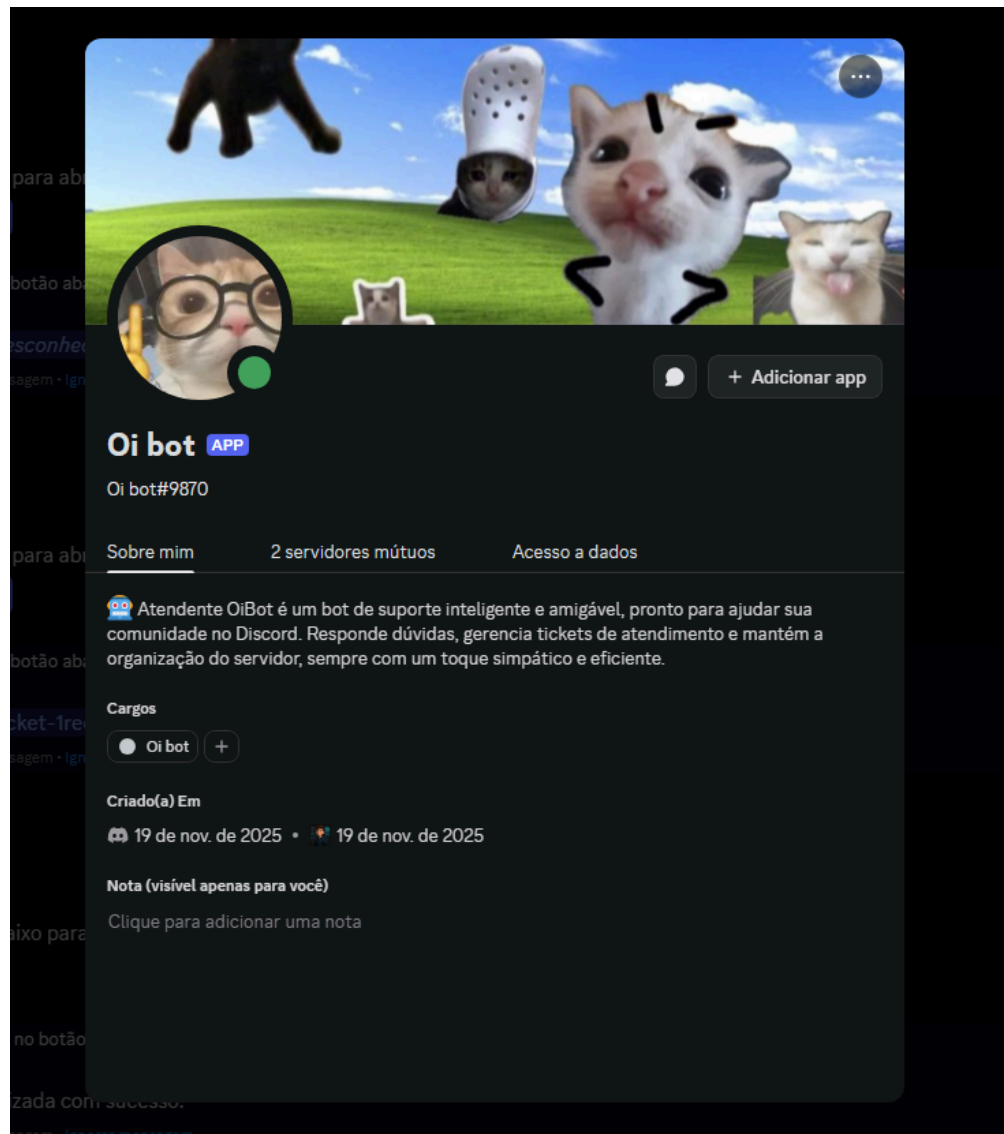
— Cache (Redis)

