

Aura

1. Início

1.1. Introdução

Por que utilizar o AURA?

A plataforma **AURA** possibilita a criação de chatbots com capacidades variadas, que podem ser personalizados de acordo com o perfil do negócio ao qual irá atender.

A facilidade disponibiliza assistentes virtuais como um canal de atendimento direto para o recebimento de solicitações ou para a realização de serviços pré-determinados automatizando o fluxo de negócios.

Os ganhos são a redução do tempo de atendimento, menor envolvimento humano, redução de custos, garantia de investimentos e maior celeridade para os processos.

Personalização

Os bots criados podem ser configurados em sua aparência e estilo, tipos de saudação e outras características visuais, permitindo completa identificação e aderência da plataforma ao cliente.

O **AURA** também pode ser utilizado para diversas finalidades de atendimento, seja ativo ou reativo, segundo a necessidade da organização. A elucidação de dúvidas, o registro de solicitações e a automatização de processos entrega valor ao negócio de maneira fácil e eficiente.

Os mecanismos de linguagem de atendimento e os canais de atendimento também são totalmente personalizáveis, ou seja, você terá a liberdade para determinar através de quais

meios o assistente poderá ser acessado, como aplicativos de mensagens instantâneas (Whatsapp, Telegram, Microsoft Teams, Facebook Message), acesso web, através de portal, ou por meio de ferramentas de gestão.

Funcionalidades

O sistema oferece diversas funcionalidades voltadas para a otimização da interação com o chatbot e a análise de dados. Entre as principais funcionalidades estão a capacidade de anexar arquivos durante a conversa, reiniciar a sessão com o chatbot, expandir a interface para tela cheia, ou minimizar o chat.

Além disso, o usuário pode enviar transcrições por e-mail, imprimir o histórico ou exportá-lo em PDF. Também é possível ditar textos ou ouvir a sua leitura no Chrome, com opções de ajuste para a conversão de palavras e símbolos em áudio. A plataforma suporta a exibição de mensagens do pipeline no chat de atendimento humano, a conversão de áudio para texto (STT), e a reprodução de texto em áudio (TTS).

A análise de sentimentos, quando configurada, ajuda a identificar o tom da conversa, enquanto notificações indicam quando alguém está digitando. O tempo de sessão do chatbot e o tempo de corte da sessão podem ser configurados para gerenciar a duração das interações.

No desenho do fluxo, é possível determinar o ponto onde o transbordo para um atendimento humano é adequado, com possibilidade de herança do histórico da conversa permitindo atendimento mais eficaz e personalizado.

Controle de desempenho e gestão

No aspecto de análise, o sistema oferece um dashboard completo que apresenta dados em tempo real, como o número de conversas não atendidas, a assertividade das respostas do robô, e o total de mensagens. O dashboard também permite a análise de intenções e entidades, identificação do canal de comunicação, índice de confiança, e a visualização detalhada de cada mensagem. Além disso, o painel possibilita o agrupamento e ordenação dos dados, a exportação das mensagens, e a visualização de estatísticas diárias, como o total de mensagens e conversas por dia. O usuário pode ainda selecionar o período desejado para a pesquisa de dados.

O sistema oferece uma ferramenta para criar relatórios personalizados, com opções de classificação detalhada, gráficos e tabelas para análise. Além disso, permite gerenciar arquivos diretamente na plataforma, incluindo a criação, renomeação, exclusão e download de arquivos e pastas, além de recarregar a página para refletir as mudanças realizadas. Na área de uso e faturas, é possível gerenciar itens de cobrança, registrar consumo e gerar faturas, além de acompanhar o uso por meio de gráficos.

Demais funcionalidades

Outras funcionalidades incluem auditoria das atividades, atualização de perfil e senha, acesso a manuais de usuário, e configurações de idioma. A plataforma também permite automação de abertura de chamados e comunicação segura entre diferentes ambientes via VPN.

1.2. Lista de Funcionalidades do Citsmart Aura

Chatbots

Funcionalidades:

- **Novo Chatbot:** Permite ao usuário configurar múltiplos workspaces para definir os comportamentos desejados.
- **Excluir Chatbot:** Remove workspaces que não são mais necessários.
- **Dashboards:** Visualiza métricas importantes sobre as interações do chatbot.
 - **Download de dados:** Possibilita o download de dados selecionados por canal.
- **Listagem de históricos de Conversa:** Possibilita a visualização de todas as interações realizadas do chat.
 - **Ver conversa:** Funcionalidade para exibir a conversa encontrada na pesquisa, permitindo também a busca por palavras-chave dentro da conversa.

Comportamentos

Descrição:

Funcionalidade para registrar, gerenciar e configurar novos plugins, controlando como o sistema interage com diferentes eventos, interlocutores e messageiros.

Funcionalidades:

- **Registrar Novo Comportamento:** Permite ao usuário realizar o cadastro de um novo plugin.
- **Editar Parâmetros do Comportamento:** Possibilita ajustar os parâmetros de plugins já registrados.
- **Excluir Comportamento:** Remove plugins não utilizados.
- **Alterar interlocutores:** Modifica os interlocutores envolvidos nas interações controladas pelo plugin.

- **Configurar Mensageiros:** Define os canais de comunicação utilizados pelo plugin.
- **Configurar Eventos:** Permite configurar eventos que acionam respostas automáticas do sistema.

Atendimento Humano

Descrição:

Recebe e gerencia solicitações de atendimento provenientes de diversos canais como **Webchat, WhatsApp, Telegram, Microsoft Teams, Facebook Messenger e Slack.**

Atendimentos

- **Em espera:**
 - **Realizar atendimento:** Permite ao operador assumir a solicitação.
- **Em atendimento:**
 - **Chat:** Permite ao operador acessar o chat e interagir diretamente com o usuário.
- **Tela de interação com o Usuário:**
 - **Emoticons:** Permite que o operador envie figuras para expressar algo.
 - **Upload de arquivos:** Envio de arquivos e documentos durante o atendimento.
 - **Microfone:** Permite ao atendente enviar uma mensagem de áudio para o usuário.
 - **Atalhos:** Acesso aos atalhos públicos e privados previamente criados.
 - **Transferir atendimento:** Permite transferir a solicitação para outra fila ou atendente sem encerrar o atendimento.
 - **Finalizar Atendimento:** Confirma o encerramento do atendimento.
 - **User Info:** Permite o registro de informações do usuário.
 - **Histórico de conversas:** Opção para visualizar as conversas anteriores.
 - **Histórico de edição:** Exibe alterações feitas nas informações do usuário.
- **Finalizados:** Exibe o histórico de conversas do operador.

Filas

Descrição:

Permite a criação, configuração e gerenciamento das filas de atendimento.

Funcionalidades:

- **Adicionar:** Criação de uma nova fila de atendimento.
- **Editar:** Ajuste das configurações da fila.
- **Excluir:** Remoção de filas não necessárias.

Atalhos de Texto

Descrição:

A funcionalidade de atalhos de texto permite agilizar o atendimento humano por meio de respostas pré-definidas.

Funcionalidades:

- **Atalhos Globais:** Disponíveis para todos os atendentes, garantindo padronização nas respostas.
- **Atalhos Privados:** Personalizados para cada atendente, permitindo maior flexibilidade na comunicação.
- **Excluir:** Remoção de atalhos desnecessários.

NLU

Descrição:

Criação de Workspaces e Skills, com ambiente visual e de código para desenvolvimento de fluxos de conversas.

Funcionalidades:

Workspaces

- **Criar workspace:** Permite a criação de um novo ambiente de trabalho.
- **Renomear workspace:** Altera o nome de um workspace existente.
- **Excluir workspace:** Remove permanentemente um workspace.
- **Pesquisar:** Permite pesquisar workspaces criados.

Skills

- **Criar nova skill:** Adiciona uma nova skill ao workspace
- **Renomear skill:** Modifica o nome de uma skill existente.
- **Excluir skill:** Remove uma skill do workspace.

Editor Visual

- **Pesquisar:** Localiza informações dentro da skill.
- **Adicionar estado:** Insere um novo estado no fluxo conversacional.
- **Propriedades:** Permite configurar as propriedades do fluxo.
- **Exemplos:** Define exemplos de interações para treinar o fluxo.
- **Entidades:** Define as entidades usadas na conversa.
- **Testar:** Permite testar o fluxo antes de ativá-lo.
- **Nome do fluxo:** Define ou altera o nome do estado.
- **Fluxo global:** Permite que o estado seja acessado de qualquer ponto do bot.
- **Bloqueia avanço:** Impede que o fluxo prossiga sem determinada condição.
- **Menu:** Define nome da opção no fluxo.
- **Detectores:** Os detectores são palavras para chamar o estado.
- **Mensagens:** Define textos de resposta do bot.
- **Texto de dúvida:** Mensagem exibida quando a entrada do usuário não é clara.
- **Texto de dúvida completa:** Mensagem alternativa para esclarecer dúvidas mais complexas.
- **Automações:** Configura ações dentro do fluxo.

Editor de Código

- **Upload:** Permite enviar um arquivo de código para o editor.
- **Download:** Permite baixar o código editado para o dispositivo.
- **Testar:** Executa o código para verificar seu funcionamento.

Questionários e Ações

Descrição:

Ferramenta para criar interações conversacionais personalizadas via scripts em Lua ou editor visual baseado em Blockly.

Funcionalidades:

- **Adicionar novo workspace:** Cria um novo ambiente de trabalho.
- **Sincronizar todos:** Sincronizar os workspaces com o git.
- **Editar nome:** Altera o nome do workspace.
- **Visualizar logs:** Exibe os registros de problemas apresentados.
- **Excluir workspace:** Remove permanentemente o workspace.
- **Download:** Permite baixar todo o workspace.

Editor visual

- **Adicionar blocos:** Insere novos blocos no fluxo de trabalho.
- **Remover blocos:** Exclui blocos do fluxo.
- **Testar pipeline:** Executa o pipeline para validar o funcionamento.
- **Organizar todos os blocos:** Organiza os blocos de forma estruturada.
- **Visualizar todos os blocos:** Exibe todos os blocos presentes no fluxo.
- **Mapa do fluxo:** Exibe um mapa visual do fluxo e suas interações.

Código

- **Adicionar arquivos ou pastas:** Insere arquivos ou pastas no projeto.
- **Excluir arquivos ou pastas:** Remove arquivos ou pastas do projeto.

- **Testar:** Executa o código para verificar seu funcionamento.

Widget Personalizado

Descrição:

Interfaces configuráveis que permitem personalizações como cores, imagens e nomes.

Funcionalidades:

- **Adicionar novo widget:** Insere um novo widget na interface.
- **Excluir widget:** Remove um widget existente.
- **Editar widget:** Modifica as configurações de um widget.
- **Link para o chat aplicado:** Fornece um acesso direto ao chat para verificação do widget.

Widget personalizado por código

Descrição:

Possibilita alterar a configuração padrão do widget utilizando a ferramenta de edição de código, permitindo ao desenvolvedor personalizar diretamente o CSS do widget.

- **Upload:** Permite ao usuário enviar o seu widget personalizado.
- **Baixar template:** Permite baixar um modelo de um código base do widget para personalização.

Outras Funcionalidades

- **OIDC:** Integração com OpenID Connect para autenticação e autorização segura.
- **Agentes Cognitivos:** Utiliza inteligência artificial para realizar atendimentos e interações automatizadas.

- **Horário de Atendimento:** Define e gerencia os horários disponíveis para interação com o bot.
- **STT (Speech-to-Text):** Tecnologia que converte fala em texto, permitindo que o sistema transcreva mensagens de áudio para texto automaticamente.
- **TTS (Text-to-Speech):** Tecnologia que converte texto em fala, possibilitando que o sistema leia mensagens em voz alta para o usuário.
- **Timeout:** Define o tempo máximo que o chatbot deve aguardar uma resposta do usuário.

1.3. Papéis

- **Usuário final:** Usuário que conversa com o sistema por meio de um canal de acesso.
- **aura_attendant:** Usuário com as permissões necessárias para responder ao usuário final por meio do respectivo canal de acesso.
- **aura_supervisor:** Usuário com permissões para acessar relatórios e associar usuários a filas de atendimento.
- **aura_developer:** Usuário com as habilidades e permissões necessárias para criar, customizar e configurar fluxos conversacionais.

Usuários podem desempenhar mais de um papel conforme as necessidades específicas do negócio. Ex: Um supervisor pode ser cadastrado para fazer atendimentos como um atendente.

Permissões

Chatbots (aura_developer, aura_supervisor)

- comportamentos (aura_developer)
- dashboard (aura_supervisor)
- histórico (aura_supervisor)
- remover (aura_developer)
- novo (aura_developer)

Disparo (aura_supervisor)

- Templates (aura_supervisor)
- Envios (aura_supervisor)
- Contatos (aura_supervisor)

Atendimento Humano (aura_supervisor, aura_attendant)

- Atendimento (aura_attendant)
- Filas (aura_supervisor)
- Atalhos de Texto (aura_supervisor, aura_attendant)
 - Atalhos Globais (aura_supervisor)
 - Atalhos Privados (aura_attendant)

Máquina de estados (aura_developer)

Widgets (aura_developer)

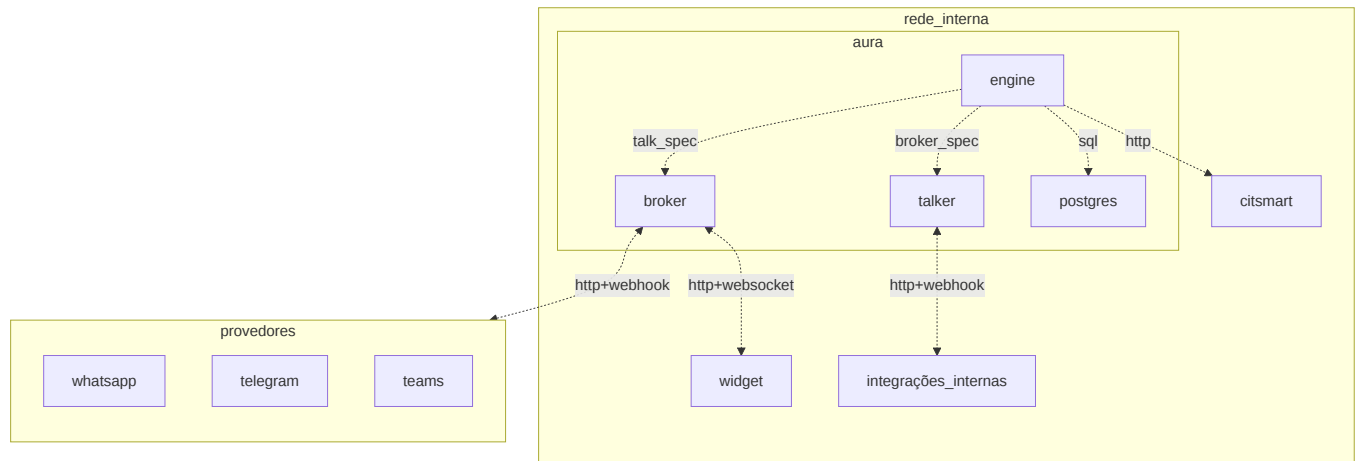
Questionários e Ações (aura_developer)

Ajuda (Todos)

Sobre (Todos)

1.4. Permissões de rede

No diagrama abaixo, apresentamos os componentes da solução. Cada um deles pode requerer permissões de rede adicionais para que suas funcionalidades sejam plenamente habilitadas.



- **Aura:** É o componente central do chat e requer permissões de rede interna para ser acessado pelo ambiente low-code do Citsmart.
- **Citsmart:** Deve estar acessível pela Aura na rede interna, permitindo que a Aura possa abrir chamados e executar outras integrações.
- **Widget:** Deve estar disponível na rede interna para que os usuários internos possam se comunicar com o chat.
- **Integrações Internas:** A Aura precisa de permissões de rede para acessar sistemas internos que o negócio identificar como necessários para integração.
- **Provedores:** Provedores como WhatsApp, Telegram, Microsoft Teams e Slack requerem que a Aura esteja exposta à internet para que possam estabelecer conexões com o webhook e notificar sobre novas mensagens enviadas pelos usuários. Esses provedores também exigem que a Aura tenha permissões para realizar requisições à internet, permitindo que o chat informe os usuários sobre novas mensagens.

Os sistemas da rede interna podem ser expostos à internet conforme as necessidades específicas do negócio. Exemplo: para que o widget esteja

acessível em um site público, é necessário que o componente Aura esteja exposto.

2. Instalação

2.1. Manual de Instalação de um novo ambiente

1. Pré-requisitos

Antes de realizar a instalação de um ambiente, certifique-se de que os seguintes pré-requisitos estão atendidos:

- **Acesso VPN:** Caso necessário, para conectar ao cluster do ambiente que será disponibilizado o Citsmart Aura.
- **Cluster Kubernetes:** Um cluster Kubernetes em execução, que pode ser Minikube, GKE, EKS, ou qualquer outra solução Kubernetes compatível.
- **kubectli:** Ferramenta de linha de comando para interagir com o cluster Kubernetes, já configurada para acessar o cluster. Siga as instruções no [site oficial do Kubernetes](#) para a instalação.
- **Banco de Dados:** PostgreSQL, no mínimo versão 16, com o usuário e banco de dados criados conforme a seção de configuração do banco de dados.
- **Domínio:** Necessário domínio personalizado configurado e disponibilizado.
- **CITSmart:**
 - front-manager: @0.34.0-2
 - backend: front-manager-api:2.10.2
 - lowcode: hyper-lowcode:1.8.6-RELEASE
 - citsmart: hyper-itsm-enterprise:CitSmart-CitsmartX-1.7.0

2. Criação do Namespace para o Projeto

Antes de iniciar a implantação dos serviços, é necessário criar um namespace para o projeto. O namespace isola todos os recursos, facilitando o gerenciamento e a organização do ambiente.

2.1 Criando o Namespace

Para criar o namespace, utilize o comando abaixo.

Substitua `aura-blocks` **pelo nome desejado para o seu projeto, se preferir outro nome:**

Text



```
kubectl create namespace aura-blocks
```

Este comando criará o namespace chamado `aura-blocks`, onde todos os recursos do projeto serão implantados.

Nota:

Ao longo deste manual, será utilizado o padrão `aura-blocks` para facilitar o acompanhamento dos exemplos. Caso você utilize um nome diferente, lembre-se de substituir `aura-blocks` em todos os comandos e exemplos apresentados.

2.2 Especificando o Namespace nos Manifests

Nos arquivos de manifesto YAML referentes a recursos de namespace, como Deployment, Service, ConfigMap e similares, certifique-se de que todos os recursos possuem o campo `namespace` com o valor correto.

Exemplo esperado nos trechos de YAML:

Text



```
metadata:  
  namespace: aura-blocks
```

Se tiver escolhido um nome diferente de namespace, substitua `aura-blocks` pelo nome utilizado em seu ambiente.

Importante:

Sempre utilize o mesmo nome de namespace em todos os arquivos de recursos associados ao seu ambiente para garantir que todos os componentes funcionem corretamente.

Dica para Atualizar o Namespace nos Manifestos

Se optar por um nome de namespace diferente de `aura-blocks`, será necessário atualizar os arquivos de manifesto (`*.yaml`) para refletir esse novo nome.

Você pode automatizar essa tarefa antes de aplicar os manifests, executando o comando a seguir no diretório onde estão armazenados:

Text



```
NAMESPACE="seu-namespace-aqui" && find . -type f -name "*.yaml" -exec bash -c
```

Substitua `seu-namespace-aqui` pelo nome que escolheu para o namespace.

Atenção:

Nem todos os recursos do Kubernetes utilizam o campo `namespace:` – por exemplo, arquivos de definição de `Namespace`, `PersistentVolume` e recursos do tipo `ClusterRole` não possuem esse campo ou são de escopo global. Sempre revise seus arquivos YAML, especialmente se estiver usando manifests que abrangem diferentes tipos de recurso.

Para localizar rapidamente onde o campo aparece, use:

Após rodar o comando de substituição, é recomendável revisar os arquivos alterados antes de prosseguir com a aplicação.

Text



```
grep -r 'namespace:' .
```

3. Configuração do Banco de Dados

Antes de prosseguir, crie o usuário e o banco de dados no PostgreSQL:

Substitua `<user>` e `<pass>` pelas informações necessárias para a criação do usuário.

SQL



```
CREATE USER <user> WITH PASSWORD '<pass>' CREATEDB LOGIN;  
CREATE DATABASE aura_blocks OWNER <user>;  
GRANT CREATE ON DATABASE aura_blocks TO <user>;
```

4. Obtendo manifestos

Os manifestos podem ser obtidos no arquivo [aura-blocks-manifests.tar.bz2](#) oferecidos junto ao presente manual.

Bash



```
tar -xf aura-blocks-manifests.tar.bz2
```

Caso queira, também é possível obter os manifestos originais no repositório oficial do projeto Citsmart Aura:

Bash



```
git clone http://gitlab.centralit.io/aura/infra/kubernetes.git  
cd kubernetes/template
```

5. Estrutura dos Diretórios dos Manifestos

Abaixo está a estrutura dos diretórios dos manifestos.


```
.
├─ attendant
│   ├── configmap.yaml
│   ├── deployment.yaml
│   ├── secret.yaml
│   └─ service.yaml
├─ dash
│   ├── configmap.yaml
│   ├── deployment.yaml
│   ├── secret.yaml
│   └─ service.yaml
├─ engine
│   ├── configmap.yaml
│   ├── deployment.yaml
│   ├── secret.yaml
│   └─ service.yaml
├─ ingress.yaml
├─ lunaris
│   ├── deployment.yaml
│   └─ service.yaml
├─ persistentvolumeclaim.yaml
├─ proactive
│   ├── configmap.yaml
│   ├── deployment.yaml
│   ├── secret.yaml
│   └─ service.yaml
├─ proxy
│   ├── configmap.yaml
│   ├── deployment.yaml
│   └─ service.yaml
├─ teams-server
│   ├── configmap.yaml
│   ├── deployment.yaml
│   └─ service.yaml
├─ wanderson
│   ├── configmap.yaml
│   ├── deployment.yaml
│   ├── secret.yaml
│   └─ service.yaml
└─ widget
    ├── configmap.yaml
    ├── deployment.yaml
    ├── secret.yaml
    └─ service.yaml
```

6. Configuração dos Manifestos do Kubernetes

Nesta seção, detalharemos o que deve ser alterado em cada arquivo para configurar corretamente os serviços.

6.1 Configuração dos Volumes Persistentes (PVC)

Alguns serviços do projeto **Citsmart Aura**, como `lunaris`, `wanderson`, `dash`, `engine` e `widget` requerem armazenamento persistente de dados (ex: logs, workspaces). Para isso, utiliza-se um **PersistentVolumeClaim (PVC)** que reserva espaço de armazenamento no cluster Kubernetes.

O arquivo `persistentvolumeclaim.yaml` define a configuração desse volume compartilhado.

⚠ **Atenção:** Antes de aplicar o PVC no cluster, é necessário ajustar campos obrigatórios como o `storageClassName`, o tipo de acesso (`accessModes`) e o tamanho do volume (`storage`).

6.1.1 persistentvolumeclaim.yaml

Campo	Descrição
<code>storageClass</code> <code>Name</code>	Define a classe de armazenamento que será usada para provisionar o volume. Esse valor depende do provisionador instalado no cluster. Exemplos comuns: <code>"standard"</code> , <code>"nfs-client"</code> , <code>"balanced"</code> , <code>"ssd"</code> . Substitua <code><storageClassName></code> pelo nome correto de sua configuração.
<code>accessModes</code>	Define o modo de acesso ao volume. Neste caso, o valor <code>ReadWriteMany</code> permite que vários pods acessem o volume simultaneamente em leitura e escrita — necessário para serviços que compartilham espaço como <code>lunaris</code> , <code>wanderson</code> , <code>dash</code> , <code>engine</code> e <code>widget</code> . Verifique se seu <code>StorageClass</code> suporta esse modo.
<code>resources.requests.storage</code>	Define a capacidade mínima de armazenamento a ser reservada para o volume. O valor padrão é <code>50Gi</code> . Recomenda-se um mínimo de 20Gi , ajustável conforme o ambiente e carga esperada.

Exemplo com os ajustes recomendados:

YAML

```

storageClassName: nfs-client
accessModes:
  - ReadWriteMany
resources:
  requests:
    storage: 50Gi

```

📌 **Recomendação:** Para ambientes de produção ou homologação com múltiplos serviços, recomenda-se reservar ao menos **20Gi de espaço**, podendo ser ajustado conforme a necessidade (ex: `50Gi`, `100Gi`, etc.).

Após os ajustes, aplique o PVC com o comando:

Bash



```
kubectl apply -f persistentvolumeclaim.yaml
```

▮ Você pode verificar se o volume foi corretamente provisionado com:

Bash



```
kubectl get pvc -n aura-blocks
```

6.2 Configuração da Engine

6.2.1 engine/configmap.yaml

O arquivo `engine/configmap.yaml` define variáveis de ambiente essenciais para o funcionamento do serviço **engine**, controlando desde as conexões com o banco de dados até rotas internas de API e autenticação.

Abaixo está a descrição detalhada de cada uma dessas variáveis:

Variável	Descrição
DB_TYPE	Define o tipo de banco de dados utilizado. Valor esperado: <code>postgres</code> .
DB_NAME	Nome do banco de dados utilizado pela aplicação (ex: <code>aura_blocks</code>).
DB_PORT	Porta de comunicação com o banco de dados (ex: <code>"5432"</code> para PostgreSQL).
DB_HOST	Hostname ou IP onde o banco de dados está hospedado. Substitua <code><db_host></code> conforme necessário.
DB_USER	Nome de usuário para autenticação no banco de dados. Substitua <code><db_user></code> conforme necessário.
HTTP_PORT	Porta HTTP onde o serviço engine será exposto internamente. Valor padrão: <code>"6060"</code> .
HTTP_BASE	Prefixo base da API da engine. Deve coincidir com a configuração do Ingress (ex: <code>"/engine/"</code>).
DOCS_PATH	Caminho relativo para acesso à documentação do Citsmart Aura (ex: <code>"/docs"</code>).
MANUALS_PATH	Caminho relativo para acesso a manuais do Citsmart Aura (ex: <code>"/manuals"</code>).
TESTS_PATH	Caminho relativo para acesso ao roteiro de testes Citsmart Aura (ex: <code>"/roteiro_testes"</code>).
BASIC_AUTH_USERNAME	Nome de usuário para autenticação básica na rota de documentação (<code>/docs</code>).
ATTACHMENT_PATH	Caminho interno relativo para armazenamento de anexos (ex: <code>"/engine/attachments"</code>).
LUNARIS_URL	Caminho interno relativo para integrar com o serviço Lunarix (ex: <code>"/lunarix"</code>).
FINAL_URL	URL pública final para acesso à engine (ex: <code>https://domain.com/engine/</code>).
USER	Usuário utilizado para autenticação da API. Valor padrão: <code>"lowcode"</code> .

Variável	Descrição
KEYCLOAK_URL	URL para obtenção de informações de usuário via Keycloak. Essa URL inclui o <code>realm</code> e o caminho do endpoint <code>userinfo</code> do OpenID Connect.
DEFAULT_MANIFESTS	Lista padrão de manifestos (em JSON) a serem carregados pela engine ao iniciar. Cada item contém: Nome, URL do manifesto e tipo (<code>Talker</code> , <code>Broker</code> , <code>Event</code> , etc.). Abaixo está a estrutura completa:

JSON	
	<pre>[["Máquina de estados", "https://domain.com/wanderson/plugin/manifest", "Talker"], ["Questionários e ações", "https://domain.com/lunaris/.well-known/lunaris-config", "Event"], ["Botmaker", "https://domain.com/lunaris/get/botmaker/manifest", "Broker"], ["Atendimento Humano", "https://domain.com/attendant/manifest", "Talker"], ["Gupshup", "https://domain.com/lunaris/get/gupshup/manifest", "Broker"], ["Messenger", "https://domain.com/lunaris/get/messenger/manifest", "Broker"], ["Positus", "https://domain.com/lunaris/get/positus/manifest", "Broker"], ["Slack", "https://domain.com/lunaris/get/slack/manifest", "Broker"], ["Teams", "https://domain.com/teams_server/manifest", "Broker"], ["Telegram", "https://domain.com/lunaris/get/telegram/manifest", "Broker"], ["Twilio", "https://domain.com/lunaris/get/twilio/manifest", "Broker"], ["Widget", "https://domain.com/widget/manifest", "Broker"], ["OIDC", "https://domain.com/lunaris/get/oidc/manifest", "Event"], ["UserInfo", "https://domain.com/lunaris/get/userinfo/manifest", "Userinfo"]]</pre>

⚠ **Atenção:** Todos os valores com `https://domain.com` devem ser substituídos pelo domínio **real** onde a aplicação será instalada (ex: `https://meusistema.empresa.com.br`).

6.2.2 engine/secret.yaml

Este arquivo contém variáveis sensíveis utilizadas pelo serviço **engine**. As informações estão codificadas em Base64 e não devem ser versionadas em repositórios públicos. As variáveis definidas são:

Variável	Descrição
<code>DB_PASS</code>	Senha de acesso ao banco de dados PostgreSQL utilizada pela aplicação.
<code>BASIC_AUTH_PASSWORD</code>	Senha para autenticação básica no acesso à rota <code>/docs</code> da documentação.
<code>PASS</code>	Senha utilizada para autenticação da API.

- Substitua senhas codificadas com:

Bash

```
echo -n 'minha_senha' | base64
```

⚠ **Atenção:** Todos os valores estão codificados em Base64. Para editar ou revisar, utilize ferramentas para codificar/decodificar com segurança.

Para ambientes sensíveis, considere usar `SealedSecrets` ou `SOPS`.

6.2.3 engine/deployment.yaml

O `deployment.yaml` é responsável por definir o deployment do serviço `engine`. Certifique-se de que o **namespace** e as **imagens** do container estão corretas e que os **ConfigMaps** e **Secrets** estão sendo referenciados corretamente.

6.3 Configuração dos Outros Serviços

6.3.1 attendant

attendant/configmap.yaml

O arquivo `attendant/configmap.yaml` define as variáveis de ambiente utilizadas pelo serviço **attendant**. Este item refere-se **exclusivamente** à configuração do `ConfigMap`. As variáveis disponíveis são:

Variável	Descrição
<code>DB_TYPE</code>	Tipo de banco de dados utilizado (ex: <code>postgres</code>).
<code>DB_NAME</code>	Nome do banco de dados acessado pela aplicação (ex: <code>aura_blocks</code>).
<code>DB_PORT</code>	Porta de comunicação com o banco de dados (ex: <code>"5432"</code> para PostgreSQL).
<code>DB_HOST</code>	Hostname ou serviço interno onde o banco de dados está acessível (ex: <code>postgresql</code>).
<code>DB_USER</code>	Usuário do banco de dados.
<code>HTTP_PORT</code>	Porta HTTP em que o serviço será exposto internamente (ex: <code>"6060"</code>).
<code>HTTP_BASE</code>	Caminho base da API do serviço <code>attendant</code> . Deve coincidir com a configuração do Ingress (ex: <code>"/attendant/"</code>).
<code>ENGINE_URL</code>	URL do serviço engine utilizado pelo <code>attendant</code> (ex: <code>"https://domain.com/engine/"</code>).
<code>BASE_URL</code>	URL base da aplicação (ex: <code>"https://domain.com"</code>).
<code>AMBIENTE</code>	URL base do Citsmart, onde será disponibilizada a aplicação. (ex: <code>"https://hml-cocriar.cithyper.click"</code>)
<code>SLEEP_TIME</code>	Intervalo (em segundos) para processamento das mensagens das filas de atendimento. Ex: <code>"120"</code> representa 2 minutos.
<code>USER</code>	Usuário utilizado para autenticação da API. Valor padrão: <code>"lowcode"</code> .
<code>KEYCLOAK_URL</code>	URL para obtenção de informações do usuário via Keycloak (OpenID Connect - endpoint <code>/userinfo</code>).

attendant/secret.yaml

O arquivo `attendant/secret.yaml` armazena informações sensíveis utilizadas pelo serviço. Os dados estão codificados em Base64.

Variável	Descrição
<code>PASS</code>	Senha codificada em Base64 para autenticação da API.
<code>DB_PASS</code>	Senha do banco de dados associada ao usuário definido em <code>DB_USER</code> .

⚠ **Atenção:** As informações armazenadas no `Secret` devem ser tratadas com confidencialidade. Evite exposição em logs ou repositórios.

6.3.2 lunaris

lunaris/deployment.yaml

O arquivo `lunaris/deployment.yaml` define o deployment do serviço **lunaris**. Um dos pontos mais importantes a ser ajustado está no campo `command` , que define os argumentos utilizados na inicialização do container.

Campo	Descrição
<code>command</code>	<p>Define os parâmetros passados ao iniciar o serviço Lunar. Os principais valores que precisam ser alterados são:</p> <ul style="list-style-type: none"> - <code>-u</code> (URL pública do serviço Lunar): deve ser substituída por uma URL válida do domínio onde o serviço será disponibilizado. - <code>-o</code> (URL do Keycloak para autenticação OpenID Connect): também precisa ser ajustada para refletir o domínio real do ambiente (incluindo o <code>realm</code>).

Exemplo original:

Bash

```
-u https://domain.com/lunaris
-o https://keycloak-domain.com/auth/realms/<realm>/protocol/openid-connect/us
```

Exemplo com domínio ajustado:

Bash



```
-u https://aura-blocks.centralit.com.br/lunaris  
-O https://hml-sso.cithyper.click/auth/realms/hml-cocriar/protocol/openid-con
```

⚠ **Atenção:** Substitua os valores `domain.com`, `keycloak-domain.com` e `<realm>` pelo domínio e realm reais utilizados no ambiente de homologação ou produção.

6.3.3 wanderson

wanderson/configmap.yaml

O arquivo `wanderson/configmap.yaml` define variáveis de ambiente utilizadas pelo serviço **wanderson**. Este item refere-se **exclusivamente** à configuração do `ConfigMap`.

Variável	Descrição
<code>TZ</code>	Define o fuso horário utilizado no container (ex: <code>"America/Sao_Paulo"</code>).
<code>DATABASE_HOST</code>	Hostname do banco de dados. Deve ser substituído por um valor real (ex: <code>postgresql</code>).
<code>DATABASE_PORT</code>	Porta para conexão com o banco de dados (ex: <code>"5432"</code>).
<code>DATABASE_USERNAME</code>	Nome do usuário de acesso ao banco de dados.
<code>DATABASE_NAME</code>	Nome do banco de dados utilizado (ex: <code>aura_blocks</code>).
<code>DATABASE_MIN_POOL_SIZE</code>	Tamanho mínimo do pool de conexões.
<code>DATABASE_MAX_POOL_SIZE</code>	Tamanho máximo do pool de conexões.
<code>PORT</code>	Porta interna utilizada pelo serviço (ex: <code>"80"</code>).
<code>LOG_LEVELS</code>	Níveis de log habilitados no serviço (ex: <code>['"error"', "warn", "log", "debug"]'</code>).
<code>CORS_ALLOWED_ORIGIN</code>	Origem permitida para requisições CORS (ex: <code>"*"</code> permite todas).
<code>CORS_ALLOWED_METHODS</code>	Métodos HTTP permitidos via CORS (ex: <code>"*"</code>).
<code>CORS_ALLOWED_HEADERS</code>	Headers permitidos via CORS (ex: <code>"*"</code>).
<code>CORS_ALLOW_CREDENTIALS</code>	Habilita envio de cookies/autenticação via CORS (ex: <code>"true"</code>).

wanderson/secret.yaml

O arquivo `wanderson/secret.yaml` armazena informações sensíveis utilizadas pelo serviço, codificadas em Base64.


Variável	Descrição
<code>DATABASE_PASSWORD</code>	Senha utilizada em conjunto com <code>DATABASE_USERNAME</code> para autenticação no banco de dados.

wanderson/deployment.yaml

O arquivo `deployment.yaml` do serviço **wanderson** define diversos parâmetros do container, incluindo o campo `command`, que deve ser ajustado conforme o domínio do ambiente.

Parâmetro	Descrição
<code>-o</code>	Define a URL do endpoint <code>userinfo</code> do Keycloak para autenticação via OpenID Connect. Deve ser ajustada para refletir o domínio e o <code>realm</code> corretos do ambiente.
<code>-u</code>	Define a URL pública base do serviço wanderson . Essa URL deve corresponder à configuração de Ingress (por exemplo: <code>https://aura-blocks.centralit.com.br/wanderson/</code>).

Exemplo original:

Bash	
<pre>-o https://keycloak-domain.com/auth/realms/<realm>/protocol/openid-connect/userinfo -u https://domain.com/wanderson/</pre>	

Exemplo ajustado:

Bash	
<pre>-o https://hml-sso.cithyper.click/auth/realms/hml-cocriar/protocol/openid-connect/userinfo -u https://aura-blocks.centralit.com.br/wanderson/</pre>	

⚠ **Atenção:** Ambos os valores – `-O` (Keycloak) e `-u` (URL pública) – devem ser substituídos pelo domínio real do ambiente em que o serviço será executado, respeitando o caminho do Ingress e o realm de autenticação.

6.3.4 teams-server

teams-server/configmap.yaml

O arquivo `teams-server/configmap.yaml` define variáveis de ambiente utilizadas pelo serviço **teams-server**. Este item refere-se **exclusivamente** à configuração do `ConfigMap`.

Variável	Descrição
<code>HTTP_PORT</code>	Porta HTTP utilizada internamente pelo serviço (ex.: <code>"6969"</code>).
<code>BASE_URL</code>	URL base pública da aplicação. Deve ser substituída pelo domínio real da aplicação (ex.: <code>"https://aura-blocks.centralit.com.br"</code>).
<code>SUBPREFIX</code>	Caminho base da rota do serviço no Ingress (ex.: <code>"/teams_server/"</code>).
<code>URL_TO_ENGINE</code>	Endpoint utilizado para repassar mensagens recebidas ao serviço engine . Substitua <code>domain.com</code> pela URL real do engine (ex.: <code>"https://aura-blocks.centralit.com.br/engine"</code>).

⚠ **Atenção:** Os valores que utilizam `domain.com` são placeholders e **devem ser substituídos** pelo domínio oficial da aplicação no ambiente de homologação ou produção.

6.3.5 widget

widget/configmap.yaml

O arquivo `widget/configmap.yaml` define as variáveis de ambiente utilizadas pelo serviço **widget**. Este item refere-se **exclusivamente** à configuração do `ConfigMap`.

Variável	Descrição
DB_TYPE	Tipo de banco de dados utilizado (ex: <code>postgres</code>).
DB_NAME	Nome do banco de dados utilizado pelo serviço (ex: <code>widget</code>).
DB_PORT	Porta utilizada para conexão com o banco de dados (ex: <code>"5432"</code>).
DB_HOST	Hostname do banco de dados. Deve ser substituído por um valor real.
DB_USER	Nome de usuário para autenticação no banco de dados.
HTTP_PORT	Porta HTTP onde o serviço será exposto internamente (ex: <code>"8989"</code>).
HTTP_BASE	Caminho base da API do widget (ex: <code>"/widget/"</code>). Deve coincidir com a configuração do Ingress.
BASE_URL	URL base da aplicação para chamadas HTTP. Substituir <code>domain.com</code> pelo domínio real do ambiente.
BASE_WEBSOCKET_URL	URL base para conexões WebSocket (ex: <code>"ws://domain.com"</code>). Substituir pelo domínio correto.
ENGINE_URL	URL do serviço engine para integração (ex: <code>"https://domain.com/engine"</code>).
ATTACHMENT_PATH	Caminho relativo para anexos enviados/recebidos (ex: <code>"/attachments"</code>).
DOCS_PATH	Caminho interno para a documentação (ex: <code>"../..../docs/book"</code>).
REF_PATH	Caminho base para referências utilizadas pelo widget (ex: <code>"/ref/"</code>).
WIDGETS_PATH	Caminho para os recursos de widgets (ex: <code>"/widgets/"</code>).
USER	Nome de usuário utilizado internamente para autenticação da API (ex: <code>"lowcode"</code>).
TIME_KEEPALIVE	Tempo em segundos para envio do evento para manter a conexão ativa (ex: <code>"30"</code>).

⚠ **Atenção:** Todos os campos que contêm `domain.com` devem ser substituídos pelo domínio oficial da aplicação (HTTP e WebSocket).

widget/secret.yaml

O arquivo `widget/secret.yaml` armazena informações sensíveis utilizadas pelo serviço, codificadas em Base64.

Variável	Descrição
<code>DB_PASS</code>	Senha utilizada em conjunto com <code>DB_USER</code> para autenticação no banco de dados.
<code>PASS</code>	Senha codificada em Base64, utilizada para autenticação da API.

⚠ **Atenção:** Nunca armazene os valores decodificados em repositórios. Manipule os segredos com segurança e mantenha-os restritos ao ambiente controlado.

6.3.6 dash

dash/configmap.yaml

O arquivo `dash/configmap.yaml` define as variáveis de ambiente utilizadas pelo serviço **dash**, responsável por relatórios e visualizações. Este item refere-se **exclusivamente** à configuração do `ConfigMap`.

Variável	Descrição
<code>DB_TYPE</code>	Tipo de banco de dados utilizado (ex: <code>postgres</code>).
<code>DB_NAME</code>	Nome do banco de dados acessado pela aplicação (ex: <code>aura_blocks</code>).
<code>DB_PORT</code>	Porta de conexão com o banco de dados (ex: <code>"5432"</code>).
<code>DB_HOST</code>	Hostname onde o banco de dados está disponível (ex: <code>postgresql</code>).
<code>DB_USER</code>	Nome do usuário utilizado na autenticação do banco.
<code>HTTP_PORT</code>	Porta onde o serviço será exposto internamente (ex: <code>"2610"</code>).
<code>HTTP_BASE</code>	Caminho base da API do serviço <code>dash</code> (ex: <code>"/dash/"</code>). Deve estar alinhado com a configuração do Ingress.
<code>REPORTS_PATH</code>	Caminho interno utilizado para armazenamento e acesso aos relatórios gerados (ex: <code>"/reports"</code>).
<code>GIT_ENABLED</code>	Define se a funcionalidade de integração com repositório Git está habilitada (<code>true</code> ou <code>false</code>). Quando habilitado, o serviço tentará sincronizar relatórios com o repositório definido.
<code>GIT_REPO_URL</code>	URL do repositório Git que será utilizado para armazenar e versionar os relatórios (ex: <code>https://gitlab.com/empresa/relatorios.git</code>).
<code>GIT_BRANCH</code>	Nome do branch padrão no repositório Git onde os relatórios serão sincronizados (ex: <code>main</code>).
<code>GIT_USERNAME</code>	Nome de usuário que será utilizado para autenticação no repositório Git.

dash/secret.yaml

O arquivo `dash/secret.yaml` armazena informações sensíveis do serviço, codificadas em Base64.

Variável	Descrição
<code>DB_PASS</code>	Senha utilizada em conjunto com <code>DB_USER</code> para autenticação no banco de dados.
<code>GIT_PASSWORD</code>	Senha ou Personal Access Token (PAT) utilizado para autenticação no repositório Git. Devido à segurança, recomenda-se sempre utilizar PAT em vez de senha simples.

⚠ **Atenção:** Os segredos devem ser tratados com segurança e não devem ser versionados em repositórios públicos.

6.3.7 proactive

proactive/configmap.yaml

O arquivo `proactive/configmap.yaml` define as variáveis de ambiente utilizadas pelo serviço **proactive**. Este item refere-se **exclusivamente** à configuração do `ConfigMap`.

Variável	Descrição
<code>DB_TYPE</code>	Tipo de banco de dados utilizado (ex: <code>postgres</code>).
<code>DB_NAME</code>	Nome do banco de dados utilizado pelo serviço (ex: <code>aura_blocks</code>).
<code>DB_PORT</code>	Porta utilizada para conexão com o banco de dados (ex: <code>"5432"</code>).
<code>DB_HOST</code>	Hostname do banco de dados. Deve ser substituído por um valor real.
<code>DB_USER</code>	Nome de usuário para autenticação no banco de dados.
<code>HTTP_PORT</code>	Porta HTTP onde o serviço será exposto internamente (ex: <code>"8989"</code>).
<code>HTTP_BASE</code>	Caminho base da API do proactive (ex: <code>"/proactive/"</code>). Deve coincidir com a configuração do Ingress.
<code>HTTP_USER</code>	Usuário utilizado para autenticação da API. Valor padrão: <code>"lowcode"</code> .
<code>GUPSHUP_APIKEY</code>	Chave da API da gupshup. Substituir o valor pela chave real do ambiente
<code>GUPSHUP_APPID</code>	ID do APP da gupshup. Substituir o valor pela chave real do ambiente
<code>GUPSHUP_SOURCE</code>	Telefone vinculado ao workspace na gupshup. Substituir o valor pela chave real do ambiente
<code>GUPSHUP_SOURCE_NAME</code>	Workspace na gupshup. Substituir o valor pela chave real do ambiente
<code>KEYCLOAK_URL</code>	URL para obtenção de informações de usuário via Keycloak. Essa URL inclui o realm e o caminho do endpoint userinfo do OpenID Connect.

proactive/secret.yaml

O arquivo `proactive/secret.yaml` armazena informações sensíveis utilizadas pelo serviço, codificadas em Base64.

Variável	Descrição
<code>DB_PASS</code>	Senha utilizada em conjunto com <code>DB_USER</code> para autenticação no banco de dados.
<code>HTTP_PASSWORD</code>	Senha codificada em Base64, utilizada para autenticação da API.

⚠ **Atenção:** Nunca armazene os valores decodificados em repositórios. Manipule os segredos com segurança e mantenha-os restritos ao ambiente controlado.

6.3.8 pgadmin

pgadmin/configmap.yaml

O arquivo `pgadmin/configmap.yaml` define as variáveis de ambiente utilizadas pelo serviço **pgAdmin**, responsável por oferecer uma interface gráfica para administração do banco de dados PostgreSQL.

Variável	Descrição
<code>TZ</code>	Fuso horário do container. Exemplo: <code>"America/Sao_Paulo"</code> .
<code>SCRIPT_NAME</code>	Caminho base opcional (prefixo) da aplicação no Ingress. Exemplo: <code>"/pgadmin"</code> .
<code>PGADMIN_DEFAULT_EMAIL</code>	E-mail do usuário administrador criado automaticamente no primeiro acesso. Exemplo: <code>"admin@empresa.com"</code> .
<code>MAX_LOGIN_ATTEMPTS</code>	Número máximo de tentativas de login antes do bloqueio do usuário. Padrão: <code>3</code> .
<code>PGADMIN_LISTEN_ADDRESS</code>	Endereço de rede no qual o pgAdmin vai escutar. Normalmente <code>0.0.0.0</code> para aceitar todas as conexões.

pgadmin/secret.yaml

O arquivo `pgadmin/secret.yaml` armazena informações sensíveis utilizadas pelo serviço, codificadas em Base64.

Variável	Descrição
<code>PGADMIN_DEFAULT_PASS</code> <code>WORD</code>	Senha associada ao <code>PGADMIN_DEFAULT_EMAIL</code> para acesso à interface do pgAdmin.

6.4 Configuração do Ingress

O arquivo `ingress.yaml` define as regras de roteamento HTTP externo para os serviços do projeto **aura-blocks**, utilizando um Ingress Controller como Traefik ou NGINX. Ele garante que cada caminho (`/engine` , `/widget` , etc.) seja corretamente encaminhado para o serviço correspondente dentro do cluster.

⚠ **Atenção:** Esse arquivo exige configurações obrigatórias para funcionar corretamente em cada ambiente. A seguir, estão os campos que **devem ser ajustados**:

6.4.1 ingress.yaml

Campo	Descrição
<code>ingressClassName</code>	Define qual controlador de Ingress será responsável por gerenciar as regras. Deve ser substituído por um valor existente no cluster (ex: <code>"traefik"</code> ou <code>"nginx"</code>).
<code>host</code>	Nome do domínio público onde os serviços serão acessados. Substitua <code>domain.com</code> pelo domínio real da aplicação (ex: <code>aura-blocks.centralit.com.br</code>).
<code>paths</code>	Cada caminho HTTP (<code>/engine</code> , <code>/widget</code> , etc.) é roteado para o respectivo serviço Kubernetes com sua porta correspondente. Esses caminhos devem estar alinhados com os valores de <code>HTTP_BASE</code> definidos nos <code>ConfigMaps</code> .
<code>tls.secretName</code>	Nome do <code>Secret</code> que contém o certificado TLS para habilitar HTTPS. O segredo deve estar criado no namespace <code>aura-blocks</code> , com o nome correspondente ao domínio utilizado. Por exemplo: <code>aura-blocks-centralit-tls</code> .
<code>tls.hosts</code>	Lista de domínios que serão protegidos via HTTPS. Deve corresponder ao domínio configurado no campo <code>host</code> .

Exemplo de valores substituídos corretamente:

YAML

```

...
ingressClassName: traefik
rules:
- host: aura-blocks.centralit.com.br
...
tls:
- secretName: aura-blocks-centralit-tls
  hosts:
    - aura-blocks.centralit.com.br

```

❗ **Importante:** Certifique-se de que o `Secret` de TLS já exista no namespace `aura-blocks`, e que tenha sido criado a partir de um certificado válido para o domínio desejado. Isso é essencial para permitir a navegação segura via HTTPS.

Após realizar os ajustes necessários, aplique o Ingress com o comando:

Bash



```
kubectl apply -f ingress.yaml
```

7. Deployment do Ambiente

Com todas as configurações ajustadas, você pode aplicar os manifestos ao cluster. Navegue até o diretório correto e execute o seguinte comando:

Bash



```
kubectl apply -f <CAMINHO_DO_MANIFESTO_MODIFICADO>
```

7.1 Aplicando os Manifests de Kubernetes

Agora, aplique os manifestos ao cluster. Repita os seguintes comandos para cada serviço:

7.1.1 engine

Bash



```
kubectl apply -f engine/configmap.yaml  
kubectl apply -f engine/secret.yaml  
kubectl apply -f engine/deployment.yaml  
kubectl apply -f engine/service.yaml
```

7.1.2 attendant

Bash



```
kubectl apply -f attendant/configmap.yaml
kubectl apply -f attendant/secret.yaml
kubectl apply -f attendant/deployment.yaml
kubectl apply -f attendant/service.yaml
```

7.1.3 lunaris

Bash



```
kubectl apply -f lunaris/deployment.yaml
kubectl apply -f lunaris/service.yaml
```

7.1.4 wanderson

Bash



```
kubectl apply -f wanderson/configmap.yaml
kubectl apply -f wanderson/secret.yaml
kubectl apply -f wanderson/deployment.yaml
kubectl apply -f wanderson/service.yaml
```

7.1.5 teams-server

Bash



```
kubectl apply -f teams-server/configmap.yaml
kubectl apply -f teams-server/deployment.yaml
kubectl apply -f teams-server/service.yaml
```

7.1.6 widget

Bash



```
kubectl apply -f widget/configmap.yaml
kubectl apply -f widget/secret.yaml
kubectl apply -f widget/deployment.yaml
kubectl apply -f widget/service.yaml
```

7.1.7 dash

Bash



```
kubectl apply -f dash/configmap.yaml
kubectl apply -f dash/secret.yaml
kubectl apply -f dash/deployment.yaml
kubectl apply -f dash/service.yaml
```

7.1.8 proactive

Bash



```
kubectl apply -f proactive/configmap.yaml
kubectl apply -f proactive/secret.yaml
kubectl apply -f proactive/deployment.yaml
kubectl apply -f proactive/service.yaml
```

7.1.9 pgadmin

Bash



```
kubectl apply -f pgadmin/configmap.yaml
kubectl apply -f pgadmin/secret.yaml
kubectl apply -f pgadmin/deployment.yaml
kubectl apply -f pgadmin/service.yaml
```

8. Verificação do Deployment

Após aplicar todos os manifests, é essencial validar se os recursos foram criados corretamente e estão funcionando como esperado.

8.1 Verificar Recursos Criados no Namespace

Bash



```
kubectl get all -n aura-blocks
```

- Lista **pods, services, deployments, replicaset**s, etc.
- Confirme se **todos os pods estão com STATUS** `Running` ou `Completed` e **READY** como `1/1` ou `x/x`.

8.2 Verificar Eventos e Condições de Falha

Bash



```
kubectl get events -n aura-blocks --sort-by=.metadata.creationTimestamp
```

- Mostra **eventos recentes**, como falhas de agendamento, erro de imagem, problemas com PVC, etc.
- Use este comando para **debug de erros de criação**.

8.3 Diagnóstico de um Pod Específico

Bash



```
kubectl describe pod <nome-do-pod> -n aura-blocks
```

Para listar os nomes dos pods:

Bash



```
kubectl get pods -n aura-blocks
```

8.4 Verificar o Ingress

Bash



```
kubectl describe ingress -n aura-blocks
```

- Valida se o Ingress está roteando corretamente os caminhos (`/engine` , `/widget` , etc).
- Verifique:
 - `Address` : IP externo ou LoadBalancer configurado
 - `Rules` : correspondem aos paths e hosts definidos
 - `TLS` : status do certificado

8.5 Validar a URL Pública dos Serviços

Bash



```
curl -I https://aura-blocks.centralit.com.br/engine/manuals/
```

- Verifica se o serviço `engine` está **acessível via HTTPS**.

8.6 Verificar Logs dos Pods

Engine:

Bash



```
kubectl logs -l app=engine -n aura-blocks --tail=100
```

Lunaris:

Bash



```
kubectl logs -l app=lunaris -n aura-blocks --tail=100
```

8.7 Verificar Volume Persistente

Bash



```
kubectl get pvc -n aura-blocks  
kubectl describe pvc <nome-do-pvc> -n aura-blocks
```

8.8 Consultar Variáveis de Ambiente do Pod

Bash



```
kubectl exec -it <nome-do-pod> -n aura-blocks -- printenv
```

8.9 Verificar Readiness de Todos os Pods

Bash



```
kubectl get pods -n aura-blocks -o custom-columns="POD:metadata.name,STATUS:s
```

2.2. Manual de Atualização de um Ambiente Existente

1. Pré-requisitos

Antes de realizar a atualização de um ambiente, certifique-se de que os seguintes pré-requisitos estão atendidos:

- **Acesso VPN:** Necessário para conectar ao cluster do Aura-Blocks.
- **Cluster Kubernetes:** O cluster Kubernetes existente deve estar funcionando corretamente.
- **kubectI:** Ferramenta de linha de comando para interagir com o cluster.
- **Imagens Atualizadas:** Verifique as novas versões das imagens Docker que deseja aplicar na atualização.

2. Listando e Trocando o Namespace

Antes de realizar qualquer operação, é importante verificar os namespaces disponíveis no cluster e garantir que você está utilizando o namespace correto para o projeto.

2.1 Listar os Namespaces no Cluster

Utilize o comando abaixo para listar todos os namespaces no cluster Kubernetes:

Bash



```
kubectl get namespaces
```

Este comando exibirá todos os namespaces disponíveis. Certifique-se de identificar o namespace correto do projeto (ex: `aura-blocks`).

2.2 Definindo o Namespace para o Projeto

Para garantir que todos os comandos subsequentes serão executados no namespace correto, você pode configurar o namespace padrão a ser utilizado:

Bash



```
kubectl config set-context --current --namespace=aura-blocks
```

Este comando irá definir o namespace `aura-blocks` como o namespace padrão para todos os comandos kubectl executados na sessão atual.

3. Procedimento de Atualização

A atualização de um ambiente existente é feita principalmente pela troca das **tags das imagens Docker** nos manifests de Kubernetes. Abaixo estão os passos simples para realizar essa tarefa.

3.1 Atualizando as Imagens nos Manifests

3.1.1 Identificando os Manifests para Atualizar

Os arquivos de `Deployment` são os principais pontos onde as tags das imagens precisam ser atualizadas. Certifique-se de que os seguintes arquivos estão sendo modificados:

- `template/engine/deployment.yaml`
- `template/lowproxy/deployment.yaml`
- `template/lowtalk/deployment.yaml`
- `template/lunaris/deployment.yaml`
- `template/wanderson/deployment.yaml`

3.1.2 Atualizando a Tag da Imagem

Abra o arquivo do `Deployment` correspondente e modifique a seção `image`, substituindo a tag antiga pela nova tag da imagem.

Exemplo de atualização no arquivo `deployment.yaml` do serviço `engine`:

YAML



```
containers:
  - name: engine
    image: aura-blocks/engine:<nova_tag>
    ports:
      - containerPort: 6060
```

Repita esse processo para os outros serviços (lowproxy, lowtalk, lunaris, wanderson), atualizando suas tags conforme necessário.

3.1.3 Aplicando as Mudanças

Depois de modificar as tags, aplique os manifests atualizados no cluster com o seguinte comando para cada serviço:

Bash



```
kubectl apply -f template/<serviço>/deployment.yaml
```

Exemplo:

Bash



```
kubectl apply -f template/engine/deployment.yaml
```

3.2 Reiniciando os Serviços

Após aplicar as mudanças, reinicie os serviços para garantir que os novos pods sejam iniciados com as imagens atualizadas. Use o comando `kubectl rollout restart` para cada serviço:

Bash



```
kubectl rollout restart deploy <deploy>
```

Exemplo:

Bash



```
kubectl rollout restart deploy env-engine  
kubectl rollout restart deploy env-lowproxy  
kubectl rollout restart deploy env-lowtalk  
kubectl rollout restart deploy env-lunaris  
kubectl rollout restart deploy env-wanderson
```

3.3 Verificando o Status

Após reiniciar os deployments, verifique o status dos recursos para garantir que os pods foram reiniciados corretamente com as novas versões das imagens:

Bash



```
kubectl get pods -n aura-blocks
```

Certifique-se de que os novos pods estão no estado `Running`.

3.4 Rollback (Se Necessário)

Caso algo não funcione como esperado após a atualização, você pode realizar um rollback para a versão anterior dos deployments. O rollback retorna o deployment à versão anteriormente aplicada.

Para realizar o rollback, utilize o comando:

Bash



```
kubectl rollout undo deploy <deploy>
```

Exemplo de rollback para o serviço `engine`:

Bash



```
kubectl rollout undo deploy env-engine
```

Repita o comando para os outros serviços conforme necessário, retornando as versões anteriores das imagens.

4. Conclusão

Com esses passos, o ambiente será atualizado com as novas versões das imagens. Acompanhe o comportamento dos serviços e, caso necessário, utilize o rollback para reverter mudanças problemáticas.

2.3. Manual de Criação e Adição de Menus

Este documento tem como finalidade servir como um guia prático e objetivo para a criação, configuração e adição de menus

Esta funcionalidade está disponível apenas para o papel de **Desenvolvedor**.

Geralmente, a **URL do Citsmart** instalada no OCI está localizada no repositório GitLab da equipe. Entretanto, em algumas situações, pode não estar disponível. Por isso, antes de cadastrar os menus, certifique-se de ter as seguintes informações:

- 1 - URL do Citsmart
- 2 - Perfil de administrador
- 3 - Valor da senha da engine

Caso não possua algum desses **itens obrigatórios**, solicite à equipe de infraestrutura.

Este manual está dividido em seções para facilitar a localização das informações. As seções são:

Cadastrar e Configurar uma Aplicação □

Antes de criar menus, você precisa cadastrar e configurar uma aplicação. Se ainda não tem uma, este é o seu primeiro passo.

Criação dos Menus □

Se você já configurou sua aplicação, aqui é ensinado como criar os menus.

Exportar e Importar Pacotes □

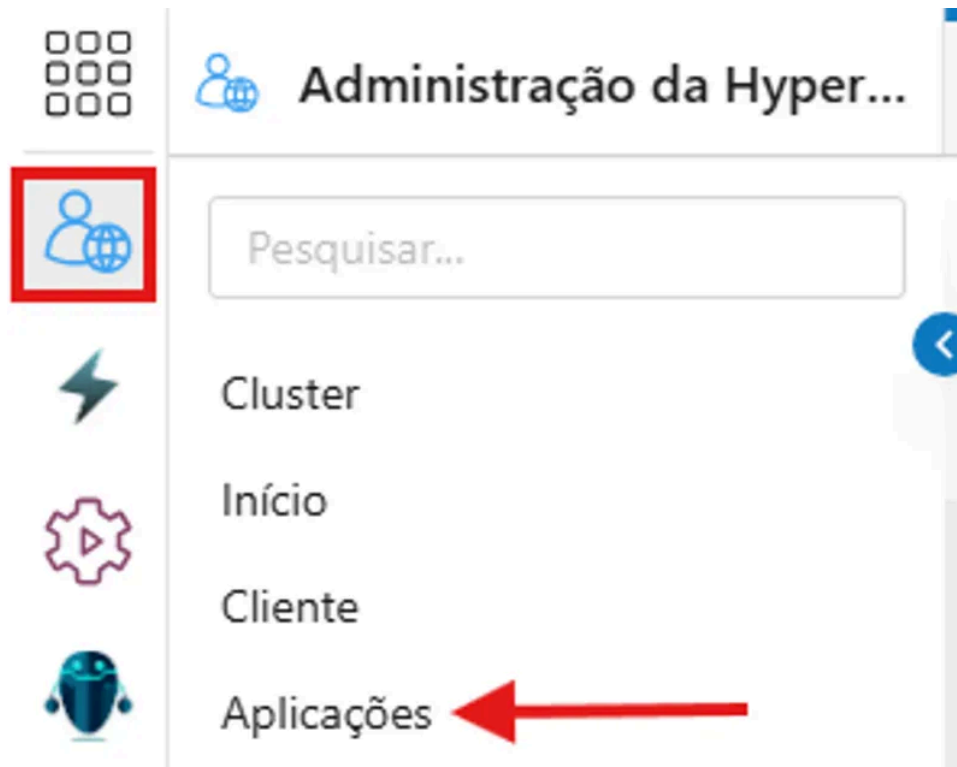
Aprenda como exportar e importar pacotes para otimizar a configuração da sua aplicação e agilizar o processo de replicação de menus e dados entre ambientes.

Menu Antigo/FrontManager Antigo □

Em alguns ambientes, o Front Manager pode estar com versões antigas. Essa seção mostra como proceder nesses casos específicos.

Criar uma aplicação





















1. No menu lateral, acesse **Administração da Hyper Platform > Aplicações**.



1. Na aba, selecione a opção **Criar**.

Administração da Hyper... Aplicações X

Administrar Aplicação + Criar Exportar CSV Buscar Platform Application

ID da Aplicação	Nome	Descrição	Situação	Tipo	Cliente	Ações
1	@hyper/todo-mvc-app-react	Itens a fazer APP (Exemplo React)	I	P		 
2	@hyper/todo-mvc-app-angular	Todo MVC Angular	I	P		 
4	@centralit/citsmart	CitSmart	A	C	hml-tjto	 
5	@varti/reserva-imobiliaria	Reserva Imobiliária	A	C	hml-valstec	 
6	@hyper/todo-mvc-app-angularjs	Todo MVC AngularJS	I	P		 
7	@citsmart/itsm	ITSM	A	P	hml-tjto	 
8	@hyper/lowcode	Low Code	A	P		 
9	@hyper/selfservice	Self-Service & Espaços de Trabalho	A	P		 
10	@hyper/sistemaportico	Pórtico Digital	A	P	hml-dp	 
11	@hyper/servicedesk	Service Desk	A	P		 

Mostrando 1 até 10 de (51)

Primeiro Anterior 1 2 3 4 5 Próximo Último

2. Preencha o formulário

- **Nome:** @hyper/<Nome da sua aplicação>

- **Descrição:** <Descrição da sua aplicação>
- **Situação:** Ativa
- **Renderizar:** Legado - Iframe
- **Tipo:** Gerenciada pela plataforma
- **Ícone:** Ícone no formato PNG ou SVG

Administrar Aplicação

> Nome *

@hyper/nome-exemplo

Descrição *

Essa é a descrição exemplo de uma aplicação.

Cliente

Informe Cliente

Situação * Renderizar * Tipo *

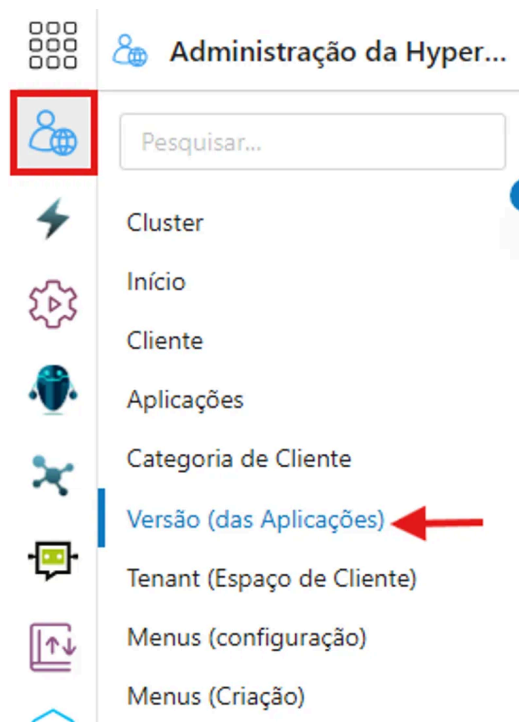
Ativa Legado - Iframe Gerenciada pela plataforma

Observação

Ícone Imagem

Gravar Limpar Cancelar





1. Novamente no menu lateral, acesse **Administração da Hyper Platform > Versão (das Aplicações)**



2. Na página, localize sua aplicação pelo **Nome** e, em seguida, na coluna **Ações**, clique no ícone de edição.

Administração da Hyper... Versão (das Aplicações) X

Selecionar uma Aplicação + Criar Exportar CSV exemplo

ID da Aplicação	Nome	Descrição	Situação	Tipo	Cliente	Ações
1	@hyper/todo-mvc-app-react	Itens a fazer APP (Exemplo React)	I	P		 
67	<u>@hyper/nome-exemplo</u>	Essa é a descrição exemplo de uma aplicação.	A	P		 

Mostrando 1 até 2 de (2)

Primeiro Anterior 1 Próximo Último

3. Clique em **Adicionar**.

Administração da Hyper... Versão (das Aplicações) X

Selecione uma Aplicação

Versões

Versao	URL	Nome Base	Ações
--------	-----	-----------	-------

+ Adicionar

Gravar Cancelar

4. Preencha o formulário.

O modelo a seguir apresenta um exemplo de aplicação criada com base na estrutura padrão utilizada pela equipe **Citsmart Aura**. Para personalizações adicionais, entre em contato com um supervisor ou consulte a documentação oficial da plataforma.

Modelo base:

Os campos **Maior**, **Menor**, **Correção** e **Revisão** compõem, em conjunto, a versão.

Por exemplo: ao inserir os valores 1, 2, 4 e 6, a versão resultante será **1.2.4.6**.

O campo Nome da Base corresponde ao segmento que aparece após a URL padrão do Citsmart. Assim, em uma URL fictícia, seria representado como: `http://url-exemplo.citsmart/<nome-da-base>`

Após concluir o preenchimento clique em **"Gravar"**, tanto no formulário quanto na tela de Versão (das Aplicações).

Versões

ID da Aplicação

67

Maior * Menor * Correção Revisão Situação *

1 0 0 0 Ativo ▼

URL * Nome da Base * URL de Validação da Licença

%MAIN_URL_DNS%/ /exemplo

Observação

Opções do Aplicativo

Gravar Limpar Fechar

Administração da Hyper... Versão (das Aplicações) X

Selecionar uma Aplicação

Versões

+ Adicionar

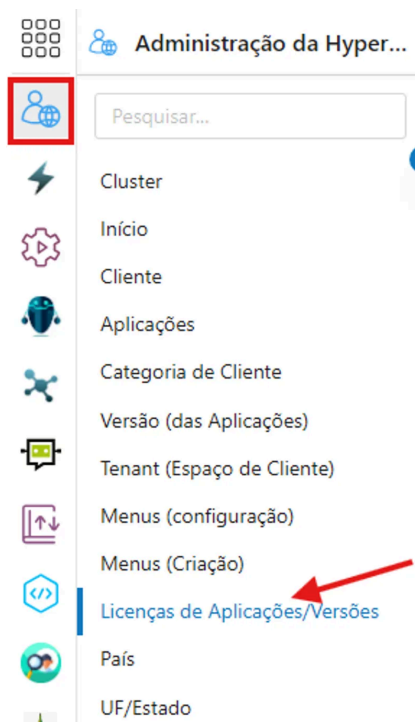
Versao	URL	Nome Base	Ações
	%MAIN_URL_DNS%/	/exemplo	

Mostrando 1 até 1 de (1)

Primeiro Anterior 1 Próximo Último

Gravar Cancelar

5. Mais uma vez no menu lateral, acesse **Administração da Hyper Platform > Licenças de Aplicações/Versões**.



6. Clique em **Criar**.

Administração da Hyper... Licenças de Aplicações/Versões X

Licença + Criar Exportar CSV Buscar License

UUID da Licença	Tenant	Versão do Aplicativo da Plataforma	Licenciado para	Data de registro	Número de usuários	Status	Data de Início	Data de expiração	Ações
N9TT-9G0A-B7FQ-RANC	democentralit.cithyper.click	100138	demo	28/04/2021 17:25:13	100	A	28/04/2021 17:25:24		
QK6A-JI6S-7ETR-0A6C	democentralit.cithyper.click	2	run2biz	28/04/2021 20:31:52	100	1	28/04/2021 20:31:52		
SXFP-CHYK-ONI6-S89U	centraldeservicoshomolog.tjto.jus.br	1	tenant2	29/04/2021 09:35:11	100	I	29/04/2021 09:35:33		
manutenção predial	democentralit.cithyper.click	100035	democentralit	17/03/2023 00:00:00	1000	A	17/03/2023 10:00:00		
7ETI-UIL2-9WAX-XHYO	democentralit.cithyper.click	6	run2biz	03/05/2021 14:46:43	500	I	03/05/2021 14:46:50		
8ETI-UIL2-9WAX-XHYO	democentralit.cithyper.click	7	run2biz	06/05/2021 19:46:11	500	I	06/05/2021 19:46:32		
9ETI-UIL2-9WAX-XHYO	democentralit.cithyper.click	8	run2biz	07/05/2021 12:53:36	500	A	07/05/2021 12:53:50		
10TI-UIL2-9WAX-XHYO	democentralit.cithyper.click	9	run2biz	11/05/2021 00:00:00	100	A	11/05/2021 00:00:00		
12TI-UIL2-9WAX-XHYO	democentralit.cithyper.click	11	run2biz	11/05/2021 00:00:00	100	A	11/05/2021 00:00:00		
13TI-UIL2-9WAX-XHYO	democentralit.cithyper.click	12	run2biz	11/05/2021 00:00:00	100	A	11/05/2021 00:00:00		

Mostrando 1 até 10 de (210)

Primeiro Anterior 1 2 3 4 5 Próximo Último

7. Preencha o formulário de licença seguindo o modelo abaixo:

O modelo a seguir apresenta um exemplo de aplicação criada com base na estrutura padrão utilizada pela equipe **Citsmart Aura**. Para personalizações adicionais, entre em contato com um supervisor ou consulte a documentação oficial da plataforma.

Após concluir o preenchimento clique em **"Gravar"**.

Administração da Hyper... Licenças de Aplicações/Versões X

Licença

Licenciado para *

EXEMPLO-LICENÇA

Versão do Aplicativo da Plataforma

Aura Blocks._Version.1.0.0.0

Tenant

hml-cocriar.cithyper.click

Data Inicial *

15/08/2024

Data de expiração

Data de registro *

15/08/2024

Numero de usuarios

1.000

Situação *

Ativo

Código *

EXEMPLO-LICENÇA

Observação

Gravar

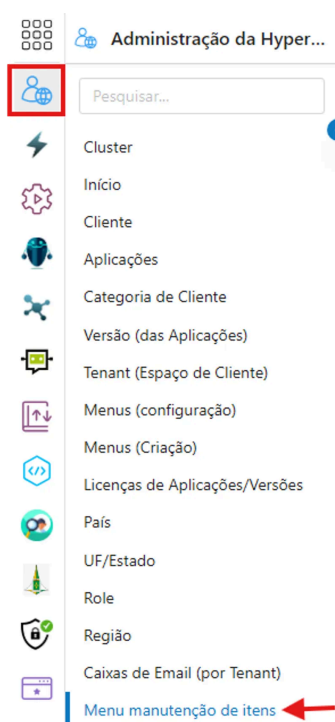
Limpar

Cancelar

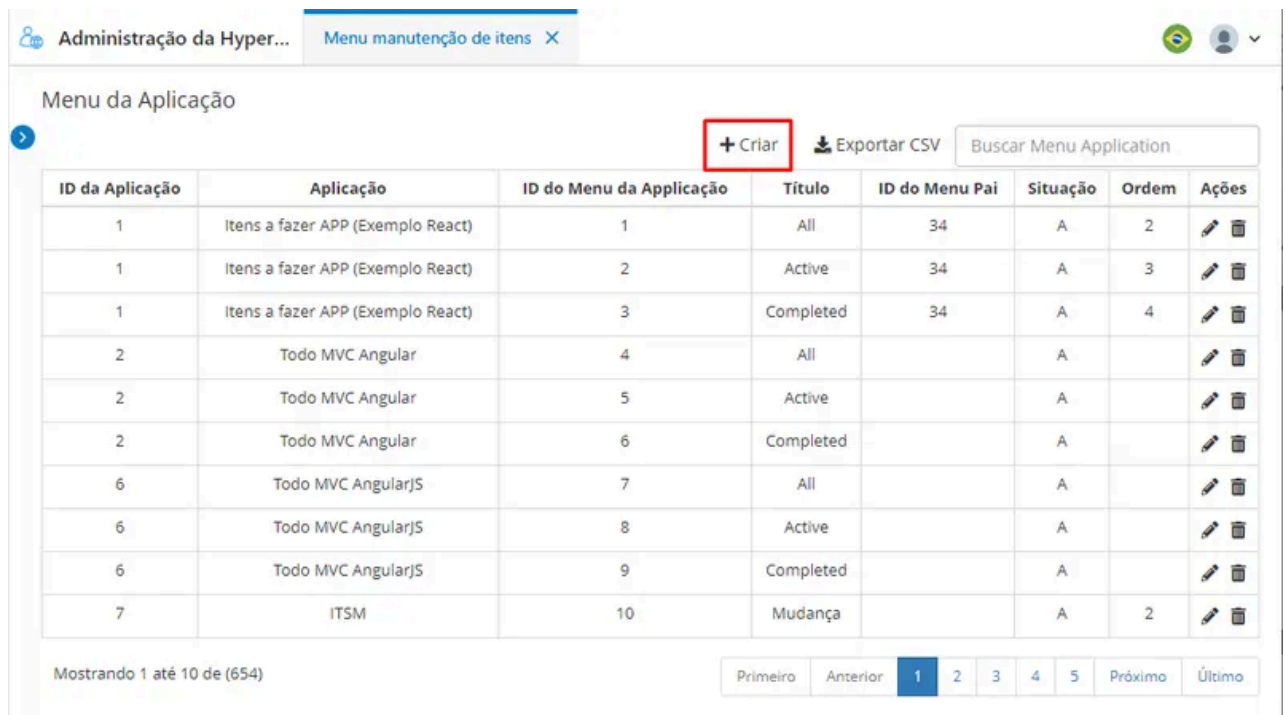
Criar Menus

- No menu lateral, acesse **Administração da Hyper Platform > Menu Manutenção de Itens**

Caso esteja utilizando uma versão antiga do Front Manager, siga esta seção.























2. Clique em **Criar**.



Administração da Hyper... Menu manutenção de itens X

Menu da Aplicação

+ Criar Exportar CSV Buscar Menu Application

ID da Aplicação	Aplicação	ID do Menu da Aplicação	Título	ID do Menu Pai	Situação	Ordem	Ações
1	Itens a fazer APP (Exemplo React)	1	All	34	A	2	 
1	Itens a fazer APP (Exemplo React)	2	Active	34	A	3	 
1	Itens a fazer APP (Exemplo React)	3	Completed	34	A	4	 
2	Todo MVC Angular	4	All		A		 
2	Todo MVC Angular	5	Active		A		 
2	Todo MVC Angular	6	Completed		A		 
6	Todo MVC AngularJS	7	All		A		 
6	Todo MVC AngularJS	8	Active		A		 
6	Todo MVC AngularJS	9	Completed		A		 
7	ITSM	10	Mudança		A	2	 

Mostrando 1 até 10 de (654)

Primeiro Anterior 1 2 3 4 5 Próximo Último

3. Preencha o formulário.

Os exemplos a seguir abrangem a criação de todos os menus utilizados pela equipe Citsmart Aura. Caso sejam necessárias alterações, utilize o modelo como referência, ajustando apenas os campos que deseja personalizar.

A imagem a seguir apresenta o formulário que deve ser preenchido para adicionar o menu, utilizando parâmetros de exemplo. Substitua esses parâmetros pelos exemplos indicados logo após a imagem. Ao final de cada modelo, clique no botão **Gravar** e, em seguida, em **Criar** para adicionar o próximo item.

Administração da Hyper... Menu manutenção de itens X

Menu da Aplicação

ID da Aplicação Aplicação

Menu Pai

Título *

Descrição

Caminho

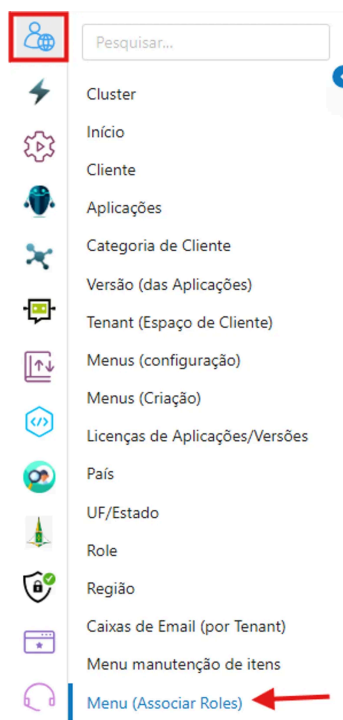
Observação

Renderizar * Situação * Ordem *

Ícone

[Base](#) | [Comportamentos](#) | [Atendimento Humano](#) | [Máquina de Estados](#) | [Widgets](#) | [Atendimentos](#) | [Filas](#) | [Atalhos de Texto](#) | [Atalhos Globais](#) | [Atalhos Privados](#) | [Questionários e Ações](#) | [Ajuda](#) | [Sobre](#)

4. No menu lateral, acesse **Administração da Hyper Platform > Menu (Associar Roles)**.



5. Na coluna **Ações**, clique no ícone para editar no cliente desejado.

Itens de Menu do Role (aurablocks)

	Título	Status
<input checked="" type="radio"/>	Comportamentos	Associado
<input type="radio"/>	Atendimento Humano	Associado
<input type="radio"/>	Wanderson	Associado
<input type="radio"/>	Widgets	Associado

Subitens do Menu

Associar

Remover

Itens de Menu do Role (aurablocks)

	Título	Status
<input type="radio"/>	Comportamentos	Associado
<input checked="" type="radio"/>	Atendimento Humano	Associado
<input type="radio"/>	Wanderson	Associado
<input type="radio"/>	Widgets	Associado

Subitens do Menu

☒Atendimentos (Associado)

☒Filas (Associado)

☒Atalhos de Texto (Associado)

☒Atalhos Globais (Associado)

☒Atalhos Privados (Associado)

Associar

Remover

Itens de Menu do Role (aurablocks)**Subitens do Menu**

	Título	Status
<input type="radio"/>	Comportamentos	Associado
<input type="radio"/>	Atendimento Humano	Associado
<input checked="" type="radio"/>	Wanderson	Associado
<input type="radio"/>	Widgets	Associado

Associar

Remover

Itens de Menu do Role (aurablocks)**Subitens do Menu**

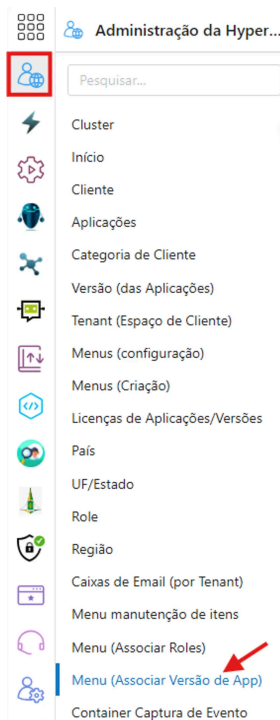
	Título	Status
<input type="radio"/>	Comportamentos	Associado
<input type="radio"/>	Atendimento Humano	Associado
<input type="radio"/>	Wanderson	Associado
<input checked="" type="radio"/>	Widgets	Associado

Associar

Remover

Após associar todos os itens, clique em **Gravar**.

7. Por fim, no menu lateral, acesse **Administração da Hyper Platform > Menu (Associar Versão de App)**.



8. Dentro da página, pesquise por **aura** e na coluna **Ações**, clique no ícone para editar.

Menu (Associar Versão de App) X

Selezione uma Aplicação

+ Criar Exportar CSV aura

ID da Aplicação	Nome	Descrição	Situação	Tipo	Cliente	Ações
59	@hyper/aura-blocks	Aura Blocks	A	P		 
65	<u>@hyper/aurablocks</u>	Citsmart Aura	A	P		 

Mostrando 1 até 2 de (2)

Primeiro Anterior 1 Próximo Último

9. Marque a versão e o item de menu desejado e clique em **"Associar"**. Caso seja necessário associar outros itens, repita o procedimento.

Menu (Associar Versão de App) X

Selecione uma Aplicação

Descrição *

Citsmart Aura

Versão

1.0.0.0

Itens de Menu

	Título	Full Version
<input type="radio"/>	Chatbots	1.0.0.0
<input type="radio"/>	Máquina de estados	1.0.0.0
<input checked="" type="radio"/>	Atendimento Humano	1.0.0.0
<input type="radio"/>	Widgets	1.0.0.0
<input type="radio"/>	Questionários e Ações	1.0.0.0
<input type="radio"/>	Sobre	1.0.0.0
<input type="radio"/>	Ajuda	Nao atribuido

Subitens do Menu (Atendimento Humano)

- ☒ Filas (1.0.0.0)
- ☒ Atendimentos (1.0.0.0)
- ☒ Atalhos de Texto (1.0.0.0)
 - ☒ Atalhos Privados (1.0.0.0)
 - ☒ Atalhos Globais (1.0.0.0)
- ☒ Atendimento (1.0.0.0)

Associar Remover

Gravar Cancelar

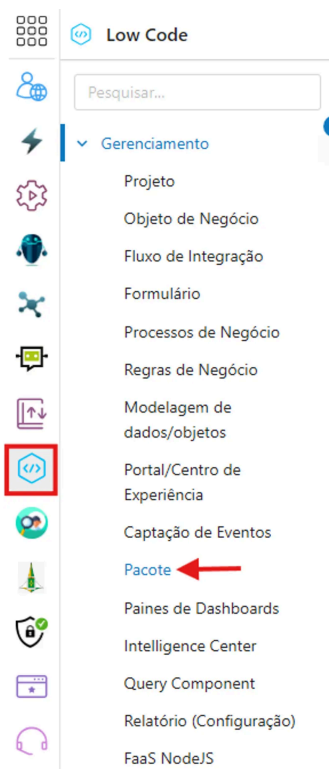
Após concluir, clique no botão **"Gravar"** para salvar as alterações.

Importar e Exportar pacotes

Exportar | Importar

Exportar



















































1. Em um ambiente **Citsmart** já configurado, acesse o menu lateral: **Lowcode > Pacote**.



1. Localize o pacote que deseja exportar (geralmente o mais recente ou uma versão estável) e clique no botão **"Exportar"**.

Low Code Pacote X Sobre X

Importar + Criar Buscar Pacote

Nome ↑	Descrição	Versão	Ações
aura_07_05_2025_0_1_6_rc22	aura_07_05_2025_0_1_6_rc22	1.0	    
aura_09_04_2025_0_1_5	aura_09_04_2025_0_1_5	1.0	    
aura_10_04_2025_0_1_5_p1	aura_10_04_2025_0_1_5_p1	1.0	    
aura_10_04_2025_0_1_5_p2	aura_10_04_2025_0_1_5_p2	1.0	    
aura_156	Pacote para AURA 156 em 02-05-2025	1.0	    
<u>aura_27_05_2025_0_1_6_rc25</u>	aura_27_05_2025_0_1_6_rc25	1.0	    
AuraBlocks	AuraBlocks	1.0	    
novo_hyper	novo_hyper	4.0	    
widget_07_05_2025_0_1_6_rc22	widget_07_05_2025_0_1_6_rc22	1.0	    
widget_27_05_2025_0_1_6_rc25	widget_27_05_2025_0_1_6_rc25	1.0	    

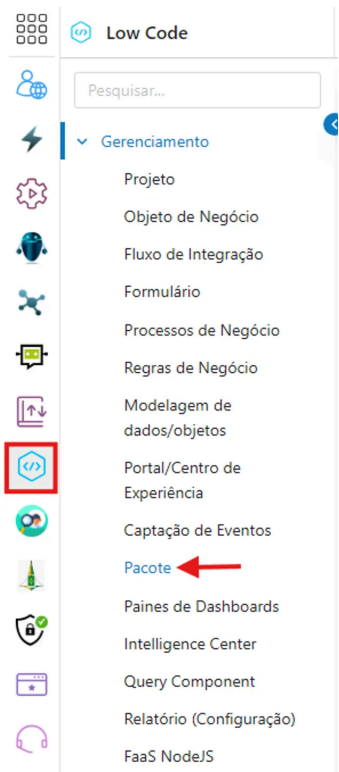
Mostrando 1 até 10 de (11) itens Por página: 10 50

« < 1 2 > »

Um pacote compactado (`.zip`) será baixado e poderá ser utilizado em importações futuras.

Importar

1. No ambiente **Citsmart** em que deseja realizar a importação, acesse o menu lateral: **Lowcode > Pacote**.



1. Clique no botão **“Importar”**.

Low Code

Pacote X Sobre X

⌂

🔄

👤

▼

Pacote

📁 Importar

+ Criar

Buscar Pacote

Nome ↑	Descrição	Versão	Ações
aura_07_05_2025_0_1_6_rc22	aura_07_05_2025_0_1_6_rc22	1.0	✎ 📄 📁 🗑
aura_09_04_2025_0_1_5	aura_09_04_2025_0_1_5	1.0	✎ 📄 📁 🗑
aura_10_04_2025_0_1_5_p1	aura_10_04_2025_0_1_5_p1	1.0	✎ 📄 📁 🗑
aura_10_04_2025_0_1_5_p2	aura_10_04_2025_0_1_5_p2	1.0	✎ 📄 📁 🗑
aura_156	Pacote para AURA 156 em 02-05-2025	1.0	✎ 📄 📁 🗑
aura_27_05_2025_0_1_6_rc25	aura_27_05_2025_0_1_6_rc25	1.0	✎ 📄 📁 🗑
AuraBlocks	AuraBlocks	1.0	✎ 📄 📁 🗑
novo_hyper	novo_hyper	4.0	✎ 📄 📁 🗑
widget_07_05_2025_0_1_6_rc22	widget_07_05_2025_0_1_6_rc22	1.0	✎ 📄 📁 🗑
widget_27_05_2025_0_1_6_rc25	widget_27_05_2025_0_1_6_rc25	1.0	✎ 📄 📁 🗑

Mostrando 1 até 10 de (11) itens

Por página: 10 50

« < 1 2 > »

1. Selecione o arquivo `.json` e clique em **Importar**.

Para obter o arquivo `.json` de uma exportação, é necessário extrair o conteúdo do arquivo `.zip`.

Em sistemas Windows, clique com o botão direito sobre o arquivo compactado e selecione a opção **“Extrair tudo...”**.

Após a extração, localize o arquivo `.json` dentro da pasta descompactada.

Este arquivo é o que deve ser utilizado no processo de importação.


☒ **Substituir destino (se houver)**

☐ **Substituir destino mesmo com versão mais recente**

☐ **Executar DDL**

Arquivo

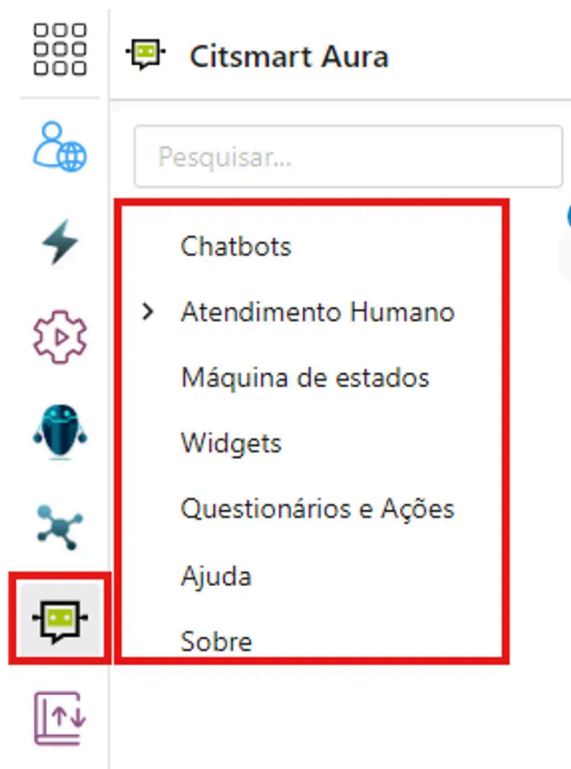
Nome	Tamanho
package_aura_27_05_2025_0_1_6_rc25_v1.0.json	1,61 MB

 **Importar**

Fechar

1. Após selecionar o arquivo, confirme a importação acessando: **menu lateral > menu da sua aplicação**.

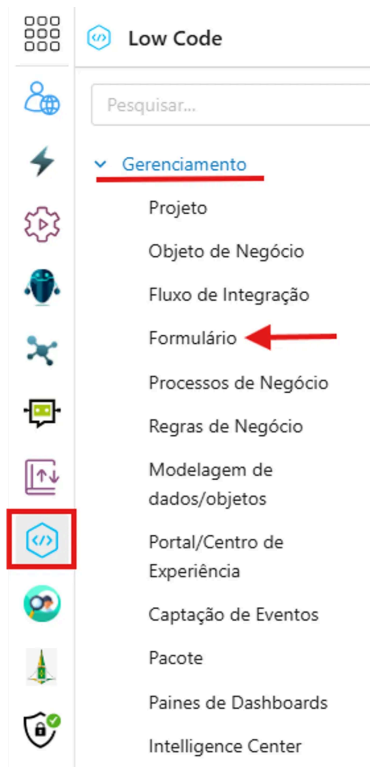
Atenção: verifique se todos os menus estão funcionando corretamente e se apontam para os locais desejados.



Menu Antigo

Em alguns clientes, o Front Manager pode estar com versões desatualizadas. Esta seção abrange esses casos.

1. No menu lateral, acesse **Low Code>Gerenciamento>Formulário**

































1. Na barra de pesquisa, procure pelo item desejado e clique no ícone de editar.

Normalmente, os itens de versões antigas começam com o prefixo "Hyper".

Low Code Chatbots X Formulário X

Formulário Copiar + Criar hyper

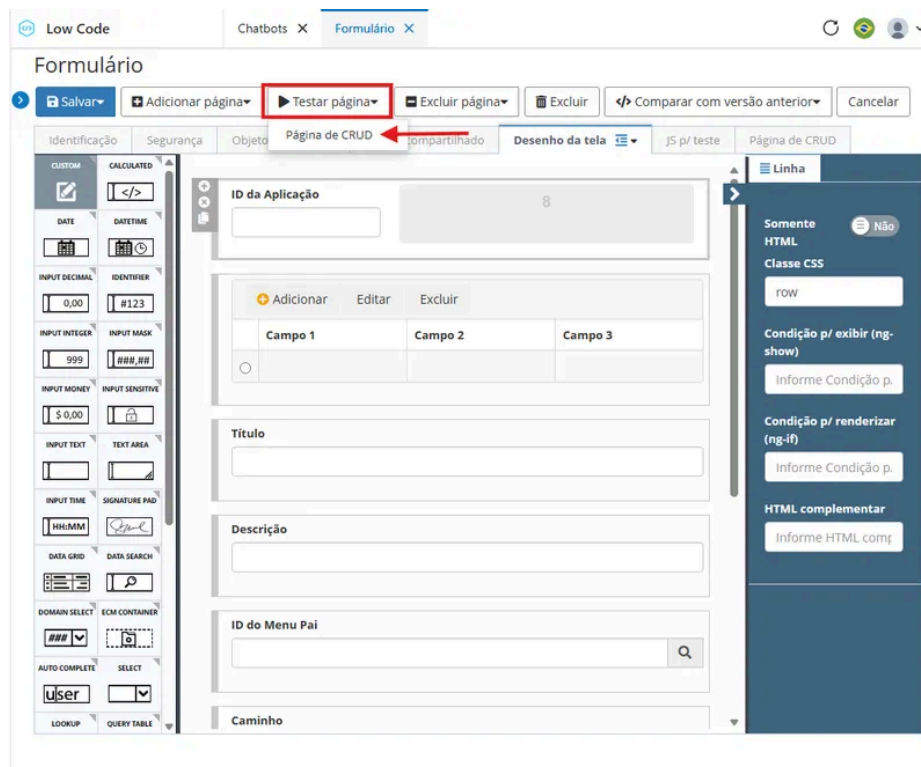
Projeto	Nome	Descrição	Versão	Ações
Hyper Platform Application Registration	adminAplicacao	LABEL.HYPER.MANAGEAPPLICATION	13.0.5	  
Hyper Platform Application Registration	adminMenu	LABEL.HYPER.MANAGEMENUS	14.0.0	  
Hyper Platform Application Registration	adminVersaoAplicacao	LABEL.HYPER.SELECTAPPLICATION	12.0.0	  
Hyper Platform Application Registration	cluster	LABEL.HYPER.CLUSTERREGISTRATION	18.0.0	  
Hyper Platform Application Registration	cluster_space_client	LABEL.HYPER.TENANT	25.0.0	  
Hyper Platform Application Registration	config_menu	LABEL.HYPER.MENUCONFIGURATION	14.0.0	  
Hyper Platform Application Registration	hyperClient	LABEL.HYPER.CLIENT	14.0.0	  
Hyper Platform Application Registration	hyperClientCategory	LABEL.HYPER.CLIENTCATEGORY	15.0.0	  
Hyper Platform Application Registration	hyperCountry	LABEL.HYPER.COUNTRY	13.0.0	  
Hyper Platform Application Registration	hyperMenuApplication	LABEL.HYPER.MENUAPPLICATION	28.0.0	  

Mostrando 1 até 10 de (26) itens Por página: 10 50

< 1 2 3 >

1. Dentro da página, selecione Testar Página e escolha a versão que deseja testar.

Sempre que possível, utilize a opção de formulário padrão. Caso ela não esteja disponível, teste com a página de CRUD.



Link Rápido

Criação dos Menus

2.4. Manual do Administrador

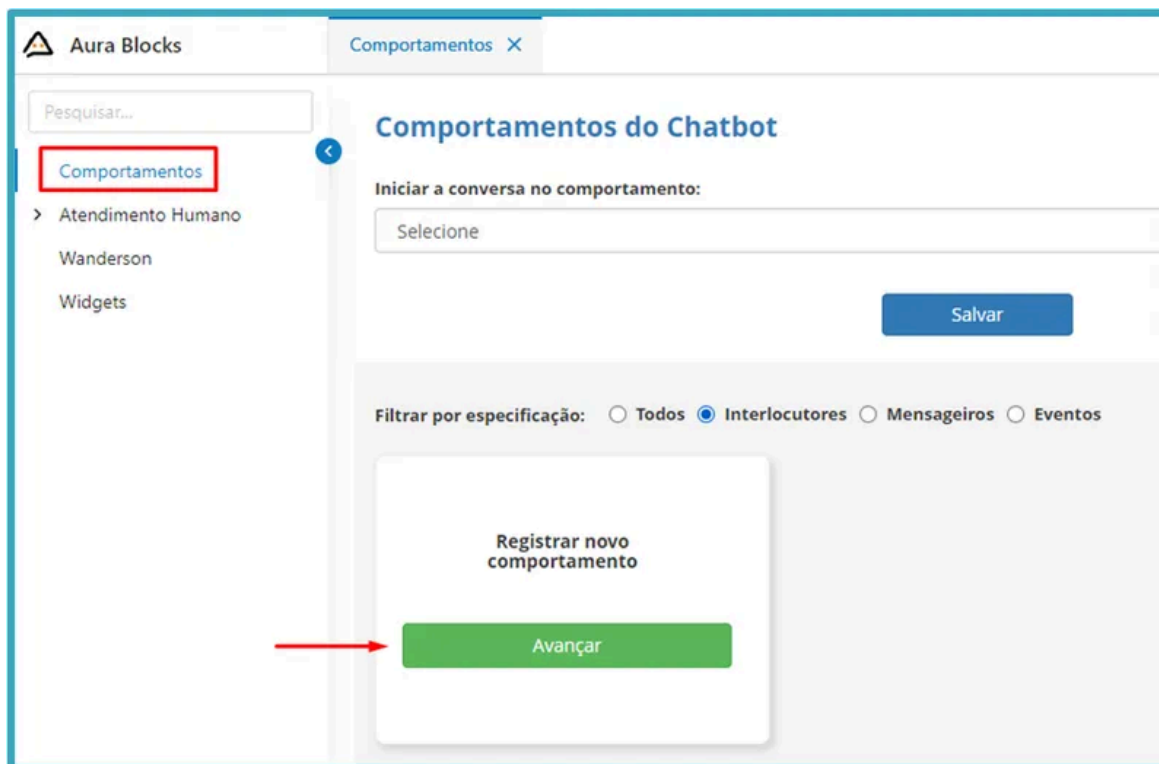
Este manual oferece orientações para que os administradores possam configurar e registrar plugins, contribuindo para a melhoria da experiência do usuário e a otimização do desempenho da plataforma.

Comportamentos

A seção **Comportamentos** é onde você pode registrar e configurar os plugins da plataforma Aura Blocks. Nessa área, você consegue criar novos comportamentos, configurar **Interlocutores**, **Mensageiros** e definir **Eventos**. Todos os plugins cadastrados ficam centralizados aqui para fácil acesso e gerenciamento.

Registrar Comportamento

Para registrar um novo comportamento, acesse o menu **Comportamentos**. Na tela, localize a seção **Registrar novo comportamento** e clique no botão **Avançar** para prosseguir. Veja a imagem abaixo como referência:



Em seguida, preencha o campo **Nome do Comportamento** com o nome desejado e selecione o **Manifesto** que ira utilizar. Cada manifesto oferece parâmetros diferentes. Quando você escolhe um manifesto, a URL é preenchida automaticamente, mas você pode personalizá-la se optar pela opção Personalizado.

A imagem mostra a interface de 'Registrar novo comportamento'. No topo, há um botão de voltar (seta para esquerda) e o título 'Registrar novo comportamento'. Abaixo, há um campo de texto rotulado 'Nome do Comportamento:' com o valor 'Novo Comportamento' e uma seta vermelha apontando para ele. Segue-se a seção 'Selecione o manifesto: *' com um menu suspenso contendo 'Botmaker' e uma seta vermelha apontando para o menu. À direita, há o campo 'Ou digite a URI do manifesto: *' com o valor 'https://aura-blocks.centralit.com.br/lowpro'. Na base, há dois botões: 'Avançar' (azul) e 'Cancelar' (vermelho), com uma seta vermelha apontando para o botão 'Avançar'.

Parâmetros

Cada manifesto requer o preenchimento de parâmetros específicos. Abaixo, você encontrará um exemplo dos parâmetros necessários para cadastrar a **Botmaker**:

- **Texto para Botões:** Mensagem exibida antes das opções no WhatsApp.
- **ID do Canal:** Identificador único atribuído a um canal específico.
- **Token de Acesso:** Código exclusivo gerado pela plataforma **Botmaker** que autentica e autoriza a comunicação entre aplicativos.

Confira a imagem abaixo para referência:

Registrar novo comportamento

Nome do Comportamento:

Novo comportamento

Selecione o manifesto: *

Botmaker

Cancelar

Ou digite a URI do manifesto: *

https://aura-blocks.centralit.com.br/lowpro:

Parâmetros ?

Texto para Botões: *

Selecione

ID do canal *

centralit-whatsapp-556132473172

Token de acesso *

.....

Avançar

Os manifestos disponíveis possuem os seguintes parâmetros para preenchimento:

Personalizado - Permite configurar a URL manualmente.

- **Parâmetros:** Depende da URL fornecida.

Wanderson

- **Selecione o Workspace:** Escolha o Workspace desejado.
- **Escolha uma skill:** Selecione uma skill cadastrada para configurar o comportamento.

Lunaris

- **Qual o nome do workspace?:** Selecione um workspace previamente cadastrado.
- **Qual o nome da pipeline?:** Escolha uma pipeline cadastrada para ser vinculada.

Atendimento Humano

- **Fila de Atendimento:** Selecione a fila de atendimento à qual o comportamento estará vinculado.

Gupshup

- **Título das opções:** Defina o texto que será exibido antes das opções de interação.
- **Número de telefone da conta:** Informe o número vinculado à conta Gupshup.

- **Canal do App:** Escolha o canal de integração do app.
- **Nome do App Gupshup:** Especifique o nome do app Gupshup.
- **Chave da API:** Insira a chave da API Gupshup para autenticação.

Facebook Messenger

- **Facebook ID:** Forneça o ID da página vinculada ao Messenger.
- **Token de Acesso:** Insira o token de acesso gerado pelo Facebook Messenger.

Positus

- **Título das opções:** Defina o texto que será mostrado antes das opções.
- **Código do número WhatsApp:** Informe o código associado ao número do WhatsApp.
- **Token de Acesso:** Insira o token de acesso para autenticar a integração com o Positus.

Slack

- **Slack ID:** Forneça o ID do workspace Slack para vinculação.
- **Token de Acesso:** Insira o token de acesso para autorização no Slack.

Twilio

- **SID da Conta:** Forneça o SID (Account SID) da sua conta Twilio.
- **Token de Acesso:** Insira o token de acesso gerado pela plataforma Twilio.
- **Contato:** Defina o número de contato que será utilizado nas interações.

Telegram

- **Qual o token do seu BOT?:** Código gerado pelo Telegram que identifica e autoriza o BOT a funcionar.
- **Qual o ID do chat?:** Número que identifica o chat onde o BOT vai atuar.

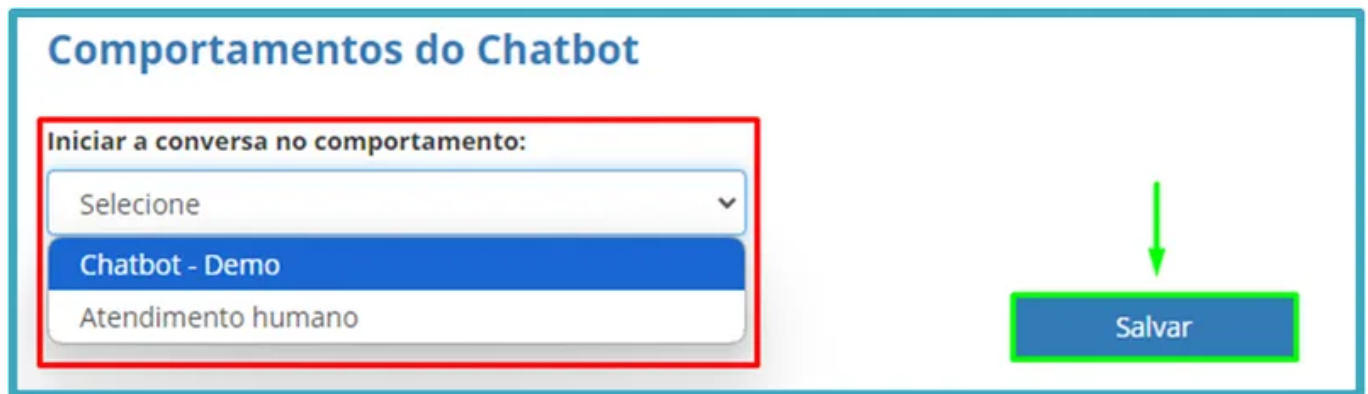
Microsoft Teams

- **Nome do Chatbot:** Nome que aparecerá no Microsoft Teams quando o BOT interagir com os usuários
- **ID do App:** Código exclusivo que identifica o aplicativo do BOT dentro do Microsoft Teams.

- **ID do Locatário:** Código representa a organização à qual o Microsoft Teams está vinculado.
- **Segredo do App:** Senha gerada pelo Microsoft Teams autorizando o seu funcionamento.

Configurações

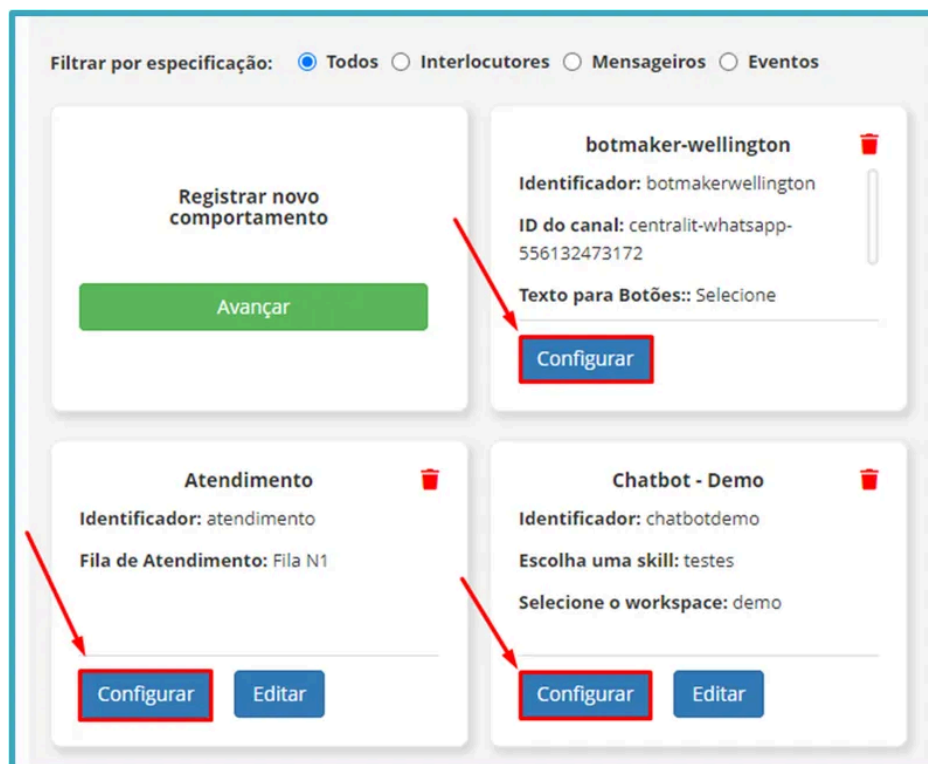
A opção **Iniciar a conversa no comportamento** permite definir qual skill será utilizada. Ao configurá-la, a skill escolhida inicia assim que o comportamento é acionado, garantindo que a conversa comece com o fluxo desejado. Confira a imagem abaixo para referência:



Atualizar Parâmetros

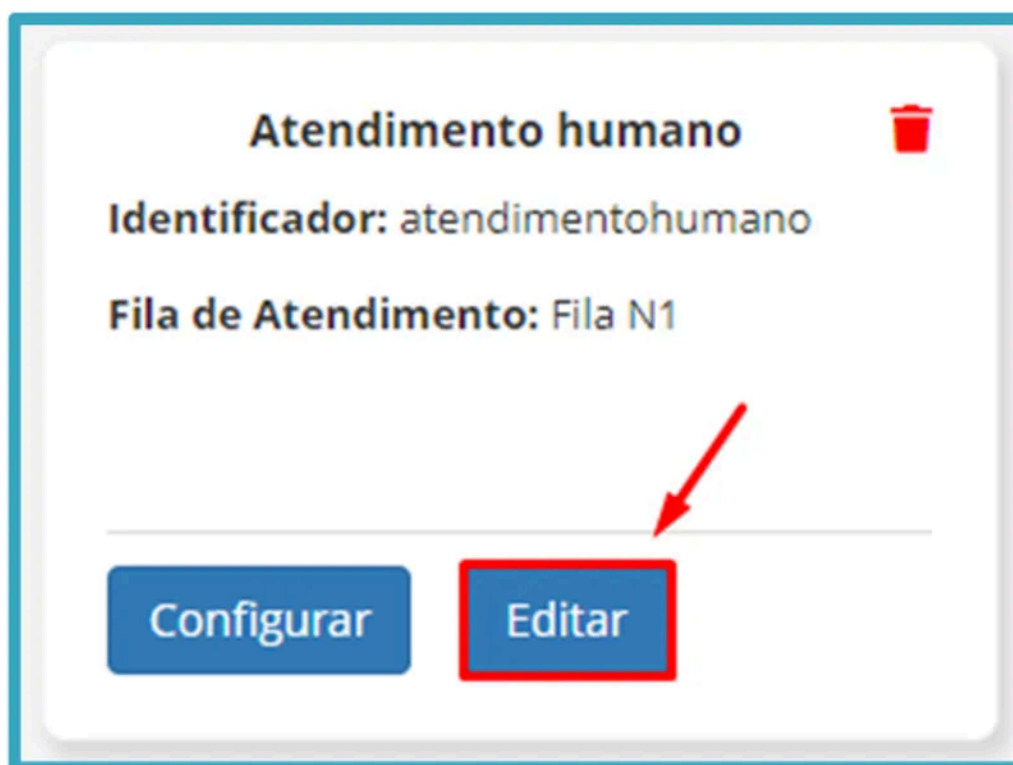
Após cadastrar um comportamento, você pode realizar novas configurações a qualquer momento. Basta selecionar o comportamento desejado e clicar em **Configurar**. Os parâmetros serão apresentados para que você os preencha novamente, permitindo ajustes conforme necessário.

Confira a imagem abaixo para referência:



Editar

Também é possível editar determinados comportamentos. Para isso, basta localizar o botão **Editar** no comportamento desejado e clicar nele. Você será direcionado para a área específica desse comportamento, onde poderá realizar as edições necessárias. Veja abaixo como o botão é apresentado:



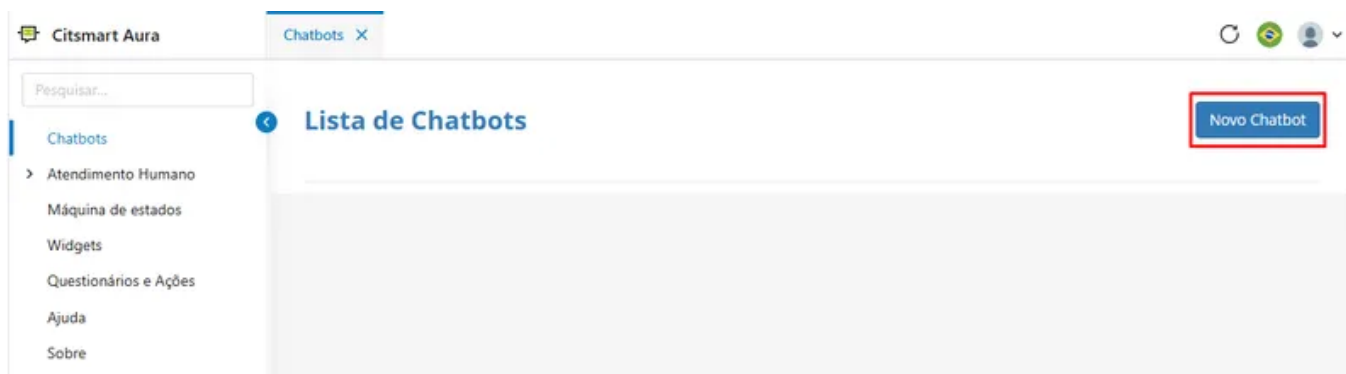
2.5. Comportamentos

Este manual tem como objetivo orientar o processo de criação de um chatbot e o registro adequado dos comportamentos.