Componente	Serviço AWS sugerido	Função	Estimativa mensal (USD)
Load Balancer	ALB	Distribuir tráfego para backends	\$25
Backend (API)	ECS/Fargate ou EKS	Container FastAPI / Gunicorn	\$60
Banco de Dados	Documento DB /MongoDB Atlas	Persistência (conversas; configs)	\$80
Cache / Sessão	ElastiCache (Redis)	Sessões, contexto curto	\$30
Serviço de Interface	GPU EC2(opcional) ou API externa	Se Hospedar modelo local	\$200 (se GPU) / \$0 (se usar API externa)
Filas /Async	SQS	Processo assíncronos	\$10
Armazenamento estático	S3 + CloudFront	Frontend e assets	\$10
Secrets	Secrets Manager	Gerenciar chaves	\$5
Monitoramento	CloudWatch / Datadog	Logs / metricas / alertas	\$30
Total estimado (sem GPU)	_____	_____	≈\$250 / mes

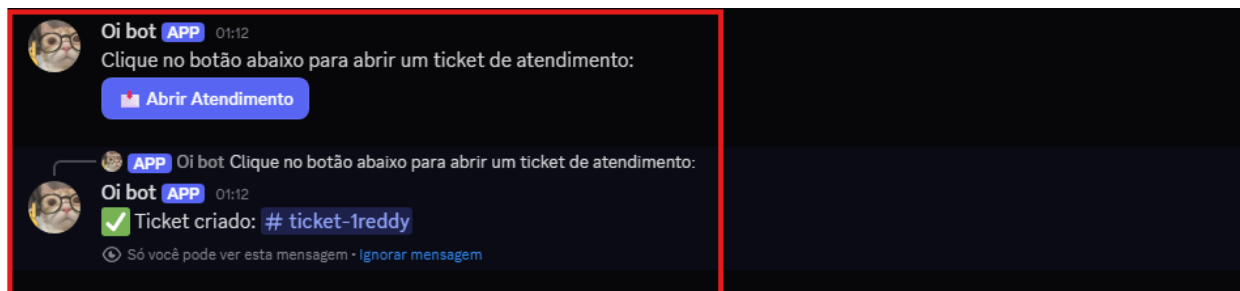
Diagrama de Funcionamento de Hardware

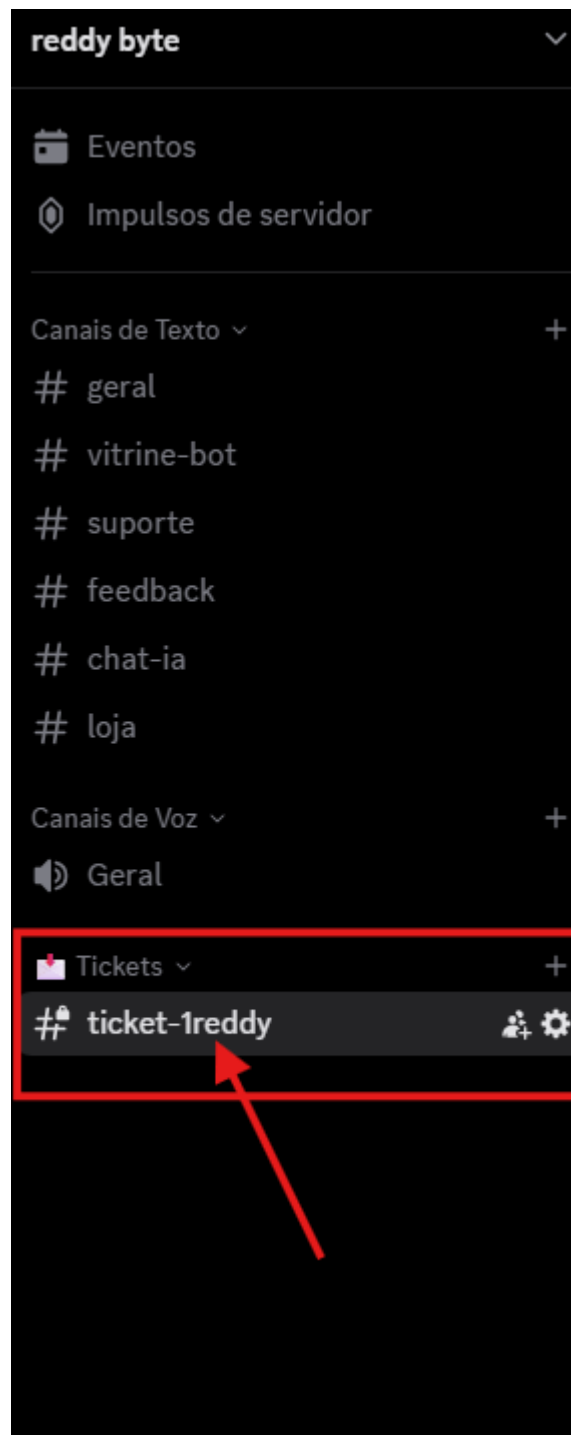


5. Protótipo de Interface

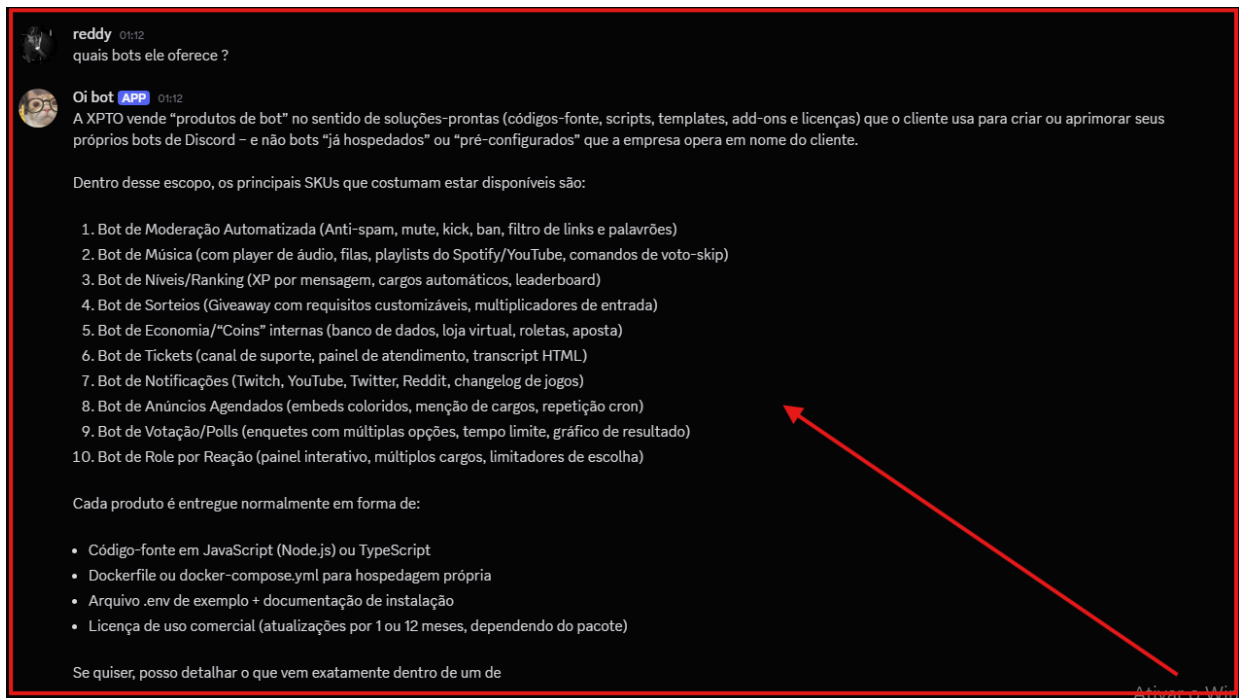


Apresentação do bot

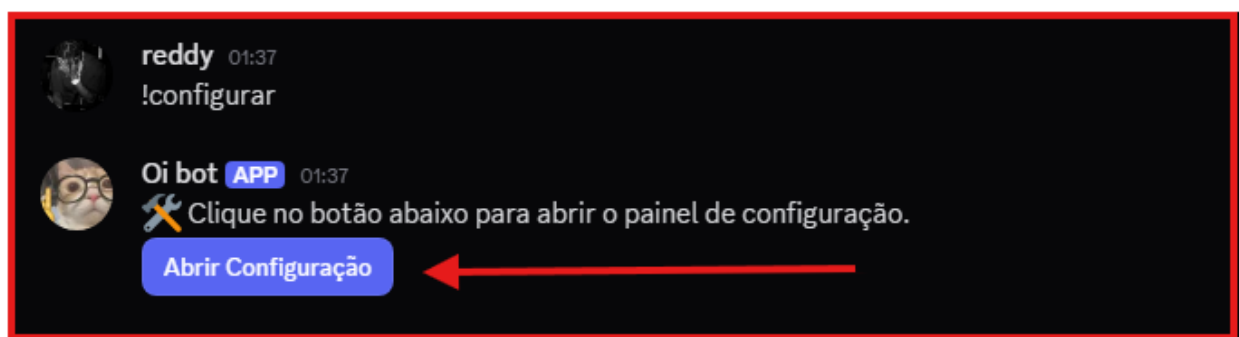




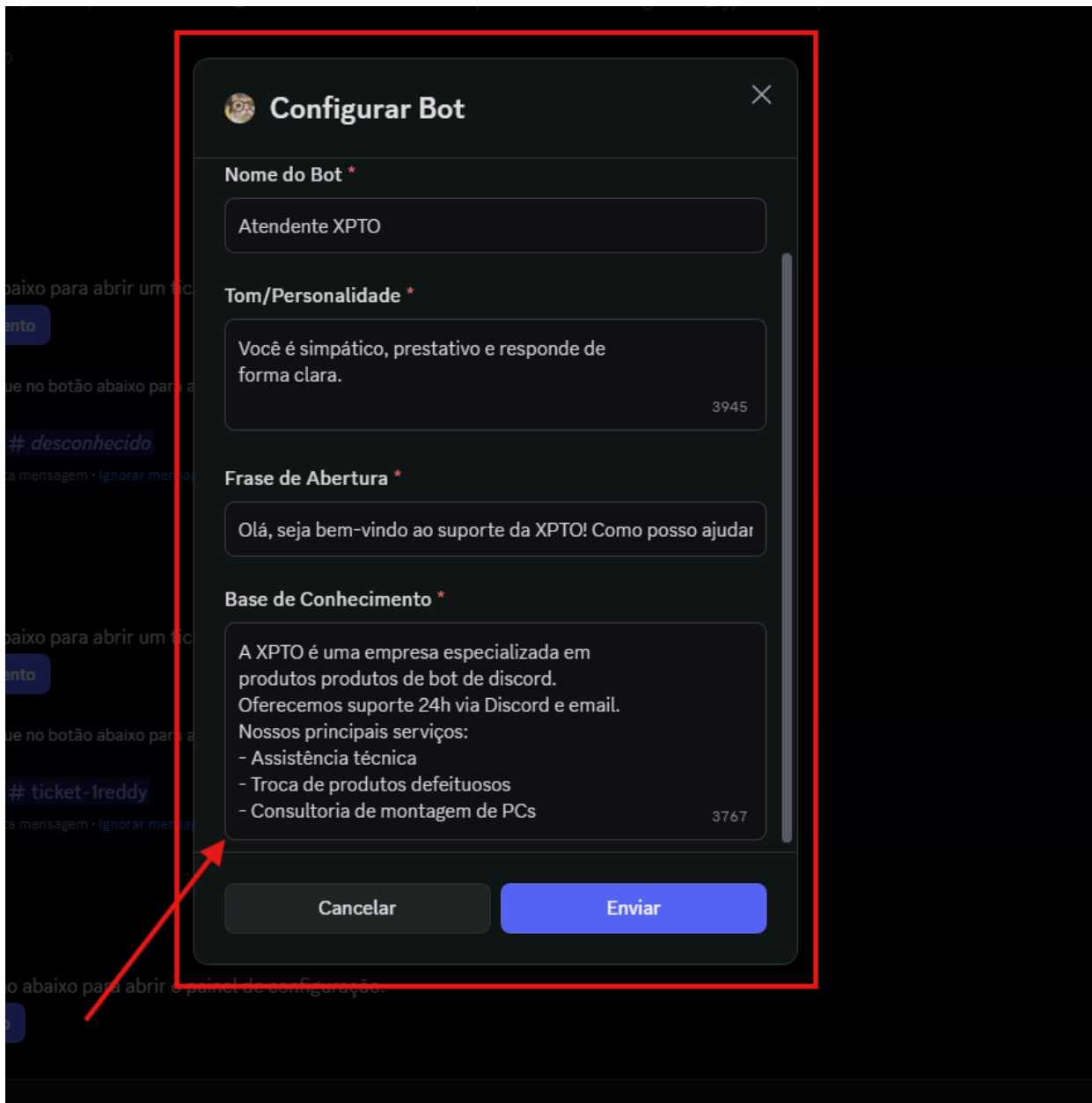
Criação de um ticket para possibilitar um atendimento organizado e segmentado no Discord.