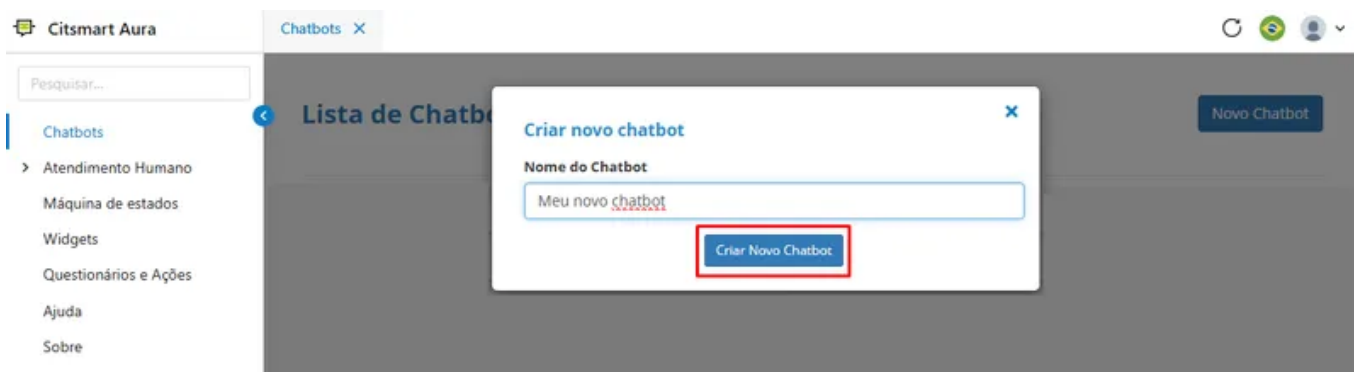
Criar novo chatbot

Esta funcionalidade está disponível apenas para o papel de **Desenvolvedor**.

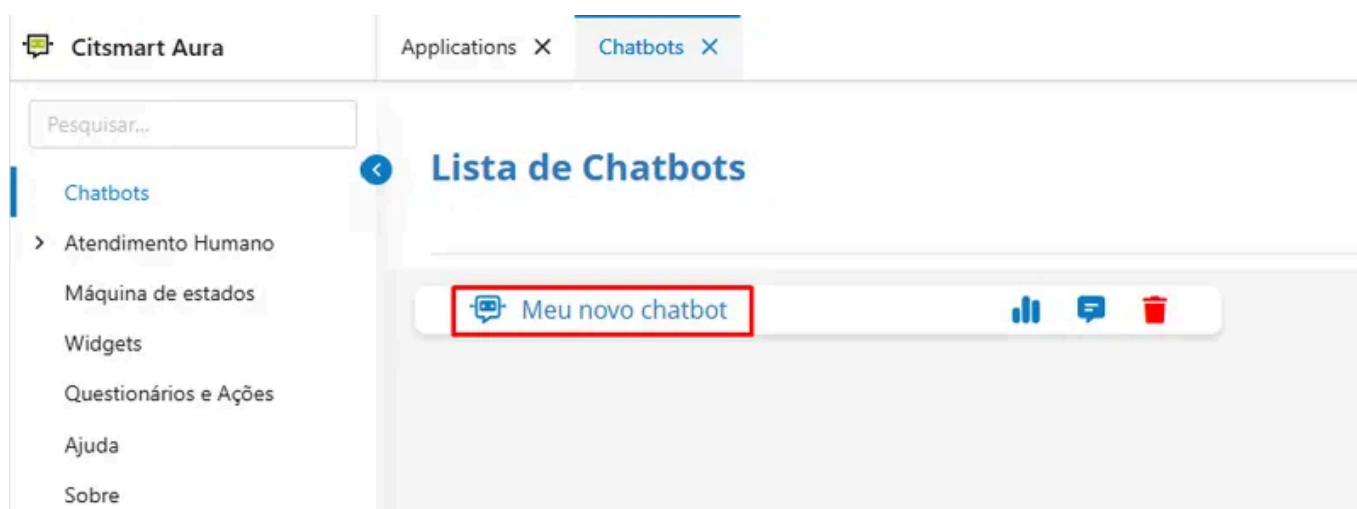
1. Acesse o menu "**CITSmart Aura**" e clique na opção "**Chatbots**".



1. Clique em "**Novo Chatbot**", insira o nome desejado e clique no botão "**Criar Novo Chatbot**".



1. Após criar o chatbot, clique sobre seu nome para acessar as funcionalidades.



Dashboard

Esta funcionalidade está disponível apenas para o papel de **Supervisor**.

Na opção "**Dashboard**", é possível visualizar métricas importantes sobre as interações do chatbot.

Listagem de histórico de conversas

Esta funcionalidade está disponível apenas para o papel de **Supervisor**.

Acesse a "**Listagem de históricos de conversas**" para visualizar todas as interações realizadas com o chatbot.

Histórico de Conversas

Selecione a data inicial: dd/mm/aaaa --:-- Seleccione a data final: dd/mm/aaaa --:-- Canais: Total de Conversas: 3 widget

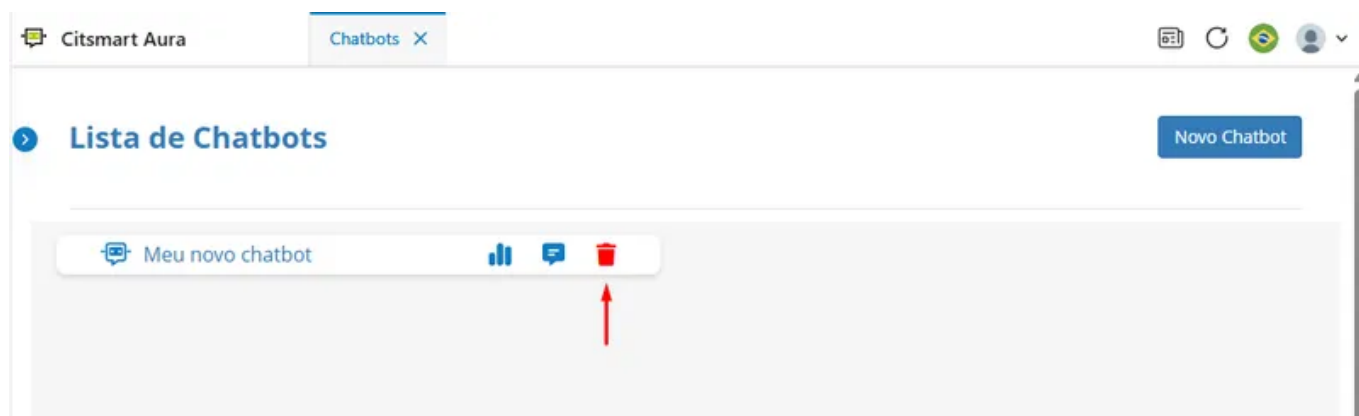
Pesquisar

Canal	Cliente	Data	Mensagens	Duração
widget	ch:5Tmv4TndKEW7_qQ6ow07DA==	03/04/2025	2	00:00:13
widget	ch:ki7iC7cTUKMNe6RSL2DF_Q==	03/04/2025	1	00:00:00
widget	ch:nWcWy7M8okjGtYOfvXqGYg==	03/04/2025	8	00:01:39

Excluir

Esta funcionalidade está disponível apenas para o papel de **Desenvolvedor**.

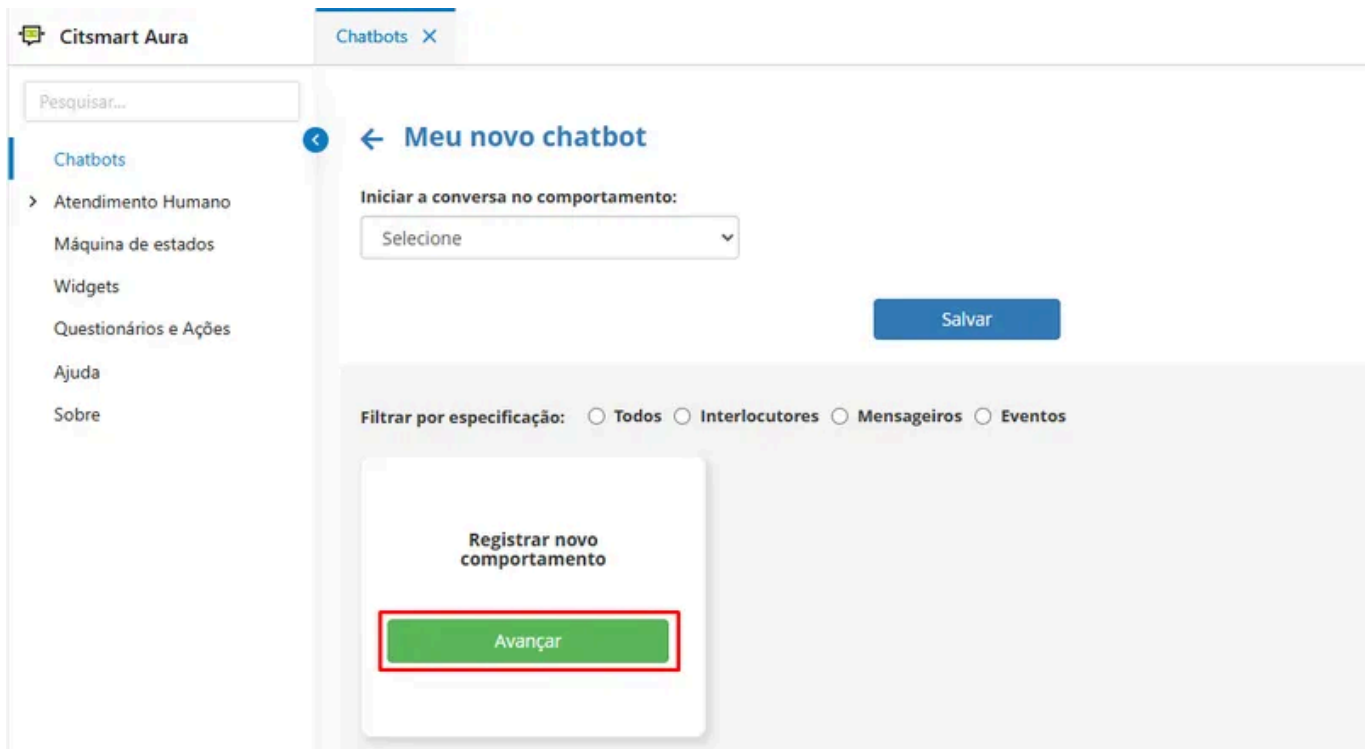
1. Para excluir um chatbot, clique no ícone "**Excluir**" e uma mensagem de confirmação será exibida.
 - Para confirmar a exclusão, clique em "**Sim**"
 - Para cancelar, clique em "**Não**".



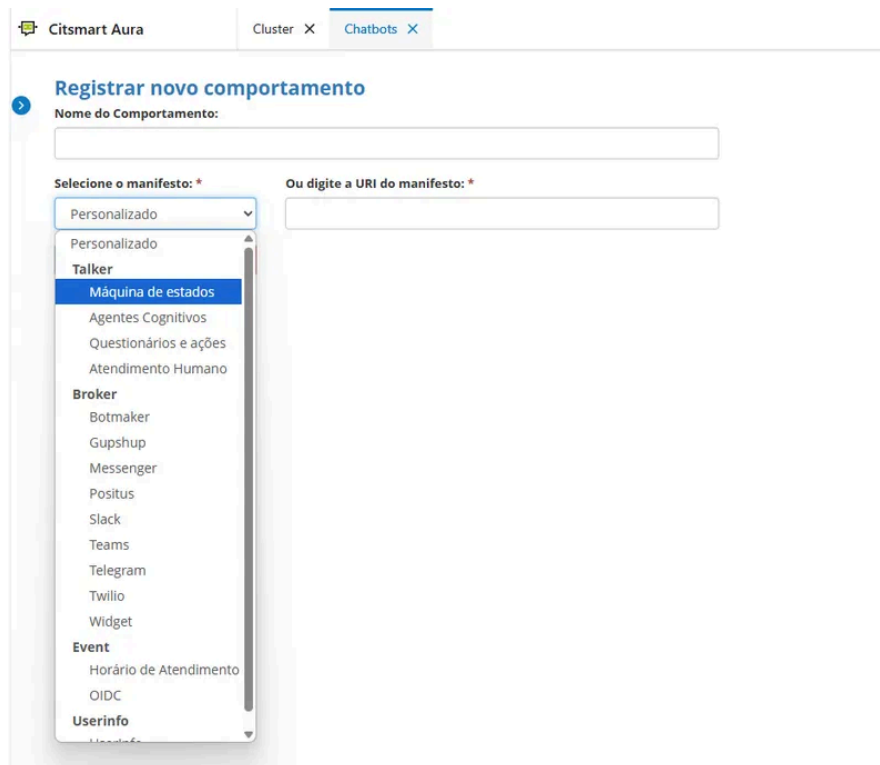
Registrar comportamentos

Esta funcionalidade está disponível apenas para o papel de **Desenvolvedor**.

1. No chatbot criado, localize a opção "**Registrar novo comportamento**" e clique em "**Avançar**".



1. Escolha o tipo de comportamento a ser cadastrado.
No exemplo abaixo, vamos cadastrar o comportamento "Máquina de estados".



1. O sistema solicitará as seguintes informações:
 - Nome: Meu novo chatbot

- Manifesto: Máquina de estados
- Login: máquina
- Senha: 123456

Citsmart Aura Chatbots X

Pesquisar...

Chatbots

Atendimento Humano

Máquina de estados

Widgets

Questionários e Ações

Ajuda

Sobre

Registrar novo comportamento

Nome do Comportamento:

Novo fluxo

Selecione o manifesto: *

Máquina de estados

Ou digite a URI do manifesto: *

https://aura-blocks.centralit.com.br/lunaris_dev/get/wander

Usuário: *

maquina

Senha: *

Avançar Cancelar

1. Após preencher os dados, clique em "**Avançar**".

Informações adicionais

Cada comportamento possui critérios e parâmetros específicos para sua configuração. Por isso, é essencial acessar informações adicionais para garantir uma configuração precisa e eficiente. Confira abaixo os links com orientações específicas para cada canal:

- [Botmaker](#)
- [Facebook Messenger](#)
- [Gupshup](#)
- [Positus](#)
- [Slack](#)
- [Microsoft Teams](#)
- [Telegram](#)
- [Twilio](#)

- **Manual do widget**

2.6. Backup

Esta documentação detalha métodos recomendados para realizar backup das bases de dados e dos volumes persistentes (PVC) da plataforma Aura. Cobre abordagens manuais e uso de ferramentas de backup, sem detalhar ou citar nomes de softwares específicos.

1. Backup de Bancos de Dados (PostgreSQL)

Existe apenas uma base de dados PostgreSQL por ambiente (Produção, Teste, Homologação, etc.), necessário para o funcionamento da plataforma Aura:

- **Nome padrão:** `aura_blocks`
-

1.1. Métodos de Backup do PostgreSQL

1.1.1. Backup Manual com pg_dump (linha de comando)

O método mais comum e recomendado para backup lógico do PostgreSQL utiliza comandos nativos da ferramenta, geralmente executados a partir do pod ou da máquina onde está o banco.

Exemplo A: Dump direto do pod para o host local

Text



```
kubectl exec -it <namespace>/<postgresdb-pod> -- pg_dump -U <db_user> <db_name>
```

Este comando executa o `pg_dump` diretamente dentro do pod e redireciona o resultado (`backup.sql`) para o seu host local, sem necessidade de copiar depois.

- **Prós:** Prático, evita etapas extras.
- **Contras:** O dump é transferido pela rede; em bancos muito grandes, avalie limitações de banda e possíveis timeouts do `kubectl`.

Exemplo B: Dump gravado no pod e copiado depois

Text



```
kubectl exec -it <namespace>/<postgresdb-pod> -- bash -c "pg_dump -U <db_user>  
kubectl cp <namespace>/<postgresdb-pod>:/tmp/backup.sql ./backup.sql
```

Use este método se preferir salvar o dump no pod e copiar para o host em seguida.

1.1.2. Backup via Interface Gráfica

Se preferir interface gráfica, utilize ferramentas de administração de bancos de dados compatíveis com PostgreSQL, seguindo a documentação oficial dessas ferramentas.

1.1.3. Backup Avançado

Ferramentas especializadas podem ser utilizadas para automatizar, agendar e aprimorar backups em ambientes críticos ou com alto volume de dados, permitindo, por exemplo, backups incrementais, upload em nuvem, gerenciamento de retenção, entre outros recursos.

- Consulte a documentação das ferramentas de sua escolha para instalação, configuração e restore.

1.1.4. Backup do Volume do PostgreSQL (PVC)

Também é possível realizar backup do volume do banco utilizando métodos manuais (como cópia de arquivos) ou ferramentas de backup de volumes persistentes.

Importante: Para garantir a consistência do backup, recomenda-se pausar o banco ou executá-lo em modo de backup durante o processo.

1.2. Restauração

1.2.1. Restauração Manual

Importante: Antes de restaurar o backup, certifique-se de que a base de dados (`<db_name>`) já está criada no PostgreSQL. O comando `psql` não cria a base automaticamente durante a restauração.

Text



```
# Exemplo: criando a base de dados se ainda não existir
kubectl exec -it <namespace>/<postgresdb-pod> -- psql -U <db_user> -c "CREATE
```

Depois, prossiga com a restauração.

Exemplo A: Restauração a partir do host local direto do pod

Text



```
cat ./backup.sql | kubectl exec -i <namespace>/<postgresdb-pod> -- psql -U <d
```

Exemplo B: Copia o backup para o pod e depois é feita a restauração

```
kubectl cp ./backup.sql <namespace>/<postgresdb-pod>:/tmp/backup.sql
kubectl exec -it <postgresdb-pod> -- psql -U <db_user> -d <db_name> -f /tmp/b
```

1.3. Boas Práticas

- Automatize backups periódicos em produção.
- Mantenha múltiplas cópias (offsite/nuvem/local).
- Sempre teste o restore em ambiente de homologação.
- Documente o processo de restauração.

1.4. Comparativo de Métodos

Método	Consistênci a	Automatizáv el	Offsite	Complexida de	Recuperaçã o
Comandos nativos	Sim	Sim	Sim	Baixa	Fácil
Ferramentas especializadas	Sim	Sim	Sim	Média	Avançada
Backup do volume (PVC)	Sim*	Sim	Sim	Média	Rápida

*Consistência depende do banco estar pausado ou em modo de backup durante o procedimento.

2. Backup de Volumes Persistentes (PVC)

Os volumes persistentes (PVC) estão descritos nos manifestos do Kubernetes, geralmente no arquivo `persistentvolumeclaim.yaml`. Exemplo de PVC:

YAML



```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: storage
  namespace: aura-blocks
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 50Gi
  storageClassName: <storageClassName>
```

A escolha do `storageClassName` impacta diretamente as opções de backup disponíveis.

2.1. Métodos de Backup de PVC

2.1.1. Backup Manual

Realize backup copiando arquivos diretamente do volume do pod para outro local seguro.

Exemplo A: Usando `kubectl cp`

Text



```
kubectl cp <namespace>/<pod>:/caminho/dados ./backup-local
```

Substitua `<namespace>`, `<pod>` e o caminho do volume conforme sua aplicação.

Exemplo B: Usando container temporário + tar

Text



```
kubectl run backup-job --rm -i -t --restart=Never \  
  --image=alpine \  
  --overrides='{ "spec": { "volumes": [ { "name": "data", "persistentVolumeClaim"
```

sh

Dentro do shell:

Text



```
tar czvf /tmp/backup.tar.gz /caminho/dados
```

Em outro shell, copie o arquivo do pod:

Text



```
kubectl cp <namespace>/backup-job:/tmp/backup.tar.gz ./backup.tar.gz
```

Dicas:

- Compacte arquivos grandes (`tar` , `gzip`).
- Faça backup regularmente e teste a restauração.

2.2. Comparativo dos Métodos

Método	Suporte a Volumes	Backup Offsite	Automação	Complexidade	Volume Grande
Manual (<code>cp</code> , <code>tar</code>)	Sim	Sim	Baixa	Baixa	Médio
Ferramentas de backup	Sim	Sim	Alta	Média/Alta	Ótimo

3. Recomendações Finais

- Realize backup periódico dos dados críticos.
- Prefira automação para ambientes produtivos.
- Sempre teste a restauração!
- Documente o processo de restauração para cada método.
- Utilize múltiplas estratégias para resiliência máxima.

Referências

- [Backup PostgreSQL](#)
- [PgAdmin Backup](#)

3. Interlocutores

3.1. Atendimento humano

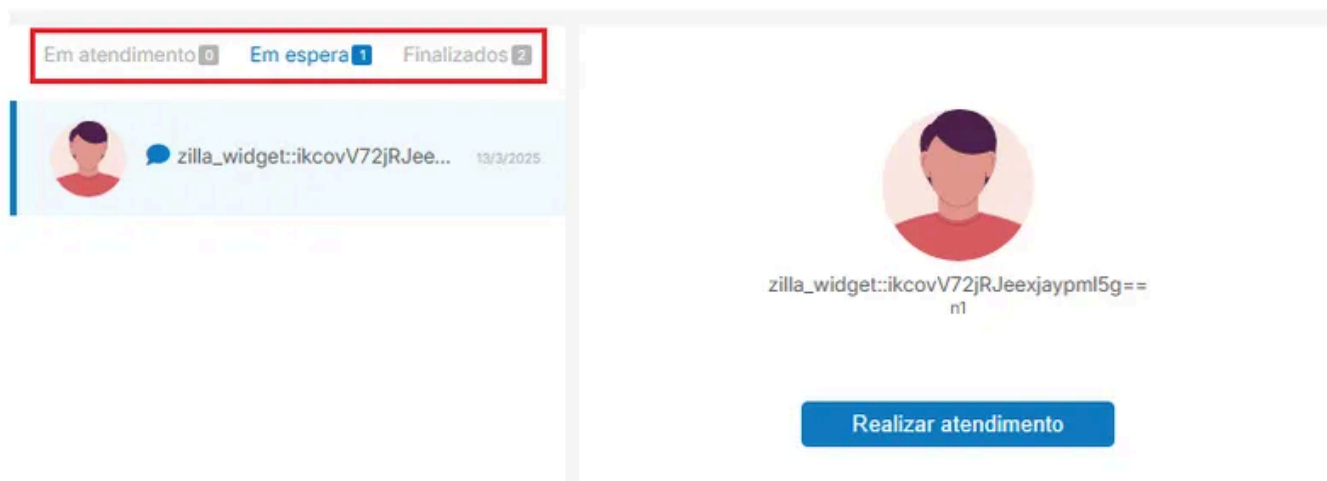
Este manual fornece orientações sobre como utilizar as funcionalidades da ferramenta de atendimento, incluindo a gestão de filas e a criação de atalhos de texto.

Atendimentos

Esta funcionalidade está disponível apenas para o papel de **Atendente**.

A área de atendimento humano é composta por três seções principais, cada uma desempenhando um papel crucial na gestão de interações com os usuários. Abaixo estão as descrições de cada seção:

Atendimentos

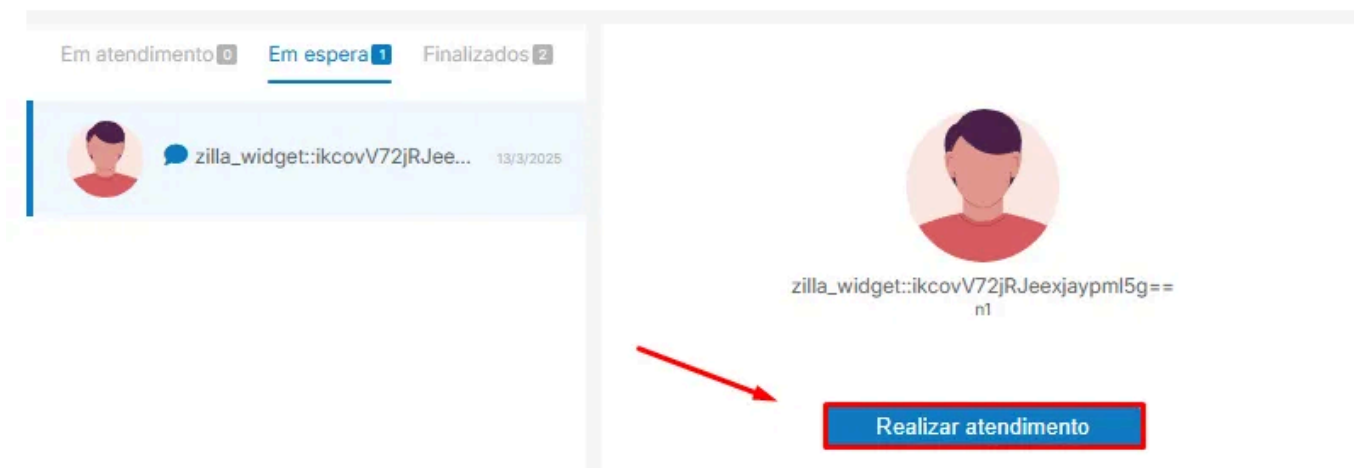


- **Em atendimento** - Exibe os atendimentos que já foram capturados e estão em progresso.
- **Em espera** - Mostra os atendimentos que ainda não foram capturados e estão disponíveis para atendimento.
- **Finalizados** - Apresenta o histórico de atendimentos concluídos pelo atendente.

Em espera

Para capturar um atendimento disponível, clique na opção "**Realizar atendimento**", como mostrado na imagem abaixo:

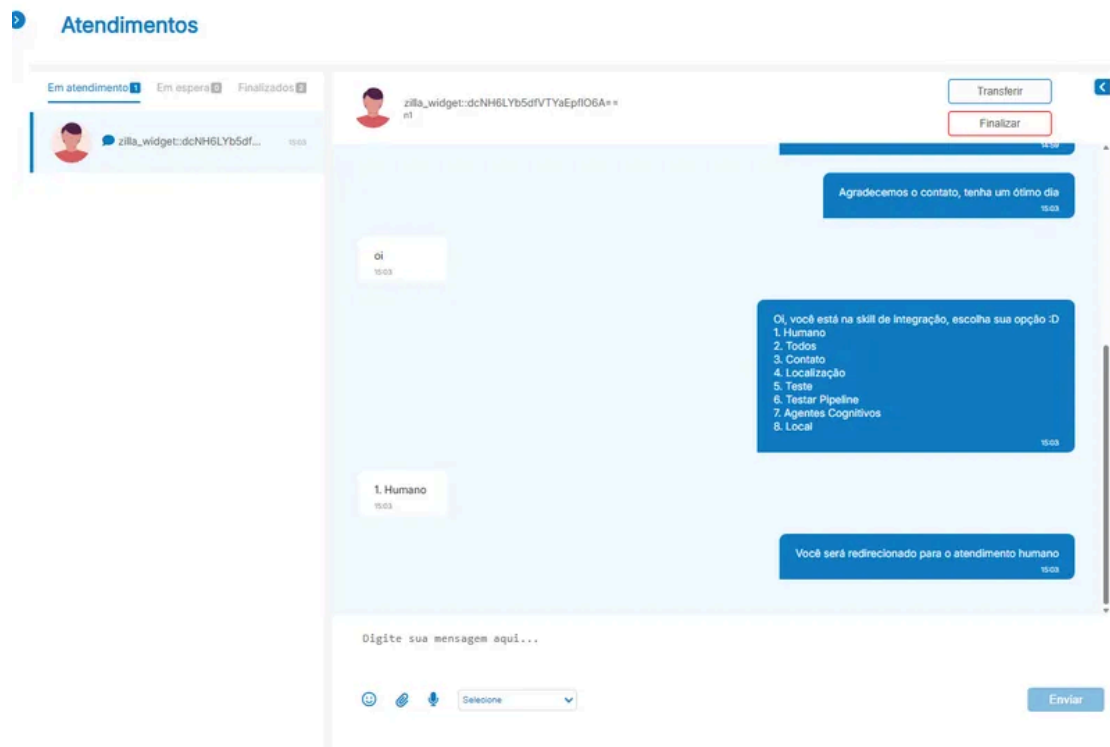
Atendimentos



Ao selecionar essa opção, a solicitação será movida para a seção **"Em atendimento"**, ficando acessível apenas para o atendente que efetuou a captura.

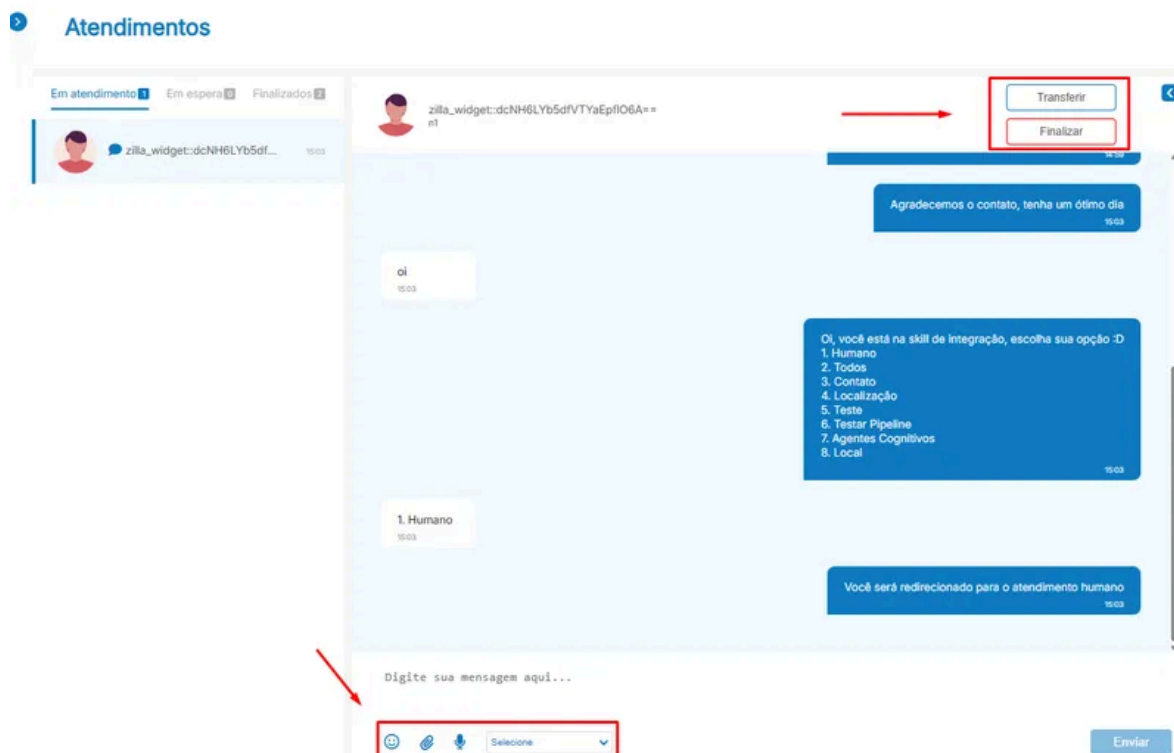
Em atendimento

Ao clicar em **"Realizar atendimento"**, o atendimento será automaticamente movido para a aba **"Em Atendimento"**, onde o atendente poderá prestar um suporte personalizado ao usuário que necessita de assistência.



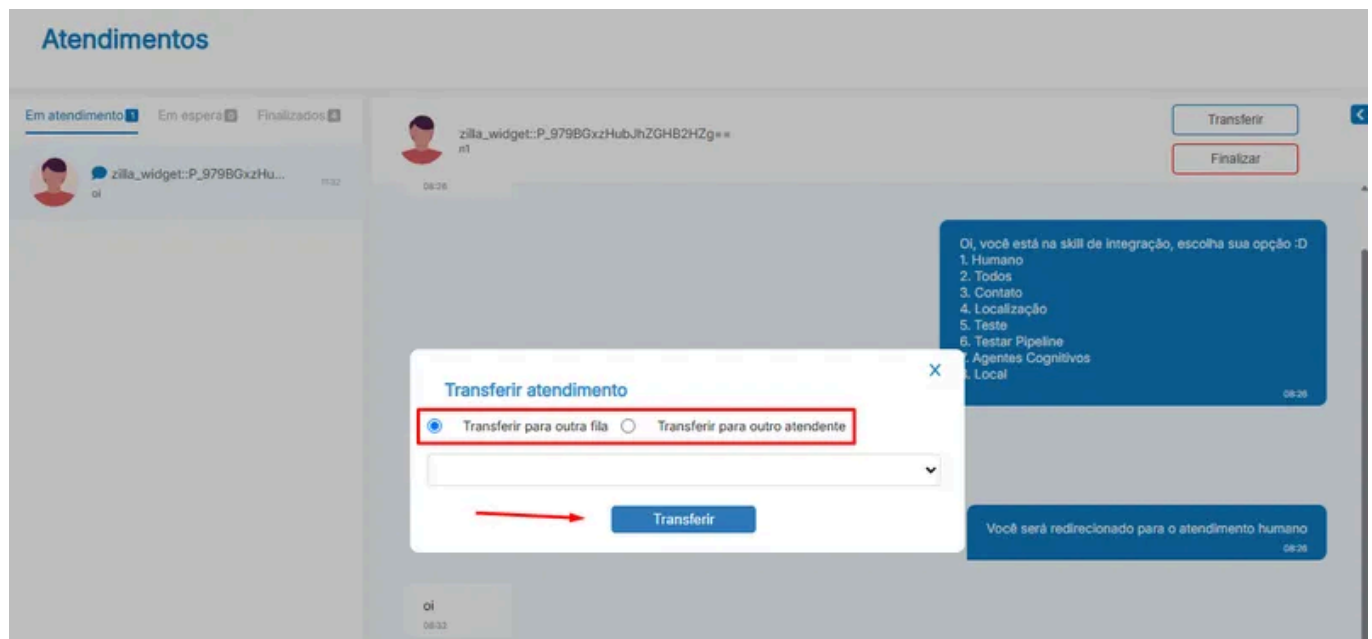
Tela de interação com o usuário

Nesta tela, o atendente pode se comunicar diretamente com o usuário em tempo real, trocando mensagens, enviando arquivos de áudio, imagens (PNG, JPG, JPEG), documentos (PDF, Word, Excel) e emojis, além de utilizar atalhos de texto para respostas rápidas.



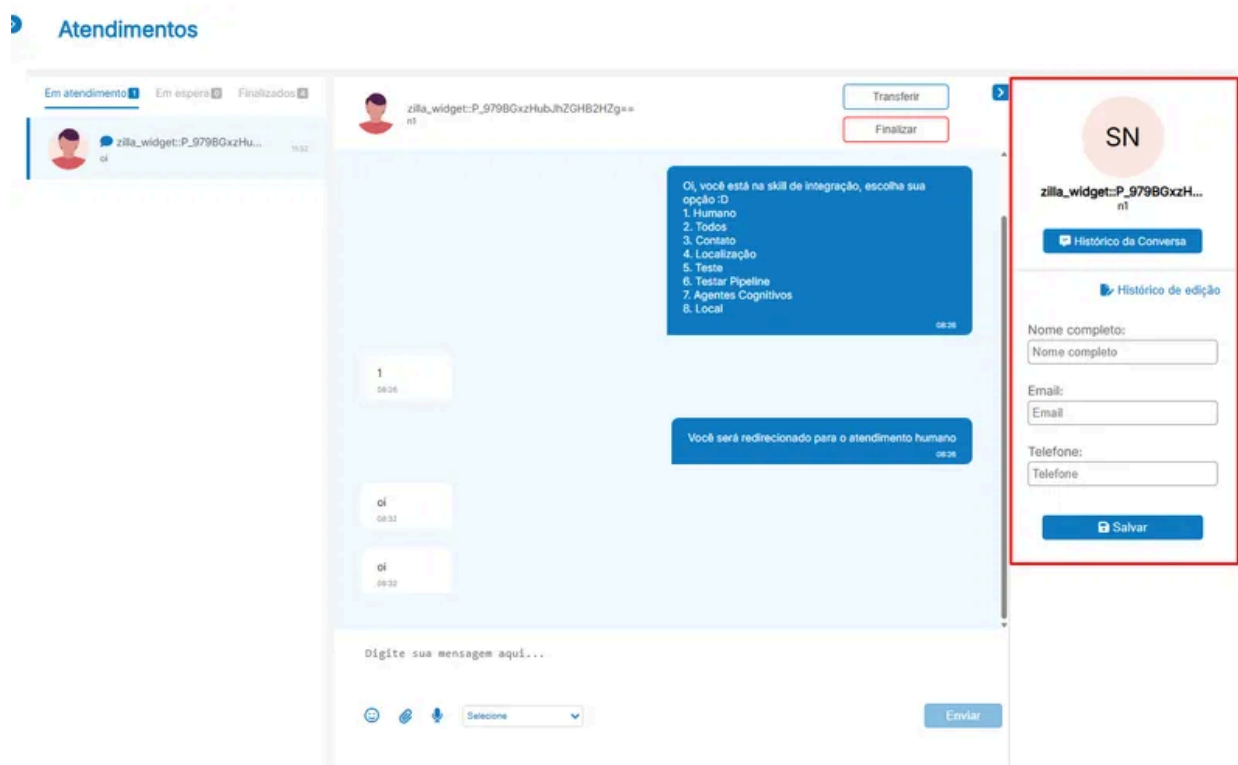
Veja as opções disponíveis nessa tela:

- **Transferir:** Durante o atendimento é possível transferir a solicitação da seguinte maneira:
 - **Transferir para outra fila:** Permite transferir o atendimento de uma fila para outra fila, garantindo que o usuário seja encaminhado ao setor mais adequado para sua demanda.
 - **Transferir para outro atendente:** Possibilita a transferência do atendimento para um atendente específico, assegurando um suporte mais eficiente e direcionado.



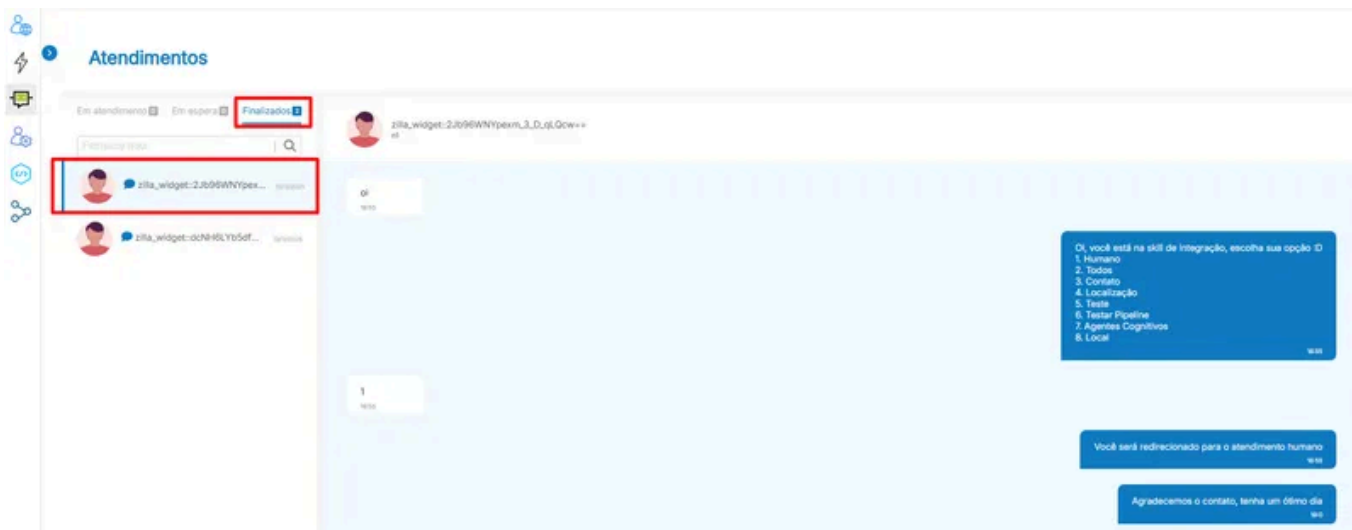
- **Finalizar:** Encerra o atendimento específico, registrando o histórico da conversa e mantendo o atendente disponível para continuar outros atendimentos simultaneamente.
- **Emojis:** Essa opção permite que atendentes e usuários utilizem emojis para tornar a comunicação mais interativa, expressiva e dinâmica.
- **Upload de arquivos:** Possibilita enviar arquivos e documentos durante o atendimento.
- **Upload de áudio:** Permite o envio de arquivos de áudio durante o atendimento humano, facilitando a comunicação e proporcionando uma alternativa eficiente para esclarecer dúvidas ou fornecer informações de forma mais dinâmica.
- **Atalhos:** Possibilita enviar os atalhos de texto criados.

- **User info:** Ao abrir o menu lateral, serão exibidas as informações do usuário, incluindo nome completo, e-mail, telefone e o canal de acesso utilizado. Além disso, poderão ser apresentadas outras informações relevantes, como o histórico de interações.



Finalizados

Esta seção exibe o histórico dos atendimentos concluídos, permitindo que você revise as mensagens trocadas e os detalhes de cada atendimento para acompanhamento e análise. Para auditar, basta clicar no atendimento desejado ou usar a opção de pesquisa na lupa ("pesquise aqui") para localizar o atendimento que deseja revisar.



Configurando o atendimento humano no Chatbot

Para configurar o atendimento humano no seu chatbot, é preciso adicionar o comportamento **Atendimento humano** e, em seguida, associar um mediador (broker). Abaixo, você vai encontrar o passo a passo para:

1. Registrar o comportamento do **atendimento humano**.
2. Vincular o **mediador (broker)**.
3. Integrar bot + atendimento humano com a **Máquina de Estados**.

Pré-requisitos

Antes de iniciar, verifique se os seguintes itens já estão configurados:

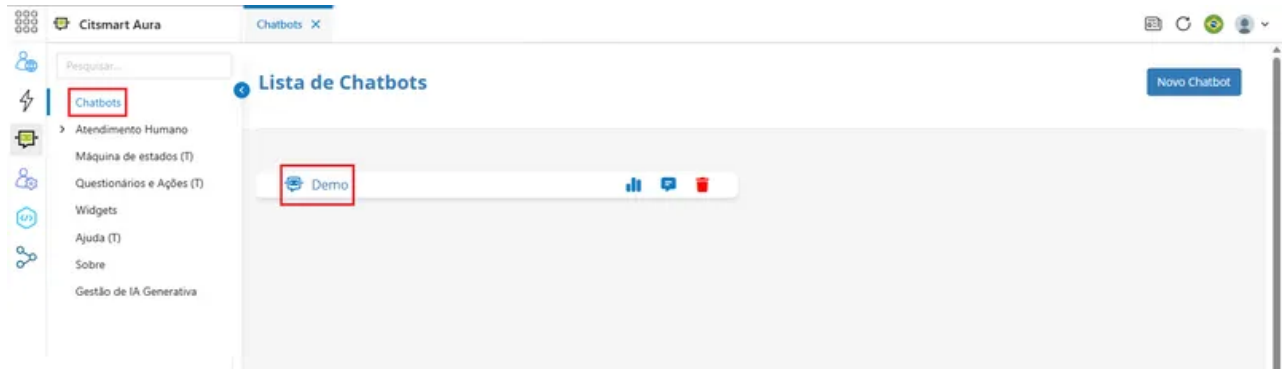
- **Fila de atendimentos:** Crie uma fila de atendimentos e cadastre os atendentes.
- **Chatbot:** Crie um chatbot ou acesse um já existente.
- **Máquina de estados (Skill):** Certifique-se de ter criado uma skill para registrar no chatbot. Veja como criar sua NLU [neste guia](#).
- **Mediador (Broker):** É necessário escolher ao menos um broker para ser registrado no chatbot. Os brokers disponíveis são:

botmaker , gupshup , messenger , positus , slack , teams , telegram ,

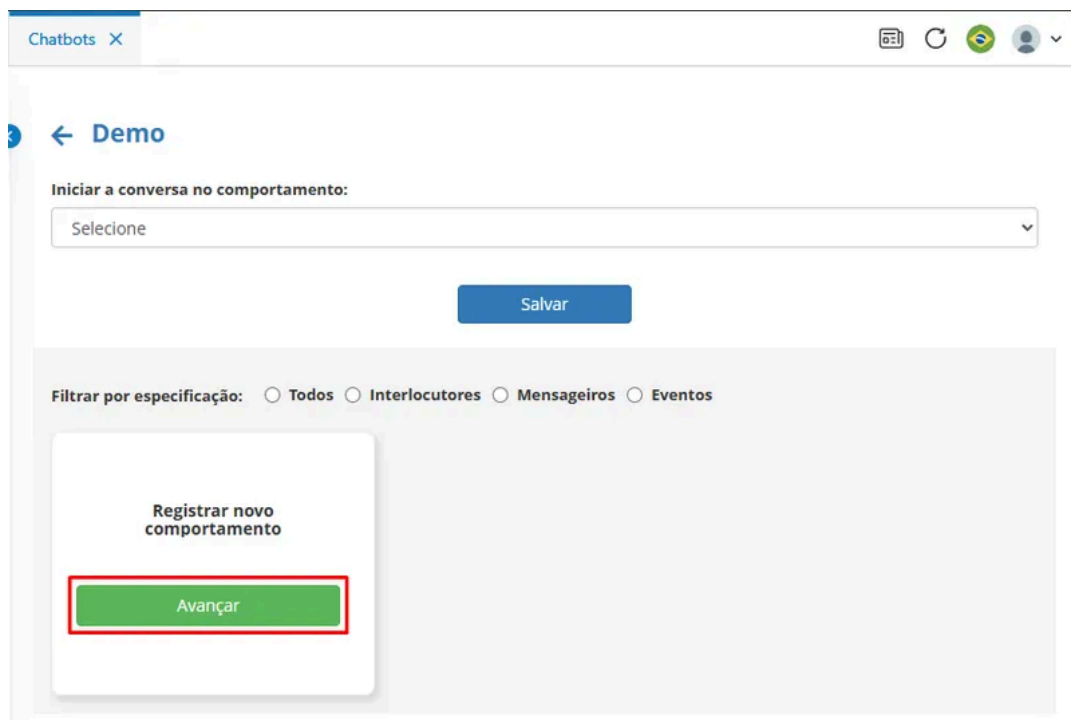
`twilio` e `widget`. Neste exemplo, usaremos o **Widget**. Se ainda não tiver um, crie-o seguindo este [tutorial](#).

Passo 1 – Registrar atendimento humano no chatbot

1. No menu **Chatbot**, selecione o bot que deseja configurar.



2. Localize a opção **Registrar novo comportamento** e depois clique em **Avançar**.



3. Na tela de cadastro, preencha:

Citsmart Aura Chatbots X

Registrar novo comportamento

Nome do Comportamento:

Atendimento Humano

Selecione o manifesto: * Ou digite a URI do manifesto: *

Atendimento Hum v Ou digite a URI do manifesto: * https://aura-blocks.centralit.com.br/attendant_test/

Parâmetros ?

v.0.1.6.rc20

Fila de Atendimento *

N1 v

Concluir Cancelar

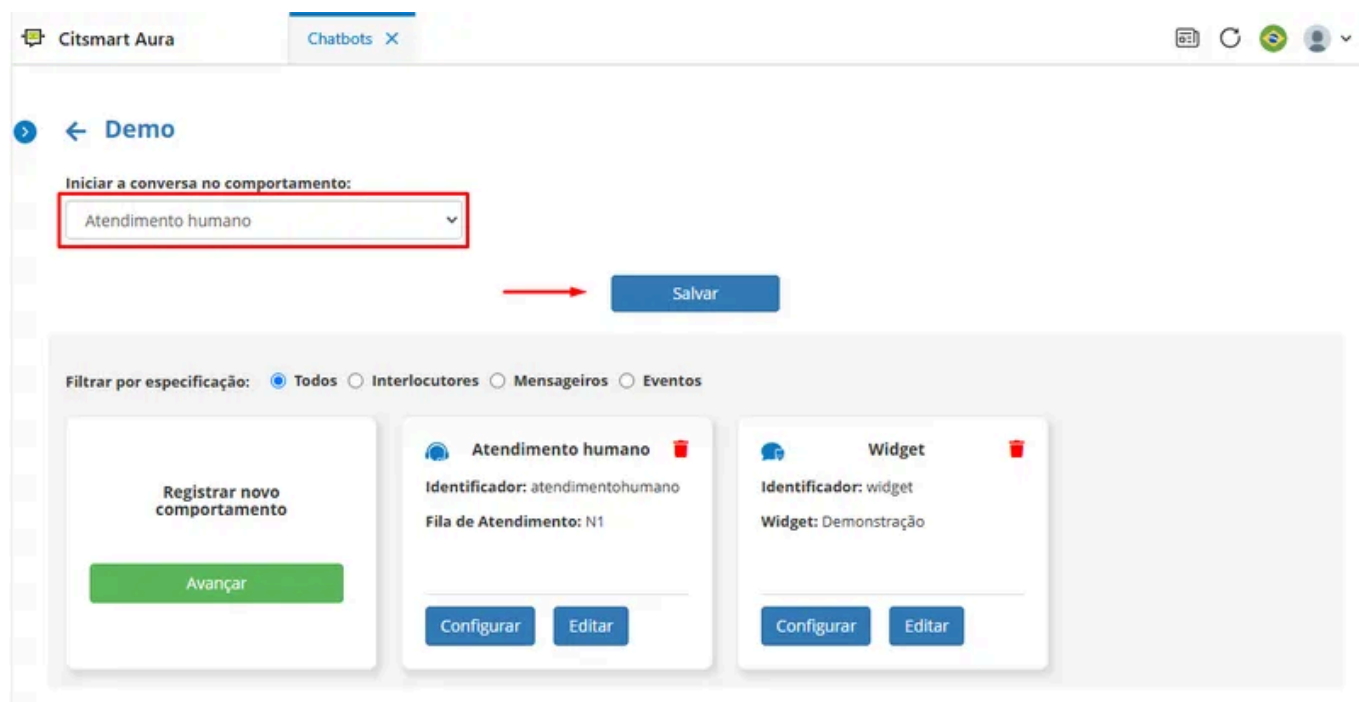
Passo 2 – Vincular o Mediador (Broker)

Observação: Crie o broker (no nosso exemplo, o Widget) antes de registrar no chatbot. Veja como em [widget](#).

1. No mesmo chatbot, localize a opção **Registrar novo comportamento** e clique em **Avançar**.
2. Informe o **Nome do comportamento**.
3. Em **manifesto**, selecione **Widget** e avance.
4. Em **Parâmetros**, escolha o widget criado e conclua.

Pronto! Após a configuração do broker, já é possível transferir o usuário para o atendimento humano.

Observação: Verifique em **Iniciar a conversa no comportamento** se o **Atendimento humano** está selecionado. Caso contrário, escolha-o e clique em **Salvar**.



Integração Bot + Atendimento humano + Máquina de estados

Para que o atendimento humano funcione integrado ao bot, é necessário registrar uma máquina de estados e configurar o estado responsável pela transferência. Siga os passos abaixo para realizar essas configurações.

Passo 1 – Registrar máquina de estados (Skill)

Observação: Crie uma skill antes de registrar no chatbot. Consulte as instruções [aqui](#).

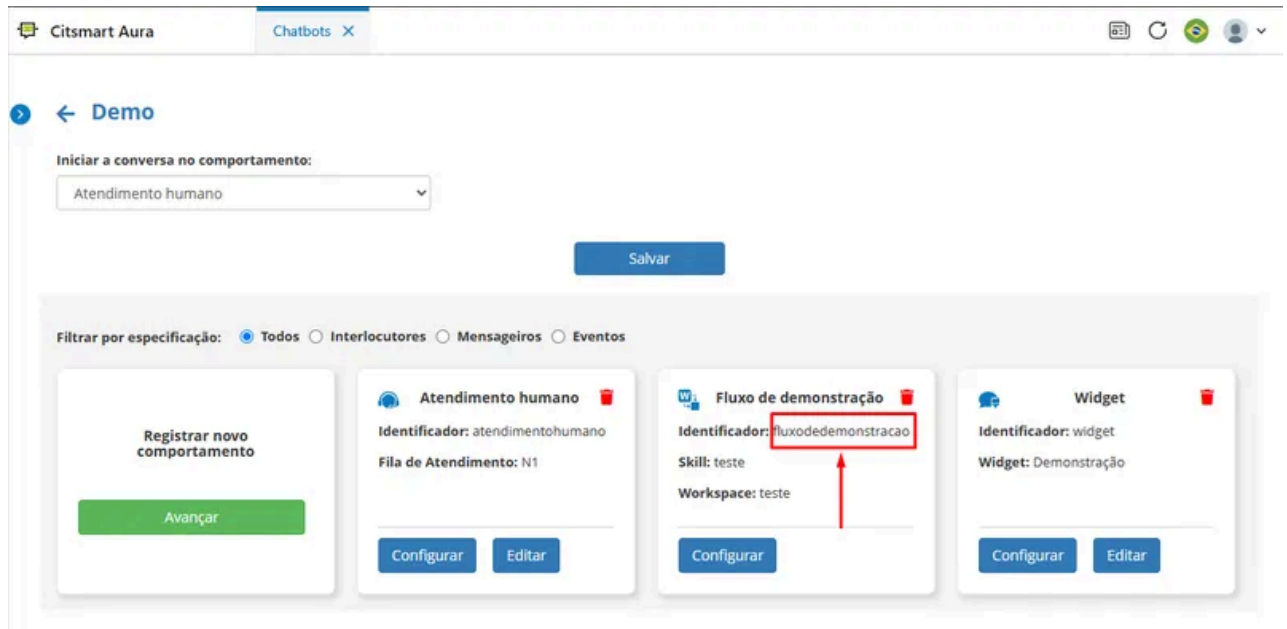
1. No chatbot, localize a opção **Registrar novo comportamento** e clique em **Avançar**.
2. Informe o nome do comportamento a ser criado.
3. Em **Manifesto**, selecione **Máquina de estados** e avance.
4. Em **Parâmetros**:
 1. Escolha o **Workspace**.
 2. Selecione a **Skill**.
 3. Finalize em **Concluir**.

A máquina de estados foi registrada em seu chatbot

Passo 2 – Configurar fila de atendimentos no fluxo

Agora é preciso configurar a fila de atendimentos **dentro do fluxo** da Máquina de Estados, para isso, siga os passos abaixo:

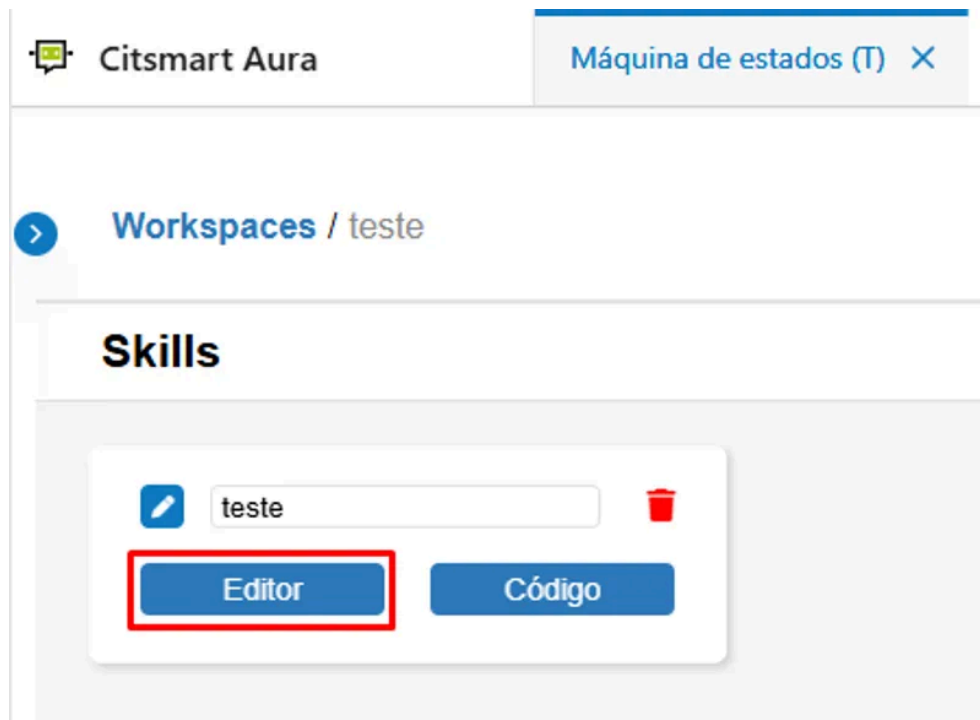
1. Na lista de comportamentos do chatbot, localize o comportamento **Máquina de estados** e copie o **Identificador**.



2. Navegue para a funcionalidade **Máquina de estados** e na funcionalidade skills, selecione umas das opções abaixo para realizar a edição:

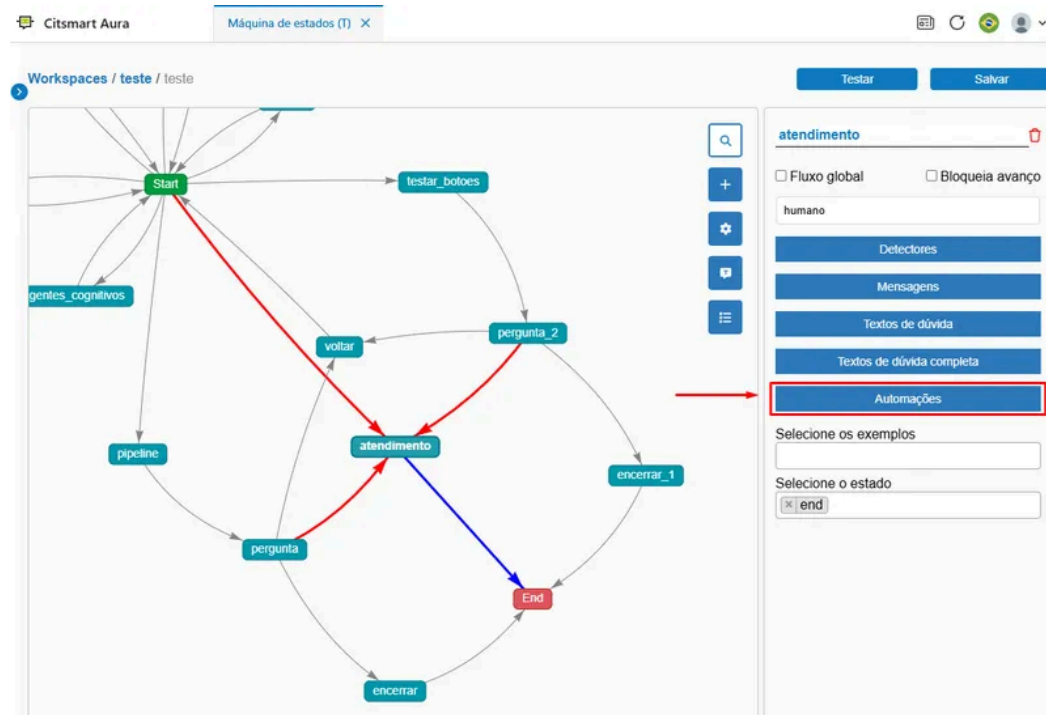
Utilizando o "Editor visual"

1. Na skill, clique em **Editor**.

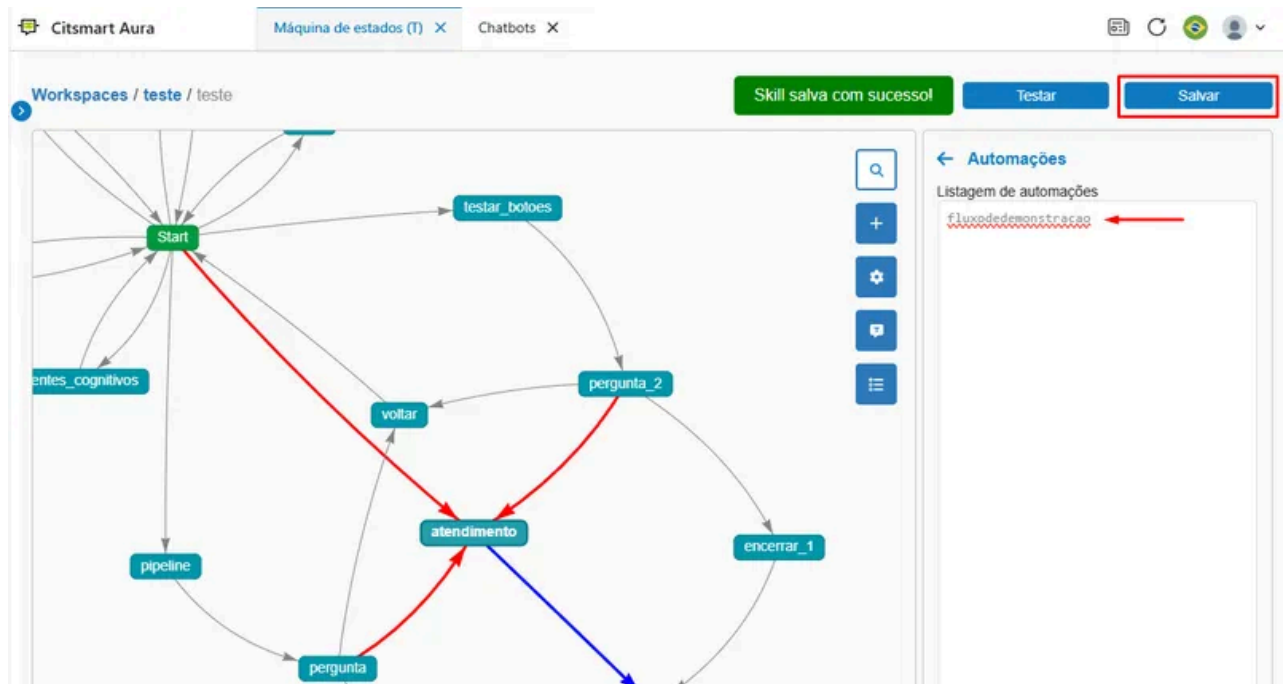


2. Selecione o estado onde ocorrerá o atendimento humano.

3. No painel lateral, vá em **Automações**.

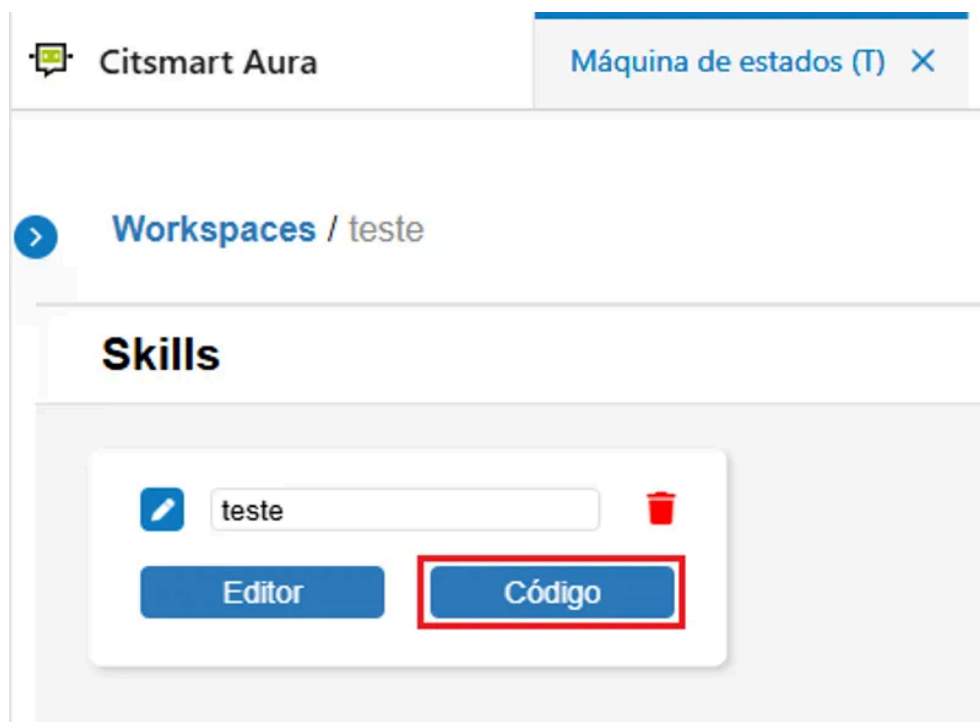


4. Cole o **Identificador** no campo e clique em "**Salvar**".



Utilizando o "Editor de código"

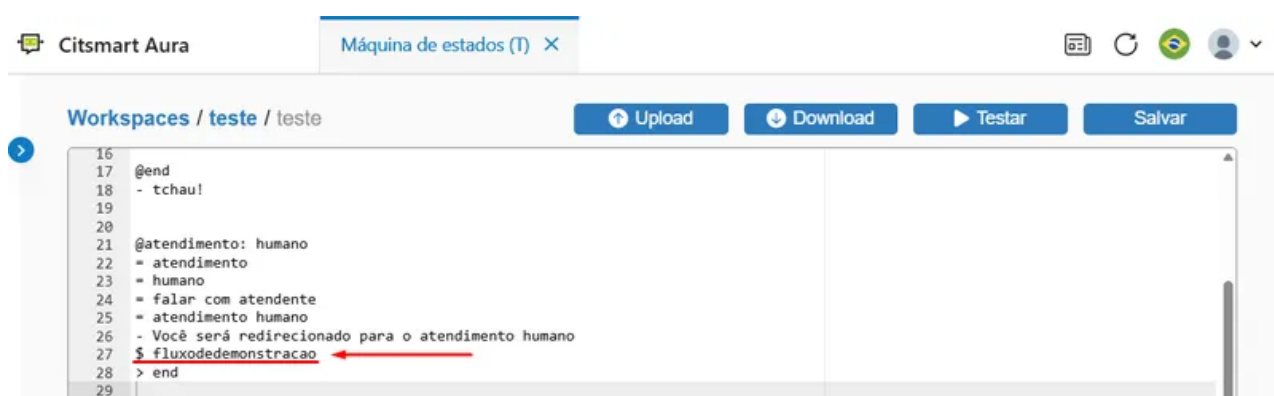
1. Na skill, clique em **Código**.



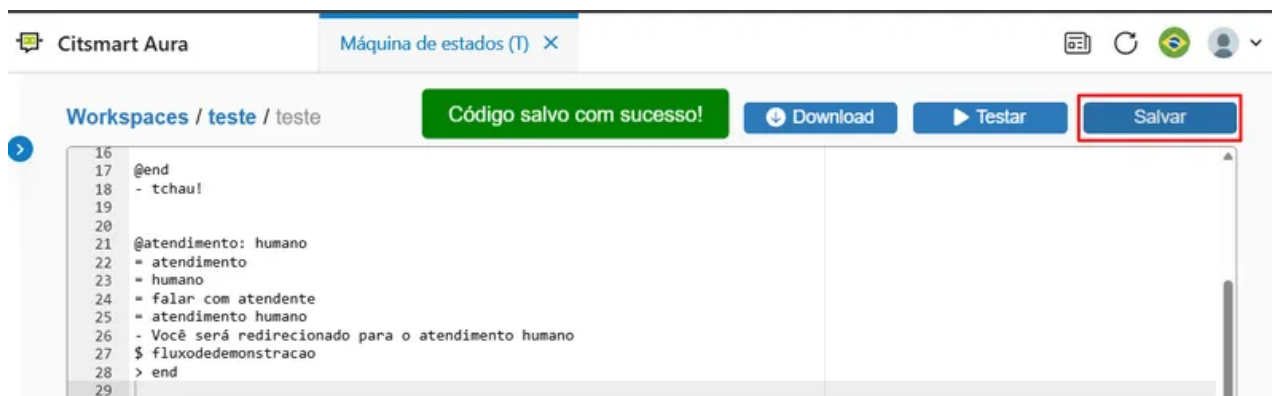
2. Localize (ou pesquise com CTRL + F) o estado escolhido para realizar o transferência.



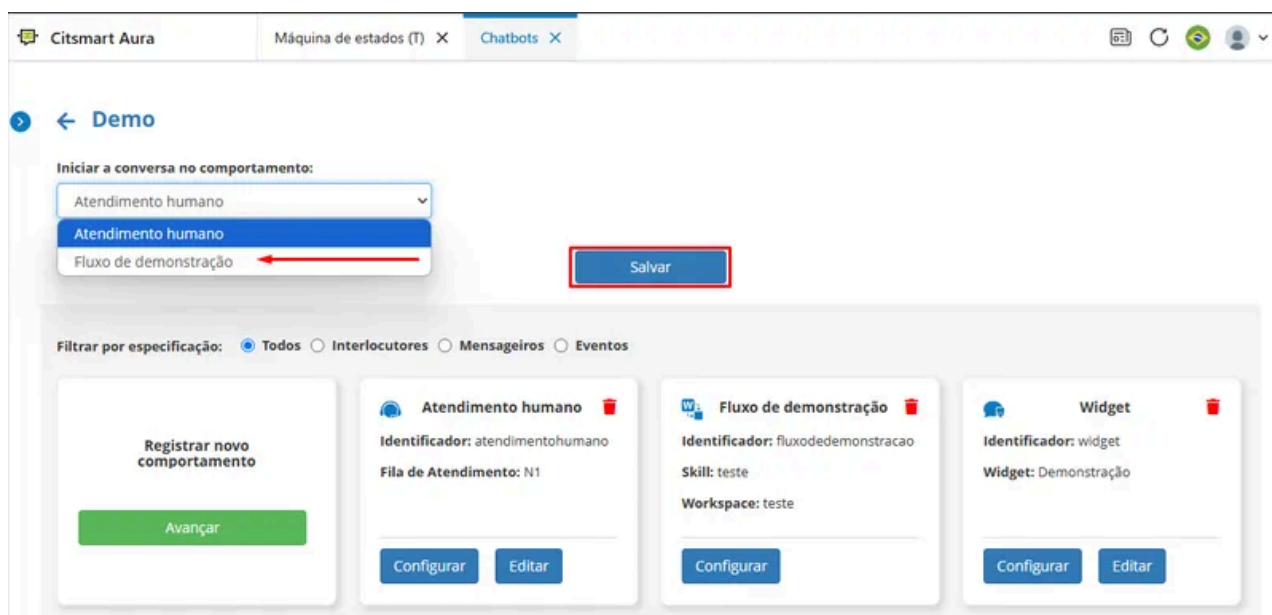
3. Insira as informações \$ <Identificador> dentro do estado responsável pela transferência, conforme imagem:



4. Conclua as alterações clicando em **Salvar** no canto superior direito da tela.



- Por fim, retorne à tela de **Listagem de comportamentos** e, na opção **Iniciar a conversa no comportamento**, selecione o **fluxo de demonstração**. Isso fará com que o chatbot seja iniciado diretamente pelo bot. Em seguida, clique em **Salvar**.



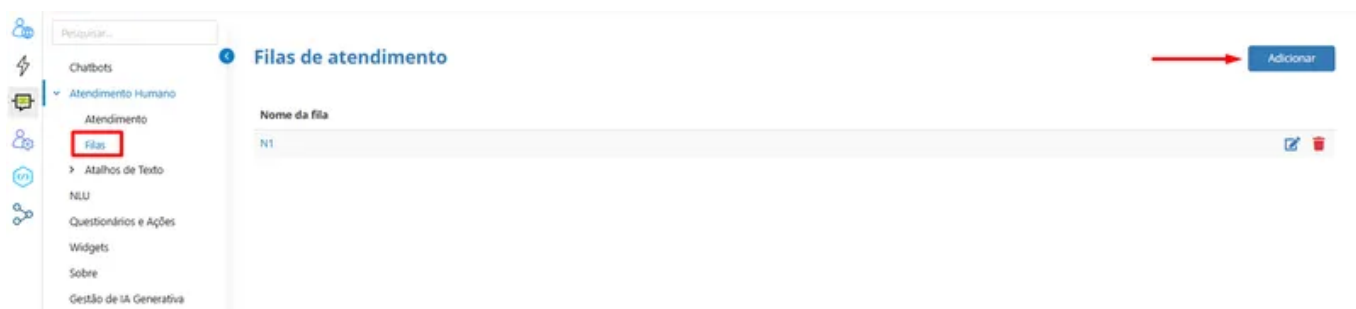
Filas

Esta funcionalidade está disponível apenas para o papel de **Supervisor**.

A opção **"Filas"** permite criar, configurar e gerenciar filas de atendimento, facilitando a distribuição e organização das solicitações recebidas.

Adicionar

Para criar uma fila, clique na opção **"Adicionar"**, conforme a imagem abaixo:

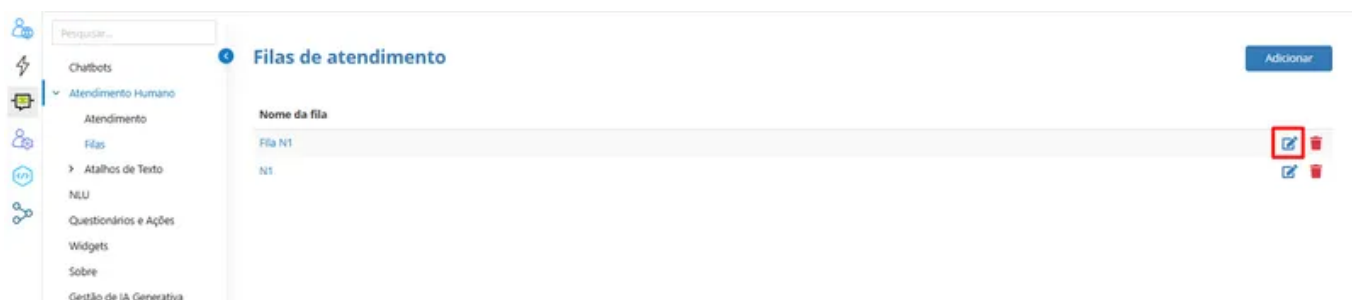


Após acessar a tela, insira o **"Nome da fila"**, selecione os usuários que farão parte daquela fila de atendimento e clique em **"Salvar"** para concluir o processo. Conforme imagem abaixo:



Editar

O sistema permite realizar edições no campo **"Usuários"**. Para isso, clique na opção **"Editar"**, conforme mostrado na imagem abaixo:



Após fazer as alterações necessárias, clique em **"Salvar"** para concluir.

The screenshot shows the 'Adicionar Fila' (Add Queue) form. On the left is a sidebar with a search bar and a menu containing: Chatbots, Atendimento Humano (selected), Máquina de estados, Widgets, Questionários e Ações, Ajuda, and Sobre. The main area has a title 'Adicionar Fila' with a back arrow and a 'Salvar' (Save) button. The form contains three fields: 'Nome da fila' (Queue Name) with the value 'Fila N1', 'Usuários' (Users) which is an empty text area, and 'Mensagem de captura (deixe vazio para desativar a funcionalidade)' (Capture message (leave empty to deactivate the functionality)) which is also an empty text area.

Excluir

Para remover uma fila que não é mais necessária, clique na opção **"Excluir"**, conforme mostrado na imagem abaixo:

The screenshot shows the 'Filas de atendimento' (Queues) list. The sidebar is similar to the previous image but includes 'Gestão de IA Generativa' at the bottom. The main area has a title 'Filas de atendimento' and an 'Adicionar' (Add) button. It displays a table with one queue: 'Fila N1'. To the right of the queue name are two icons: a pencil (edit) and a trash can (delete). The trash can icon is highlighted with a red square, indicating the 'Excluir' (Delete) action.

Em seguida, confirme a ação clicando em **Sim** na janela de confirmação:



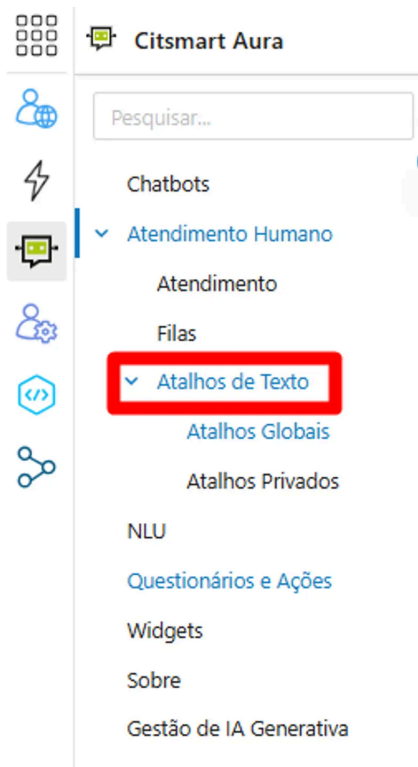
Atalhos de Texto Globais e Privados

A funcionalidade "**Atalhos Globais**" está disponível para os usuários que possuem o papel de "supervisor" e "**Atalhos Privados**" está disponível para o perfil "atendente".

Os atalhos de texto têm como propósito agilizar o atendimento ao usuário, permitindo o uso de respostas e procedimentos previamente configurados, sem a necessidade de buscar informações nas bases de conhecimento. É possível criar atalhos de respostas rápidas que podem ser compartilhados globalmente entre todos os operadores ou configurados como privados, acessíveis apenas pelo operador que os criou.

Adicionar Atalho

Para adicionar um atalho, seja **global ou privado**, escolha a opção correspondente, conforme imagem abaixo:



Em seguida, preencha os campos **Nome do Atalho** e **Conteúdo do Atalho**, e finalize clicando em **Adicionar Atalho**, conforme ilustrado na imagem:

A screenshot of the 'Adicionar Atalho Global' form. At the top is a tab labeled 'Atalhos Globais' with a close icon. Below it is a back arrow icon. The form has two main sections: 'Nome do Atalho:' with a text input field containing 'Bem vindo', and 'Conteúdo do Atalho:' with a larger text area containing 'Olá!' and 'Meu nome é Pedro. Irei fazer o seu atendimento hoje. Como posso ajudá-lo?'. At the bottom are two buttons: 'Adicionar Atalho' (highlighted with a red box and a red arrow) and 'Voltar'.

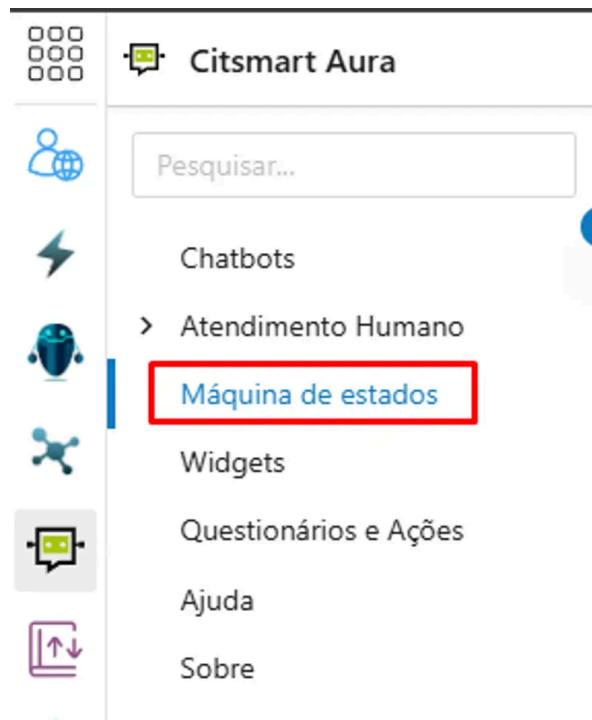
3.2. Máquina de estados

Este manual orienta como configurar, criar e editar uma máquina de estados, explicando a definição de estados, uso de detectores, entidades, mensagens automáticas, textos de dúvida e automações. Assim, você poderá definir como a máquina interage com os usuários e quais respostas devem ser apresentadas com base nas entradas.

Esta funcionalidade está disponível apenas para o papel de **Desenvolvedor**.

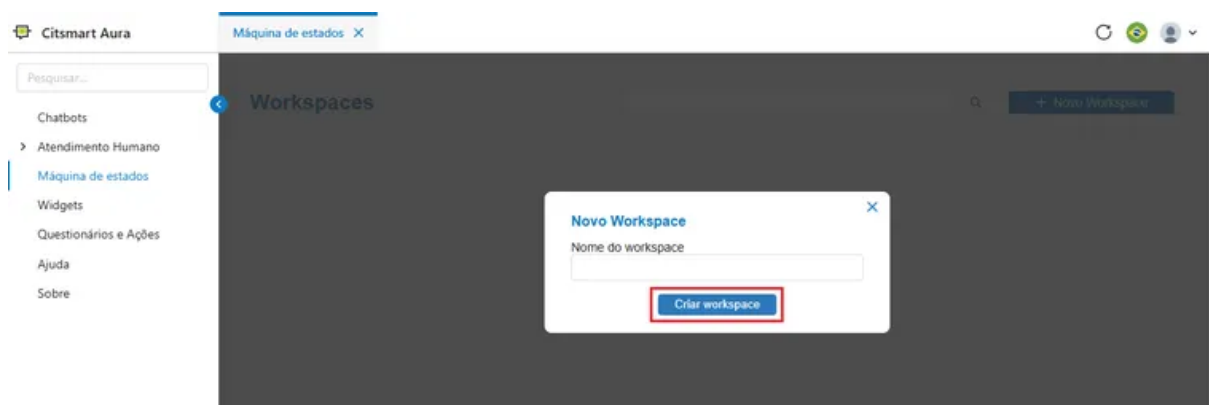
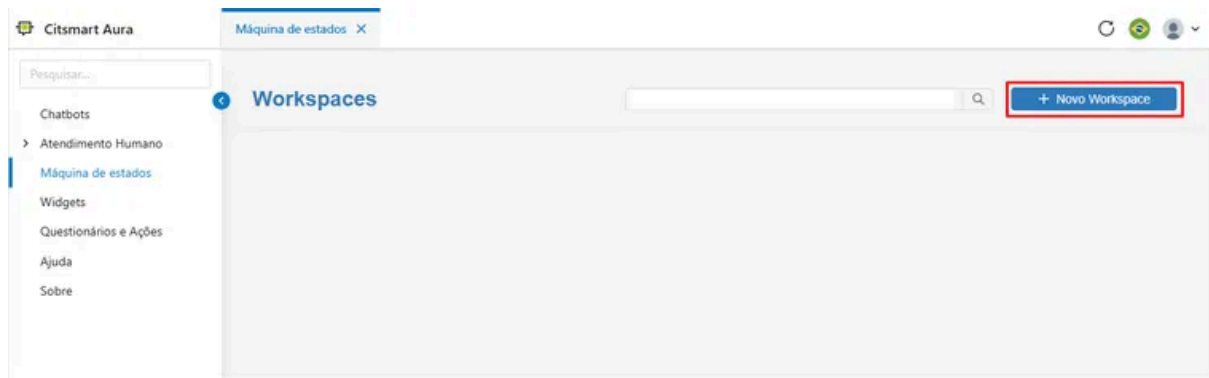
Criar máquina de estados

1. No menu lateral, selecione a opção **Máquina de estados**.

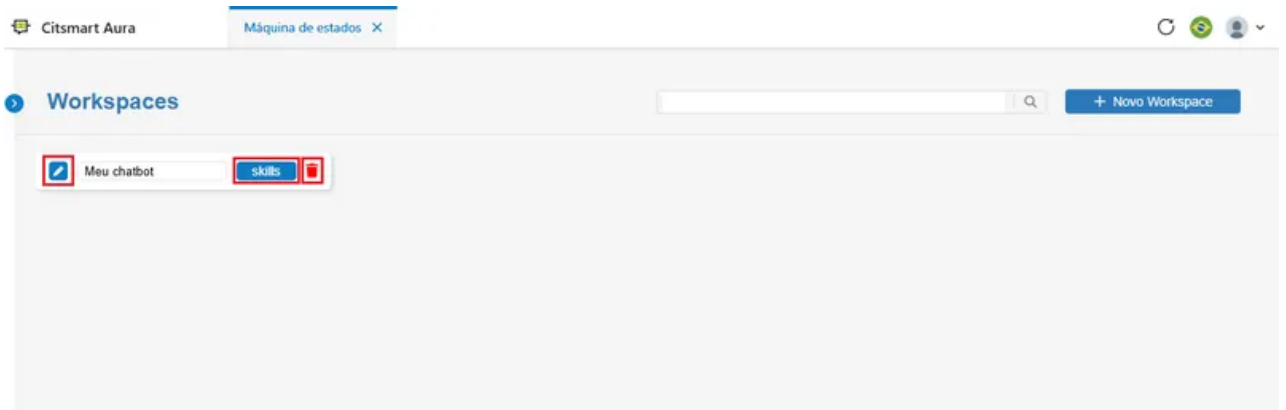


2. Clique em **Novo workspace**.

Uma janela será exibida. Informe o nome desejado e clique em **Criar workspace**.

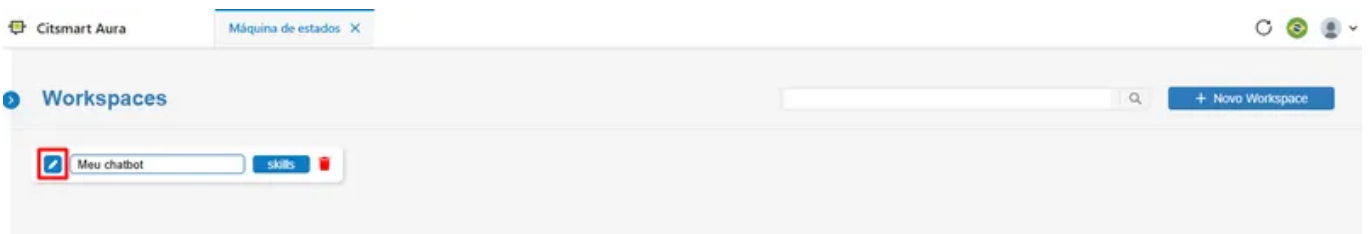


3. Dentro do workspace, é possível editar o nome, criar/editar skills ou excluir a máquina.



Editar nome do workspace

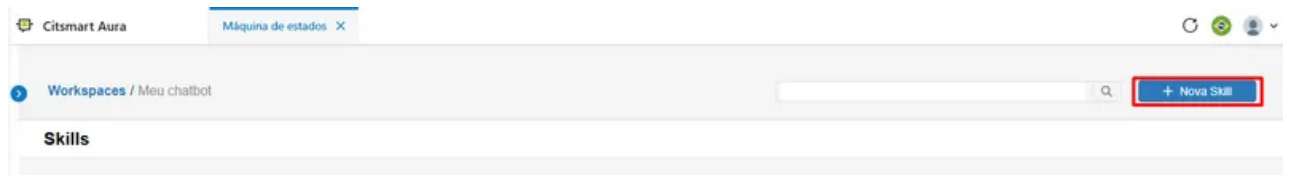
Clique no ícone de lápis, digite o novo nome e clique fora da caixa para salvar automaticamente.



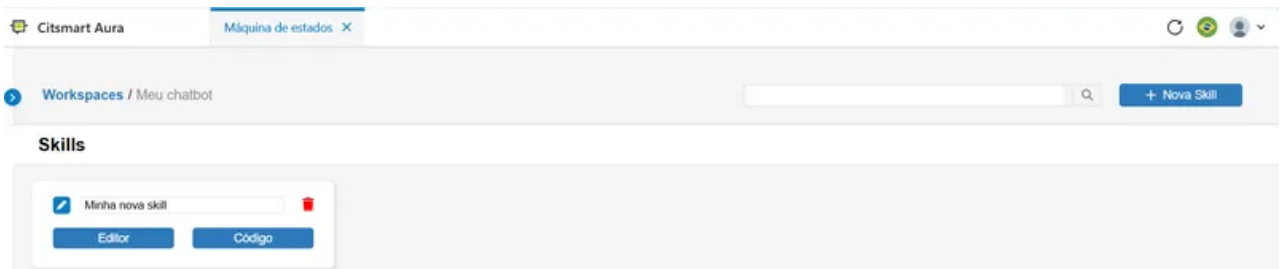
Criar skill

1. Clique em **Nova skill**.

2. Na janela que abrir, informe o nome e clique em **Criar skill**.



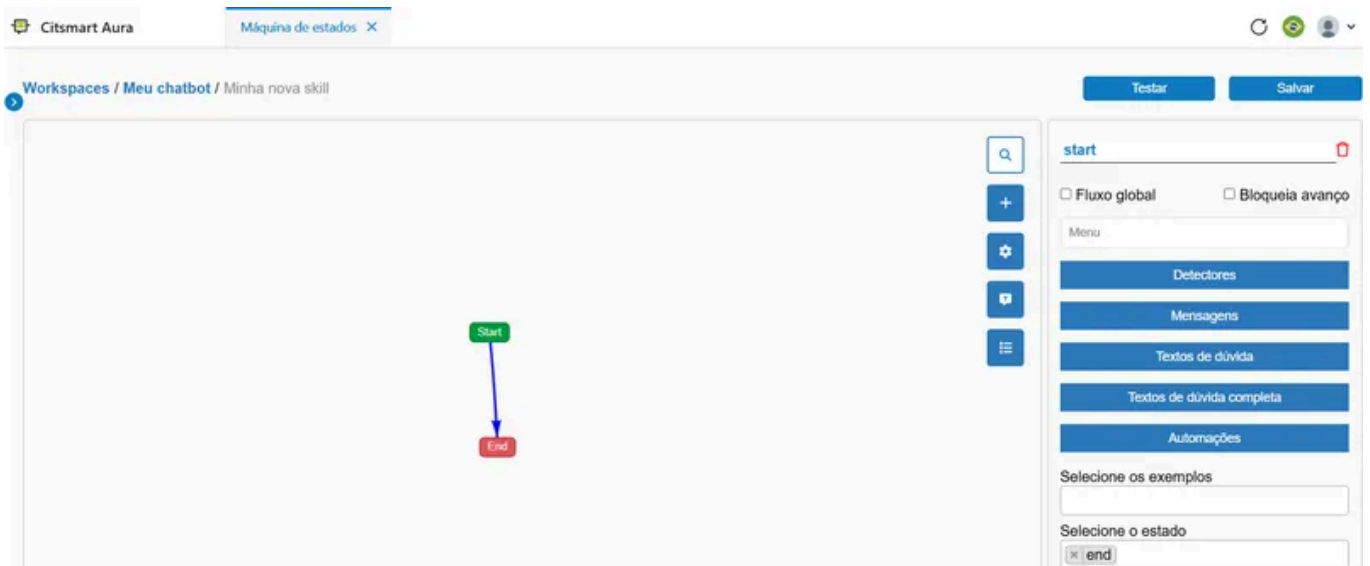
3. A skill pode ser configurada no editor visual ou no editor de código. Ela define o fluxo de conversa, os estados, detectores e mensagens.



Editor visual

Permite criar e organizar o fluxo da conversa conectando visualmente os estados da skill.

Ao abrir, haverá dois estados iniciais: **Start** e **End**.



Recursos do editor

- **Pesquisar estados:** encontre rapidamente qualquer estado criado.

- **Adicionar novo estado:** estados representam os pontos de interação da conversa.



- **Propriedades:** defina configurações avançadas. [Veja mais aqui.](#)



- **Exemplos:** cadastre palavras ou frases que funcionem como detectores.

Citsmart Aura Máquina de estados X

Workspaces / Meu chatbot / Minha nova skill

Testar Salvar

Adicionar Exemplos

Novo exemplo

End Start novo_estado_0

```
graph LR; Start --> novo_estado_0; novo_estado_0 --> End;
```

Citsmart Aura Máquina de estados X

Workspaces / Meu chatbot / Minha nova skill

Testar Salvar

Adicionar Exemplos

Novo exemplo

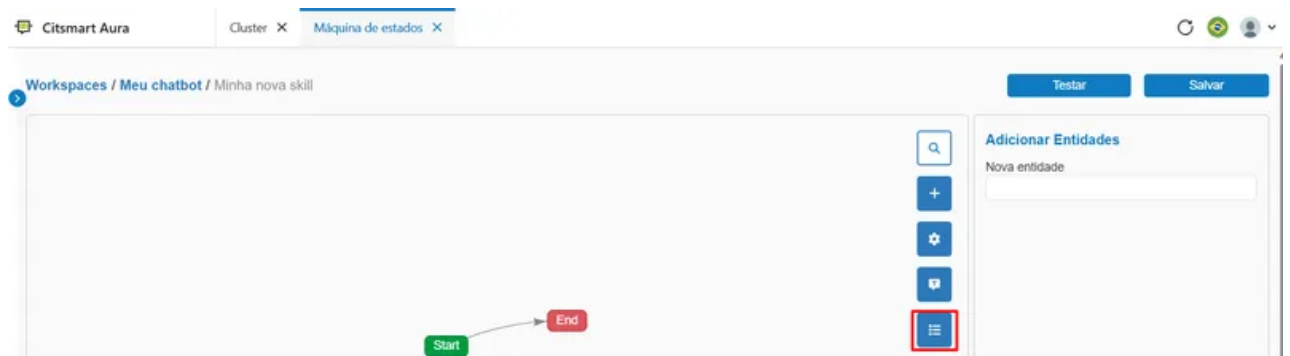
Oi

End Start novo_estado_0

```
graph LR; Start --> novo_estado_0; novo_estado_0 --> End;
```

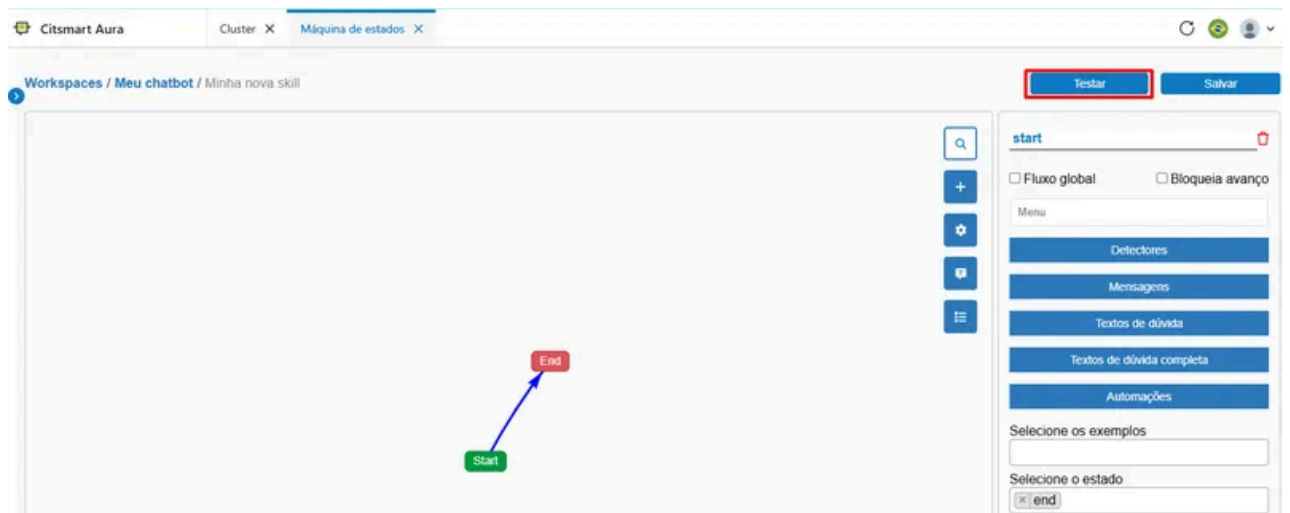


- **Entidades:** grupos de palavras com significado semelhante. Ajudam a identificar termos nas mensagens.

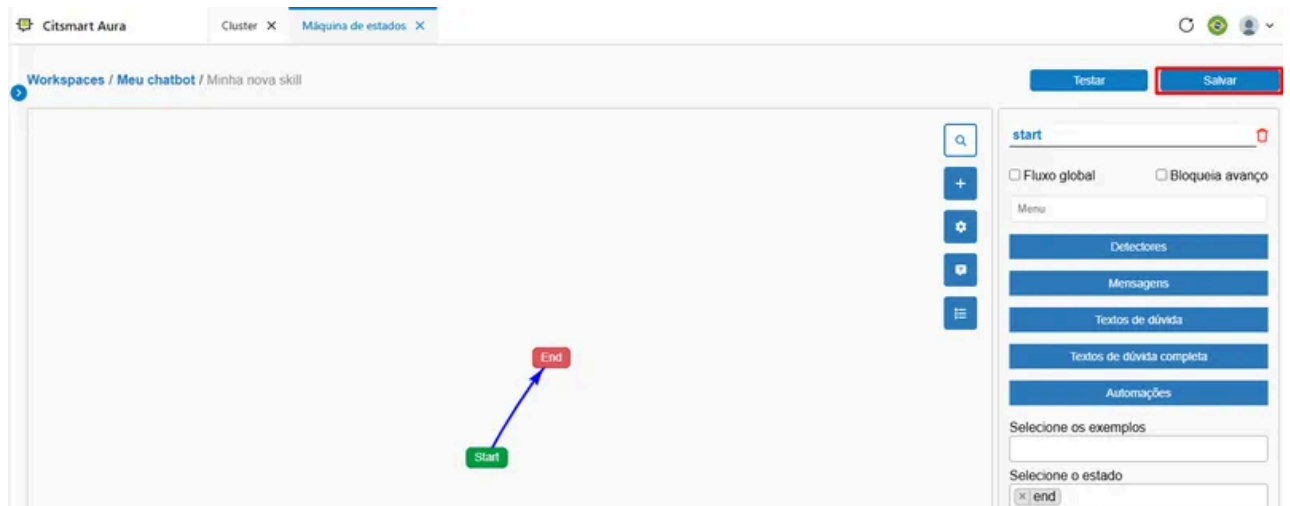




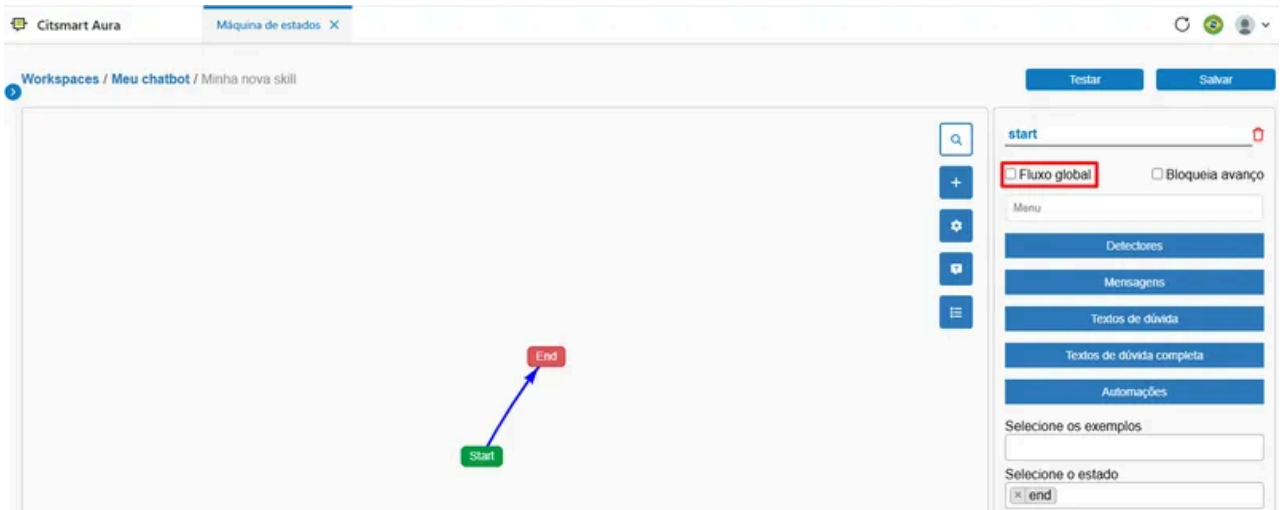
- **Testar:** simule o fluxo para verificar o comportamento configurado.



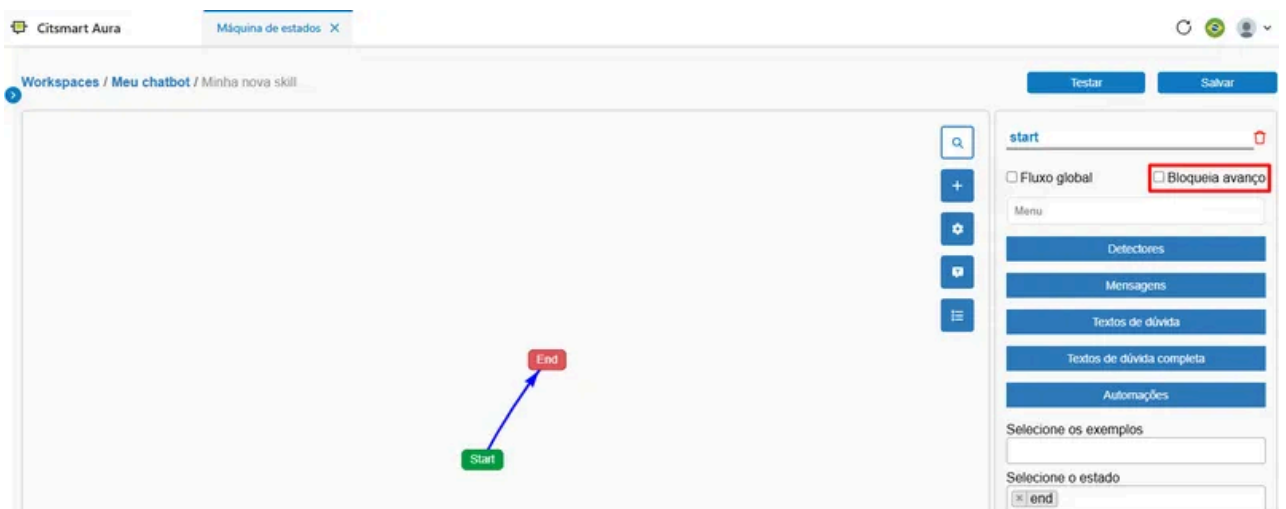
- **Salvar:** salve as alterações realizadas.



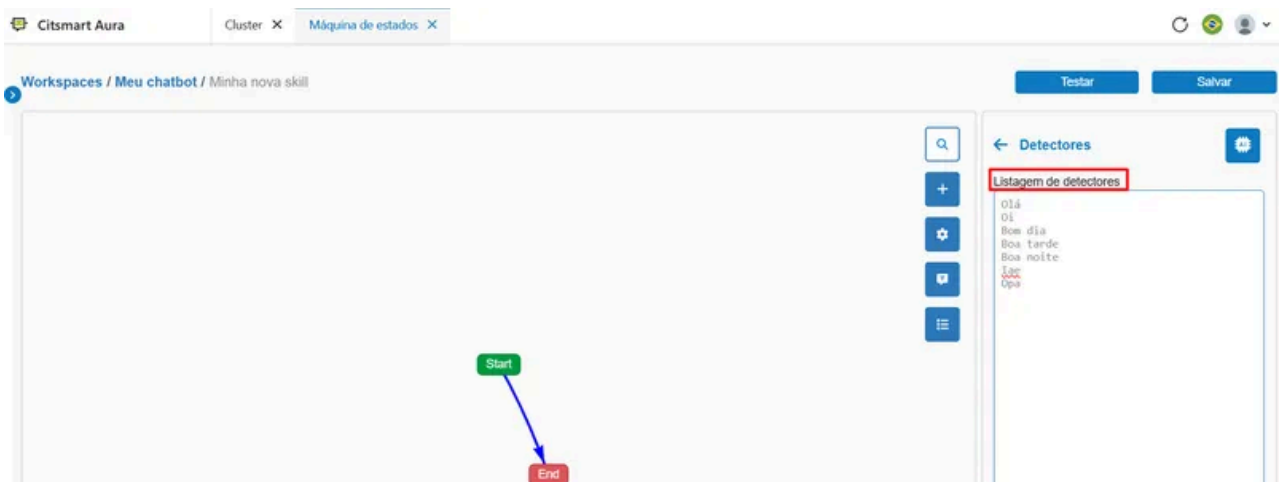
- **Fluxo global:** permite ativar o estado de qualquer ponto da conversa, caso os detectores sejam correspondidos.



- **Bloqueia avanço:** o sistema só avança após a resposta do usuário.

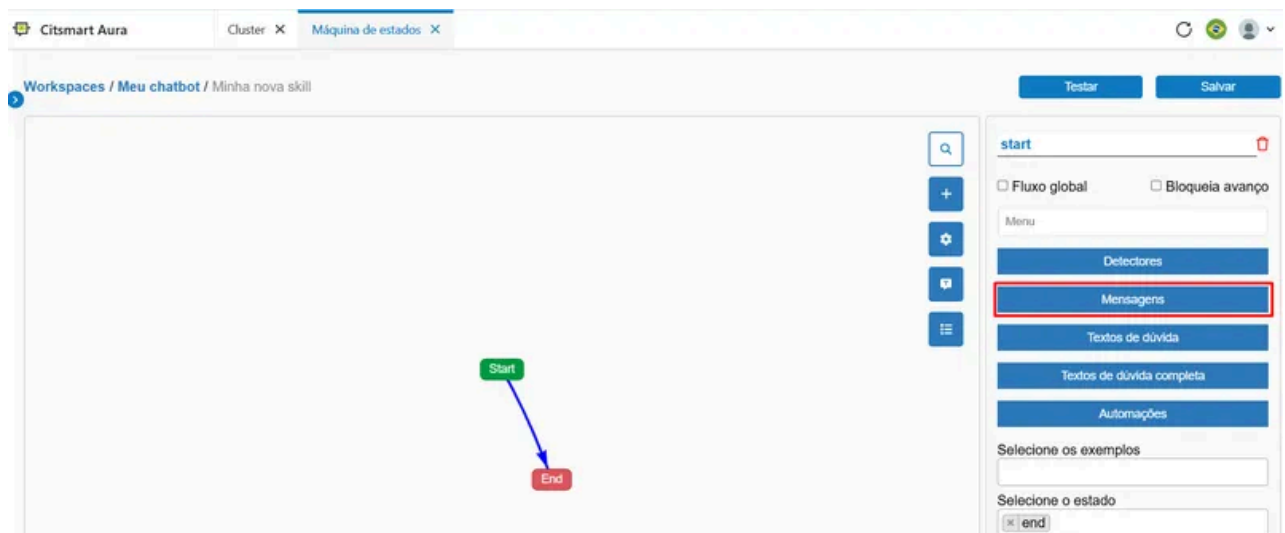
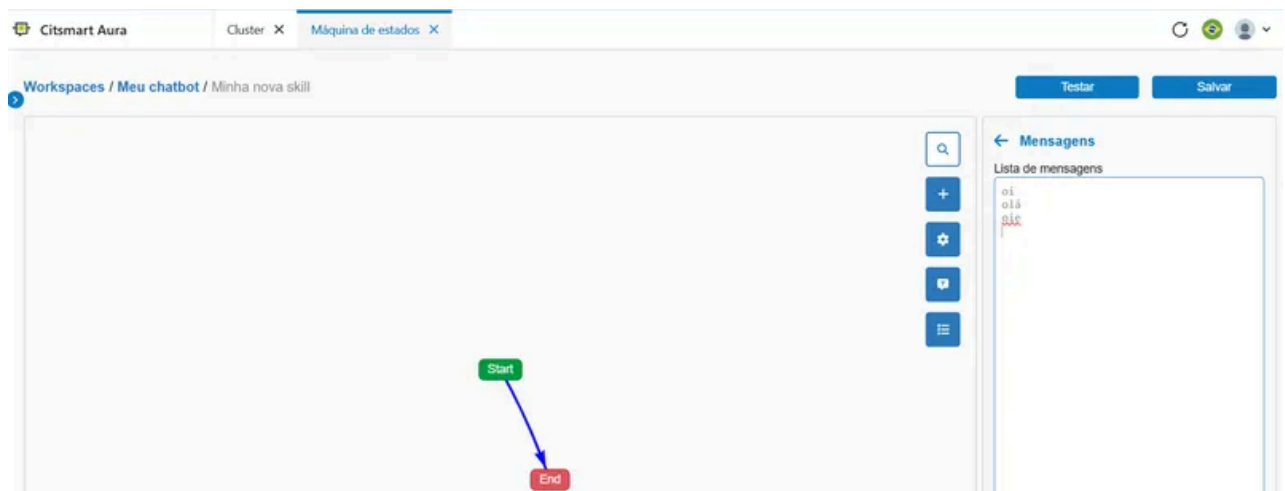


- **Detectores:** palavras ou expressões que ativam um estado.



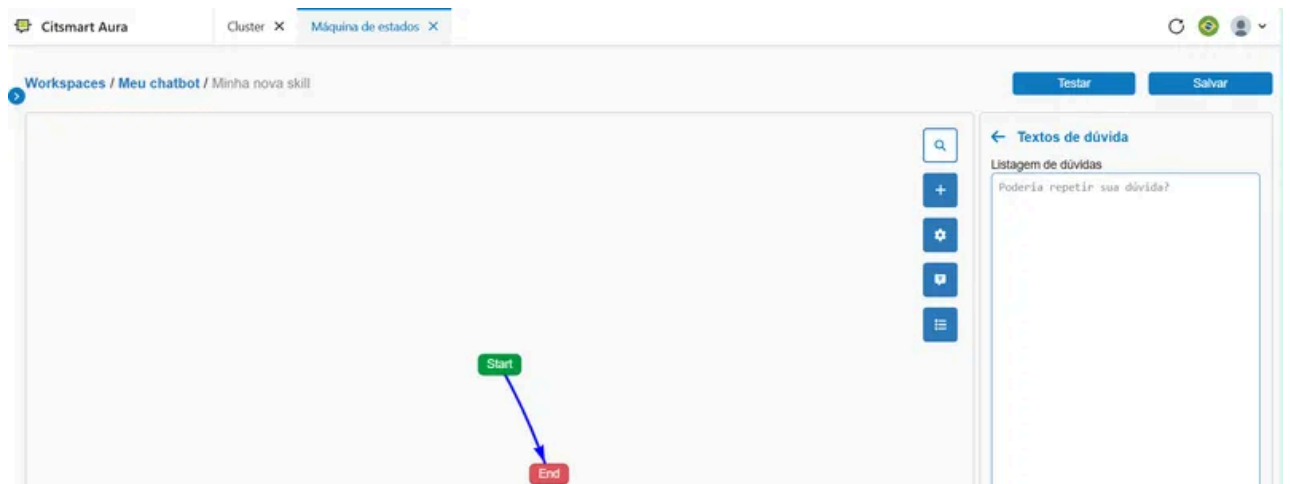
- **Mensagens:** textos que o sistema exibe automaticamente quando o estado é ativado.

Se houver mais de uma, uma delas será escolhida aleatoriamente.

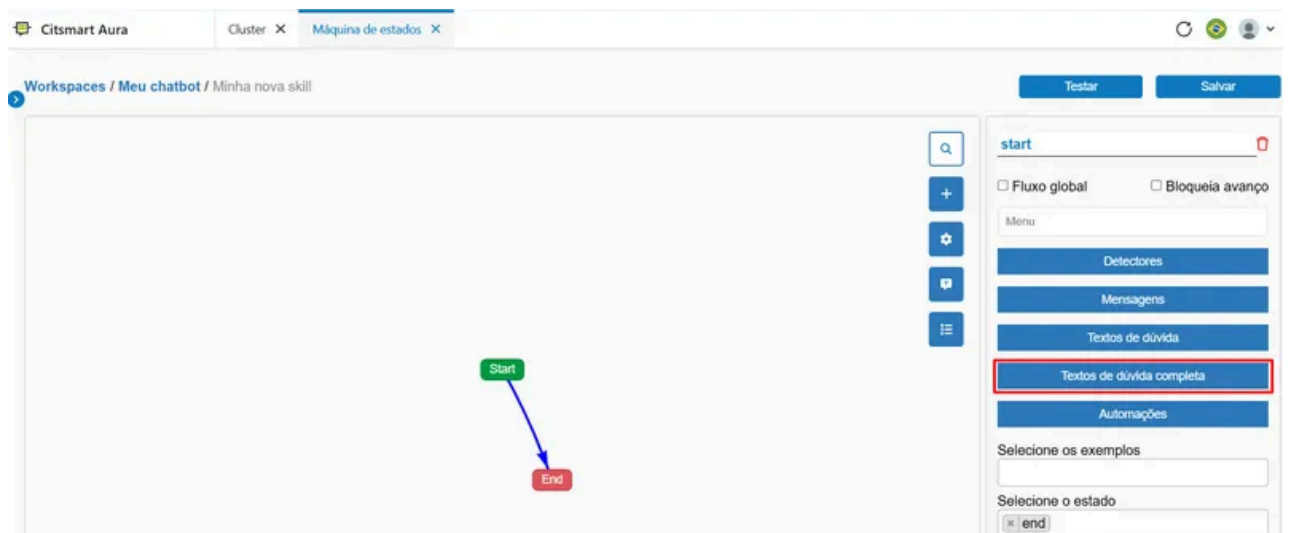


- **Textos de dúvida:** usados quando há mais de um estado possível e a intenção não está clara.