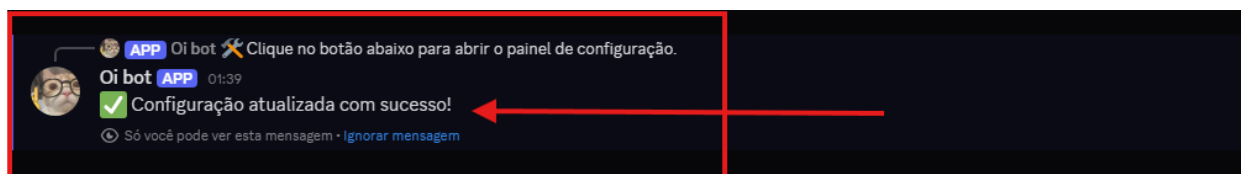
A resposta foi elaborada de forma adequada, considerando as informações fornecidas pela empresa e a nova imagem apresentada.



Botão de configuração para personalizar o bot



painel para configurar cada coisa do bot, personalidade, nome da empresa, como ele vai e tom de personalidade



6. Critérios de Qualidade de Software

Funcionalidade

- O sistema deve entregar todas as funções previstas, como criação de bots, personalização de personalidade, integração com Discord, painel administrativo e escalonamento humano.
- As respostas do bot devem corresponder às solicitações dos usuários.
- O sistema deve encaminhar corretamente atendimentos para operadores humanos quando necessário.
- Integrações com APIs externas (Discord, Kimi-k2, Grog) devem seguir padrões RESTful.
- Compatibilidade com navegadores modernos como Chrome, Firefox e Edge.

Confiabilidade

- Operação com alta disponibilidade.
- Monitoramento contínuo via ferramentas como CloudWatch, ELK e Datadog.
- Implementação de circuit breaker em chamadas externas.
- Recuperação automática em falhas temporárias.
- Capacidade de lidar com grandes volumes de requisições sem interrupções.
- Backups automáticos e logs estruturados para auditoria.

Desempenho e Eficiência

- O sistema deve responder em até 3 segundos em condições normais.
- Escalonamento automático via ECS/EKS.
- Testes de estresse garantindo estabilidade em picos de demanda.
- Uso eficiente de GPU/CPU para processamento de IA.

- Redis como cache para reduzir latência.

Usabilidade

- Painel administrativo de fácil navegação, responsivo e acessível.
- Administração de bots e personalidades sem necessidade de conhecimento técnico avançado.
- Adequação a práticas de acessibilidade conforme diretrizes WCAG.

Segurança

- Comunicação 100% criptografada via HTTPS/TLS.
- Proteção de dados sensíveis.
- Validação de entradas e proteção contra ataques como injection, XSS, CSRF e abusos de requisições.
- Uso de tokens seguros como JWT ou OAuth2.
- Controle de permissões por função (RBAC).
- Consentimento explícito do usuário.
- Armazenamento mínimo de dados.
- Remoção automática de conversas temporárias.