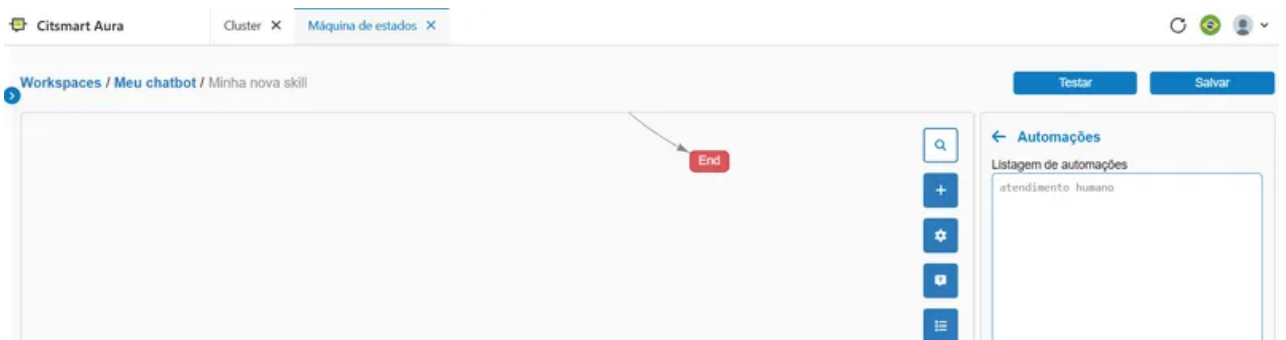
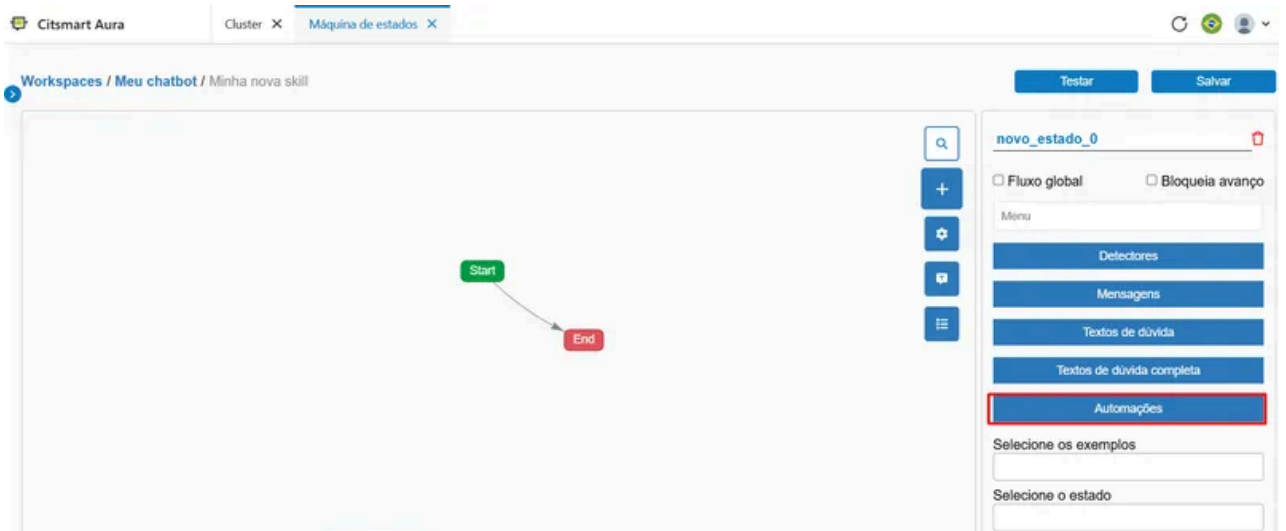
A escolha entre as frases será aleatória, se houver mais de uma.



- **Textos de dúvida completa:** exibidos quando o sistema não consegue reconhecer nenhuma intenção com precisão.



- **Automações:** ações automáticas disparadas ao ativar um estado (mensagens, chamadas de API, etc).



- **Selecione os exemplos:** use exemplos criados para configurar os detectores de outros estados.
- **Selecione os estados:** defina para onde o fluxo pode seguir.

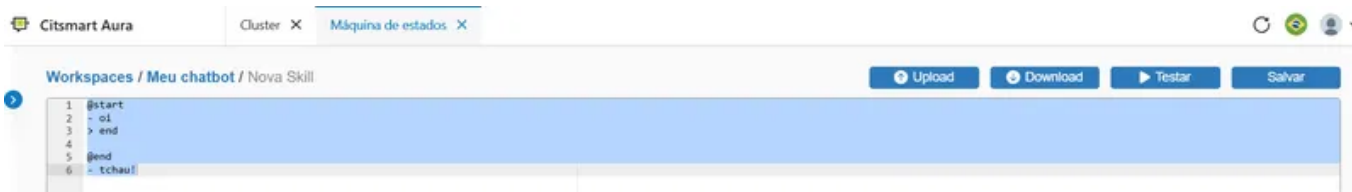
Selecione o estado

start
end
novo_estado_1



Editor de código

Permite criar e editar a skill diretamente por código, usando uma linguagem inspirada em Rust.



Recursos disponíveis:

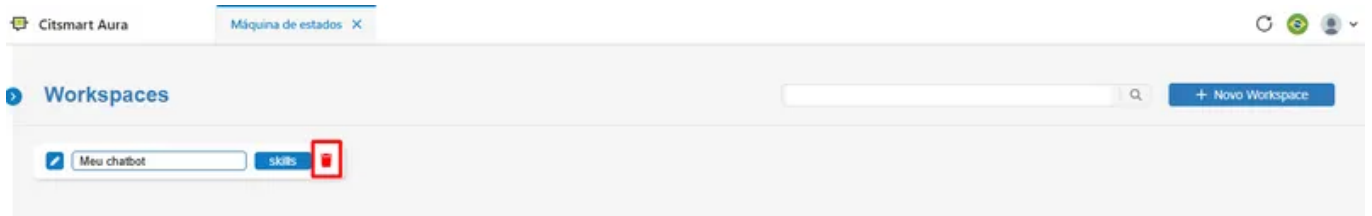
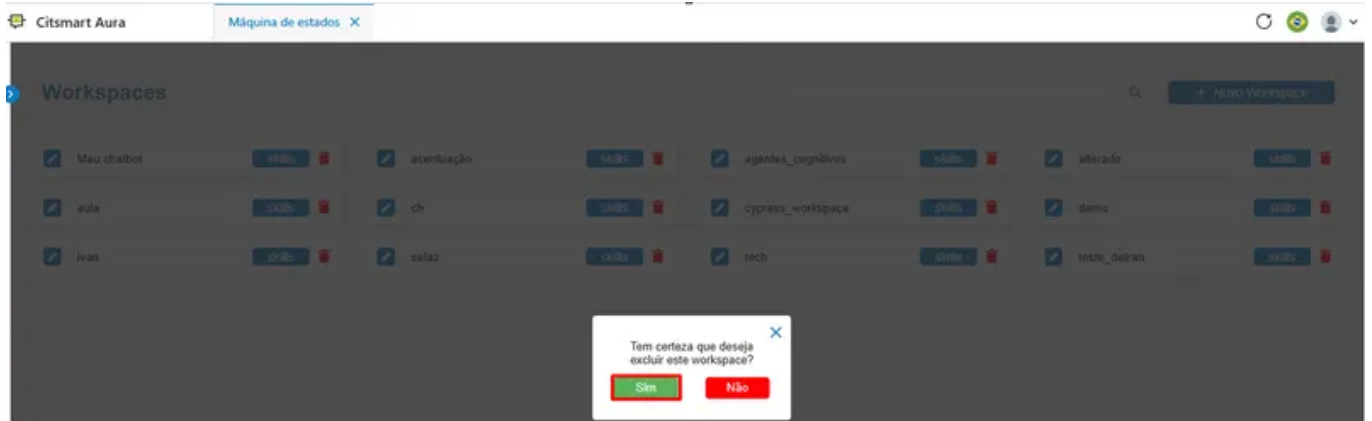
- Upload de uma skill existente
- Download da skill atual
- Testes do fluxo
- Salvamento de alterações

Consulte mais detalhes no [guia completo](#).

Excluir máquina de estado

Clique no ícone de lixeira.

Atenção: a exclusão é permanente.



3.3. Questionários e Ações

Esta funcionalidade está disponível apenas para o papel de **Desenvolvedor**.

A funcionalidade **Questionários e Ações** permite criar conversas e fluxos de interação de forma flexível, seja escrevendo **scripts em Lua** ou usando o **editor visual Blockly**. Escolha a abordagem que mais se adequa ao seu conhecimento e à complexidade do projeto.

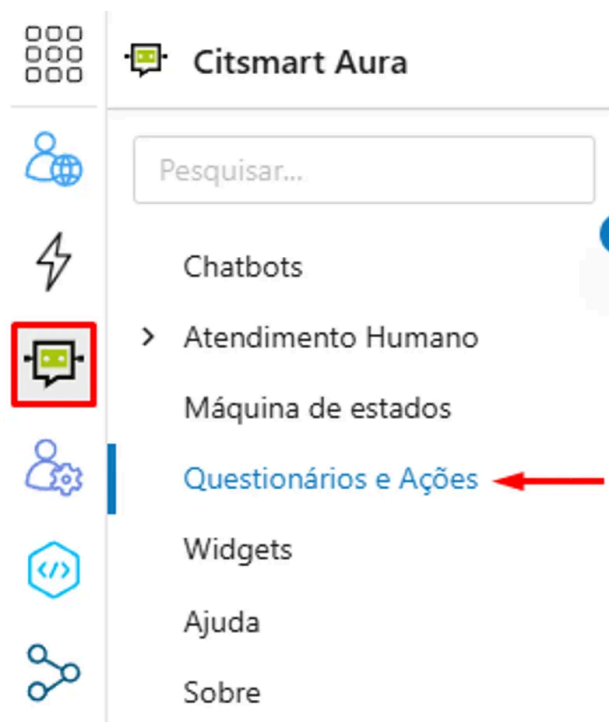
- **Editor de Código:** Ideal para usuários que preferem controle total sobre o código e fluxos personalizados.
- **Editor Visual:** Indicado para quem busca rapidez e visualização intuitiva, montando lógica por meio de blocos.

Workspaces

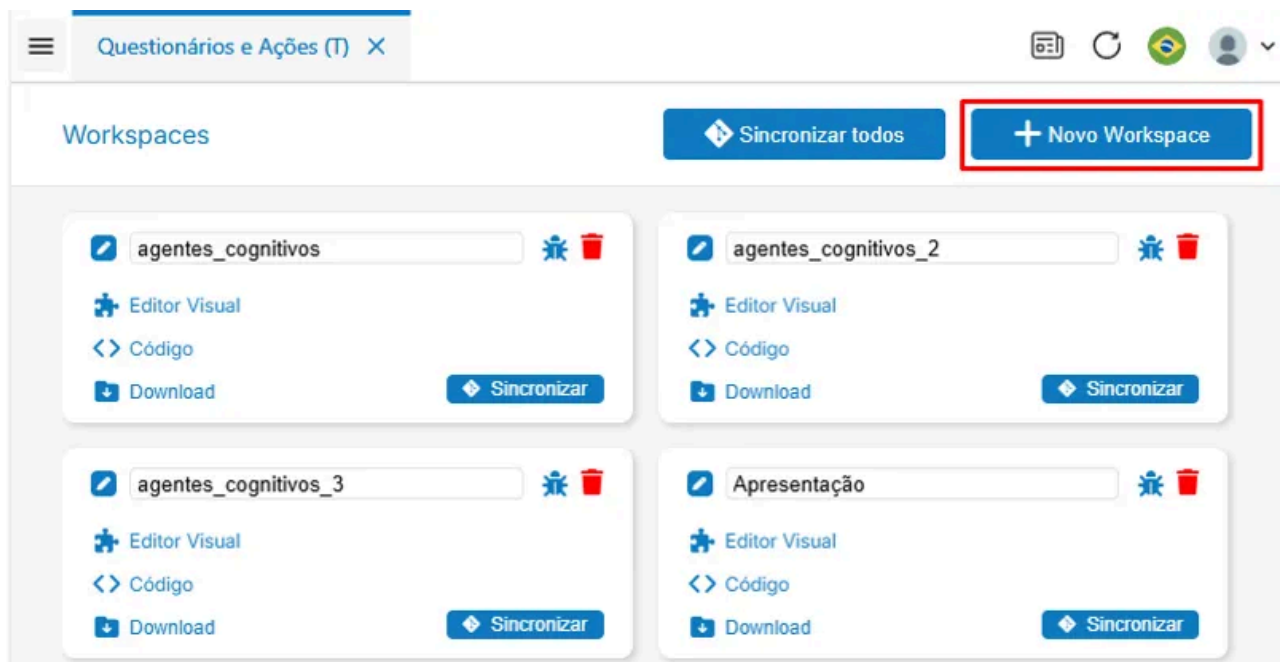
Os *Workspaces* permitem organizar seus projetos de forma prática e eficiente dentro da funcionalidade **Questionários e Ações**. Para criar um novo Workspace, siga as instruções abaixo:

Criar um workspace

1. No menu lateral, acesse **Citsmart Aura > Questionários e Ações**.



2. Na tela principal, clique em **Novo Workspace**.



3. Ao clicar em **"Novo Workspace"**, um pop-up será exibido:

Criar novo workspace

Nome do workspace

Demonstração

☒ Integrar com o GIT

Repositório

Usuário

Senha

Criar novo workspace

Campos do formulário:

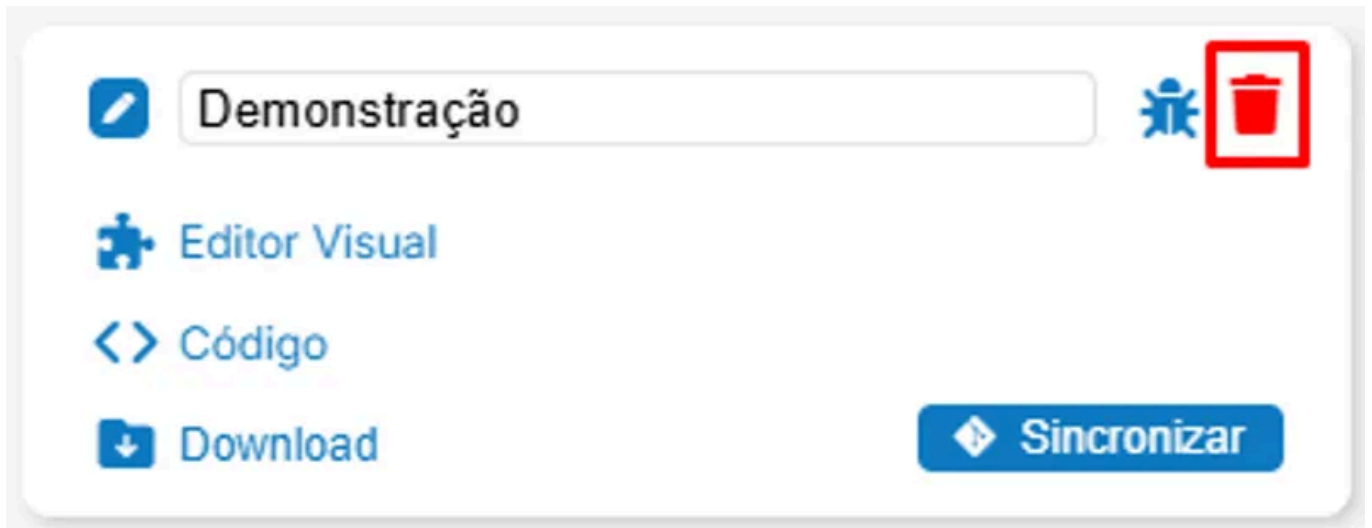
- **Nome do Workspace:** Insira um nome único e descritivo para facilitar a identificação.
- **Integrar com o GIT (opcional):** Se desejar conectar o Workspace a um repositório Git, preencha:
 - **Repositório:** URL completa do repositório.
 - **Usuário:** Seu nome de usuário para acesso ao repositório.
 - **Senha:** Senha ou token de acesso ao repositório.

Quando terminar, basta clicar em **Criar novo Workspace**.

Pronto! Seu ambiente será criado e já poderá ser utilizado.

Excluir um workspace

Para deletar um **Workspace**, clique no ícone de lixeira ao lado daquele que deseja excluir.



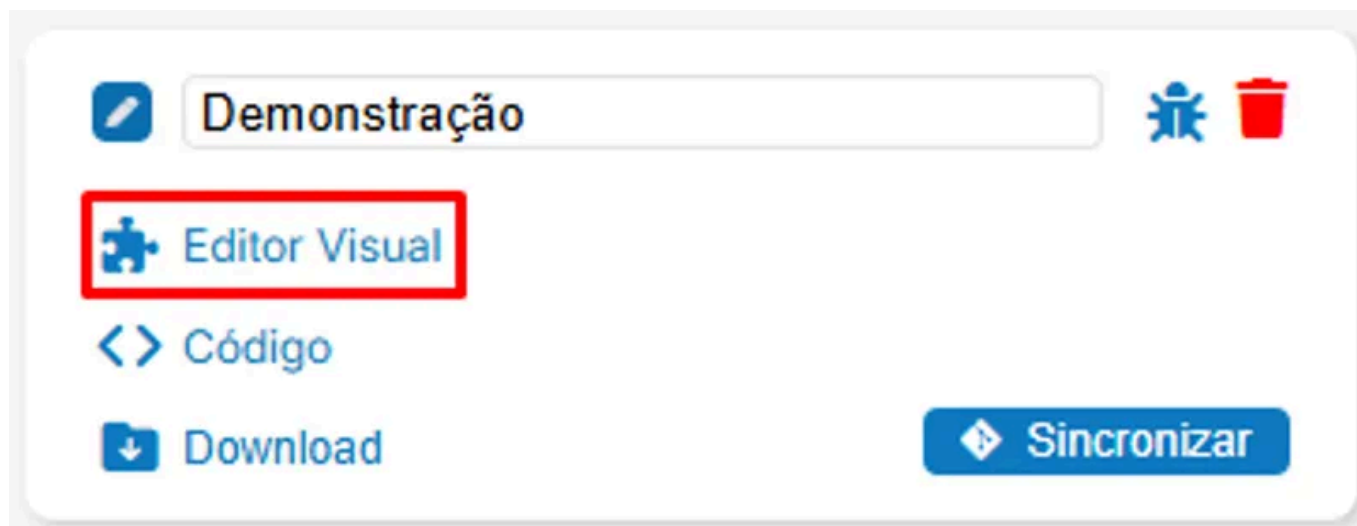
No pop-up de confirmação, clique em **Sim** para confirmar a exclusão.

Editor visual

No CitSMART Aura, o Editor Visual oferece uma forma fácil de montar seus fluxos sem escrever código. Com base na tecnologia Blockly, ele apresenta blocos coloridos que representam ações, condições, repetições, cálculos, textos, listas, variáveis e funções. Ao arrastar e encaixar esses blocos no espaço de trabalho, o sistema converte automaticamente a sequência em código funcional.

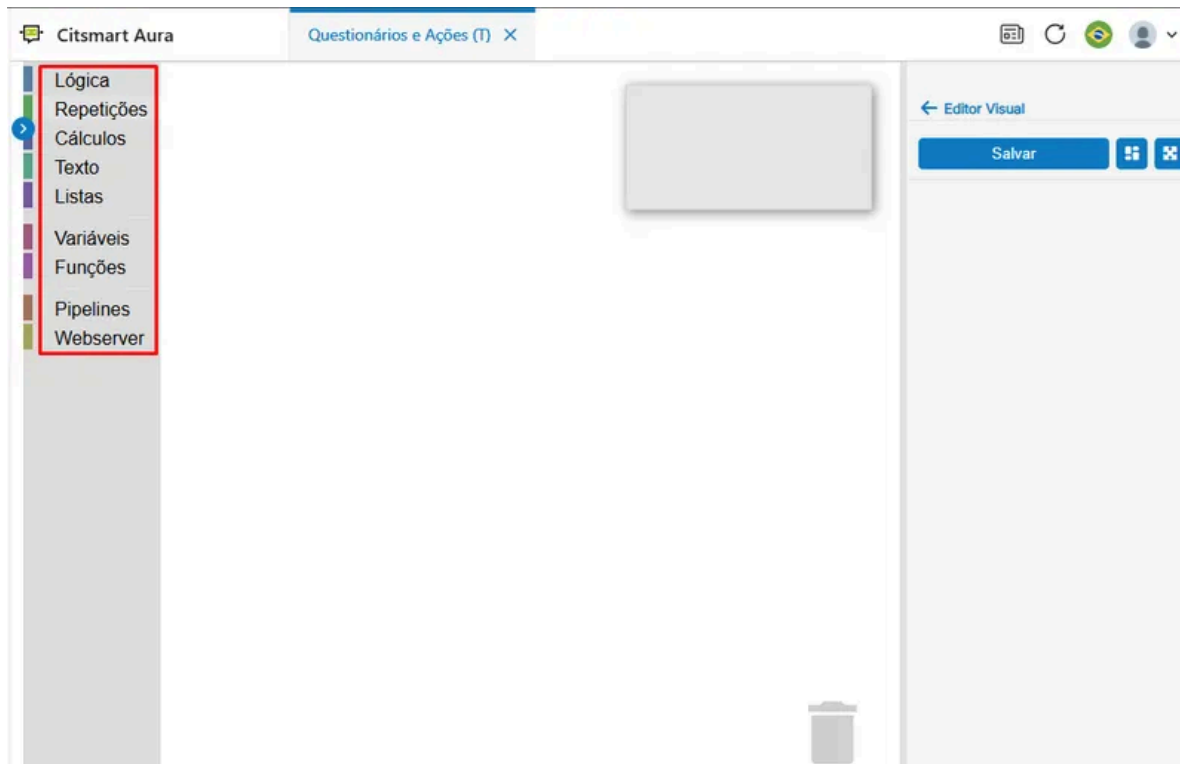
Acessar

Para acessar o editor, escolha o Workspace e clique na opção "**Editor Visual**".

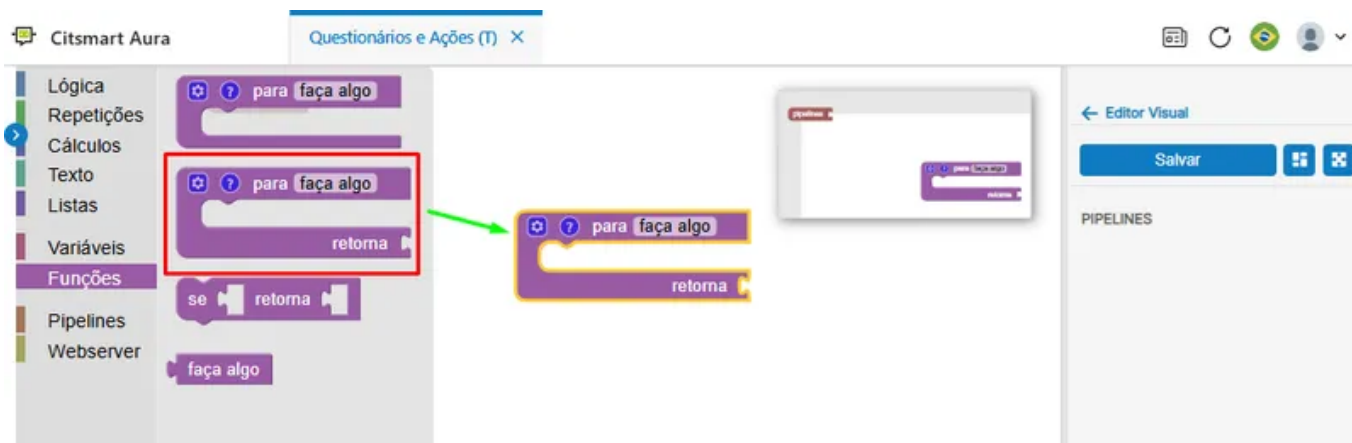


Caixa de ferramentas

A **caixa de ferramentas** organiza todos os blocos por tema, facilitando a busca pelo recurso certo. Cada categoria agrupa blocos com finalidades específicas.



Para usar um bloco, clique no nome da categoria para expandir a lista, em seguida arraste o bloco escolhido para o painel principal.



Tipos de blocos

Nesta seção, você vai conhecer os diferentes tipos de blocos disponíveis e entender para que serve cada um. Esses blocos são como peças de um quebra-cabeça que você pode montar para criar soluções no sistema, mesmo sem saber programar. Cada categoria tem uma função específica para tornar o processo simples e intuitivo.

- **Lógica:** Os blocos de lógica permitem criar condições e tomar decisões no seu projeto. Por exemplo, você pode usar "se... então" para que algo aconteça apenas quando uma condição for verdadeira. Também pode combinar condições ou verificar se algo é verdadeiro ou falso.
- **Repetições:** Esses blocos ajudam a automatizar ações que precisam ser feitas várias vezes. Você pode repetir uma tarefa um número específico de vezes ou enquanto uma condição for verdadeira, como "continue até que o valor chegue a 10".
- **Cálculos:** Os blocos de cálculo são usados para realizar operações matemáticas, como somar, subtrair, multiplicar ou dividir números. Além disso, você pode trabalhar com números aleatórios ou realizar cálculos mais avançados dependendo da necessidade do projeto.
- **Texto:** Aqui você encontra blocos para trabalhar com palavras e frases. Com eles, é possível criar textos personalizados, juntar palavras, verificar se uma palavra aparece em outra ou até contar quantos caracteres tem uma frase.

- **Listas:** Os blocos de lista ajudam a organizar informações em sequência, como uma lista de itens de compras ou números ordenados. Com eles, você pode adicionar, remover ou encontrar elementos específicos em uma lista, facilitando a manipulação de dados.
- **Variáveis:** As variáveis funcionam como caixinhas que guardam informações que podem mudar ao longo do tempo. Por exemplo, você pode criar uma variável chamada "pontos" para armazenar a pontuação de um jogo e atualizar esse valor conforme o jogo avança.
- **Funções:** Os blocos de função permitem criar conjuntos de ações que podem ser reutilizados em diferentes partes do projeto. Pense neles como uma receita: você monta uma vez e pode chamá-la quantas vezes quiser, tornando tudo mais organizado e eficiente.

Pipelines

As pipelines são uma forma poderosa de organizar fluxos de interação com o usuário, especialmente em sistemas que precisam de entradas dinâmicas e respostas interativas. No questionários e ações, criar uma pipeline é simples e intuitivo, pois tudo é feito através de blocos que representam cada etapa do processo.



Cada bloco desempenha uma função, veja abaixo:

- **Bloco "pipelines - criar lista com"**

- **Função:** Cria uma lista para armazenar pipelines. As pipelines são fluxos de interação que conectam várias etapas ou ações.
-

- **Bloco "criar pipeline com as iterações":**

- **Função:** Define um pipeline específica com um nome personalizado e as iterações que ele conterá.
 - **Parâmetros:**
 - **Nome da pipeline:** Identifica a pipeline.
 - **Iterações:** Define quais etapas (ou interações) fazem parte do fluxo.
 - **Uso:** Este é o ponto central de criação da pipeline, conectando diferentes etapas.
-

- **Bloco "nome da interação":**

- **Função:** Define as iterações dentro da pipeline.
 - **Parâmetros:**
 - **Nome da interação:** Define o nome da interação da pipeline.
 - **Mensagem de saída:** Texto ou instrução exibida ao usuário durante a interação.
 - **Mostrar interação:** Define se a interação será visível (verdadeiro) ou oculta (falso).
 - **Aceitar entrada:** Configura se a interação aceita dados do usuário.
 - **Mensagem de rejeição:** Texto exibido caso a entrada do usuário seja inválida.
 - **Uso:** Configura o comportamento de uma etapa na pipeline.
-

- **Bloco "chave - valor" -**

- **Função:** Define pares de chave-valor. Muito útil para armazenar dados ou configurar informações dentro de uma interação ou função.
- **Parâmetros:**
 - **Chave:** Nome que identifica o dado.

- **Valor:** Conteúdo associado à chave.
 - **Uso:** Esse bloco é usado para manipular e organizar dados na pipeline.
-

- **Bloco "referenciar função"**

- **Função:** Faz referência a uma função previamente definida no sistema.
 - **Parâmetros:**
 - **Nome da função:** A função que será chamada quando o bloco for executado.
 - **Uso:** Este bloco é utilizado para reaproveitar comportamentos ou lógicas já implementadas.
-

- **Bloco "função com argumentos - criar lista com" -**

- **Função:** Define uma função com argumentos que podem ser passados quando ela for chamada.
 - **Parâmetros:**
 - **Nome da função:** Identifica a função.
 - **Argumentos:** Lista de variáveis ou parâmetros que a função utiliza.
 - **Uso:** Permite criar funções reutilizáveis que podem ser configuradas dinamicamente com base nos argumentos fornecidos.
-

- **Bloco "função com argumentos (repetição)"**

- **Função:** Similar ao bloco anterior, define uma função com argumentos, mas aqui pode ser usado em outro contexto ou repetido.
-

- **Bloco "definir variável local "i" para"**

- **Função:** Cria uma variável local (visível apenas dentro do contexto onde foi definida) e atribui um valor a ela.
- **Parâmetros:**
 - **Nome da variável:** Identifica a variável (neste caso, i).
 - **Valor:** O conteúdo inicial da variável.

- **Uso:** Este bloco é útil para armazenar temporariamente valores ou realizar cálculos locais.
-

Criando uma pipeline simples

Neste exemplo, vamos criar uma pipeline básica chamada **pipeline_exemplo** que solicita três informações do usuário:

- O **Nome** do usuário,
- A **Idade** do usuário,
- E qual o **Problema** ele deseja resolver.

Além de aprender a criar a pipeline, você verá como configurar mensagens personalizadas e validar as entradas do usuário. Com este passo a passo detalhado e ilustrado, você conseguirá criar suas próprias pipelines para atender a diversas necessidades.

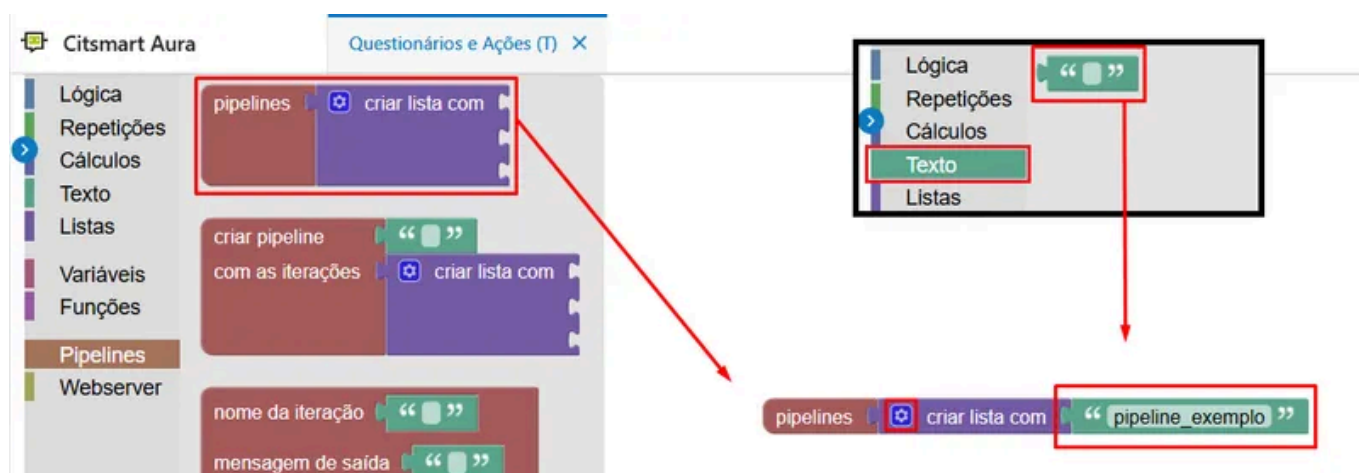
Exemplo prático

Siga as instruções abaixo para criar e configurar sua pipeline.

Passo 1: Definir nome da pipeline

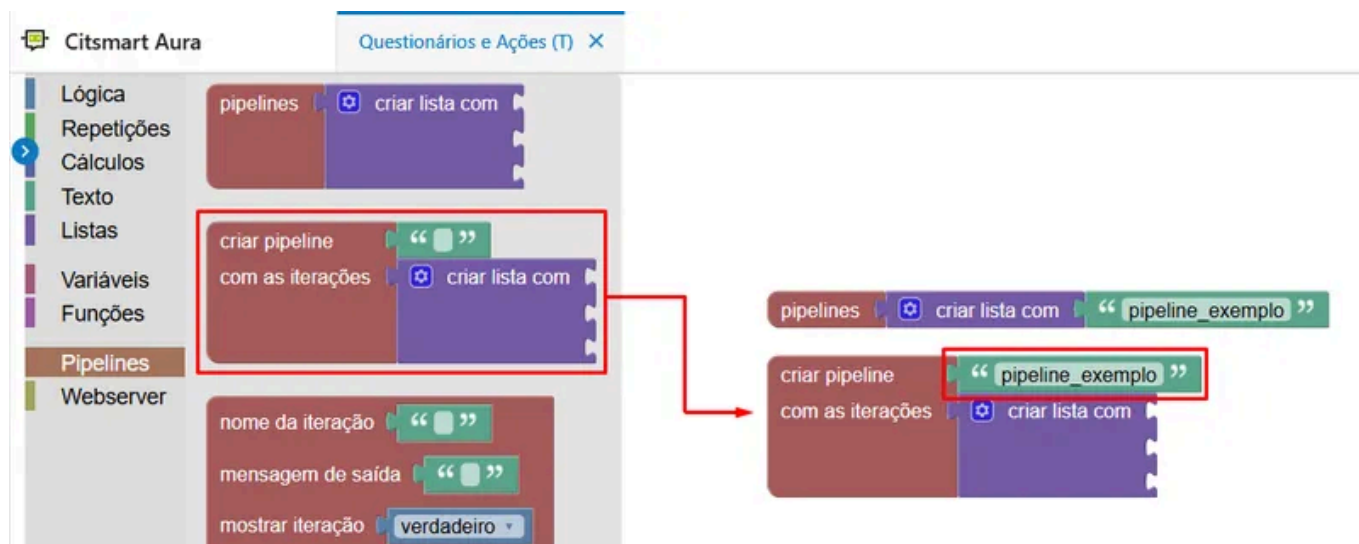
1. No menu **Pipelines**, localize o bloco "**pipelines**" > "**criar lista com**"
2. Arraste o bloco para o espaço de trabalho.
3. No menu **Texto**, arraste o primeiro bloco "**()**".
4. Conecte ao espaço disponível no bloco "**criar lista com**".
5. Clique no campo de texto do bloco e insira o nome da pipeline: `pipeline_exemplo`.

Observação: Para remover os campos de conexões que não serão utilizados, basta clicar na engrenagem e arrastar o campo "**Item**" para fora.



Passo 2: Iniciar a Pipeline

1. No menu **Pipelines**, localize o bloco "**criar pipeline com as iterações**" > "**criar lista com**".
2. Arraste o bloco para o espaço de trabalho.
3. No campo de texto do bloco, insira o nome da pipeline: `pipeline_exemplo`.

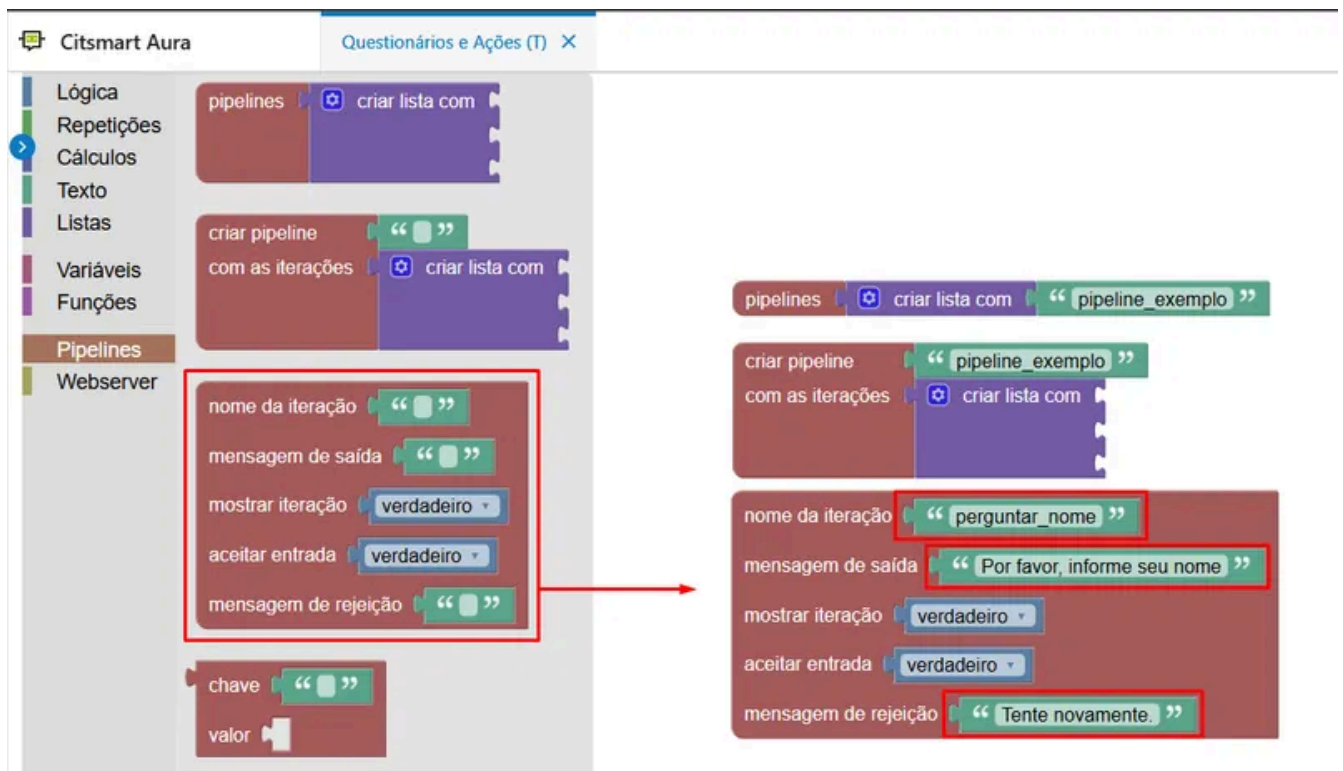


Passo 3: Configurar as Iterações

Agora, você configurará as etapas (iterações) da pipeline. Este exemplo inclui três etapas: **Perguntar o nome**, **Perguntar a idade** e **Perguntar o problema**.

Etapa 1: Perguntar o Nome

1. No menu **Pipelines**, localize o bloco "**nome da interação**"
2. Arraste-o para o espaço de trabalho.
3. Preencha os campos do bloco:
 - **Nome da interação:** Digite `perguntar_nome` .
 - **Mensagem de saída:** Escreva "**Por favor, informe seu nome**"
 - **Mostrar interação:** Selecione `verdadeiro` .
 - **Aceitar entrada:** Selecione `verdadeiro` .
 - **Mensagem de rejeição:** Escreva "**Tente novamente.**"



Etapa 2: Perguntar a Idade

1. Arraste outro bloco "**nome da interação**" para o espaço de trabalho.
2. Preencha os campos do bloco:
 - **Nome da interação:** Digite `perguntar_idade`.
 - **Mensagem de saída:** Escreva "**Qual é a sua idade?**"
 - **Mostrar interação:** Selecione `verdadeiro`.
 - **Aceitar entrada:** Selecione `verdadeiro`.
 - **Mensagem de rejeição:** Escreva "**Idade inválida. Tente novamente.**"

Observação: Utilize como referência a imagem da **Etapa 1**.

Etapa 3: Perguntar o Problema

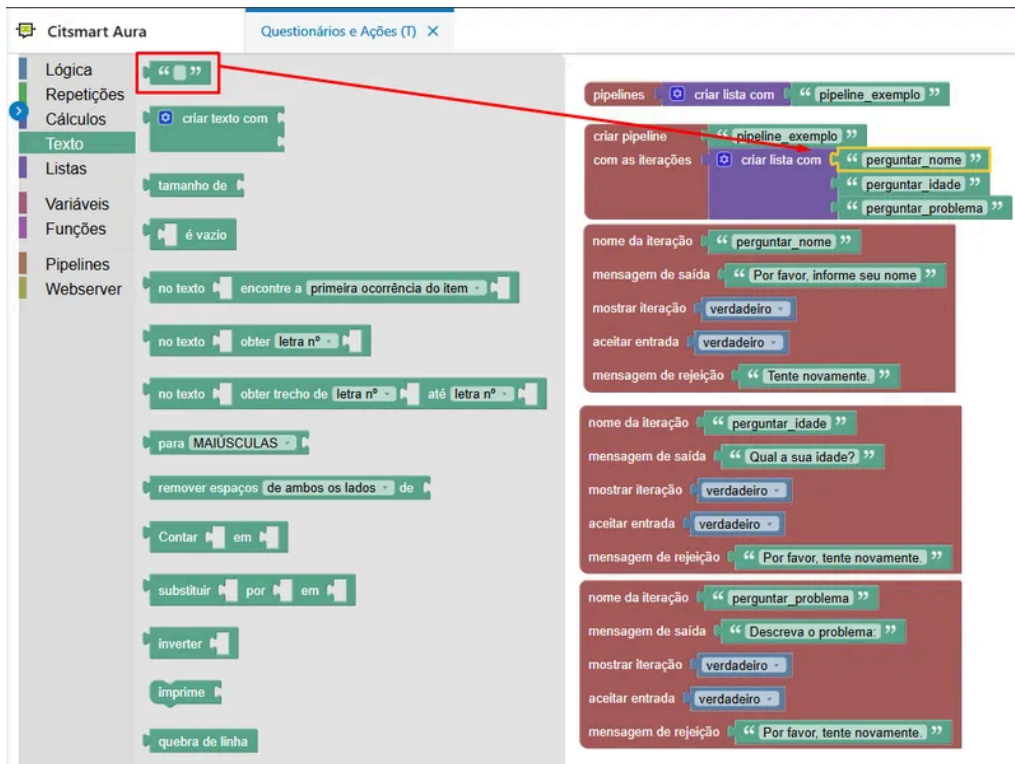
1. Adicione mais um bloco "**nome da interação**" ao espaço de trabalho.
2. Preencha os campos do bloco:
 - **Nome da interação:** Digite `perguntar_problema`.
 - **Mensagem de saída:** Escreva "**Qual é o problema que deseja resolver?**"
 - **Mostrar interação:** Selecione `verdadeiro`.
 - **Aceitar entrada:** Selecione `verdadeiro`.

- **Mensagem de rejeição:** Escreva "Entrada inválida. Por favor, tente novamente."

Observação: Utilize como referência a imagem da **Etapa 1**.

Passo 4: Conectar as Iterações à Pipeline

1. Certifique-se de que as três iterações foram criadas:
 - perguntar_nome
 - perguntar_idade
 - perguntar_problema
2. No menu **Texto**, arraste três blocos de texto " " (**campo de texto vazio**) para o espaço de trabalho.
3. Em cada bloco de texto, escreva os nomes das iterações criadas: `perguntar_nome`, `perguntar_idade`, e `perguntar_problema`.
4. Conecte os blocos de texto ao campo "criar lista com" do bloco "criar pipeline com as iterações".
5. Verifique se todas as interações estão conectadas corretamente à lista.



Checagem final

Após seguir todos os passos, revise o fluxo completo no espaço de trabalho. Certifique-se de que:

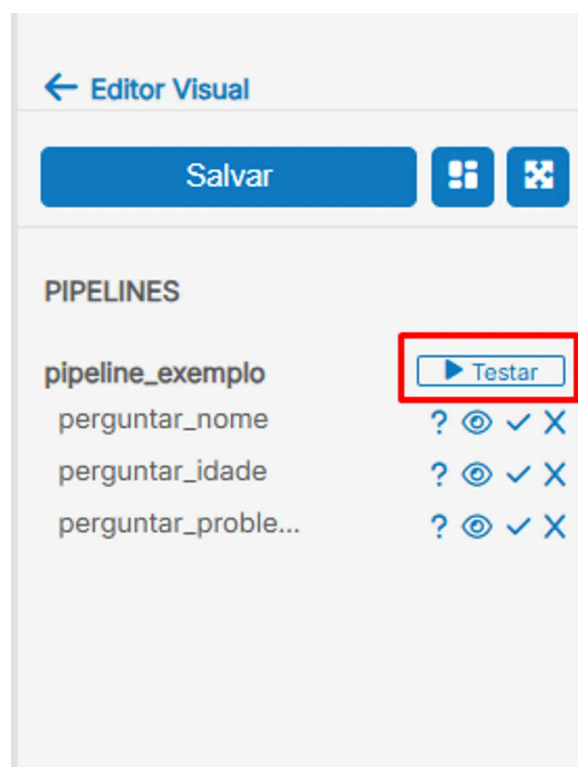
- O nome da pipeline está correto: `pipeline_exemplo`.
- As três iterações foram criadas e configuradas corretamente.
- As iterações estão conectadas ao bloco principal.

Se tudo estiver correto, sua pipeline estará funcional e pronta para uso. Ela seguirá o seguinte fluxo:

1. Pergunta o nome do usuário.
2. Pergunta a idade do usuário.
3. Pergunta qual problema o usuário deseja resolver.

Testar a pipeline

Para validar a lógica da sua pipeline, utilize o botão **Testar**, disponível no canto superior direito do editor visual.



Ao clicar na opção **Testar**, uma janela será exibida para que você possa interagir com o fluxo criado e verificar se ele está funcionando conforme o esperado.

pipeline_exemplo X

Contexto 0 registros

oi

Por favor, informe seu nome

Paulo

Qual a sua idade?

21

Descreva o problema:

A impressora parou de funcionar

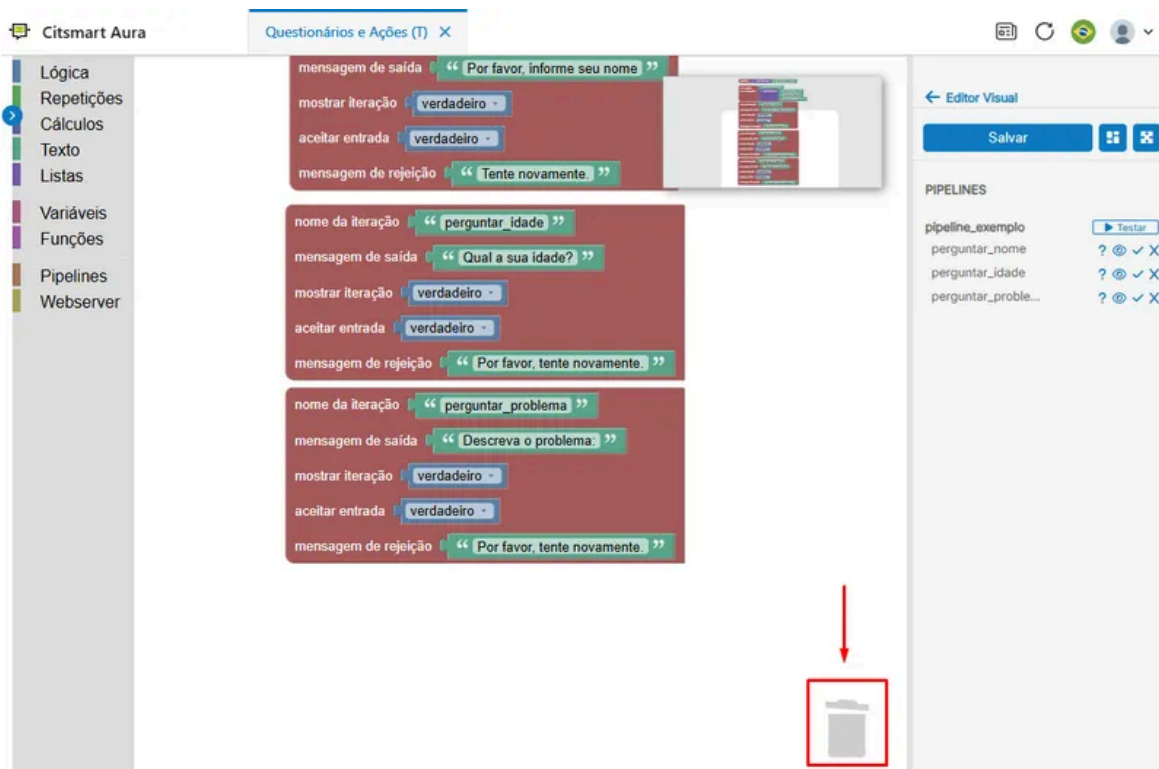
FINISHED

Webserver

Com os blocos de webserver, você pode configurar interações com servidores na web, como enviar ou receber informações. Eles são úteis para criar sistemas que se comunicam com outros serviços online ou para criar funcionalidades como APIs dentro do seu projeto.

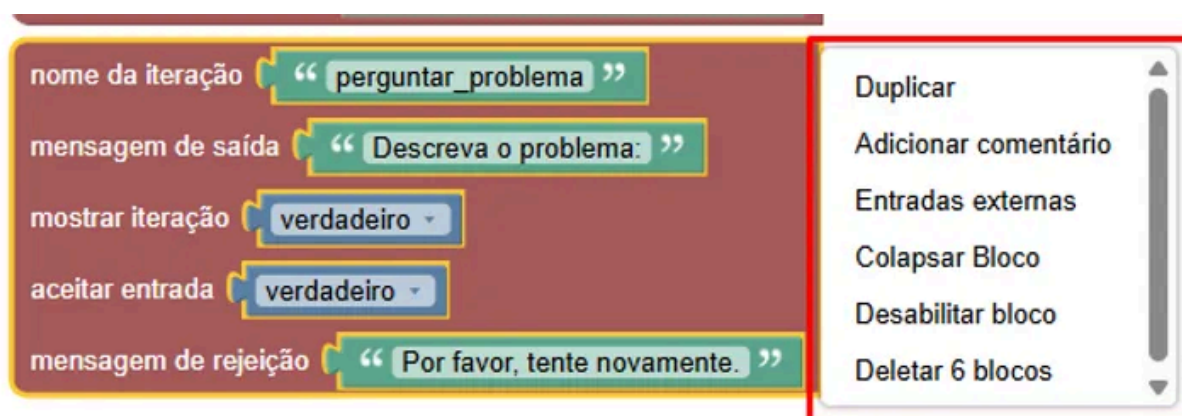
Lixeira

A lixeira permite que você remova blocos arrastando-os até ela. Se precisar, basta clicar na lixeira para abrir uma lista com os blocos descartados, permitindo que você os recupere quando necessário.



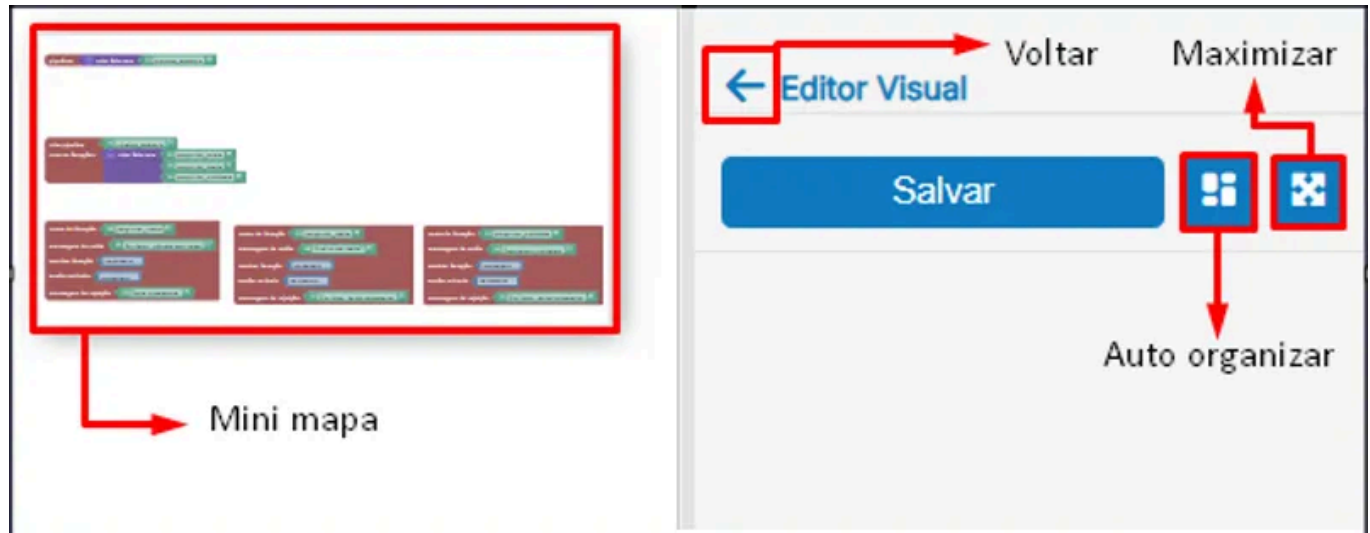
Menu de contexto

O menu de contexto é exibido quando você clica com o botão direito do mouse ou pressiona por alguns instantes certos elementos no espaço de trabalho, como o fundo ou os blocos. Ele apresenta uma lista de ações disponíveis para o elemento selecionado. Veja abaixo:



Outras ferramentas do Editor Visual

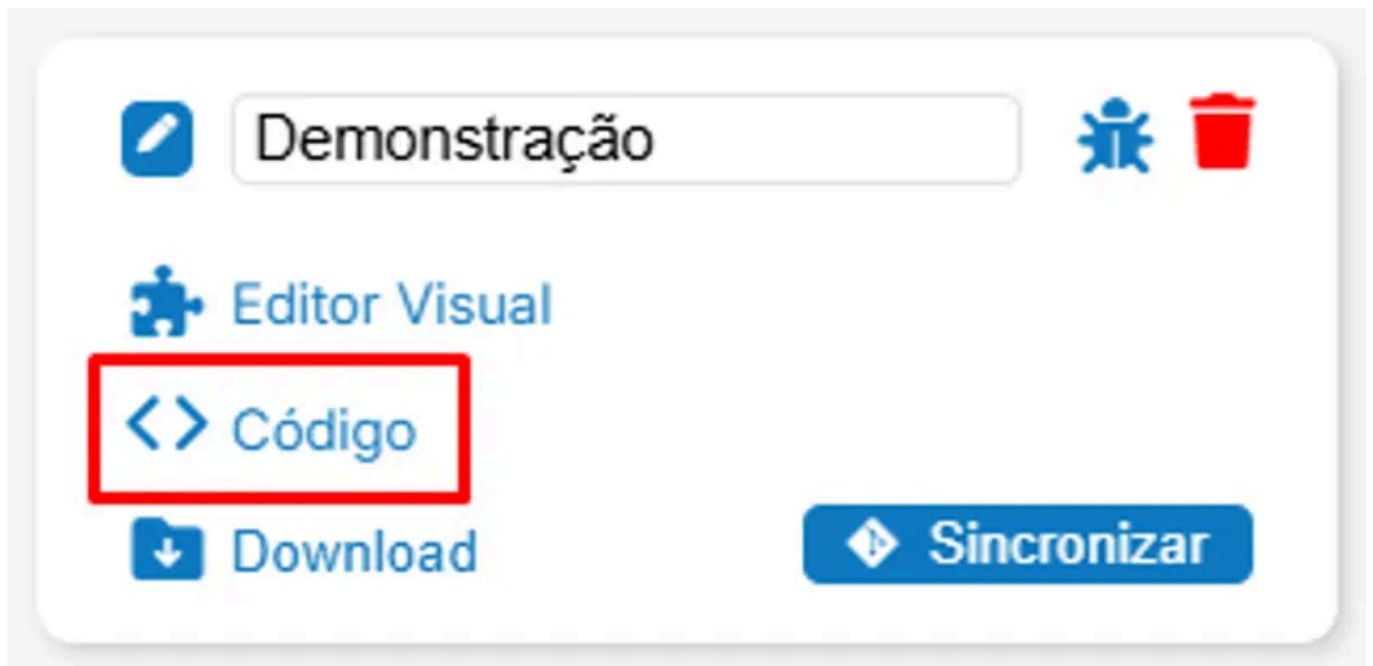
O editor visual oferece outras ferramentas que tornam a navegação e organização mais prática e eficiente. Confira os recursos apresentados na imagem abaixo:



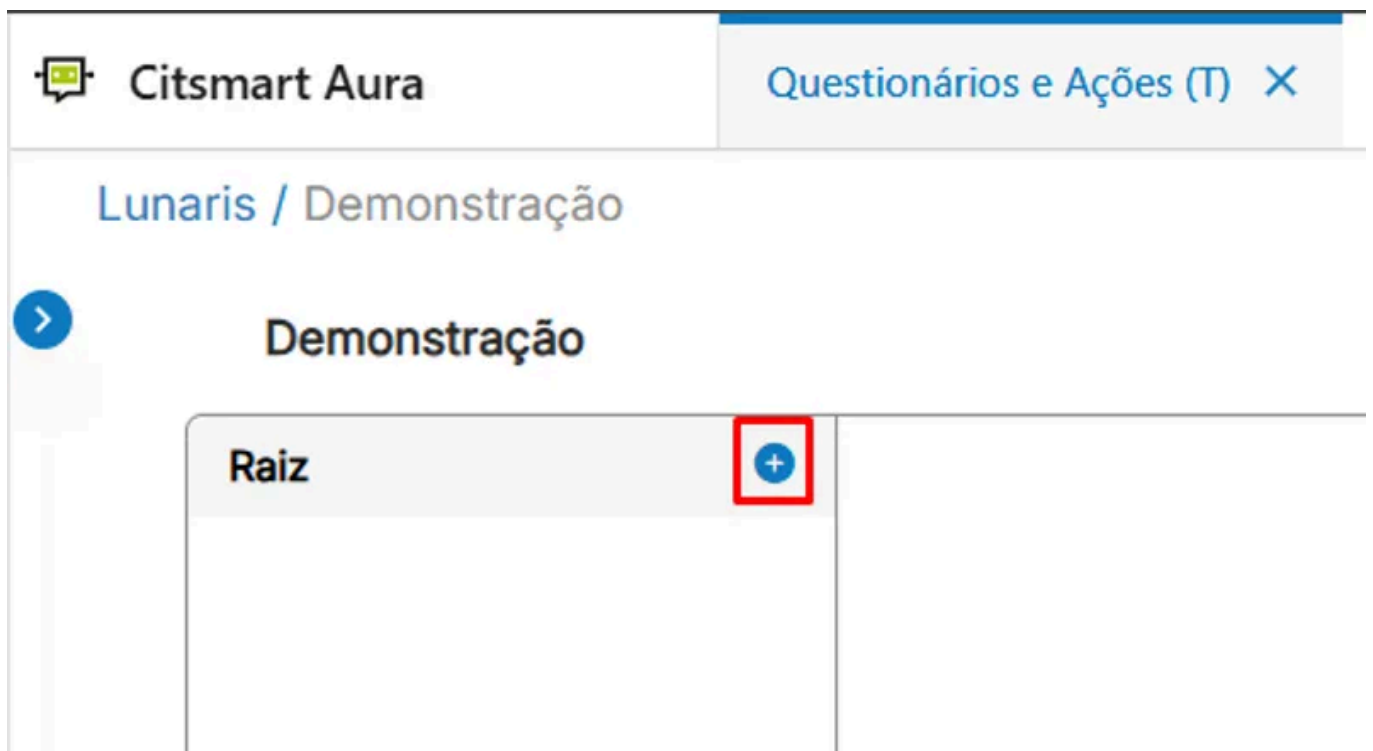
- **Mini mapa:** Exibe uma visão reduzida do espaço de trabalho, permitindo localizar rapidamente áreas específicas quando há muitos blocos distribuídos. Ideal para facilitar a navegação em projetos maiores.
- **Maximizar:** Expande os blocos para ocupar todo o espaço de trabalho, proporcionando uma visão melhor dos blocos já criados. Útil para evitar distrações e aproveitar melhor o ambiente de trabalho.
- **Auto organizar:** Realinha automaticamente os blocos no espaço de trabalho, organizando-os de forma clara e eficiente. Essa função é ideal para manter os fluxos bem estruturados.
- **Voltar:** Localizada ao lado do título "Editor Visual", a seta permite retornar facilmente ao menu principal do questionários e ações, proporcionando acesso rápido às demais funcionalidades.
- **Salvar:** Grava as alterações feitas no projeto atual, garantindo que nenhum progresso seja perdido.

Código

Para acessar o código, selecione o card desejado na lista do Workspace e clique na opção "**Código**", conforme indicado na imagem abaixo:



Para criar um "**Novo item**", clique no ícone com o símbolo de "+", localizado ao lado da palavra "**Raiz**".



Digite o nome para o item e escolha se ele será um "**Arquivo**" ou "**Pasta**". Após preencher as informações, clique em "**Salvar**" para finalizar a criação.



Criar novo item

Nome

Demonstracao

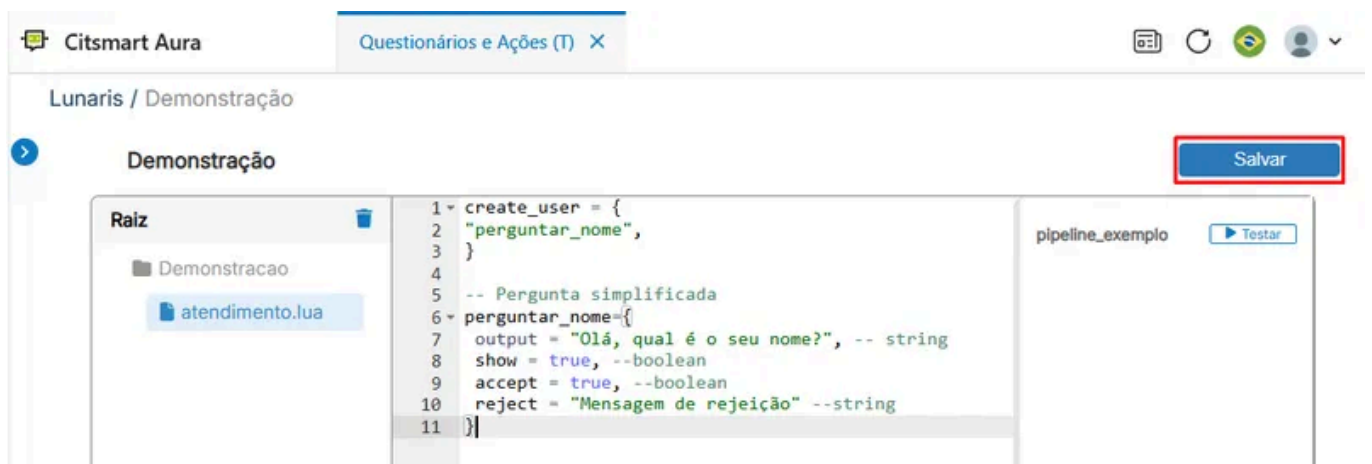
☒ Arquivo ☐ Pasta

Salvar

Editor de código

Após criar a pasta e o arquivo, a tela de edição de código será exibida, oferecendo um ambiente semelhante ao de um editor de texto avançado.

Essa tela permite que você escreva o código necessário para configurar sua pipeline. Na imagem abaixo, estão destacados a área principal para edição do código, o botão para salvar as alterações e o arquivo criado dentro da pasta, onde o código será inserido.

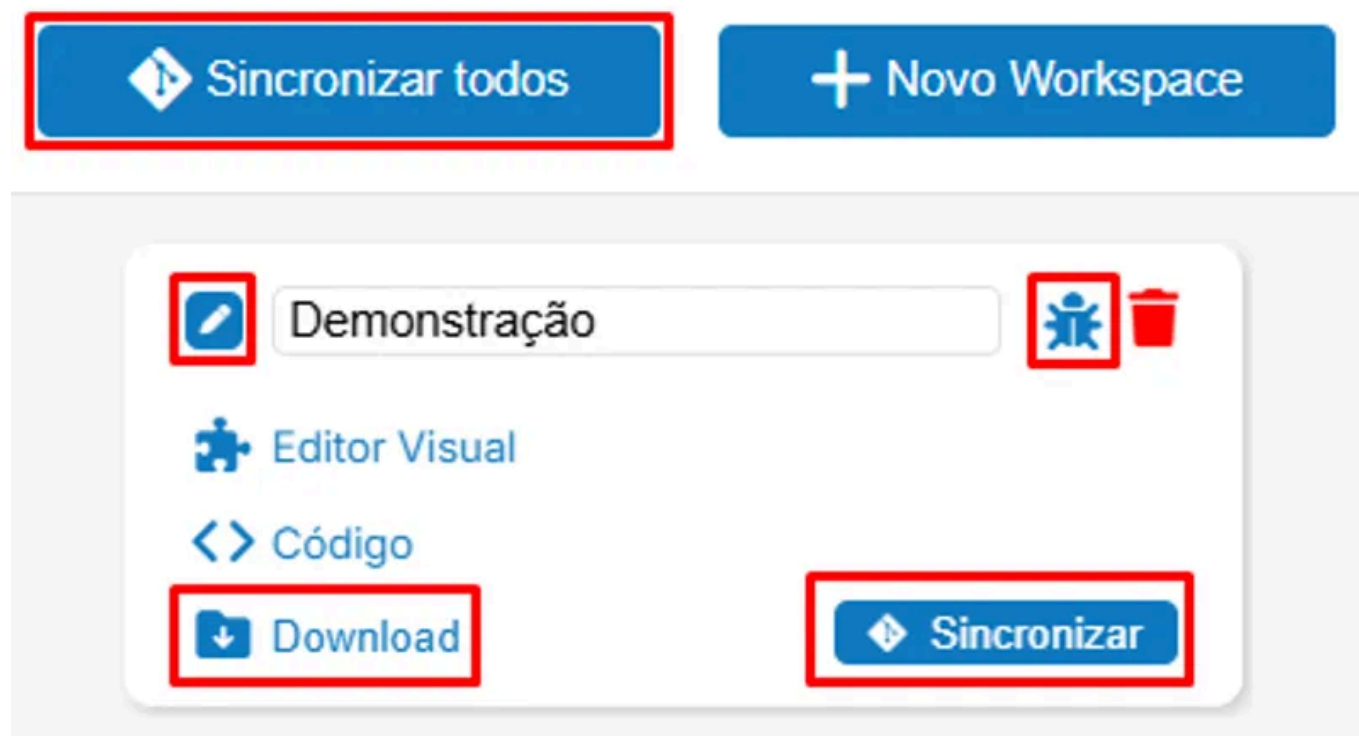


- **Área de código:** Aqui é onde você vai inserir o código da sua pipeline. Você pode digitar diretamente ou colar o código se já o tiver pronto.
- **Botão "Salvar":** Após finalizar a edição do código, clique em **"Salvar"** para garantir que suas alterações sejam aplicadas.
- **Lista de arquivos:** No painel lateral, você pode navegar entre os arquivos e pastas criados, como mostrado no exemplo.

Com essas ferramentas, você pode facilmente criar, editar e organizar as pipelines diretamente pelo sistema.

Outras Funções

O sistema de questionários e ações oferece diversas funcionalidades adicionais para facilitar a gestão e personalização dos seus Workspaces. Confira na imagem abaixo:



Funcionalidades disponíveis:

- **Sincronizar todos:** Permite sincronizar todas as alterações realizadas nos Workspaces com o repositório GIT conectado.
 - **Sincronizar (Card):** Atualiza as alterações mais recentes daquele card, garantindo que os dados estejam sempre em sincronia.
- **Editar nome:** Você pode alterar o nome do Workspace de forma simples. Basta clicar no ícone de lápis ou diretamente sobre o nome do Workspace. Após digitar o novo

nome, clique fora da caixa para salvar automaticamente a alteração.

- **Download:** Esta opção permite fazer o download completo de todas as criações realizadas no editor visual e no código. Assim, você pode manter uma cópia local do conteúdo para backup ou análise.
- **Visualizar Logs:** Clique no ícone em formato de **"bug"** para acessar o log de atividades do Workspace. Essa funcionalidade permite acompanhar de forma detalhada as alterações, sincronizações e outros eventos importantes, facilitando o monitoramento. Ao clicar, uma nova página será aberta exibindo todos os registros dos logs.

4. Customizações Avançadas

4.1. Engine local

Rodando a Engine localmente pela CLI

Obtendo o projeto

Bash



```
git clone http://gitlab.centralit.io/aura/blocks.git
```

Dentro dos repositórios do **blocks** temos uma pasta chamada **cli**. Acesse a mesma e compile o executável aura, caso ainda não tenha feito.

Bash



```
# dentro da raiz do projeto  
cd cli  
cargo install --path .
```

Aviso:

A partir desse momento o utilitário aura fica disponível para ser utilizado a partir de qualquer pasta

Configurando as variáveis de ambiente para a cli

É necessário exportar 3 variáveis de ambiente para o funcionamento do comando `aura`. As variáveis devem apontar para o local onde os respectivos recursos estão sendo executados.

Bash



```
export ENGINE_HOST=http://localhost:6060/engine
export LOWTALK_HOST=http://localhost:8989/lowtalk
export LOWWIDGET_HOST=http://localhost:4156/lowwidget
```

Aviso:

Por padrão os servidores acima já estão configurados para as portas e pontos de montagem definidos no exemplo

Inicializar servidores

Aviso:

Esse tutorial assume que o suporte as linguagens de programação utilizadas pelo servidor já estão funcionando.

Garanta que os comandos `cargo`, `go` e `npm` estão funcionando.

Aviso:

Garanta que o banco de dados está em execução e devidamente configurado nos arquivos `main.sh` dos respectivos projetos GO

Bash



```
# dentro da raiz do projeto
cd engine
sh scripts/main.sh

# dentro da raiz do projeto, em um novo terminal
cd lowtalk
sh scripts/main.sh

# dentro da raiz do projeto, em um novo terminal
cd lowwidget
sh scripts/main.sh

# dentro da raiz do projeto, em um novo terminal
cd wanderson
cargo run -- server -d ./workspace_examples

# dentro da raiz do projeto, em um novo terminal
cd lunaris
cargo run -- server -w ./workspace_examples
```

Resumo do comando aura

Após a instalação podemos ver os comandos disponíveis dentro da **aura**.

Ajuda

Bash



```
# Ajuda do comando aura
aura --help
```

Workspace

Bash



```
# Ajuda do subcomando workspace
aura workspace --help

# Retorna a lista de workspaces
aura workspace list

# Remove o workspace passado como parametro (coluna name da lista)
aura workspace remove wpname

# Cria um novo workspace a partir do lábel passado como parâmetro
aura workspace new "Label do Workspace"
```

Plugin

Bash



```
# Lista os plugins registrados em um workspace
aura list wpname

# Inspecciona as configurações de um plugin registrado
aura inspect wpname plname

# Remove um plugin registrado
aura remove wpname plname

# Registra um plugin
# Workflow: as propriedades podem ser passadas conforme a solicitação do plug
aura register wpname "label do plugin" "url_do_manifesto" "propriedade0=valor

# Configura talker padrão e a ordem dos eventos
aura conf wpname -d nometalkerpadrao -o primeiroplugin -o segundoplugin
```

Fila

Bash



```
# Lista as filas
aura attendant list

# Remove uma fila
aura attendant remove flname

# Cria uma nova fila
aura attendant new "Label da Fila" -a "email_do_agente1" -a "email_do_agenten

# Abre o atendimento humano para o agente passado no parâmetro
aura attendant open "email_do_agente"
```

Widget

Bash



```
# Lista os widgets
aura widget list

# Remove um widget
aura widget remove widget_name

# Cria uma novo widget
# Existem parâmetros adicionais que podem ser obtidos com --help
aura widget new "Label do Widget"

# Abre o widget passado como parâmetro
aura widget open "nome_do_widget"
```

4.2. Máquina de Estados - Detalhes

Este é um manual mais avançado das capacidades da máquina de estados. Os exemplos mostrados aqui pertencem ao editor de código de uma skill.

- `#` Adiciona um comentário;
- `/` Caractere de escape
- `@` Estado local;
- `@+` Espera por um input;
- `@@` Estado global;
- `-` Adiciona um texto;
- `=` Adiciona um exemplo;
- `%` Exemplo externo;
- `>` Avança para um estado;
- `!` Frase apresentada, caso não haja um entendimento;
- `?` Frase apresentada, caso haja uma desambiguação;
- `^` Permite ocultar menu (botão);
- `{{VARIABLE DE AMBIENTE}}` Variáveis de ambiente;
- `$PIPELINE_INTERCEPTOR: PASTA_DO_CLIENTE/NOME_DA_PIPELINA/` Chamada de pipeline;
- `$HUMAN_ATTENDANCE_INTERCEPTOR: ID_FILA` Chamada de fila de atendimento humano;
- Paginação
 - `more_options_label` Configura mensagem para ver mais opções;
 - `max_menu_options` Quantidade de menus apresentada por vez;
- `min_word_size` Tamanho mínimo para ser considerado uma palavra;

Estado ou Intenções

Operadores Especiais

Não entendimento (!)

O **operador (!)**, mostrará de forma aleatória uma das **frases apresentadas**, caso não haja um entendimento.

Desambiguação (?)

Outro operador muito similar ao similar ao anterior, é **operador (?)**, onde este **mostrará de forma aleatória uma das frases apresentadas**, caso haja uma **desambiguação**.

Texto (-)

Para adicionar um texto a um fluxo, use o operador (-). Vale lembrar que **se o fluxo tiver mais de um operador de texto (-), um abaixo do outro, todos no início da linha. A máquina de estados escolherá de forma aleatória um dos textos.**

Text



```
@start
= um exemplo aleatorio

- Olá, Seja bem-vindo(a)!
Eu sou o assistente virtual da Reduce XYZ. Como posso ajudar?
> end

@end
# Se seu fluxo tiver mais de um operador de texto ( - ), um abaixo do outro
como no exemplo. A máquina de estados escolherá de forma aleatória um dos tex
- Se precisar de ajuda, basta me chamar.
Reduce xyz agradece o contato!
- Reduce xyz está sempre a disposição para ajudar e agradecemos o contato!
```

```
~/codes/cetralit/wanderson on master ?3 wanderson examples/operador
```

```
- Olá, Seja bem-vindo(a)!
```

```
Eu sou o assistente virtual da Reduce XYZ. Como posso ajudar?
```

```
- Reduce xyz está sempre a disposição para ajudar e agradecemos o contato!
```

```
start..end 1.000
```

```
~/codes/cetralit/wanderson on master ?3 wanderson examples/operador
```

```
- Olá, Seja bem-vindo(a)!
```

```
Eu sou o assistente virtual da Reduce XYZ. Como posso ajudar?
```

```
- Se precisar de ajuda, basta me chamar.
```

```
Reduce xyz agradece o contato!
```

```
start..end 1.000
```

Exemplo (=)

O operador de exemplo (=) permite cadastrar exemplos a intenções e entidades para serem analisados e interpretados pela nossa NLU.

Exemplo Externo (%)

O operador de exemplo externo (%), permite cadastrar **de forma externa exemplos (=)**, que podem ser **utilizados por mais de uma intenção ou entidade**. Através **do caractere (&)**.

Utilizar essa abordagem reduz o número de linhas e facilita a adesão de novos exemplos, após o processo de curadoria.

Exemplos Externos devem ser cadastrados em uma única pasta, todos os arquivos nesta pasta devem conter apenas, este operador.

Text



```
%chit_chat_saudacao
= Alô
= Boa noite
= Boa tarde
= Bom dia
= Bom dia, prezados
= Hello
= Hi
= Oi
= Olá
= Saudação
```

Text



```
@start
# Uso do exemplo externo ( % )
&chit_chat_saudacao
= um exemplo aleatório

- Olá, Seja bem-vindo(a)!
Eu sou o assistente virtual da Reduce XYZ. Como posso ajudar?
> end

@end
&chit_chat_saudacao
- Se precisar de ajuda, basta me chamar.
Reduce xyz agradece o contato!
- Reduce xyz está sempre a disposição para ajudar e agradecemos o contato!
```

Variável de ambiente ({{variavel_ambiente}})

Este é um operador que permite **acessar variáveis externas de clientes**, para **incorporar** ao nosso **fluxo de conversa**. O exemplo **mais comum** é a variável, **{{username}}** do **Citsmart**.

Caractere de escape (/)

O caractere de escape permite, anular um operador da máquina de estados que seria interpretado de alguma maneira, para ser usado num texto.

Text



```
@start
- {{saudacao_tempo}}
Olá {{user_name}}. Como vai você?
Por favor acesse o perfil \@CentralIT

> end

@end
- Até a próxima!
```

Text



```
- Bom dia
Olá user_name. Como vai você?
ePor favor acesse o perfil @CentralIT

- Até a próxima!

start..end 1.000
```

Intenções Locais (@)

Intenções ou estados locais são estados que só podem ser acessados quando um jump (>) a partir do estado atual os aponta. São **categorizados** pelo

@nome_qualquer: titulo do menu e possuem a seguinte estrutura:

```
@start
- Olá, Seja bem-vindo(a)!
Eu sou o assistente virtual da Reduce XYZ. Como posso ajudar?
> atendente
> status

@atendente: Atendente
= atendente
- Um atendente irá te ajuda
> end

@status: ^ status
= status
- Seu status está OK
> end

@reduce: Reduce
= reduce
- impossivel entrar no reduce, pois nenhum estado, da jump para ele
> end

@end
- acabou
```


- Olá, Seja bem-vindo(a)!

Eu sou o assistente virtual da Reduce XYZ. Como posso ajudar?

1. map

```
start..start 1.000 > status
```

- Seu status está OK

- acabou

```
filter..end 1.000
```

```
~/codes/cetralit/wanderson on master ?3 wanderson examples/operador
```

- Olá, Seja bem-vindo(a)!

Eu sou o assistente virtual da Reduce XYZ. Como posso ajudar?

1. atendente

```
start..start 1.000 > reduce
```

- Não compreendi o que você deseja

1. atendente

```
start..start 0.000 >
```

Perceba que apesar de a intenção **@reduce** existir e **não estar ocultada**, pelo operador (^). Ela **não aparece de forma alguma no fluxo conversacional**. Isso porque o **estado atual** não aponta para o reduce.

Aguarde por input (@+)

Aguarde por input, faz com que o próximo jump, só seja executado caso o usuário digite o próximo passo. Perfeito para criar listas passo a passo. Veja o exemplo:

Text



```
@start
- oi
> passo1

@+passo1
- passo 1
> passo2

@+passo2: passo 2
- passo 2
> passo3

@passo3: passo 3
- passo 3
> end

@end
- tchau
```

Text



```
~/codes/cetralit/wanderson on master ?3 wanderson examples/operador

- oi

- passo 1
1. passo 2

start..passo1 1.000 > 1
- passo 2
1. passo 3

passo2..passo2 1.000 > 1
- passo 3

- tchau

passo3..end 1.000
```

Intenções Globais (@@)

Intenções ou estados globais, como proprio nome sugere, **podem ser acessados de qualquer etapada da conversa**. São **categorizados** pelo

`@@nome_qualquer: titulo do menu` e possuem a seguinte estrutura:

Text



```
@start
- Olá {{username}}! Seja bem-vindo(a)!
Eu sou o assistente virtual do Reduce. Como posso ajudar?

> map
> filter

#fluxo local
@map: map
= map
> end

#fluxo local
@filter: filter
= filter
> end

# fluxo global
@@fluxo_global_oculto: ^ fluxo global oculto
= fluxo global oculto
- todo fluxo global, se não ocultado pelo operador (^). Conforme a cima,
aparecera nos menus do @start
Este é o caso, do fluxo_global_não_oculto
> end

#fluxo global
@@fluxo_global_nao_oculto: fluxo global nao oculto
= fluxo global não oculto
- todo fluxo global, se não ocultado pelo operador (^). Conforme a cima,
aparecera nos menus do @start
> end

@end
- acabou
```

```
~/codes/cetralit/wanderson on master ?3 wanderson examples/operador
```

- Olá username! Seja bem-vindo(a)!

Eu sou o assistente virtual do Reduce. Como posso ajudar?

1. map
2. filter
3. fluxo global nao oculto

```
start..start 1.000 > fluxo global
```

```
start..start 0.683 >
```

```
~/codes/cetralit/wanderson on master ?3 wanderson examples/operador
```

- Olá username! Seja bem-vindo(a)!

Eu sou o assistente virtual do Reduce. Como posso ajudar?

1. map
2. filter
3. fluxo global nao oculto

```
start..start 1.000 >
```

```
~/codes/cetralit/wanderson on master ?3 wanderson examples/operador
```

- Olá username! Seja bem-vindo(a)!

Eu sou o assistente virtual do Reduce. Como posso ajudar?

1. map
2. filter
3. fluxo global nao oculto

```
start..start 1.000 > fluxo global não oculto
```

- todo fluxo global, se não ocultado pelo operador (^). Conforme a cima, aparecerá nos menus do @start

- acabou

```
fluxo_global_nao_oculto..end 1.000
```

Vale lembrar que todo **fluxo global, se não ocultado pelo operador (^)**.

Conforme a cima, **aparecerá** nos menus do **@start** Este é o caso, do **fluxo_global_não_oculto**

Entidades

Entidades na máquina de estados (*)

O operador que caracteriza uma **Entidade (*)**, seguido de seu nome `*nome` .

Para um melhor uso dessa entidade, você deve pegar exemplos (=) que são muito próximos e cadastrá-los numa mesma entidade. Veja a seguinte estrutura.

Entidades devem ser cadastrados em uma única pasta, todos os arquivos dessa pasta devem conter apenas, este operador. Vale lembrar também, que a ordem das Entidades (*) importa nos exemplos (=).

```
*veiculo
= carro
= moto
= caminhao

*marca_veiculo
= honda
= bmw
= ford
= fiat

*posse
= meu
= minha

*ipva
= imposto propriedade veículo automotor
= imposto propriedade veículos automotores
= ipva

*vontade
= quero
= preciso
= gostaria de
```

```
@start
- oi
> pagar_ipva
> transferir_carro
> end

@pagar_ipva
= *ipva *posse
- ipva

> end

@transferir_carro
# a ordem das Entidades ( * ) importa nos exemplos ( = )
= *ipva *marca_veiculo
- transferir carro

> end

@end
- tchau
```

Interceptadores (\$)

Pipeline

De forma bem resumida e abrangente o conceito de pipeline, representa as etapas do processo pelas quais algo passa até cumprir um objetivo.

Na equipe Gia, se utiliza muito pipelines para acessar api's do Citsmart para abrir chamados, preenchendo formulários via pipeline.

Como usar?

Basta colocar no fluxo que deseja invocar uma pipeline: " \$

PIPELINE_INTERCEPTOR: PASTA_DO_CLIENTE/NOME_DA_PIPELINE/ "

O nome da pasta do cliente bem como nome da pipeline pode ser encontrado no CITBot - Filebrowser

File Browser

Text



```
@mapear_pasta_rede: ^ Conceder Acesso/Mapear pasta de rede
= Acesso a pasta de rede
- Entendi que deseja atendimento sobre concessão de acesso/mapeamento de pasta de rede. está correto?

> mapear_pasta_rede_sim
> mapear_pasta_rede_nao

@mapear_pasta_rede_sim: ^ Sim
&navigation_sim

# INVOCANDO PIPELINE
$ PIPELINE_INTERCEPTOR: PASTA_DO_CLIENTE/NOME_DA_PIPELINE/

> finalizacao_fluxo_padrao

@mapear_pasta_rede_nao: ^ Não
&navigation_nao
> finalizacao_de_fluxo_em_caso_de_entendimento_incorreto
```

Atendimento Humano

A plataforma Aura permite que o bot, transfira para uma fila de atendimento humano. Bastando colocar no fluxo que deseja invocar um atendimento humano

```
$HUMAN_ATTENDANCE_INTERCEPTOR:ID_FILA .
```

Para conseguir o ID_FILA, acesse a plataforma aura ou consulte seu manual respectivamente.

Citbot

Citbot Manual


```
@@atendimento_humano: ^ Atendimento Humano
- Gostaria de falar com um de nossos analistas?
&atendimento_humano
&demanda_ao_resolvida
&nenhuma_das_opcoes
> atendimento_humano_sim
> atendimento_humano_ao

@atendimento_humano_sim: ^ Sim
&navigation_sim

# INVOCANDO ATENDIMENTO HUMANO
$HUMAN_ATTENDANCE_INTERCEPTOR:ID_FILA

> finalizacao_fluxo_padrao

@atendimento_humano_ao: ^ Não
&navigation_ao
> finalizacao_fluxo_padrao
```

Parametros de configuração

Recomenda-se que os parâmetros citados nesse tópico, sejam passados num arquivo chamado: **"01_settings"**.

Paginação

Os parâmetros de paginação permitem, mostrar uma quantidade específica de menus na tela. Sendo o `more_options_label`, a mensagem que permite ver mais opções, `max_menu_options`, a quantidade de menus apresentada por vez.


```
# paginated
more_options_label = Ver mais
max_menu_options = 3

@start
? estou em dúvida entre as opções
! não entendi absolutamente nada
- Esses são os parametros de paginação, sendo o more_options_label, a
mensagem que permite ver mais opções.
O max_menu_options, a quantidade de menus apresentada por vez.

> menu
> atendimento
> status
> reclamação
> novo_pedido

@manu: Menu
- Menu
= menu
> start

@atendimento: Atendimento
- Atendimento
= atendimento
> start

@status: Status
- Status
= status
> start

@reclamação: Reclamação
- Reclamação
= reclamacao
> start

@novo_pedido: Novo Pedido
- Novo pedido
= novo
= pedido
> start

@@end: Sair
= tchau
```

```
= bye
- tchau
```

Text



```
- Esses são os parametros de paginação, sendo o more_options_label, a
mensagem que permite ver mais opções.
```

```
0 max_menu_options, a quantidade de menus apresentada por vez.
```

1. Menu
2. Atendimento
3. Status
4. Sair
5. Ver mais

```
start..start 1.000 > 5
```

```
- Opções:
```

1. Reclamação
2. Novo Pedido
4. Sair
5. Ver mais

```
start..start 1.000 > 4
```

```
- tchau
```

```
end..end 0.000
```

Tamanho minimo de palavra

Como o próprio nome sugere, o parâmetro `min_word_size`, especifica o **número mínimo de letras**, que uma palavra deve ter, para a máquina de estados considerar ela como válida.

Vale lembrar que por padrão o `min_word_size` é igual a 2, caso queira alterar esse valor, basta passar o parâmetro.

Emojis nos exemplos (=)

Para passar emoji nos exemplos, `min_word_size` deve ser igual a 0.

Text



```
min_word_size = 0

@start
- Oi. Como você está? 
> triste
> feliz

@triste: Triste
= 😞
- Que pena
> end

@feliz: Feliz
= 😊
- um dia lindo um bom lugar pra ler um livro
> end

@end
- tchau
```

Text



```
~/codes/cetralit/wanderson on master !2 ?3 wanderson examples/emoji.pagode

- Oi. Como você está? 
1. Triste
2. Feliz

start..start 1.000 > c
- - um dia lindo um bom lugar pra ler um livro

- tchau

feliz..end 1.000
~/codes/cetralit/wanderson on master !2 ?3
```

Rank Options

A máquina de estados trabalha com referência por estado, palavras-chave e número de palavras. Dando uma nota para os inputs, colocados pelo usuário. Como mostra o exemplo do debug export `WANDERSON_INFO=TRUE` . Os Ranks abaixo permitem mudar isso.

Text



```
~/codes/cetralit/wanderson on master !2 ?3 wanderson examples/emoji.pagode
```

```
☹ => triste((1.0, 1))
```

```
😊 => feliz((1.0, 1))
```

Referências por estado: 16

```
end ////////////////////////////////// 4
feliz ////////////////////////////////// 4
start ////////////////////////////////// 4
triste ////////////////////////////////// 4
```

Número de palavras chave: 2

```
1 ////////////////////////////////// 2
```

Número de estados relacionados a palavra chave: 2

```
1 ////////////////////////////////// 2
```

- Oi. Como você está? ☹

1. Triste

2. Feliz

```
start..start 1.000 > ☹
```

```
* ☹
```

Rank: 1

```
feliz ////////////////////////////////// 1
```

```
triste 0
```

- - um dia lindo um bom lugar pra ler um livro

- tchau ☹

```
feliz..end 1.000
```

ignore_below

O valor padrão desse parâmetro fará com que a máquina de estados ignore tudo abaixo de

```
ignore_below = 0.2
```

is_answer_above

O valor padrão desse parâmetro fará com que a máquina de estados somente considere como correto, tudo que for superior ou igual a `is_answer_above = 0.8`

parse_at_start

Esse parâmetro define, se no **@start**, o bot vai começar mandando mensagem

```
parse_at_start = false
```

 ou se ele vai esperar o usuário digitar

```
parse_at_start = true
```

 .

4.3. Manual do widget

Widgets são interfaces configuráveis que permitem personalizações como cores, imagens e nomes, tornando cada um deles único para o usuário. Entre as opções de personalização, você pode escolher a cor principal, adicionar um avatar, definir um título e até selecionar um papel de parede, entre outras possibilidades. Neste guia, explicaremos cada uma dessas configurações e mostraremos o processo de criação de um widget passo a passo.

O objetivo principal do widget é facilitar a comunicação entre o usuário final e nossa engine, ou até mesmo com nossa equipe de atendimento. Incorporado diretamente no site do cliente, o widget oferece praticidade e conveniência, eliminando a necessidade de acessar outros sites ou canais para interação.

Como criar um widget

No menu **CITSmart Aura**, selecione a opção **Widget**.

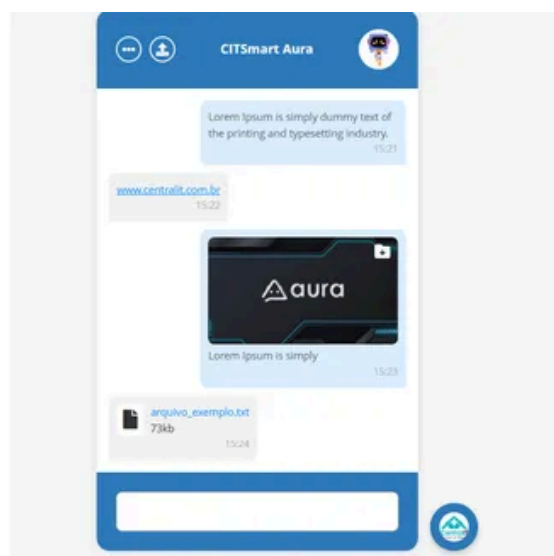
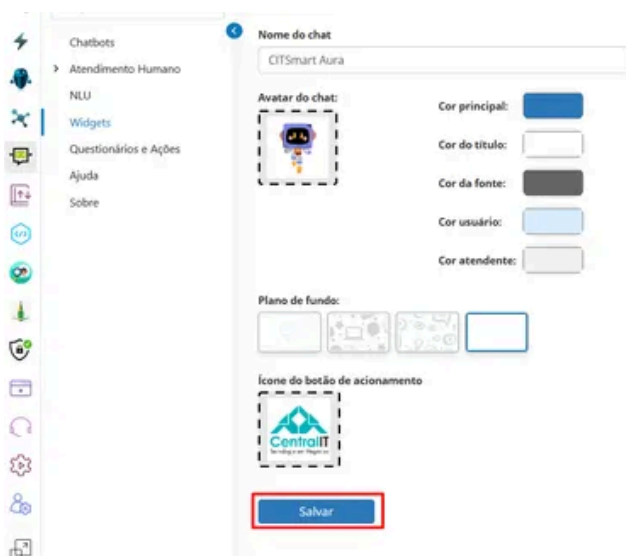


Uma página com todos os widgets já registrados será exibida. Para criar um novo, clique em **"Novo widget"**.



Personalizando seu Widget

Após clicar em "**Novo Widget**" você será redirecionado para a página de personalização.



Na página de personalização, preencha as seguintes informações para configurar o widget:

- **Nome do chat:** Nome exibido no topo do widget, ao lado do avatar.

- **Avatar do chat:** Imagem do avatar do chatbot, com tamanho máximo de **100×100 pixels**.
- **Cor principal:** Cor predominante do widget, visível no cabeçalho e rodapé.
- **Cor do título:** Cor do texto no cabeçalho do widget.
- **Cor da fonte:** Cor usada para o texto dentro do widget.
- **Cor usuário:** Cor usada para identificar as mensagens enviadas pelo usuário.
- **Cor atendente:** Cor usada para identificar as mensagens enviadas pelo atendente.
- **Papel de fundo:** É possível personalizar o papel de fundo do widget. Sendo possível escolher uma das quatro opções disponíveis para personalizar o fundo do widget.
- **Ícone do botão de acionamento:** É possível escolher entre quatro opções para personalizar o fundo do widget, com tamanho máximo de **100×100 pixels**

Nota: Para selecionar cores, clique na caixa de cor, e uma janela pop-up será exibida para escolher a cor desejada.

À medida que realiza as personalizações, uma pré-visualização aparecerá ao lado das configurações. Quando concluir, clique em **Salvar** para criar o widget.

Editando um Widget

Para editar um **Widget** existente, siga os passos:

1. Na tela de listagem de widgets, encontre o widget que deseja modificar.
2. Clique no ícone de pincel **Editar widget**.



1. Faça as alterações necessárias na tela de edição.
2. Clique em **Salvar** para aplicar e registrar as mudanças.

Excluindo um Widget

Para excluir um **Widget**, siga os passos abaixo:

1. Na tela de listagem, encontre o widget que deseja excluir.
2. Clique no ícone de lixeira ao lado do nome do widget.



1. Um modal de confirmação será exibido. Clique em **Sim** para confirmar a exclusão.

Atenção: Uma vez excluído, o widget não poderá ser recuperado. Para usá-lo novamente, será necessário criar um novo.

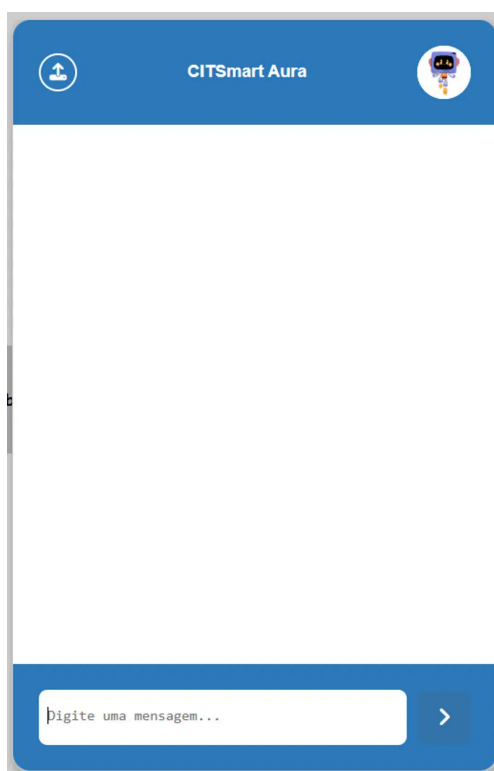
Acessando o Widget

Após configurar e criar o widget, acesse a página de listagem. Nessa tela, você poderá visualizar todos os widgets criados.

Para abrir a interface do chatbot, clique em "**Link para o chat aplicado**".



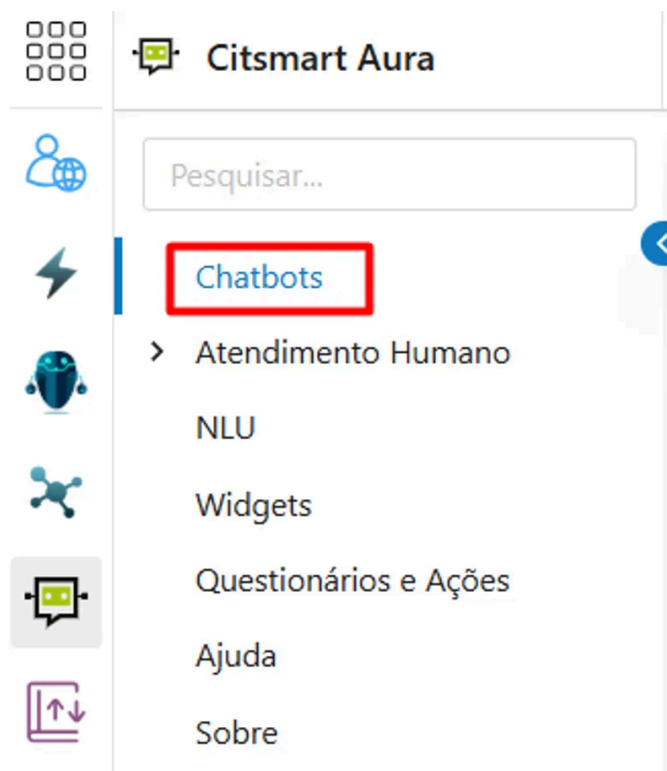
Isso abrirá um link para a interface do seu chatbot.



Criando comportamento para o Widget

Para que o widget responda às interações, é necessário registrar um comportamento. Siga os passos abaixo:

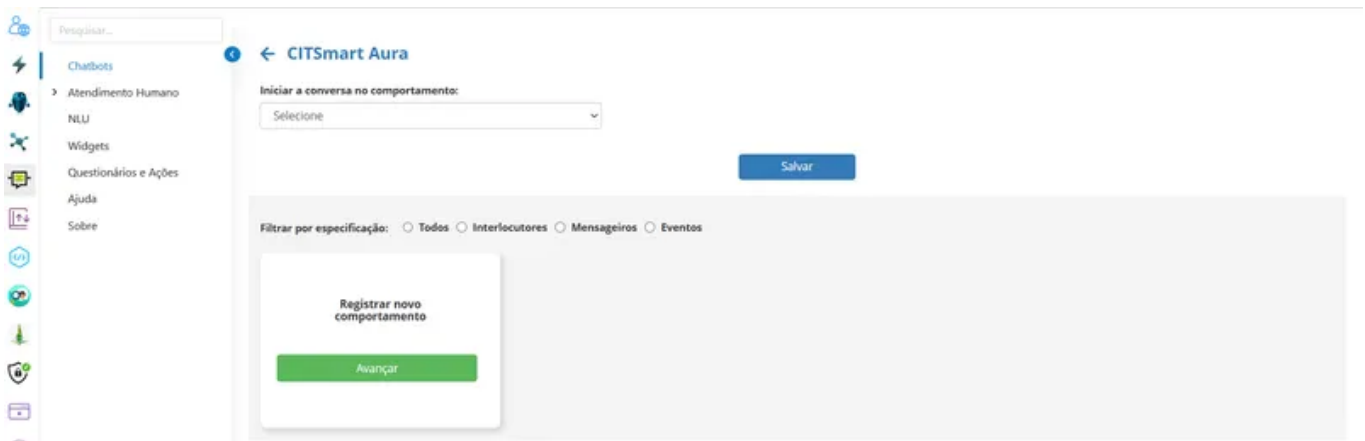
1. Acesse o menu **Chatbots**.



1. Caso ainda não possua, crie um **"Chatbot"** clicando em **"Novo Chatbot"**.



1. Insira o nome desejado e clique em **"Criar novo chatbot"**. Em seguida, clique no nome do chatbot recém-criado para acessá-lo.
2. No card **"Registrar novo comportamento"**, clique em **"Avançar"**.



1. Preencha o formulário de registro conforme indicado:

- **Nome do Comportamento:** Nome para identificar o comportamento.
- **Selecione o Manifesto:** Escolha o manifesto do tipo "**Widget**".
- **Parâmetros:** Escolha o **Widget** que deseja cadastrar.

1. Confira o preenchimento das informações e clique em "**Concluir**" para registrar o comportamento.

⚠ **Importante:** Um widget só pode ser registrado em um único comportamento por vez.

Isso significa que **não é possível utilizar o mesmo widget simultaneamente em dois ou mais chatbots.**

Para associar o mesmo visual ou configurações a múltiplos chatbots, será necessário criar widgets distintos para cada um.

Pesquisar...

Chatbots

Atendimento Humano

NLU

Widgets

Questionários e Ações

Ajuda

Sobre

Registrar novo comportamento

Nome do Comportamento:

CITSmart Aura

Selecione o manifesto: *

Widget

Ou digite a URI do manifesto: *

https://aura-blocks.centralit.com.br/widget_dev/manifest

Parâmetros ?

v.1.0.0

Widget *

CITSmart Aura

Concluir Cancelar

Atenção: Para que o widget responda às interações, também é necessário cadastrar um comportamento do tipo **Wanderson** e selecionar a skill desejada. Para mais informações sobre esse cadastro, consulte o manual do operador.

2. Após registrar os comportamentos, volte para a listagem de widgets e clique em **"Link para o chat aplicado"** referente ao widget configurado. Isso abrirá a interface do seu chatbot e será possível realizar as interações desejadas.

4.4. Bibliotecas

4.4.1. Biblioteca padrão do Lunarix

O Lunarix é um ambiente de execução Lua projetado com foco na flexibilidade e integração. Ele injeta todas as bibliotecas necessárias diretamente no escopo global Lua através de uma Virtual Machine (VM) Lua. Isso permite que funções e recursos específicos do Lunarix, como manipulação de dados, operações HTTP, codificação e segurança, sejam acessados diretamente no código Lua sem a necessidade de importação explícita.

Para remover a mensagem de advertência sobre variáveis globais, é possível personalizar a configuração do Lua com o seguinte arquivo `.luarc.json`, que define as bibliotecas globais permitidas e desabilita a advertência sobre o uso de variáveis globais:

```
{
  "diagnostics.globals": [
    "set_key",
    "http",
    "json",
    "LUNARIS_URL",
    "get_key",
    "remove_key",
    "urlencoding",
    "base64",
    "dataurl",
    "hmac"
  ],
  "diagnostics.disable": [
    "lowercase-global"
  ]
}
```

4.4.2. Logs

Este módulo é responsável pela geração de logs durante o tempo de execução da aplicação. Os logs são registrados em um arquivo, utilizando o formato chave-valor.

Atalhos

- Log

Métodos

Log

Lua



```
-- @param identifier The identifier of the log
-- @param value The value to be logged.
log(identifier, value)
```

```
function process_data(a)
    -- Perform some operations (this can be customized)
    -- For the sake of this example, we will simply pass the variable through

    log("PROCESSING", a)

    return a
end

-- Assuming the variable "a" has the value of 42:
local a = 42

-- When the function is called, the log entry generated would look like this:
-- [2025-02-13 18:03:37.137227800 -03:00 "PROCESSING"]: 42
```

4.4.3. Pipelines Lua

O que é?

O sistema de Pipelines permite criar interações conversacionais customizadas através de scripts em Lua podendo ser utilizada como uma ferramenta de linha de comando ou como um servidor. De forma simplificada pode ser entendido como um questionário que é definido através de scripts Lua.

Como funciona uma Pipeline?

Para utilizarmos o sistema, primeiramente devemos criar uma pasta em nosso sistema operacional que será utilizada para armazenar todos os nossos workspaces, cada um contem os seus respectivos scripts Lua. Após isso criamos outra pasta que vai conter o nosso projeto específico.

O nome da pasta não é restritivo, mas para fins didáticos vou me referir a ela como **workspaces**.

Exemplo: `workspaces/projeto-exemplo`

Passo um: criação da tabela de pipelines

A tabela de pipelines é especial e **deve ser definida uma única vez no workspace**, ela é responsável pela definição de quais são as pipelines presentes no workspace.

Vamos começar criando um arquivo Lua `main.lua` dentro de um workspace conforme exemplo acima.

Lua



```
-- Arquivo main.lua

PIPELINES = {
  "questionario_dados_pessoais",
  "consultar_chamado"
}
```

Acima criamos a tabela `PIPELINES` que informa ao sistema a existência de duas pipelines: `questionario_dados_pessoais` e `consultar_chamado`.

Passo dois: definição de uma pipeline

Vamos agora definir a pipeline `questionario_dados_pessoais`, para isso devemos criar a tabela própria da pipeline contendo as perguntas que ela possui.

Lua



```
-- Arquivo main.lua

PIPELINES = {
  "questionario_dados_pessoais",
  "consultar_chamado"
}

questionario_dados_pessoais = {
  "perguntar_nome",
  "perguntar_idade",
}
```

Perceba que os nomes devem corresponder nas tabelas de `PIPELINES` e em sua própria definição.

Passo tres: definição das perguntas

Cada pergunta é representada por uma tabela com o seu respectivo nome que foi definido acima, a tabela contem quatro campos:

- `output` é a resposta da pergunta atual.
- `show` define se a pergunta deve ser feita ou não
- `select` é um conjunto de possíveis escolhas mostrada ao usuário junto ao *output*. O *label* é a frase mostrada de cada opção, no qual há um número *id* associado respectivamente.
- `accept` define se a pergunta atual atende condições específicas e pode ser mostrada.
- `reject` é a resposta da pergunta atual caso o `accept` seja falso.

Exemplo completo.


```

-- Arquivo main.lua

PIPELINES = {
    "questionario_dados_pessoais",
    "consultar_chamado"
}

questionario_dados_pessoais = {
    "perguntar_nome",
    "perguntar_idade",
    "perguntar_localidade"
}

-- Pergunta simplificada
perguntar_nome = {
    output = "Olá, qual é o seu nome?", -- string
    show = true,                        -- boolean
    accept = true,                      -- boolean
    reject = "Mensagem de rejeição"    -- string
}

-- Pergunta com condições
perguntar_idade = {
    output = function(respostas)
        -- O output pode ser um função caso deseje utilizar as respostas
        -- anteriores
        return "Olá" .. respostas.perguntar_nome .. ", qual é a sua idade?"
    end,
    show = function(respostas)
        -- O show pode ser uma função caso deseje fazer alguma condição para
        -- determinar se o sistema deve fazer essa pergunta ao usuário ou não
        if respostas.perguntar_nome == "Felipe" then
            -- se a resposta da primeira pergunta for Felipe
            -- o sistema não vai perguntar sua idade
            return false
        end
        return true
    end,
    accept = function(atual, repostas)
        -- Aqui definimos condições para aceitar o input do usuário
        -- baseado na resposta atual ou anteriores
        if tonumber(atual) < 18 then
            return false
        end
        return true
    end
}

```

```

end,
reject = function(atual, respostas)
  -- Podemos usar a resposta atual e as anteriores para rejeitar
  -- a resposta do usuário
  return "Digite uma idade maior " .. respostas.perguntar_nome
end,
}

perguntar_localidade = {
  output = "Localidade:",
  show = true,
  accept = true,
  select = {
    { label = "Acre", id = 2},
    { label = "Alagoas", id = 3 },
    { label = "Amapá", id = 4},
    { label = "Distrito Federal", id = 5},
    { label = "Espírito Santo", id = 6},
    { label = "Goiás", id = 7},
    { label = "Maranhão", id = 8},
    { label = "Mato Grosso", id = 9},
    { label = "Mato Grosso do Sul", id = 10},
    { label = "Minas Gerais", id = 11},
    { label = "Pará", id = 12},
    { label = "Paraíba", id = 13},
    { label = "Paraná", id = 14},
    { label = "Pernambuco", id = 15},
    { label = "Piauí", id = 16},
    { label = "Rio de Janeiro", id = 17},
    { label = "Rio Grande do Norte", id = 18},
    { label = "Rio Grande do Sul", id = 19},
    { label = "Rondônia", id = 20},
    { label = "Roraima", id = 21},
    { label = "Santa Catarina", id = 22},
    { label = "São Paulo", id = 23},
    { label = "Sergipe", id = 24},
    { label = "Tocantins", id = 25},
  },
  reject = "Mensagem de rejeição"
}

```

Estrutura de projeto e biblioteca global

Dentro da pasta usada para guardar os workspaces, é possível criar um projeto especial com o nome `lib_global` que compartilha todo o seu conteúdo com os outros projetos e repositórios, ele pode ser usado para escrever funções utilitárias que são usadas amplamente em todos os outros projetos por exemplo.

Estrutura de exemplo:

Text

```
workspaces
├── cliente-x
│   ├── arquivo_x.lua
│   ├── arquivo_g.lua
│   └── subpasta_x
│       ├── arquivo_t.lua
│       └── arquivo_y.lua
├── cliente-y
│   ├── arquivo_t.lua
│   └── subpasta_z
│       ├── arquivo_t.lua
│       └── arquivo_y.lua
└── lib_global
    ├── arquivo_h.lua
    └── subpasta
        └── cnbbb.luaa
```

Como pode-se ver a estrutura de pastas dentro de cada projeto é livre para se decidir e usar da forma que preferir.

Bibliotecas disponíveis no Lua

Os seguintes pacotes vão estar disponíveis no ambiente de desenvolvimento através do LuaRocks.

- [lua-cjson](#)
- [http](#)
- [xml2lua](#)
- [inspect](#)
- [base64](#)

- urlencode
- openssl

Além dos pacotes externos está disponível também as seguintes bibliotecas de sistema:

- TABLE
- STRING
- COROUTINE
- PACKAGE
- MATH
- DEBUG

Formas de execução do sistema

O sistema de pipelines possui duas formas de ser executado, ferramenta de linha de comando (CLI) e servidor HTTP.

O sistema pode ser instalado localmente usar o seguinte comando no local do repositório:

```
cargo install --path .
```

Repl (CLI)

Modo de linha de comando.

Comando para se executar no modo de linha de comando:

Bash



```
cargo run repl --client <CLIENT> --pipeline <PIPELINE>
```

Descrição dos parâmetros

Bash



```
-w, --workdir <WORKDIR>  Workspaces directory, uses current dir as default
-c, --client <CLIENT>    Client name
-p, --pipeline <PIPELINE> Pipeline name
-h, --help                Print help
-V, --version             Print version
```

Server

Modo de servidor HTTP.

Comando para se executar no modo servidor:

Bash



```
cargo run server --port <PORT> --username <USERNAME> --password <PASSWORD>
```

Descrição dos parâmetros

Bash



```
-W, --workdir <WORKDIR>  Workspaces directory, uses current dir as default
-S, --sync               Synchronize with Git
-P, --port <PORT>        Server port
-u, --username           Username
-p, --password           Password
-h, --help               Print help
-V, --version            Print version
```

IMPORTANTE: para se utilizar a sincronização com git é necessário configurar o arquivo `Config.toml` dentro do seu diretório utilizado como `workdir` utilizando o seguinte modelo:

Exemplo de arquivo `Config.toml`

```
[cliente-xpto]
url = "https://github.com/centralit-governanca-corporativa/cliente-xpto.git"
username = "git-user"
password = "secret-password"

[nome-do-cliente]
url = "url-do-github"
username = "git-user"
password = "secret-password"
```

Documentação da API

A API suporta requisições para quatro recursos diferentes:

Workspaces

Recursos utilizados para manipular os workspaces.

Reload All Workspaces

- **Method:** GET
- **Endpoint:** `/workspace/reload`
- **Description:** Reloads all workspaces

Reload One Workspace

- **Method:** GET
- **Endpoint:** `/workspace/reload/{client}`
- **Description:** Reloads a specific workspace identified by the client.

Git

Recursos utilizados para se sincronizar o GIT.

Sync All Repositories

- **Method:** GET
- **Endpoint:** `/git/sync`
- **Description:** Synchronizes all Git repositories.

Sync One Repository

- **Method:** GET
- **Endpoint:** `/git/sync/{client}`
- **Description:** Synchronizes a specific Git repository identified by the client.