Manutenibilidade

- Arquitetura limpa (Clean Architecture) e MVC garantindo separação de responsabilidades.
- Estratégias de personalidade implementadas com Strategy Pattern.
- Logs estruturados e documentação clara via Swagger/OpenAPI.
- Suporte a testes unitários, de integração e testes de carga.

Portabilidade

- Possibilidade de migração entre provedores de nuvem com alterações mínimas.
- Deploy containerizado com Docker.
- Pipelines automatizados de CI/CD.

7. Testes

7.1. Plano de Testes

Objetivo: Validar requisitos funcionais e não-funcionais do OiBot.

Escopo

- Testes unitários: lógica de prompt, adaptadores Discord, regras de negócio.
- Testes de integração: integração com API de IA, Discord, DB.
- Testes de UI: painéis e formulários.
- Testes de desempenho: latência < 3s, carga máxima definida.
- Testes de segurança: XSS, CSRF, injection, autenticação.

Ambiente

- Backend: FastAPI/Flask (staging).
- DB: MongoDB staging.
- Redis: staging.
- Integrações: sandbox Kimi-k2, bot de teste no Discord.

Critérios de Aceitação

- Todos os testes unitários com > 90% de cobertura nas camadas críticas.
- Tempo médio de resposta ≤ 3 segundos (condições normais).

- Zero vulnerabilidades críticas em scan SAST.
- Retenção de logs compatível com LGPD (política aplicada).

Matriz de Testes (amostra)

- TF-01: UC03 — Integração Discord — Teste de conexão com token válido → Esperado: conexão OK.
- TF-02: UC04 — Processamento de Mensagens → Esperado: resposta válida em < 3s.
- TF-03: Segurança — Teste de SQL/NoSQL injection nos endpoints → Esperado: bloqueado.
- TF-04: Performance — 500 usuários simultâneos no chat → Esperado: sistema mantém SLA.

7.2. Roteiro de Testes

Cada caso indica passos, dados e resultado esperado.

Exemplo: Roteiro UC03 — Integrar com Discord

1. Pré-condição: token de teste válido.
2. Ação: Admin → Integrações → Discord → Inserir token → Testar.
3. Esperado: Status “Conectado”; webhook lista canais; log de conexão criado.

Exemplo: Roteiro UC04 — Processar Mensagens (fluxo feliz)

1. Pré: Bot ativo no canal #suporte.
2. Enviar mensagem: “Qual é o horário de atendimento?” (usuário teste)
3. Esperado: Bot responde com texto relevante em $\leq 3s$; log de interação gravado; consentimento verificado.

Exemplo: Roteiro UC07 — Escalonamento

1. Simular intenção “Falar com humano”.
2. Verificar enfileiramento e notificação ao atendente.
3. Esperado: Atendimento aceito e conversa transferida.

8.Referências

BRASIL. Lei nº 13.709, de 14 de agosto de 2018. Lei Geral de Proteção de Dados

Pessoais (LGPD). Disponível em:

https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm

DISCORD. Discord Developer Documentation. Disponível em:

<https://discord.com/developers/docs>

DISCORD.PY. Official Documentation. Disponível em:

<https://discordpy.readthedocs.io/>

GROQ. Groq API Documentation. Disponível em:

<https://console.groq.com/docs>

PYTHON SOFTWARE FOUNDATION. Python Documentation. Disponível em:

<https://docs.python.org/>

PLANTUML. Documentação oficial do PlantUML. Disponível em:

<https://plantuml.com/pt/>

AMAZON WEB SERVICES. AWS Documentation. Disponível em:

<https://docs.aws.amazon.com/>

PRESSMAN, Roger S.; MAXIM, Bruce R. Engenharia de Software: uma abordagem profissional. 9. ed. Porto Alegre: AMGH, 2021.