Conversation

Recursos utilizadas para interagir com as conversas.

IMPORTANTE: Esse recurso está protegido por `Basic auth`, as credenciais são as mesmas utilizadas na inicialização do servidor.

Create Conversation

- **Method:** POST
- **Endpoint:** `/conversation/spawn/{client}/{pipeline}/{conversation_uuid}`
- **Authentication:** Basic auth
- **Description:** Creates a new conversation instance within a specific pipeline and conversation_uuid for the given client.

Process Conversation

- **Method:** POST
- **Endpoint:** `/conversation/process/{conversation_uuid}`
- **Authentication:** Basic auth
- **Description:** Processes the specified conversation within the given pipeline for the client.

Body:

JSON



```
{ input: "User input here" }
```

Has Conversation

- **Method:** GET
- **Endpoint:** `/conversation/{conversation_uuid}`
- **Authentication:** Basic auth
 - **Description:** Verify if a conversation within a specific conversation_uuid exists.

Exposição de funções para Web

É possível expor funções que podem ser chamadas através de endpoints específicos que serão consumidos por outros serviços que desejem se comunicar com nosso sistema pelos protocolos http.

Para expor alguma função primeiro é necessário declarar uma tabela. Essa tabela pode ser declarada dentro de um arquivo que já contenha um fluxo conversacional das pipelines ou pode ser criado em um arquivo separado dentro da pasta do projeto.

Para nosso exemplo, criaremos um arquivo chamado webservice `workspaces/projeto-exemplo/webservice.lua`

Dentro do arquivo temos 5 (cinco) tabelas.

Lua



```
-- Arquivo webservice.lua

ALLOW_GET_ROUTES = {}
ALLOW_POST_ROUTES = {}
ALLOW_PUT_ROUTES = {}
ALLOW_PATCH_ROUTES = {}
ALLOW_DELETE_ROUTES = {}
```

Cada tabela corresponde a um método HTTP (GET, POST, PUT, PATCH e DELETE).

Agora vamos criar uma função dentro desse arquivo.

Lua



```
-- Arquivo webservice.lua

function teste()
    return "Hello Word"
end
```

Com a função criada, precisamos expô-la para que seja acessada via web. Neste caso vamos expô-la para que seja acessada pelo método GET.

Lua



```
-- Arquivo webservice.lua

ALLOW_GET_ROUTES = {
    "teste"
}
ALLOW_POST_ROUTES = {}
ALLOW_PUT_ROUTES = {}
ALLOW_PATCH_ROUTES = {}
ALLOW_DELETE_ROUTES = {}

function teste()
    return "Hello Word"
end
```

Dessa forma, podemos acessar essa função construindo a URL abaixo:

```
http://dominio-exemplo/get/projeto-exemplo/teste
```

- **dominio-exemplo:** domínio que roda o sistema
- **get:** constitui a url para informar que iremos buscar uma função declarada no ALLOW_GET_ROUTES
-- O método http também deve ser o GET
- **projeto-exemplo:** Nome do projeto em que o arquivo webservice.lua se encontra

- **teste:** nome da função que você deseja acessar.

Chamado então esse endpoint, você receberá como resposta o que foi retornado na função `Hello Word`.

IMPORTANTE

Todas as funções recebem 3 (três) parâmetros. O primeiro são os headers da requisição, o segundo os query params e o terceiro o body.

Dessa forma podemos modificar nossa função afim de receber esses parametros e realizar alguma lógica utilizando-os.

Lua



```
-- Arquivo webservice.lua

ALLOW_GET_ROUTES = {
  "teste"
}
ALLOW_POST_ROUTES = {}
ALLOW_PUT_ROUTES = {}
ALLOW_PATCH_ROUTES = {}
ALLOW_DELETE_ROUTES = {}

function teste(headers, query)
  --lógica a ser aplicada com os parâmetros
  return "Hello Word"
end
```

OBSERVAÇÃO 1: Caso você use o parâmetro query e no endpoint não seja enviado query params, ela estará vazio.

OBSERVAÇÃO 2: Caso você declare um terceiro parâmetro na função ele será entendido como o body da requisição, e no caso do método HTTP GET, que não há body, ele virá como uma string vazia.

Vamos agora criar outra função para ser chamada via método HTTP POST.

Criaremos a função `consulta_autorizacao` no mesmo arquivo que estávamos `webservice.lua`. Essa função receberá os 3 (três) parâmetros citados acima. Ela será exposta na tabela `ALLOW_POST_ROUTES` para estar disponível para acesso via HTTP.

Lua



```
-- Arquivo webservice.lua

ALLOW_GET_ROUTES = {
    "teste"
}

ALLOW_POST_ROUTES = {
    "consulta_autorizacao"
}

ALLOW_PUT_ROUTES = {}

ALLOW_PATCH_ROUTES = {}

ALLOW_DELETE_ROUTES = {}

function teste()
    return "Hello Word"
end

function consulta_autorizacao(headers, query, body)
    --lógica para consultar alguma autorização
    return "Autorizado"
end
```

O endpoint para chamar a função que criamos agora é `http://dominio-exemplo/post/projeto-exemplo/consulta_autorizacao`. Lembrando que essa função foi exposta no `ALLOW_POST_ROUTES`, sendo assim é necessário que você utilize o método HTTP POST e na construção da url, após o domínio-base, você deverá insirir `/post/` para que sua função seja chamada corretamente.

4.4.4. Audio

Biblioteca utilizada para realizar processamentos em áudio.

Atalhos

- Convert

Métodos

convert

Utiliza ffmpeg para a conversão.

Lua



```
-- @param from_type representa o formato do áudio original a ser convertido
-- @param to_type representa o formato de destino do áudio
-- @param bytes representa os bytes originais do áudio a ser convertido
-- @return bytes do áudio convertido
audio.convert(from_type, to_type, bytes)
```

Lua



```
-- example usage:
local mp3_bytes = "xxxyyyzzz..."

-- converte um áudio mp3 para ogg
local ogg_bytes = audio.convert("mp3", "ogg", mp3_bytes)
```

4.4.5. Base 64

Utilizado para codificar e decodificar valores no formato Base64.

Atalhos

- Encode
- Decode

Métodos

Decode

Lua



```
-- @param value The value to be decoded.  
-- @return string containing the decoded value  
base64.decode(value)
```

Lua



```
-- example usage:  
  
local decoded_value = base64.decode("aGVsbG8gd29ybGQh")  
  
-- hello world!
```

Encode

Lua



```
-- @param value The value to be encoded.  
-- @return string containing the encoded value  
base64.encode(value)
```

Lua



```
-- example usage:  
  
local encoded_value = base64.encode("hello world!")  
  
-- aGVsbG8gd29ybGQh
```


4.4.6. URL

Biblioteca para serializar e deserializar URLs. O uso dessa biblioteca visa evitar a edição insegura de urls que acontece quando concatenamos strings diretamente.

Atalhos

- Parse
 - Serialize
-

Métodos

Parse

Lua



```
-- @param string representando uma url
-- @return Uma table representando as estruturas que compõem a url específica
url.parse("http://a")
```

```
-- example usage:
local central_url = "http://usuario:senha@centralit.com.br/"
    .. "start/end?parametro=valor&parametro2=valor2"

local parsed = url.parse(central_url)

--[[
parsed value:

{
  host = "centralit.com.br",
  pass = "senha",
  path = { "start", "end" },
  port = 9876,
  query = {
    parametro = "valor",
    parametro2 = "valor2"
  },
  scheme = "http",
  user = "usuario"
}
--]]
```

Serialize

```
-- @param Uma table representando as estruturas que compõem a url especificada
-- @return A string da url final
url.parse(url_table)
```

```
-- example usage:
local parsed = url.parse("http://a")

parsed.port = 1234
parsed.scheme = "wss"
parsed.host = "centralit.com.br"
parsed.path = {"a", "b", "c"}
parsed.query = {
    d = "e",
    f = "g"
}
parsed.user = "aura"
parsed.pass = "123456"

local final_url = url.serialize(parsed)

-- final_url = "wss://aura:123456@centralit.com.br:1234/a/b/c?d=e&f=g"
```

4.4.7. URL Encoding

Utilizada para codificar caracteres especiais em uma URL, substituindo-os por sequências de escape no formato `%HH`, onde `HH` representa o valor hexadecimal do byte correspondente.

Atalhos

- Encode
- Decode

Encode

Lua



```
-- @param url String containing the url.  
-- @return a string containing the encoded url  
urlencoding.encode(url)
```

Lua



```
-- example usage  
  
local encoded_url = urlencoding.encode("https://httpbin.org/get")  
  
-- https%3A%2F%2Fhttpbin.org%2Fget
```

Decode

Lua



```
-- @param url String containing the url.  
-- @return a string containing the decoded url  
urlencoding.decode(url)
```

Lua



```
-- example usage  
  
decoded_url = urlencoding.decode("http%3A%2F%2Flocalhost%3A1754%2F.well-known  
-- http://localhost:1754/.well-known/lunaris-configuration
```

4.4.8. Data URL

Utilizada para embutir pequenos arquivos diretamente em documentos, representando dados como uma string codificada em Base64. Segue o formato `data:[<tipo MIME>[;base64],<dados>` de acordo com o **Fetch Standard**.

Atalhos

- **Process**
- **Create**

Métodos

Process

Lua



```
-- @param data String containing the Data URL.  
-- @return a table containing all of the the data url parts.  
dataurl.process(data)
```

Lua



```
-- example usage

local result = dataurl.process("data:text/plain;charset=iso-8859-7,%eb%fc%e3%

local body = result.body
local mimetype = result.mime.full

--[[
{
  body = "00000",
  fragment = "fragment-url-here",
  mime = {
    full = "text/plain;charset=iso-8859-7",
    parameters = {
      charset = "iso-8859-7"
    },
    subtype = "plain",
    type = "text"
  }
}
--]]
```

Create

Lua



```
-- @param mimetype String containing the mimetype.
-- @param bytes String containing the bytes of the file.
-- @return a string containing the dataurl.
dataurl.create(mimetype, bytes)
```

```
-- example usage
```

```
local data = dataurl.create("text/plain", "my text here")
```

```
-- "data:text/plain;charset=utf-8;base64,bXkgdGV4dCB0ZXJl"
```


4.4.9. Blob

Biblioteca utilizada para obtenção dos bytes de um arquivo a partir de seu caminho.

Atalhos

- Load

Métodos

load

Lua



```
-- @param string representando o caminho do arquivo
-- @return bytes do arquivo.
blob.load("./caminho_da_imagem.png")
```

Lua



```
-- example usage:

local image_bytes = blob.load("./my_image_path.png")

-- image_bytes = "000000"
```

4.4.10. Form

Biblioteca utilizada para converter bytes em um array de bytes formatado como multipart/form-data, que é o padrão para envio de arquivos e dados em requisições HTTP.

Atalhos

- Encode

Métodos

Encode

Lua



```
-- @param fields é uma table contendo os campos do formulário. Nessa table po
--           As strings são tratadas como campos de texto e as tabelas de
--           Opcionalmente, pode-se incluir "filename" e "mime" para arqu
-- @return Uma table contendo "body" e "content_type" (o cabeçalho Content-T
encode(fields)
```

```
-- example usage:

local form_data = form.encode({
  language = "pt",
  audio_file = {
    bytes = "conteúdo do arquivo em bytes",
    filename = "meuarquivo.mp3",
    mime = "audio/mpeg"
  }
})

print(form_data.body)          -- Conteúdo codificado do formulário para ser e
print(form_data.content_type) -- Content-Type para ser configurado no HEADERS
```

4.4.11. HMAC

Utilizado para gerar tokens HMAC com os algoritmos de criptografia SHA 256, SHA 512 ou SHA 1.

Atalhos

- [SHA-512](#)
- [SHA-256](#)
- [SHA-1](#)

Métodos

New SHA 256

Lua



```
-- @param secret The secret used to create the HMAC.  
-- @param input The input used to create the HMAC.  
-- @return bytes that can be transformed into base64 or hex string to be used  
hmac["SHA256"].new(secret, input)
```

Lua



```
-- example usage:

local my_hmac = hmac["SHA256"].new("secret", "input")

local hmac_base64 = base64.encode(my_hmac)
-- jYmF0Et6vTLLqjd5o9qgGeDSaaIq7BWvjnKW9wLMaMY=

local hmac_hex = x:gsub(".", function(s)
    return string.format("%02x", string.byte(s))
end)
-- 8d8985d04b7abd32cbaa3779a3daa019e0d269a22aec15af8e7296f702cc68c6
```

New SHA 512

Lua



```
-- @param secret The secret used to create the HMAC.
-- @param input The input used to create the HMAC.
-- @return bytes that can be transformed into base64 on hex string to be used
hmac["SHA512"].new(secret, input)
```

Lua



```
-- example usage:

local my_hmac = hmac["SHA512"].new("secret", "input")

local hmac_base64 = base64.encode(my_hmac)
-- Ksle03F+BCxwZKX6fDGCMM022F4G+P+Dc9BMoX42Fingn0a38VH/0Co/SMWxkSFEbkXCWI8P8d

local hmac_hex = x:gsub(".", function(s)
    return string.format("%02x", string.byte(s))
end)
-- 2ac95ed3717e042c7064a5fa7c318230cd36d85e06f8ff8373d04ca17e361629e09f46b7f1
```

New SHA 1

Lua



```
-- @param secret The secret used to create the HMAC.  
-- @param input The input used to create the HMAC.  
-- @return bytes that can be transformed into base64 on hex string to be used  
hmac["SHA1"].new(secret, input)
```

Lua



```
-- example usage:  
  
local my_sha1_hmac = hmac["SHA1"].new("secret", "input")
```

4.4.12. Functional

Métodos utilizados para programação funcional.

Atalhos

- Map
 - Filter
 - Reduce
 - Windows
-

Métodos

Map

Cria uma nova tabela preenchida com os resultados da chamada de uma função fornecida em cada elemento da tabela de chamada.

Lua



```
-- @param table Tabela que será usada na função.  
-- @param callbackFn Uma função a ser executada para cada elemento na tabela.  
-- @return table Uma nova tabela com os elementos mapeados  
table.map(table, callbackFn)
```

Lua



```
-- example usage:

local t = { 10, 20, 30}

local result = table.filter(t, function(x) return x * 2 end)

-- {20, 40, 60}
```

Filter

Cria uma cópia de uma parte de uma determinada matriz, filtrada apenas para os elementos da matriz fornecida que passam no teste implementado pela função fornecida.

Lua



```
-- @param table Tabela que será usada na função.
-- @param callbackFn Uma função a ser executada para cada elemento do array.
-- @return table Uma nova tabela com os elementos mapeados
table.filter(table, callbackFn)
```

Lua



```
-- example usage:

local t = { 90, 19, 16, 14, 1, 2, 3, 4, 5, 6, 7, 8 }

local result = table.filter(t, function(x) return x > 5 end)

-- {90, 19, 16, 14, 6, 7, 8}
```

Reduce

O método reduce() executa uma função de retorno de chamada "reduzidor" fornecida pelo usuário em cada elemento do array, em ordem, passando o valor de retorno do cálculo no elemento anterior. O resultado final da execução do reduzidor em todos os elementos do array é um único valor.


```
table.reduce(table, callbackFn(accumulator, value), initialValue)
```

- **table** Tabela que será usada na função.
- **callbackFn** Uma função a ser executada para cada elemento de uma tabela. Seu valor de retorno se torna o valor do parâmetro accumulator na próxima invocação de callbackFn. Para a última invocação, o valor de retorno se torna o valor de retorno de reduce(). A função é chamada com os seguintes argumentos:
 - **accumulator** O valor resultante da chamada anterior para callbackFn. Na primeira chamada, seu valor é initialValue se o último for especificado; caso contrário, seu valor é `table[1]`.
 - **value** O valor do elemento atual. Na primeira chamada, seu valor é `table[1]` se initialValue for especificado; caso contrário, seu valor é `table[2]`.
- **initialValue** Um valor para o qual o acumulador é inicializado na primeira vez que o retorno de chamada é chamado. Se initialValue for especificado, callbackFn inicia a execução com o primeiro valor no array como currentValue. Se initialValue não for especificado, o acumulador é inicializado com o primeiro valor da tabela e callbackFn inicia a execução com o segundo valor na tabela como currentValue.

Valor de retorno

O valor resultante da execução da função de retorno de chamada "redução" até a conclusão em todo o array.

OBSERVAÇÃO: Na linguagem de programação Lua, o índice de posição de um elemento em uma tabela (Array) começa em 1.

Lua



```
-- example usage:

local t = { 2, 3, 4, 5 }

local result = table.reduce(t, function(accum, value) return accum * value end

-- 240
```

Windows

Cria janelas contínuas de tamanho `size` de uma tabela.

Lua



```
-- @param table Tabela que será usada na função.
-- @param size Tamanho da janela
-- @return table Uma nova tabela com todas as janelas
table.windows(table, size)
```

Lua



```
-- example usage:

local t = { 1, 2, 3, 4 }

local result = table.windows(t, 3)

--[[
result:

{
    {1, 2, 3},
    {2, 3, 4}
}

]]--
```


4.4.13. Html to PDF

Utilizado para criar PDF através de HTML.

Atalhos

- From string
 - From URL
-

Métodos

From string

Lua



```
-- @param html The string containing the HTML.  
-- @returns The bytes of the generated PDF  
htmltopdf.from_string(html)
```

Lua



```
-- example usage:  
  
local html = "<html><body><h1>Meu HTML em PDF</h1></body></html>"  
  
local bytes = htmltopdf.from_string(html)
```

From url

Lua



```
-- @param url The url to get the HTML
-- @returns The bytes of the generated PDF
htmltopdf.from_url(url)
```

Lua



```
-- example usage:

local bytes = htmltopdf.from_url("https://www.google.com/")
```

4.4.14. HTTP

Utilizada para fazer requisições utilizando o protocolo HTTP.

Atalhos

- Get
 - Post
 - Put
 - Delete
 - Patch
-

Métodos

GET

Lua



```
-- @param url String containing the url.  
-- @param @optional headers Table containing the headers of the request.  
-- @param @optional body String containing the JSON encoded body.  
-- @return a table containing headers, status, text.  
http.get(url, headers, body)
```

```
-- example usage:

local request = http.get("https://httpbin.org/get")

local body = request.text
local headers = request.headers
local status = request.status

--[[
request value:

{
  headers = {
    ["access-control-allow-credentials"] = "true",
    ["access-control-allow-origin"] = "*",
    connection = "keep-alive",
    ["content-length"] = "222",
    ["content-type"] = "application/json",
    date = "Fri, 27 Dec 2024 20:55:39 GMT",
    server = "gunicorn/19.9.0"
  },
  status = 200,
  text = '{\n  "args": {}, \n  "headers": {\n    "Accept": "*/*", \n    "Host'
```

POST

```
-- @param url String containing the url.
-- @param @optional headers Table containing the headers of the request.
-- @param @optional body String containing the JSON encoded body.
-- @return a table containing headers, status, text.
http.post(url, headers, body)
```

Lua



```
-- example usage:

local request = http.post(
    "https://httpbin.org/post",
    { ["Content-Type"] = "application/json" },
    json.encode({key = "value"})
)

local body = request.text
local headers = request.headers
local status = request.status
```

PUT

Lua



```
-- @param url String containing the url.
-- @param @optional headers Table containing the headers of the request.
-- @param @optional body String containing the JSON encoded body.
-- @return a table containing headers, status, text.
http.put(url, headers, body)
```

Lua



```
-- example usage:

local request = http.put(
    "https://httpbin.org/put",
    { ["Content-Type"] = "application/json" },
    nil
)

local body = request.text
local headers = request.headers
local status = request.status
```


DELETE

Lua



```
-- @param url String containing the url.
-- @param @optional headers Table containing the headers of the request.
-- @param @optional body String containing the JSON encoded body.
-- @return a table containing headers, status, text.
http.delete(url, headers, body)
```

Lua



```
-- example usage:

local request = http.delete(
    "https://httpbin.org/delete",
    { ["Content-Type"] = "application/json" },
    nil
)

local body = request.text
local headers = request.headers
local status = request.status
```

PATCH

Lua



```
-- @param url String containing the url.
-- @param @optional headers Table containing the headers of the request.
-- @param @optional body String containing the JSON encoded body.
-- @return a table containing headers, status, text.
http.patch(url, headers, body)
```

```
-- example usage:
```

```
local request = http.patch("https://httpbin.org/patch")
```

```
local body = request.text
```

```
local headers = request.headers
```

```
local status = request.status
```

4.4.15. JSON

A biblioteca JSON fornece funções para codificar e decodificar valores JSON em Lua, permitindo a conversão entre tabelas Lua e strings JSON.

Atalhos

- [Decode](#)
- [Encode](#)

Métodos

Decode

Lua



```
-- @param value The value to be decoded.  
-- @return string containing the decoded value  
json.decode(value)
```

Lua



```
-- example usage:  
  
local decoded = json.decode('{ "xxx": "yyy" }')
```

Encode

Lua



```
-- @param value The value to be encoded.  
-- @param @optional empty_as_array Boolean that defines if an empty table sho  
-- @return string containing the encoded value  
json.encode(value, empty_as_array)
```

Lua



```
-- example usage:  
  
local encoded = json.encode({xxx = "yyy"})
```

4.4.16. Key

Utilizado para guardar valores em memória para utilização em runtime.

Atalhos

- Get
- Set
- Remove

Métodos

Get

Lua



```
-- @param name The name of the key stored  
-- @return the value associated with the key or an empty string if the key do  
get_key(name)
```

Lua



```
-- example usage:  
  
local my_value = get_key("my_key")
```

Set

Lua



```
-- @param key The key name to be stored  
-- @param value The value to be stored  
set_key(key, value)
```

Lua



```
-- example usage:  
  
set_key("my_key", 123)
```

Remove

Lua



```
-- @param name The name of the key to be removed  
-- @return the value the value that was removed or an empty string if the key  
remove_key(name)
```

Lua



```
-- example usage:  
  
local removed_value = remove_key("my_key")
```

4.4.17. Mime

Utilizado para converter extensão de arquivos em MIME type, ou um MIME type em extensão de arquivo.

Atalhos

- De extensão para MIME type
- De MIME type para extensão

Métodos

from_ext

Lua



```
-- @param ext Extensão do arquivo que deseja converter para mime type
-- @return string: MIME type correspondente à extensão
mime.from_ext(ext)
```

Lua



```
local mimetype = mime.from_ext("png")
print(mimetype) -- saída: image/png
```

into_ext

```
-- @param mimetype Mimetype utilizado para converter em extensão
-- @return string: Extensão correspondente ao MIME type
mime.into_ext(mimetype)

```lua
local ext = mime.into_ext("application/json")
print(ext) -- saída: json
```



## 4.4.18. String

---

Funções auxiliares adicionais as da Std lib String do lua.

### Atalhos

- Split

### Métodos

#### Split

Lua



```
-- @param str The string to be splited.
-- @param pattern The string to be used as pattern.
-- @returns Table containing the splited strings
string.split(str, pattern)
```

Lua



```
-- example usage:
local my_string = "my string here"

local result = string.split(my_string, " ")

-- {"my", "string", "here"}
```

## 4.4.19. Sleep

---

Utilizado para pausar o processo por determinado tempo.

### Atalhos

- Secs
  - Millis
  - Micros
  - Nanos
- 

### Métodos

#### Secs

Lua



```
-- @param duration The duration in seconds to pause the process.
sleep.secs(duration)
```

Lua



```
-- example usage:

sleep.secs(1)
```

#### Millis

Lua



```
-- @param duration The duration in milliseconds to pause the process.
sleep.millis(duration)
```

Lua



```
-- example usage:

sleep.millis(1000)
```

## Micros

Lua



```
-- @param duration The duration in microseconds to pause the process.
sleep.micros(duration)
```

Lua



```
-- example usage:

sleep.micros(1000000)
```

## Nanos

Lua



```
-- @param duration The duration in nanoseconds to pause the process.
sleep.nanos(duration)
```

```
-- example usage:
```

```
sleep.nanos(1000000000)
```

## 4.4.20. Task

---

Módulo utilizado para criar tasks assíncronas.

### Atalhos

- Spawn
- 

### Métodos

#### spawn

Ao criar um **loop infinito** dentro de uma task assíncrona, é necessário **inserir intervalos de tempo** para que outras tasks concorrentes possam ser executadas. O sistema de **servidor e pipelines** é considerado uma **task concorrente**, portanto, uma **task mal estruturada pode bloqueá-los**.

```
-- @param fn: function
-- Executa a função passada como argumento em uma nova task.
task.spawn(fn)

```lua
-- example usage:

function foo()
    while true do
        print("Executando em uma task assíncrona!")
        sleep.secs(5) -- pausa de 5 segundos inserida para permitir a execução
    end
end

task.spawn(foo)

print("Essa mensagem será exibida antes ou depois da task, dependendo do agente")
```

4.4.21. Time

Biblioteca para parsear tempo

Atalhos

- Unix
 - Parse
-

Métodos

Unix

Lua



```
-- example usage:  
  
time.unix()  
  
-- 1751394335
```

Parse

Lua



```
-- @param timestamp The timestamp in integer to be parsing  
time.parse(timestamp)
```

```
-- example usage:
```

```
time.parse(1751394335)
```

```
-- {  
  -- "hour": 18,  
  -- "year": 2025,  
  -- "month": 7,  
  -- "seconds": 35,  
  -- "month0": 6,  
  -- "minutes": 25,  
  -- "weekday0": 2,  
  -- "day": 1  
-- }
```


5. Brokers

5.1. Botmaker

A especificação abaixo serve como manual para a utilização do broker **Botmaker**.

1. Criar uma conta

Através do link abaixo, se inscreva na Botmaker.

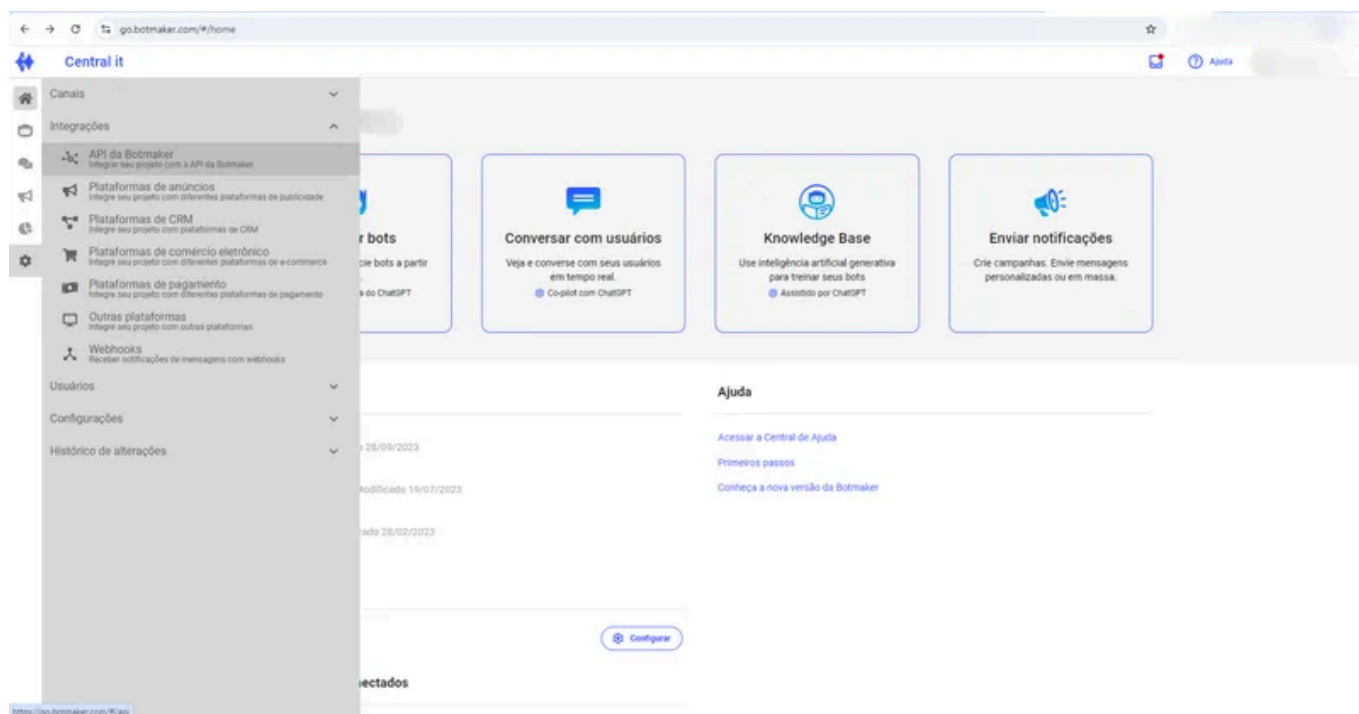
link de acesso

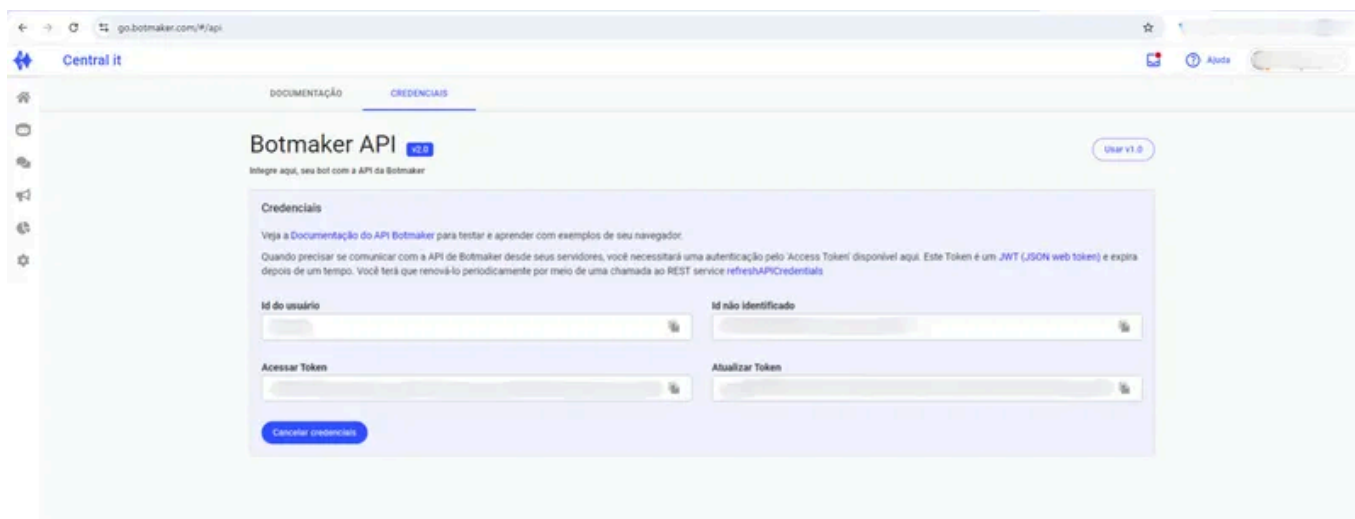
Não é possível criar uma conta de pessoa física. Dessa forma, é necessário criar uma empresarial que possua um número de whatsapp.

A screenshot of the "DADOS DA SUA EMPRESA" (Company Data) registration form. The form includes the following fields: "Em qual empresa você trabalha?" (In which company do you work?) with a text input; "Qual é o website da sua empresa?" (What is your company's website?) with a text input; "Em que país sua empresa está registrada?" (In which country is your company registered?) with a dropdown menu showing "Brasil"; and "Número de celular para contato" (Cell phone number for contact) with a dropdown for the country code (+55) and a text input for the number. A "CONTINUAR" (Continue) button is at the bottom right.

2. Obtendo as credenciais da Botmaker API

Para obter o **ID do usuário** e o **token** acesse no menu lateral no icone de "engrenagem" a opção Integrações > API da Botmaker. Na parte superior, na aba credenciais.





O ID do Canal é composto da seguinte forma: id_do_usuario-whatsapp-numero_do_canal

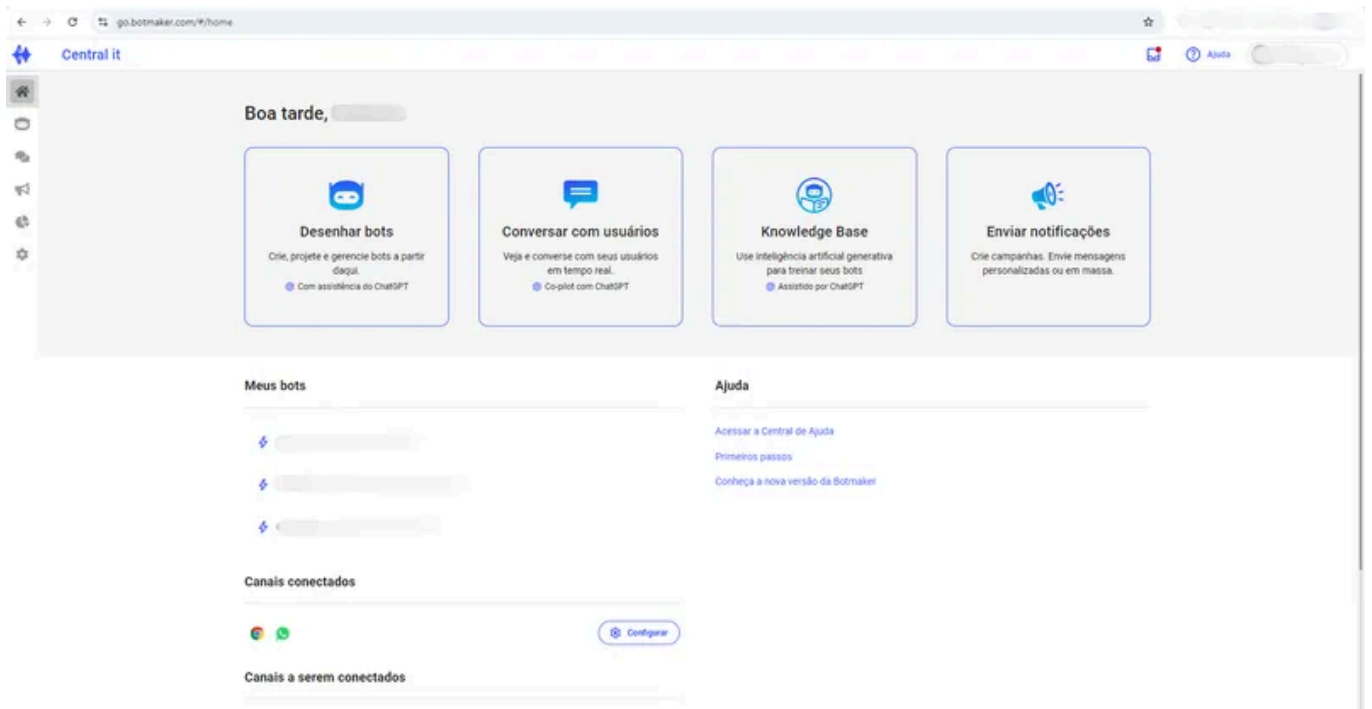
Text



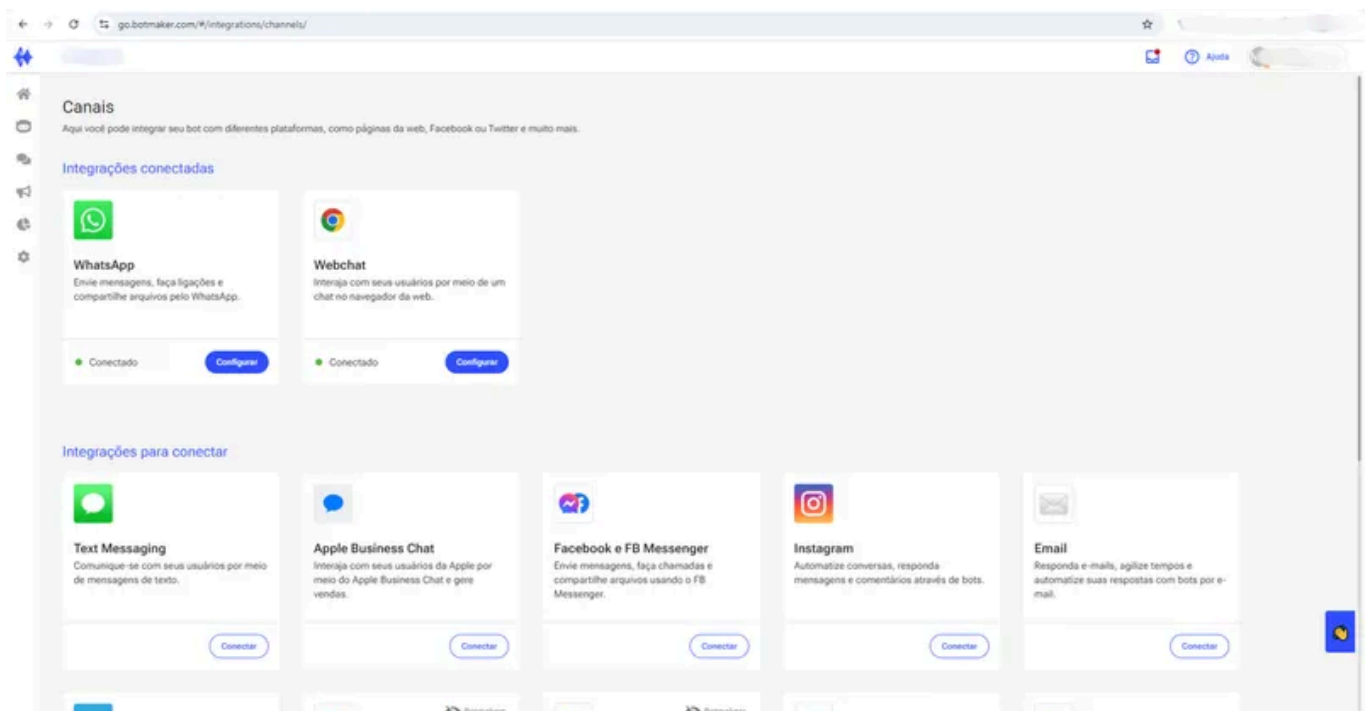
Exemplo: centralit-whatsapp-556133333333

3. Configurando um webhook

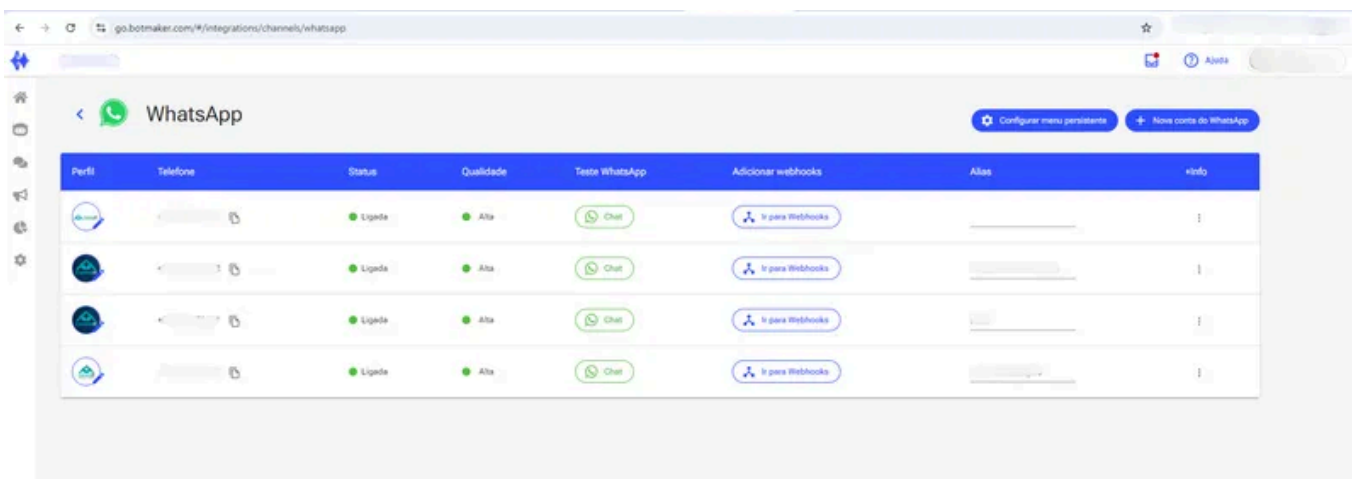
Para configurar o webhook partiremos da tela "home", e na seção "Canais conectados" clique em configurar.



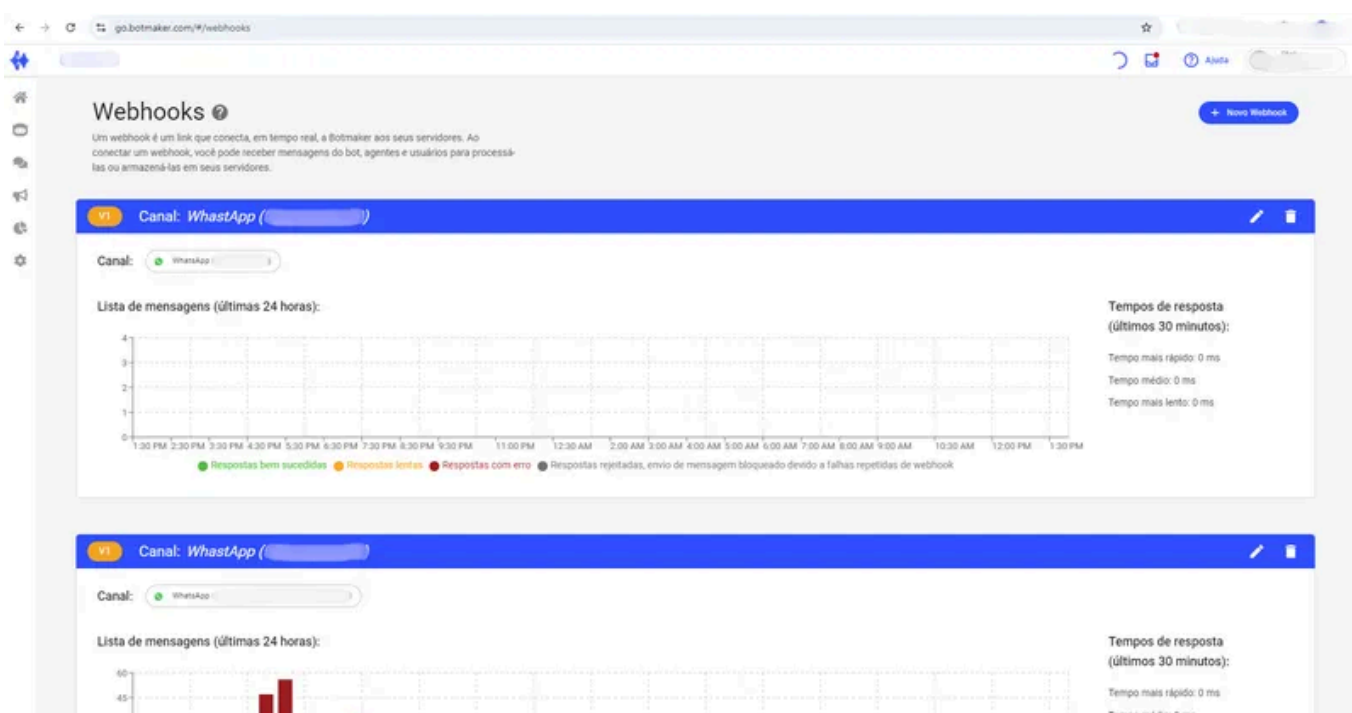
Na seção das Integrações Conectadas clique em configurar dentro do card do whatsapp.



Clique no botão "ir para webhook".

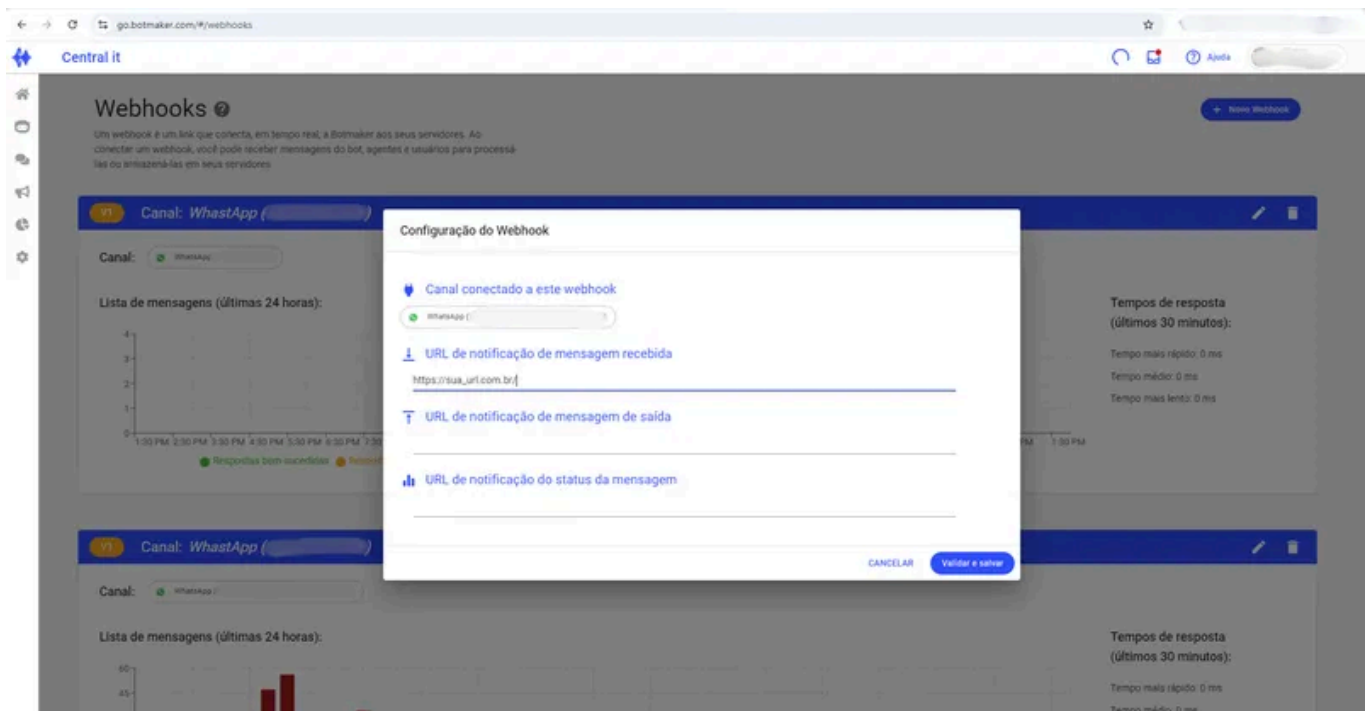


Clique no ícone "lápis" para editar o webhook selecionado.



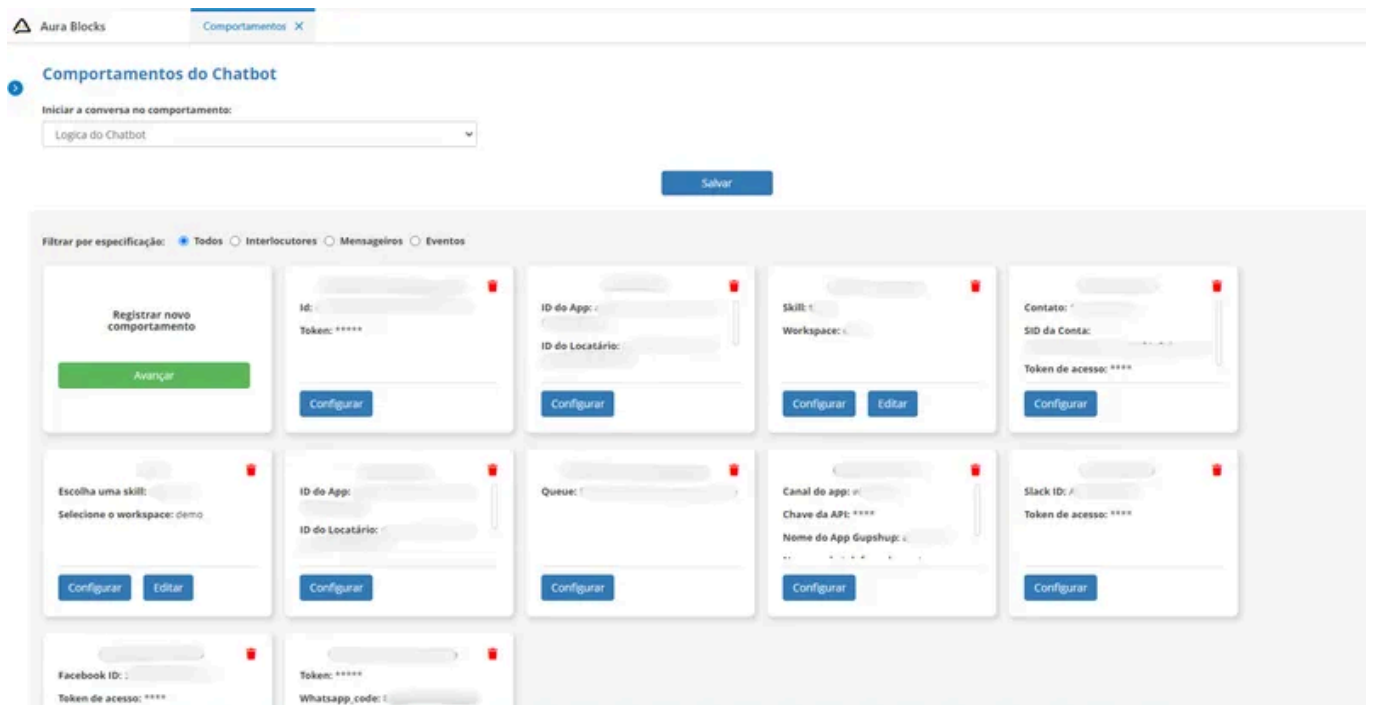
A url do webhook é composta pela url
https://{endereço_do_lunaris}/post/botmaker/receive.

Coloque a url: **https://{endereço_do_lunaris}/post/botmaker/receive** no campo "URL de notificação de mensagem recebida" e clique no botão "Validar e salvar".



4. Configurando a Botmaker dentro da Aura

Na tela de comportamentos dentro da Aura vamos registrar um novo comportamento clicando no botão verde "avançar".



Para registrar esse novo comportamento, primeiro coloque o nome que deseja para ele, selecione o manifesto "Botmaker" e clique em avançar.

Aura Blocks Comportamentos X

Registrar novo comportamento

Nome do Comportamento:

Comportamento Botmaker

Selecione o manifesto: *

Botmaker

Ou digite a URI do manifesto: *

Avançar Cancelar

Coloque o texto para os botões, ID do canal e token ambos obtidos no botmaker como apresentado no item 2 deste documento.

Para finalizar clique no botão "concluir".

Aura Blocks Comportamentos X

Registrar novo comportamento

Nome do Comportamento:

Comportamento Botmaker

Selecione o manifesto: *

Botmaker

Ou digite a URI do manifesto: *

Parâmetros ?

Texto para Botões: *

Botão 1

ID do canal *

seu_id_botmaker

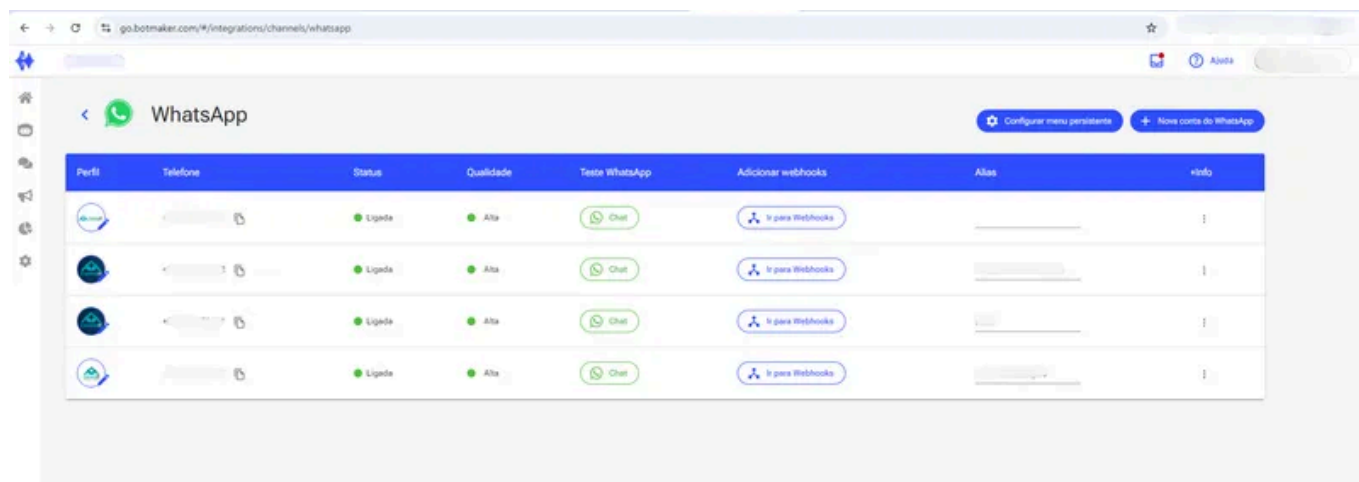
Token de acesso *

XXXXXXXXXXXX

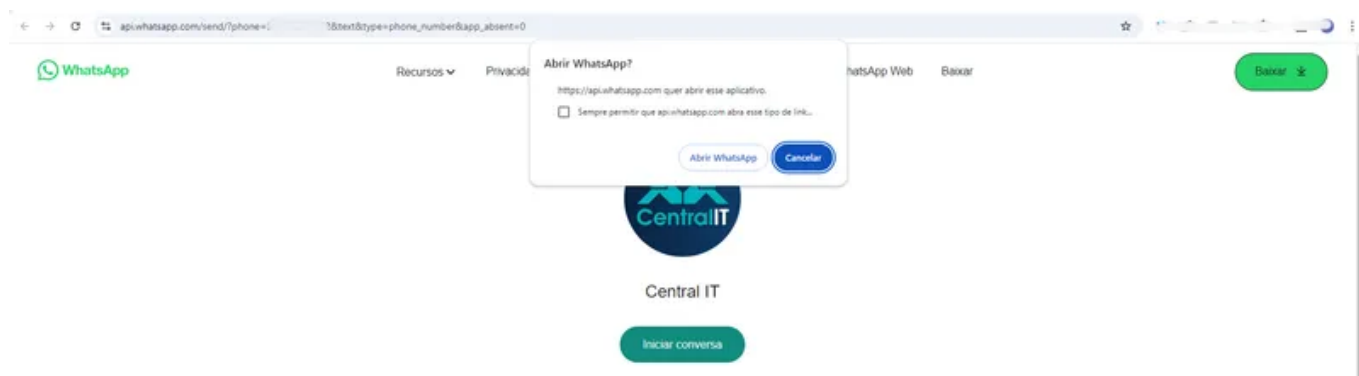
Concluir Cancelar

5. Testando o botmaker

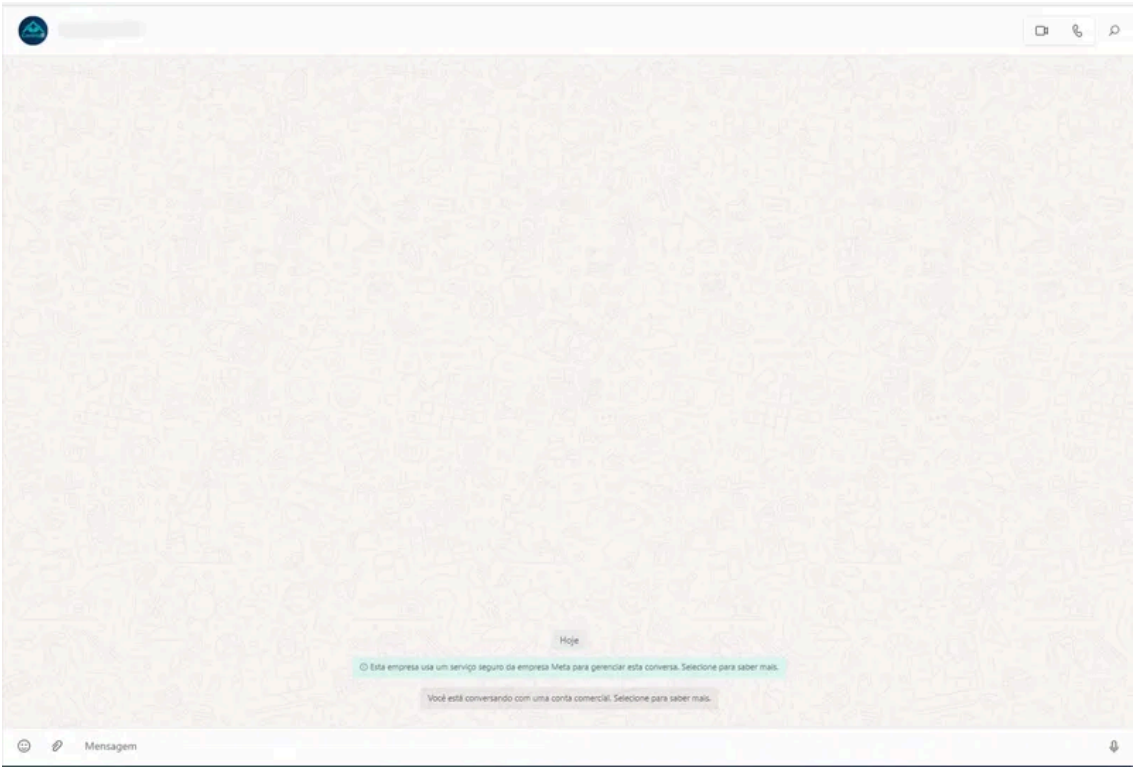
Para fazer os testes, primeiro clique em "chat" e será redirecionado para uma aba do whatsapp.



Clique em "Abrir Whatsapp".



E agora você terá uma nova conversa diretamente com o Botmaker.



5.2. Gupshup

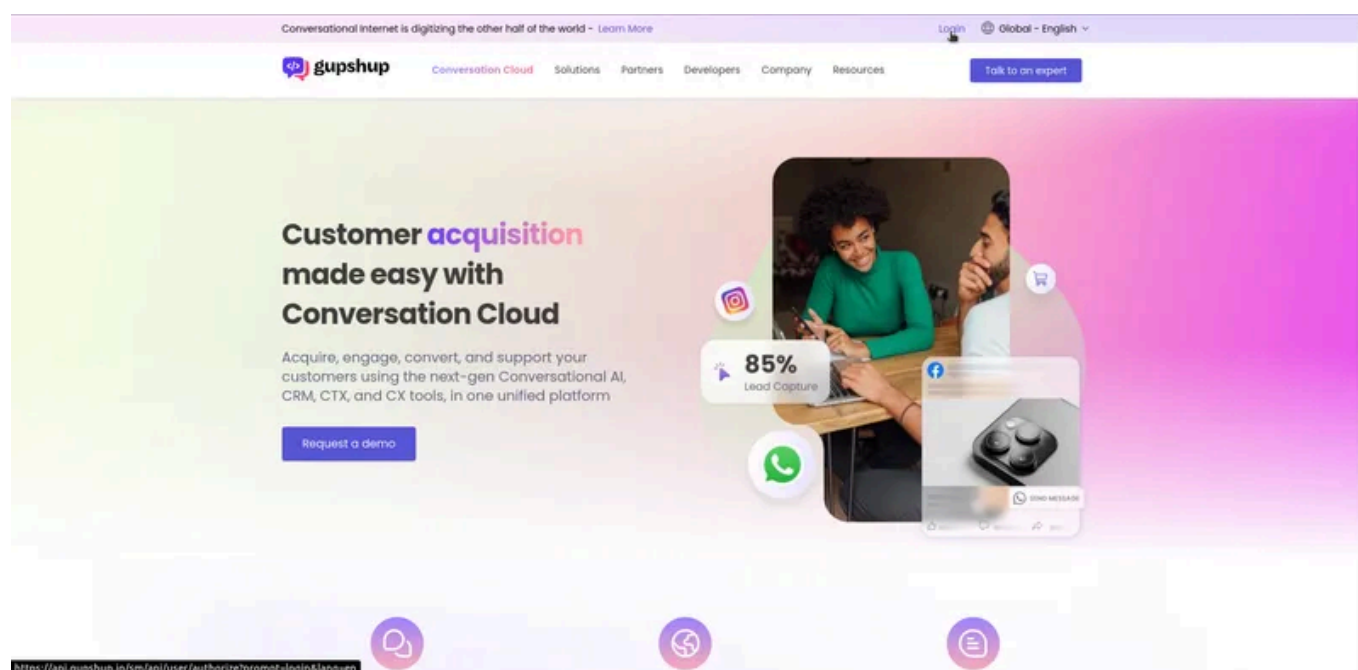
A especificação abaixo serve como manual para a utilização do broker Gupshup.

Configuração

Para cadastrar o broker Gupshup é necessário um número de Whatsapp Busines, o nome do app registrado no Gupshup e uma chave de acesso do app. Para testes o Gupshup oferece um modo Sandbox com um número habilitado a enviar e receber mensagens.

Criando uma conta

Basta acessar o Gupshup e acessar a tela de login clicando no botão superior esquerdo.




Criando um app

Para criar um app Gupshup é necessário estar logado e acessar o dashboard e clicar no botão "Create App", nomear o app. Uma vez criado o app basta acessá-lo clicando em "Overview" ao lado direito do nome do app. Lá é possível obter informações do app como sua chave de api e acessar o modo Sandbox. **Para que seja possível receber e enviar**

mensagens de um número é necessário ir na aba "Opt-ins" e habilitar o número de Whatsapp.

For all kinds of support, please write to devsupport@gupshup.io

English

WhatsAppMy Wallet4,999 USD

DashboardWhatsApp

Dashboard

View how your apps are performing and manage them directly from this dashboard.

+ Create App

Create App


Customer Id : 4000271191

ChangelogAPI docsHelp

Search by App name or Phone number

Show All

Access API




Messaging Limit : NA

Aug 28 2024 | 8:52 AM

Not live

Go Live

Access API



Messaging Limit : NA

Aug 20 2024 | 3:10 PM

Not live

Go Live

For any queries, write to devsupport@gupshup.io with your Customer ID: 4000271191


PrivacySecurityCookiesTermsDL1

Gupshup © 2024. All rights reserved.

Feedback

For all kinds of support, please write to devsupport@gupshup.io

English

WhatsAppMy Wallet4,999 USD

DashboardWhatsApp

Dashboard

View how your apps are performing and manage them directly from this dashboard.

+ Create App

Create App


Customer Id : 4000271191

ChangelogAPI docsHelp

Search by App name or Phone number

Show All

Access API




Messaging Limit : NA

Aug 28 2024 | 8:52 AM

Not live

Go Live

Access API



Messaging Limit : NA

Aug 20 2024 | 3:10 PM

Not live

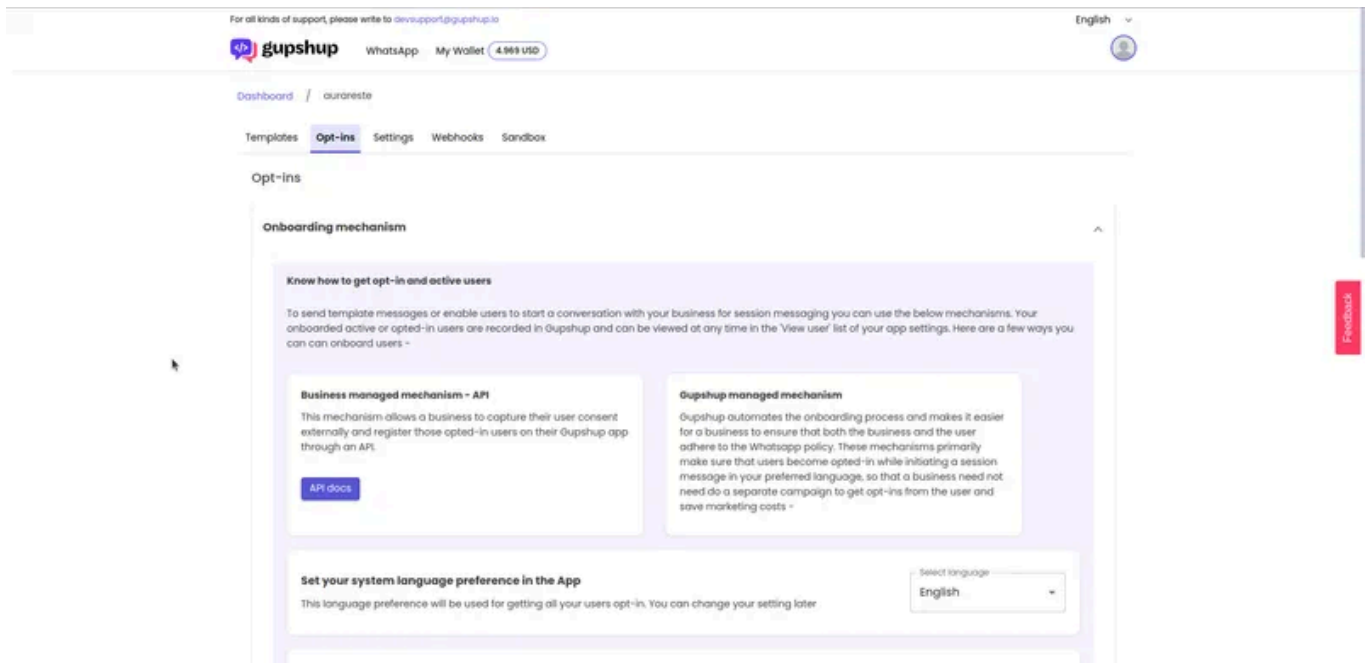
Go Live

For any queries, write to devsupport@gupshup.io with your Customer ID: 4000271191

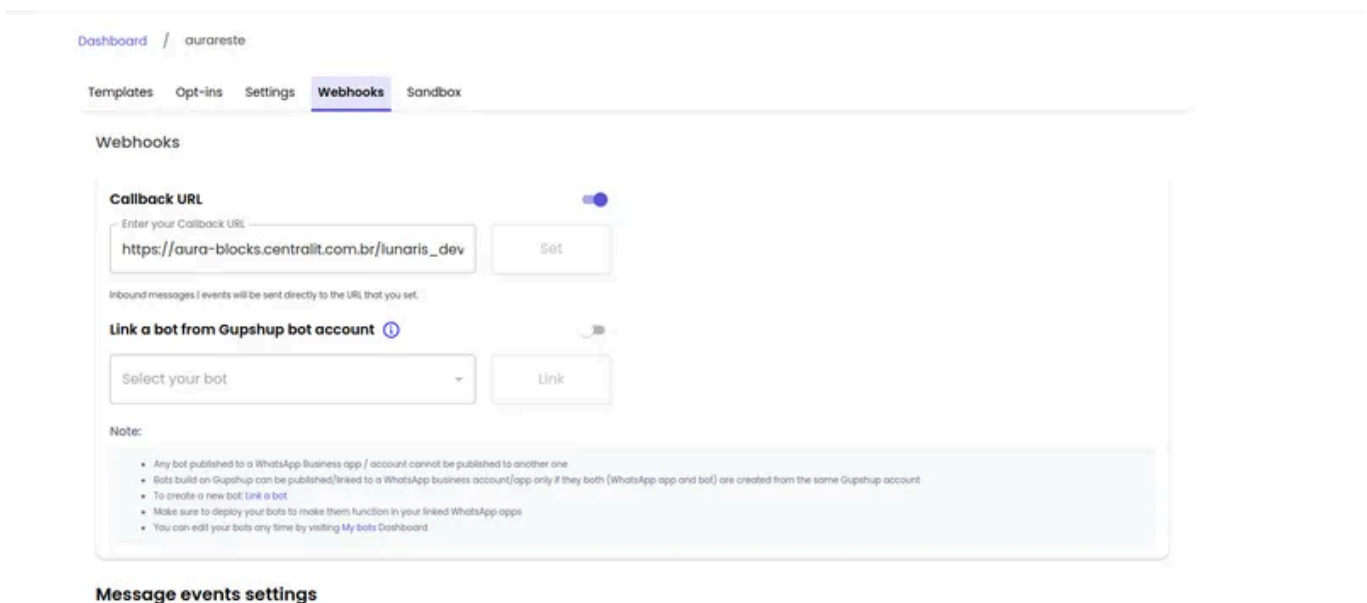
PrivacySecurityCookiesTermsDL1

Gupshup © 2024. All rights reserved.

Feedback



Registrando o webhook



Para cadastrar o endereço webhook basta acessar a aba "Webhook", inserir o endereço do webhook no campo "Callback URL" e clicar em "Set".

Text



`https://{caminho_para_lunaris}/post/{nome_do_plugin}/webhook`

Campos Necessários

Para registrar o broker Gupshup são necessários os seguintes campos:

Text



Número de telefone da conta
Canal **do** app
Nome **do** app gupshup
Chave da API

Número de telefone

Caso esteja usando o modo Sandbox do Gupshup é fornecido um número para testes.

Dashboard / aurareste

Templates Opt-ins Settings Webhooks **Sandbox**

Sandbox

Send message

Test access API

Applicable charges will be deducted from your wallet balance

Send messages to desired recipients

Sender

Sender number

917834811114

Recipient

Country code

India - 91

Recipient number *

Message

Enter message

Receive inbound messages from customers

Inbound Messages | Events

[Read API docs](#)

No inbound messages yet. List will be updated as soon as we have a message from the customers.

Request **Response**

```
curl -X POST https://api.gupshup.io/wa/api/v1/msg \
-H "Cache-Control: no-cache" \
-H "Content-Type: application/x-www-form-urlencoded" \
-H "apikey: bixbjfbxuohw9apifzx0klpsgxdnt43" \
-H "cache-control: no-cache" \
-d "channel=whatsapp&source=917834811114&destination=91&message=%78%22t
ype%22%3A%22text%22%2C%22text%22%3A%22%7D&src.name=aurareste"
```

É possível verificar o número da conta através do menu Sandbox dentro de um app

Caso não esteja usando o modo sandbox utilize o número registrado na conta Gupshup

Canal do APP

O Gupshup possui diversos canais para envio e recebimento de texto por isso é necessário escolher em qual ele irá funcionar.

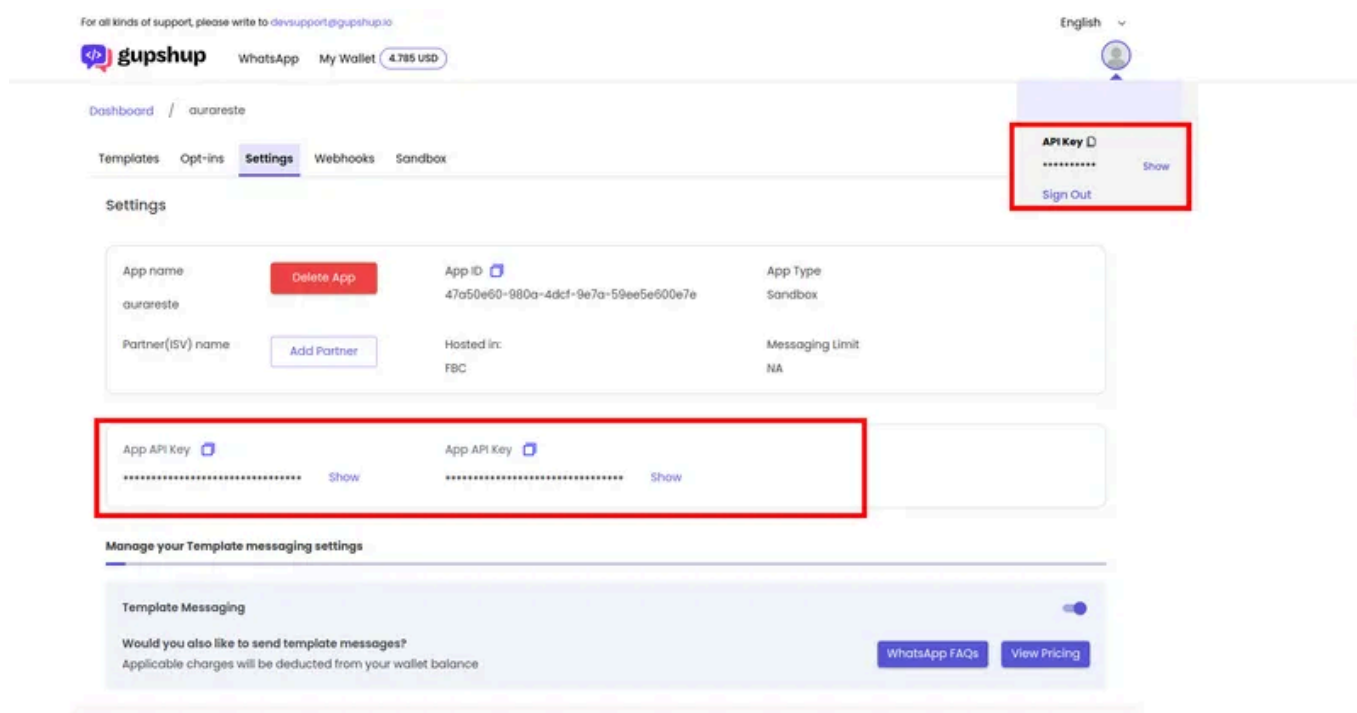
Para utilizar o Whatsapp como canal basta registrar como "whatsapp"

Nome do APP Gupshup

É o nome registrado do APP Gupshup feito nos passos anteriores.

Chave da API

Para obter a chave API basta acessar o menu "Settings" dentro de um APP Gupshup ou clicar no avatar no canto superior direito



5.3. Positus

A especificação abaixo serve como manual para utilização do broker **Postius**.

Configuração

Para cadastrar o broker Positus é necessária copiar a rota de envio de mensagens do Positus e do token api do Positus. Para testes o Positus oferece um modo Sandbox com um número já vinculado para recebimento e envios de mensagens além de uma url específica do sandbox.

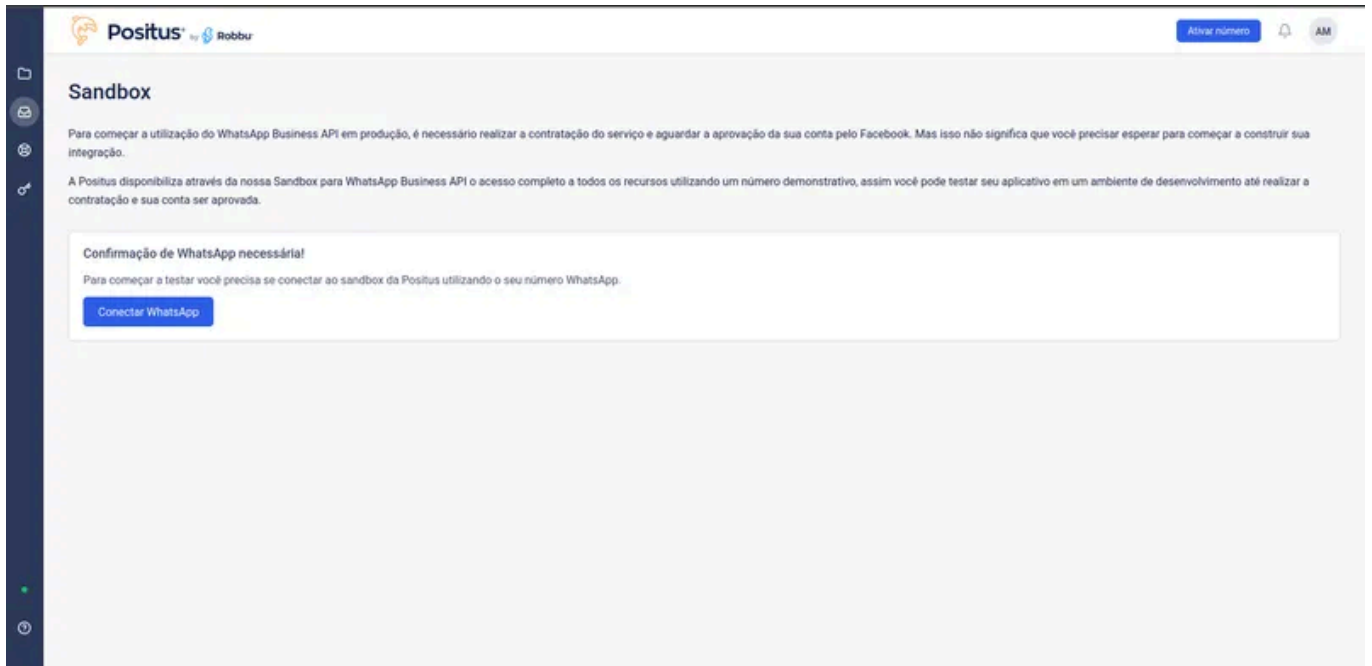
URL exemplo: **<https://api.positus.global/v2/sandbox/whatsapp/numbers/814e1f43-af0b-4ecb-a3ee-719c3481fabf/messages>**

Criando uma conta

Basta acessar o **[link para criação de contas](#)** e acessar o modo **Sandbox**

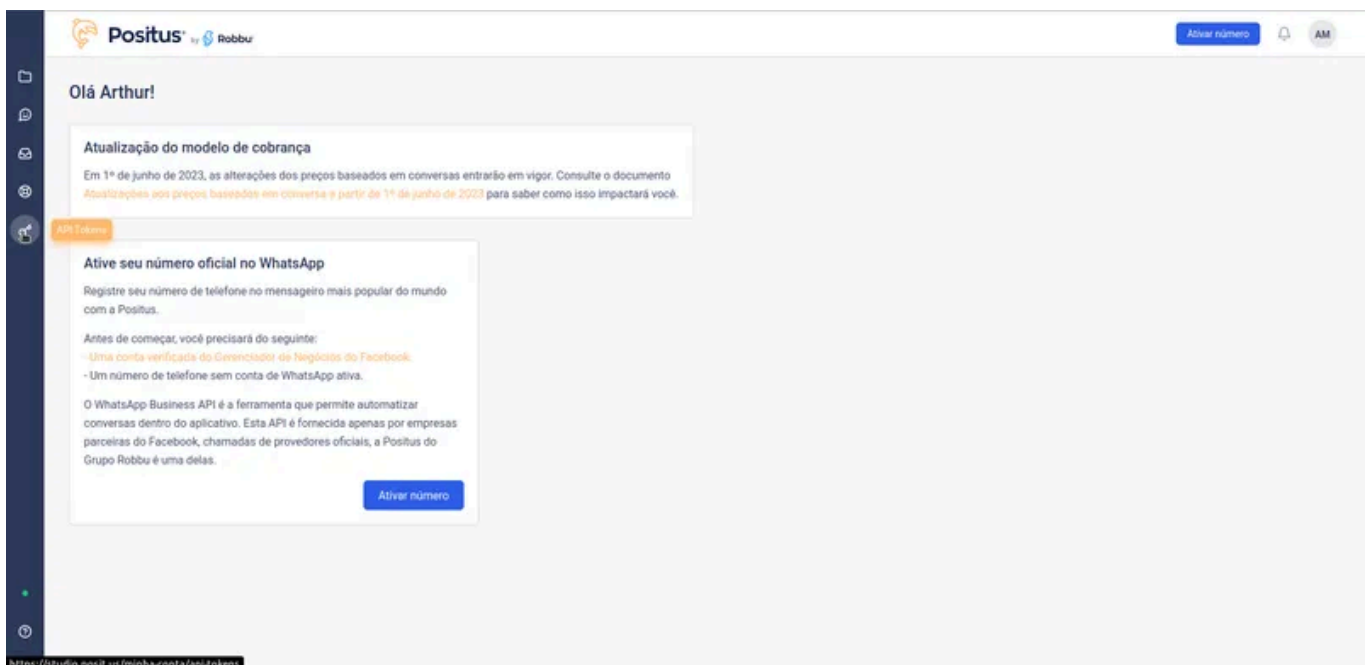
Modo Sandbox

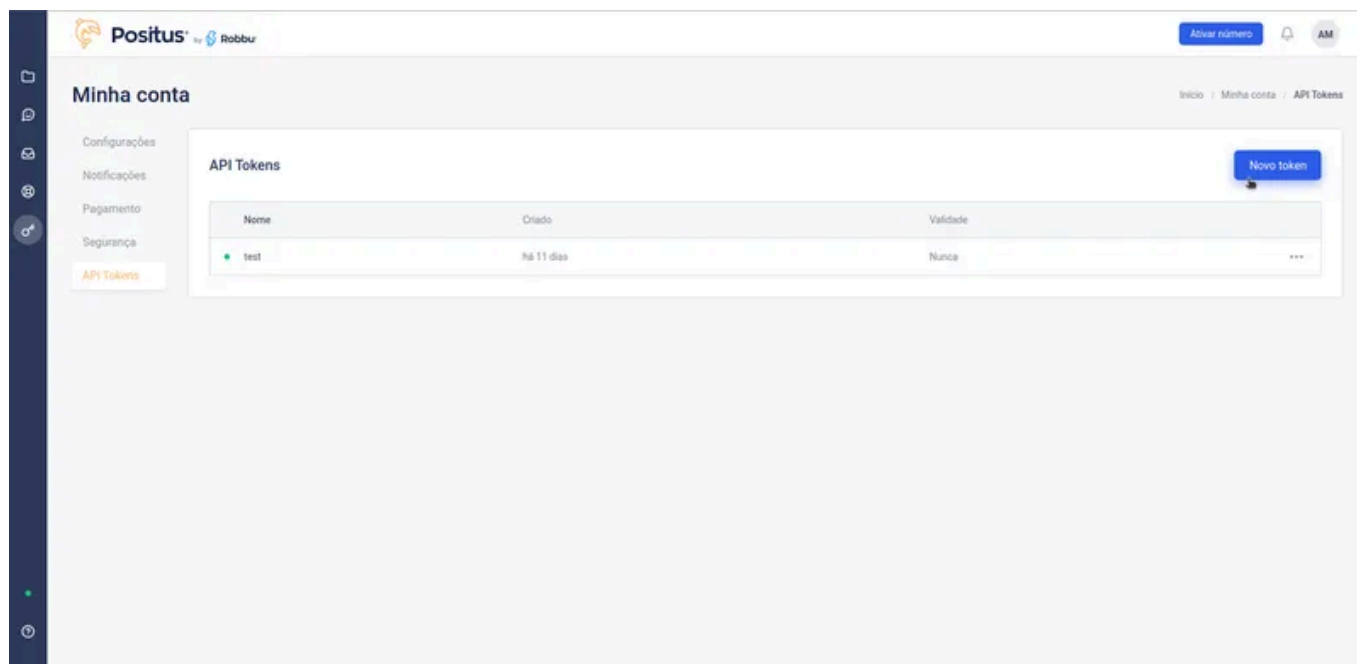
Para acessar o modo sandbox é necessário primeiro habilitar um numero de whatsapp para receber e enviar mensagens a Positus



Obtendo uma chave api

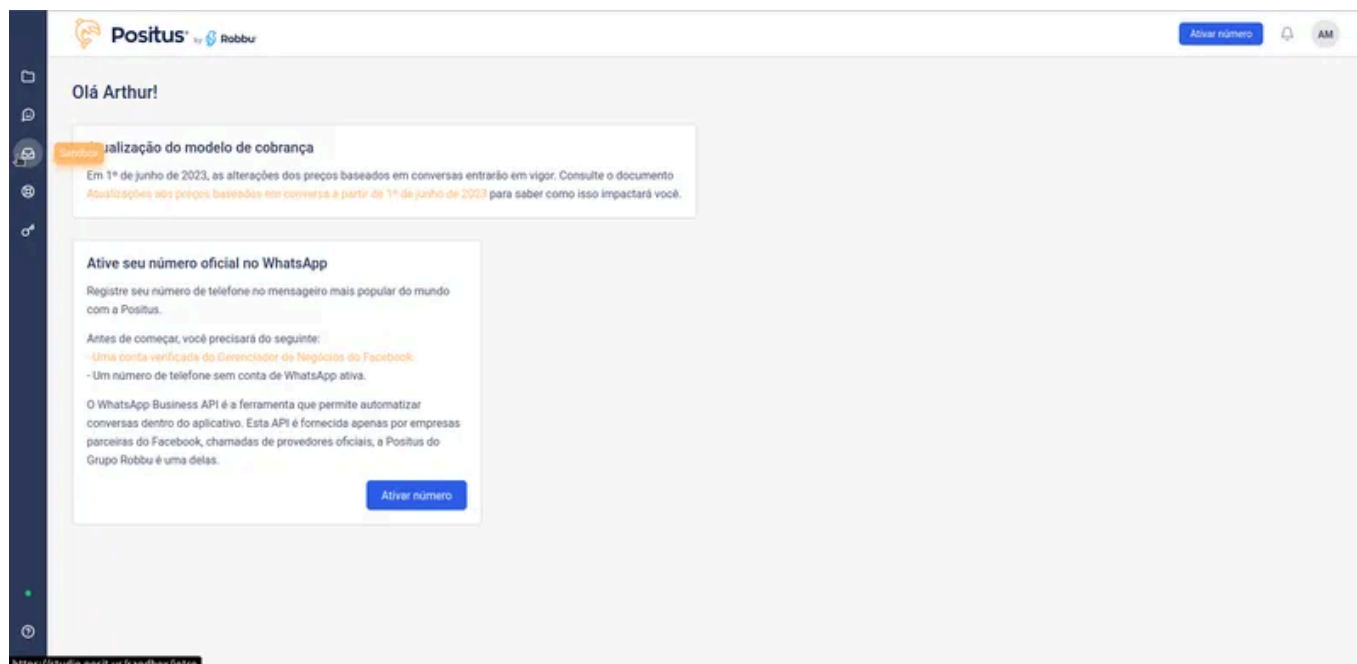
Para obter a chave api basta acessar no menu lateral a aba API Tokens ou acessar o [link](#) e clicar em "Novo token".





Registrando o webhook

Para cadastrar o endereço do webhook basta acessar no menu lateral a aba Sandbox e ir em "Webhook" ou acessar o [link](#). Uma vez registrado o webhook a plataforma irá redirecionar todas as mensagens para lá. **Para o uso do modo Sandbox é necessário habilitar os números que podem enviar ou receber mensagens.** Para isso basta ir na aba "Contatos" e enviar o código indicado para o whatsapp do número fornecido pelo Sandbox ou ler o QR da tela.



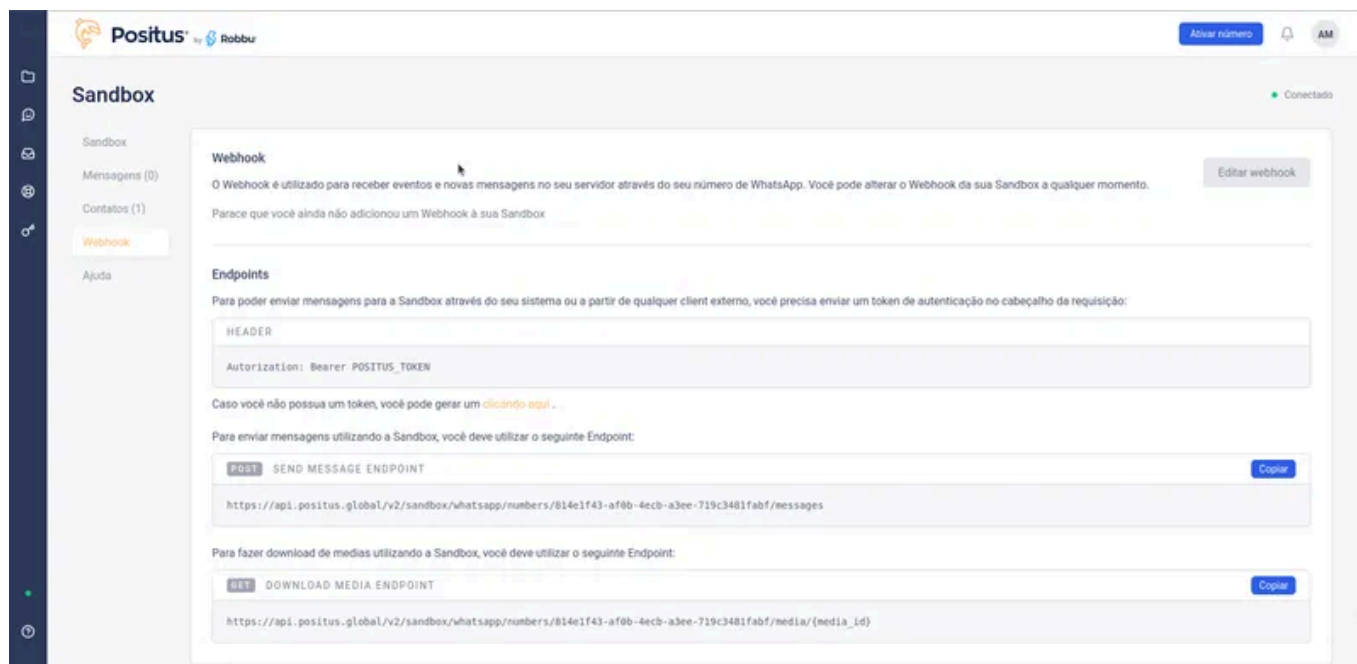
Para o funcionamento do Broker é necessária a devida configuração da rota do Webhook para isso basta clicar em "Editar Webhook" e preencher o campo com o endereço e salvar

Para o registro do webhook da Positus é necessário conter o identificador do workspace como parâmetro da URL:

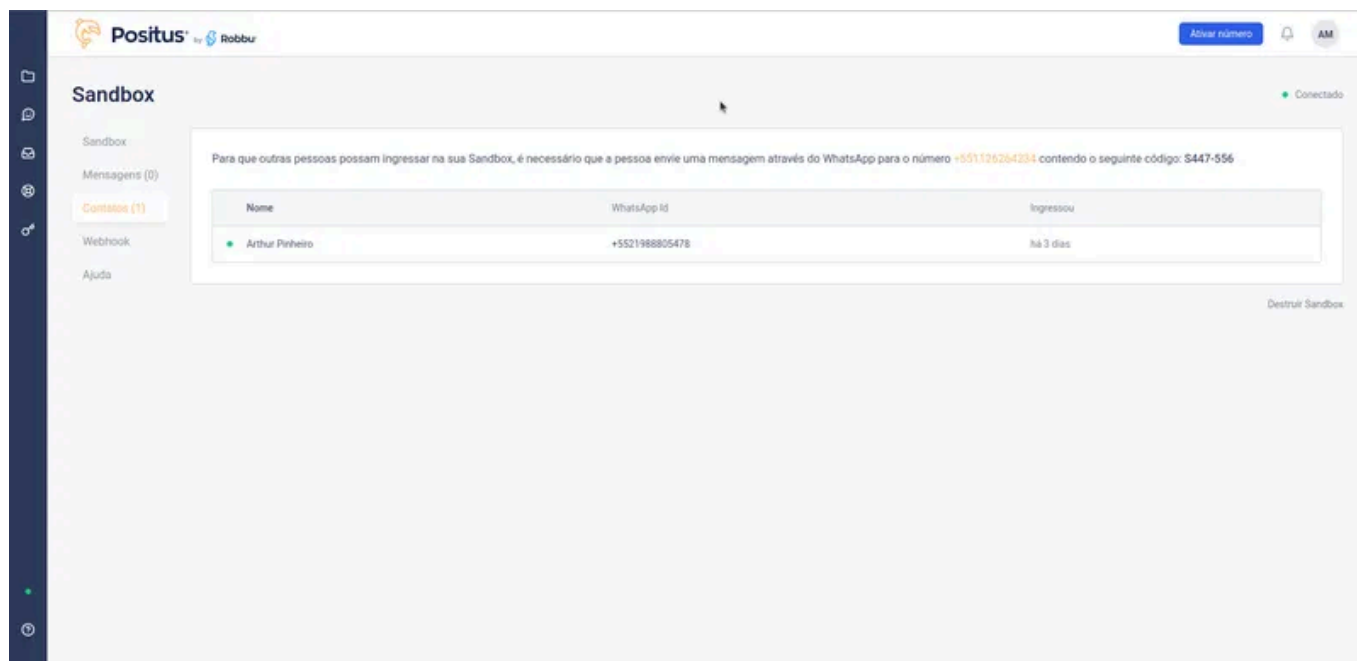
Text



```
https://{caminho_para_lunaris}/post/{nome_do_plugin}/webhook?workspace={ident
```



No menu contatos é possível verificar quais contatos estão habilitados a se comunicar com o Positus



Campos necessários

Para registrar o broker Positus são necessários os seguintes campos:

Código de número do Whatsapp
URL para envio das mensagens
Token de acesso

Código de número do Whatsapp

Todos os números de Whatsapp possuem um código único de identificação.
Para achar o código do seu número específico é preciso verificar na página do Whatsapp Business

No caso do modo Sandbox é possível achar de maneira fácil o código do número já disponibilizado ao entrar no Submenu Webhook

Webhook

Editar webhook

O Webhook é utilizado para receber eventos e novas mensagens no seu servidor através do seu número de WhatsApp. Você pode alterar o Webhook da sua Sandbox a qualquer momento.

Parace que você ainda não adicionou um Webhook à sua Sandbox

Endpoints

Para poder enviar mensagens para a Sandbox através do seu sistema ou a partir de qualquer client externo, você precisa enviar um token de autenticação no cabeçalho da requisição:

HEADER

Autorization: Bearer POSITUS_TOKEN

Caso você não possua um token, você pode gerar um [clikando aqui](#).

Para enviar mensagens utilizando a Sandbox, você deve utilizar o seguinte Endpoint:

POST SEND MESSAGE ENDPOINT

Copiar

`https://api.positus.global/v2/sandbox/whatsapp/number/814e1f43-af0b-4ecb-a3ee-719c3481fabf/messages`

Para fazer download de medias utilizando a Sandbox, você deve utilizar o seguinte Endpoint:

GET DOWNLOAD MEDIA ENDPOINT

Copiar

`https://api.positus.global/v2/sandbox/whatsapp/numbers/814e1f43-af0b-4ecb-a3ee-719c3481fabf/media/{media_id}`

URL para envio das mensagens

Existem duas possibilidades de URL para envio de mensagem

Caso esteja testando o modo Sandbox da Positus é preciso utilizar a seguinte URL:

Text



```
https://api.positus.global/v2/sandbox/whatsapp/numbers/
```

Caso não esteja usando o modo Sandbox é preciso utilizar a seguinte URL:

Text



```
https://api.positus.global/v2/whatsapp/numbers/
```

Token de acesso

Para obter o token de acesso basta executar os passos já descritos para obter a chave da API

5.4. Slack

O broker do Slack segue o padrão de cadastrar webhook para recebimento de mensagens e receber um link para envio de mensagens. A API do slack fornece certos identificadores para o cadastro do broken, como id da conta, token, signing secret e oath do usuário.

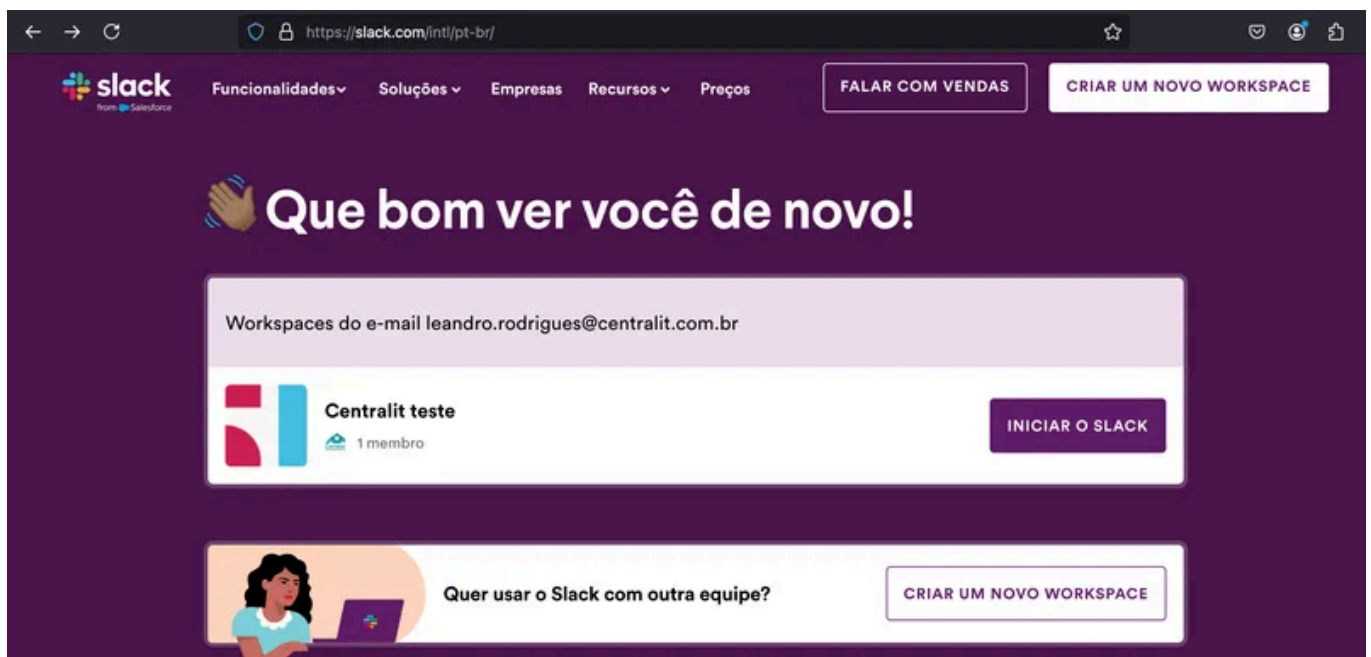
1. Criar uma conta

Através do link abaixo, se inscreva no Slack API

[link de acesso](#)

2. Criar um novo workspace

Workspace é um espaço onde se pode fazer a comunicação entre canais, mensagens diretas etc.

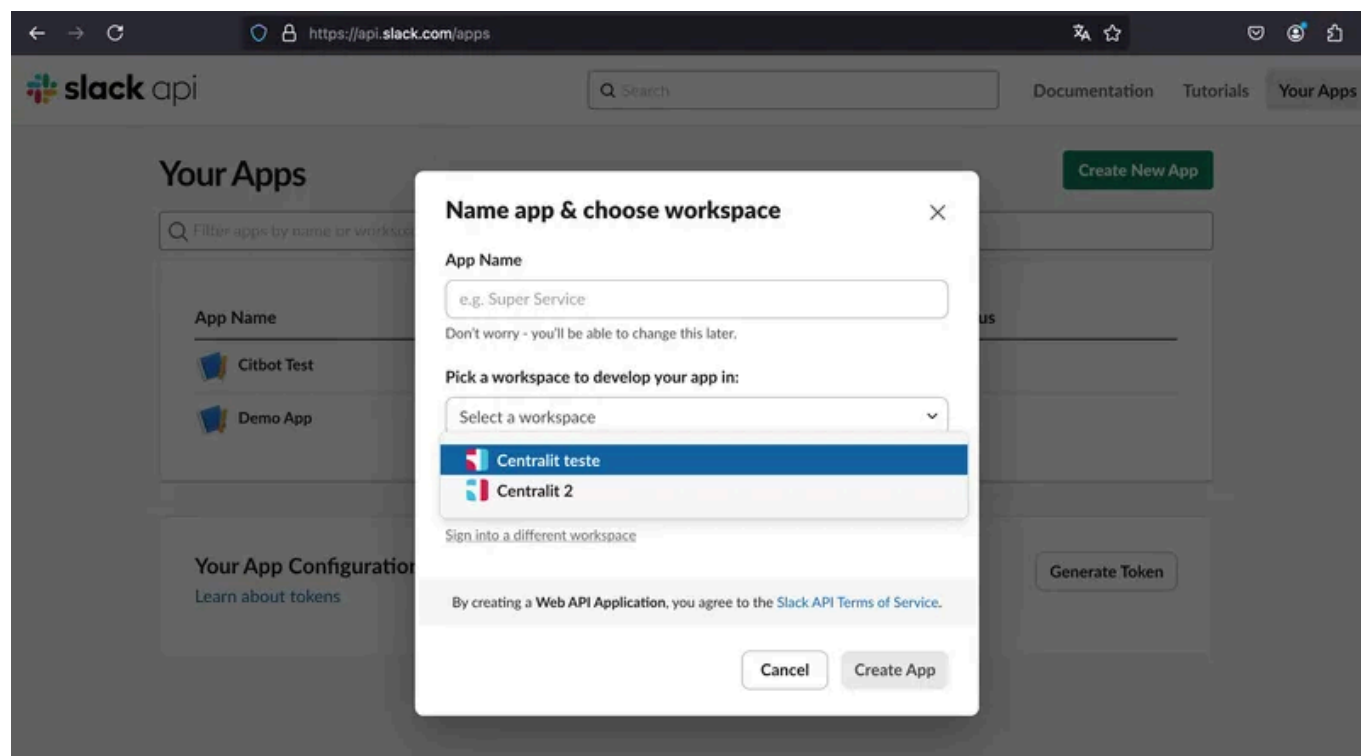


3. Criar um novo aplicativo de desenvolvedor

[link slack aplicativo](#)

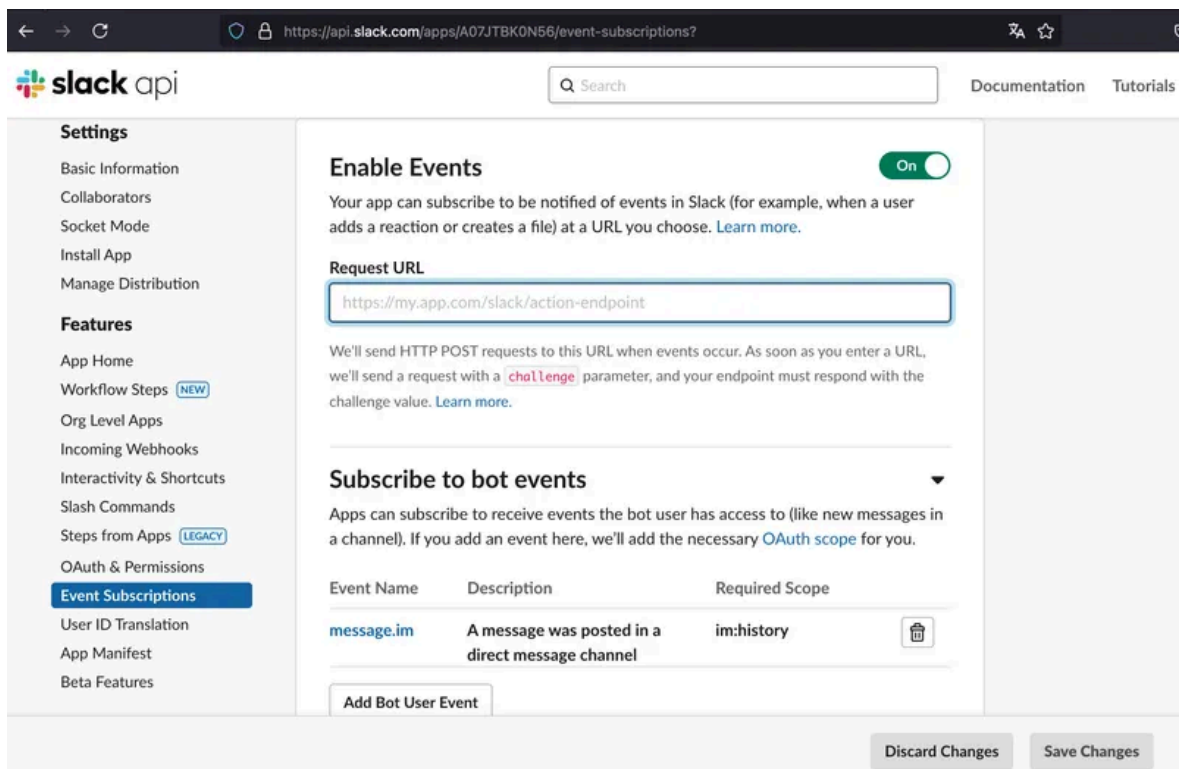
O aplicativo serve para usar e configurar ferramentas da API do slack. Na criação do aplicativo, deve selecionar o workspace criado.

Para criar selecione Seus Aplicativos e Criar Novo Aplicativo.



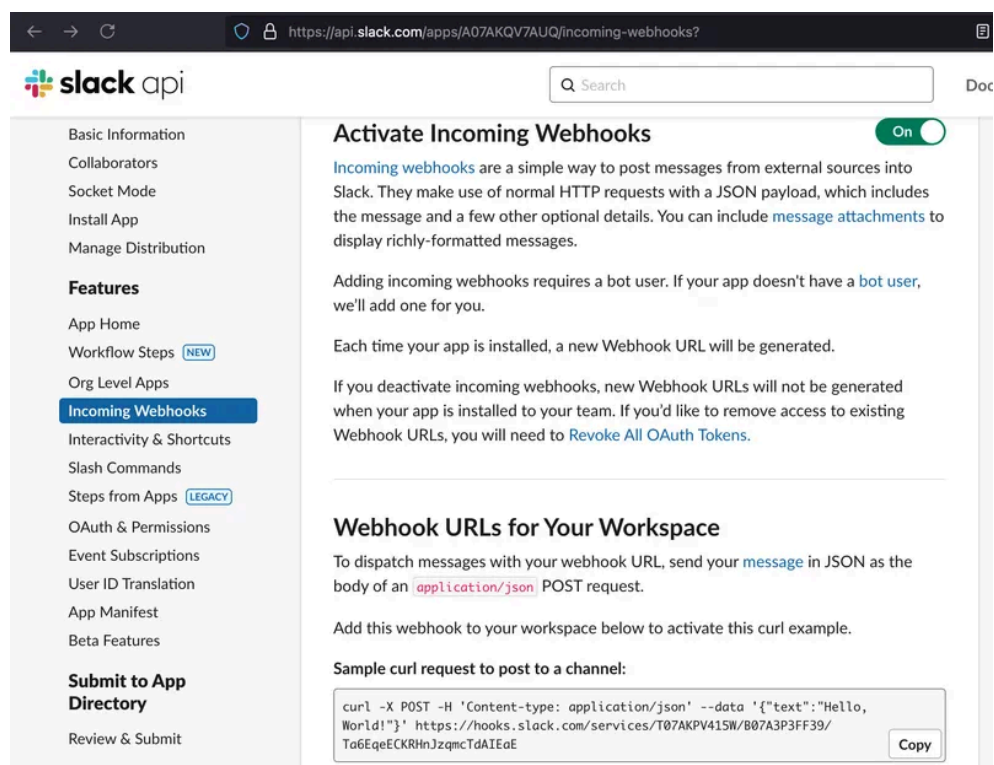
4. Configurar webhook e inscrição em eventos

Na aba de inscrição de eventos, permita os eventos. Depois, cadastre uma url para o webhook **`https://{endereço_do_lunaris}/post/slack/webhook`**. e também se inscreva em evento no recebimento de mensagens (Subscribe to bot events e Subscribe to events on behalf of users).



5. Receber link para envio de mensagem

Na aba de webhook de recebimento, ative tal funcionalidade e clique na opção de Adicionar um Novo Webhook ao Workspace. Na imagem abaixo é mostrado o link em que é possível enviar requisições de envio de mensagem. A url tem o padrão de endereço fixo (<https://hooks.slack.com/services/>) seguido do token, no caso, o token da imagem seria **T07AKPV415W/B07A3P3FF39/Ta6EqeECKRHnJzqmcTdAIEaE**.



6. Ativar Interatividade de Botões

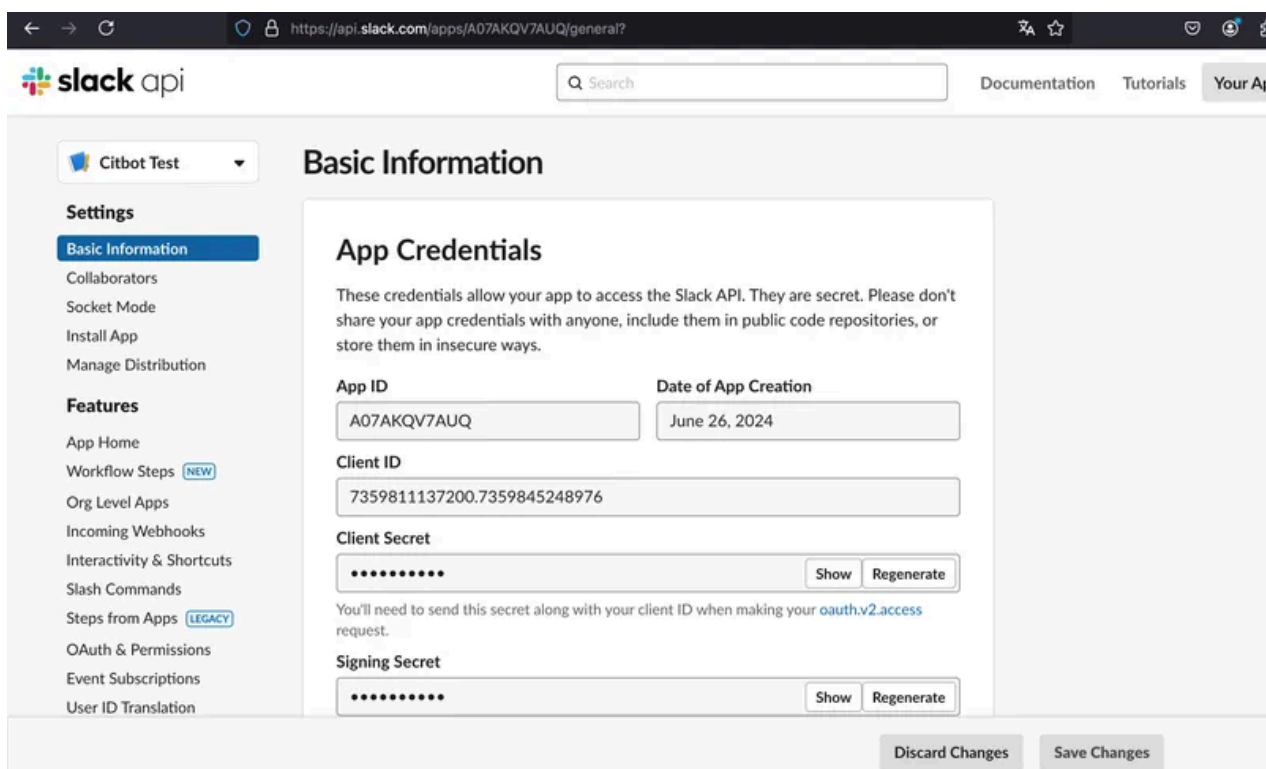
Na aba de Interatividade e Atalhos de Texto, ative a Interatividade e ponha o link do webhook parecido com o passo 4, com endereço

`https://{endereço_do_lunaris}/post/slack/webhook_interaction.`

7. Obter ID da conta

Na aba de informações básicas é mostrado o ID do aplicativo, como no caso, A07AKQV7AUQ.

O campo Signing Secret é a **Chave Secreta** que deve ser cadastrado no AURA BLOCKS.



8. Obter OAuth do Usuário

Nesta página do OAuth como pode ser vista na imagem abaixo, em User Token Scopes ative a opção *im:read*

e *files:read*. Isso permitirá que arquivos e imagens sejam recebidos pelo nosso servidor do broker slack.

Por último guarde o campo, **User OAuth Token** conforme mostra a imagem em que se inicia com "xoxp..."

← → ↻

🔒 https://api.slack.com/apps/A07AKQV7AUQ/oauth?

📄

Citbot Test

▼

Settings

Basic Information

Collaborators

Socket Mode

Install App

Manage Distribution

Features

App Home

Agents & Assistants NEW

Workflow Steps NEW

Org Level Apps

Incoming Webhooks

Interactivity & Shortcuts

Slash Commands

Steps from Apps LEGACY

OAuth & Permissions

Event Subscriptions

User ID Translation

App Manifest

Beta Features

Submit to Slack Marketplace

Review & Submit

OAuth & Permissions

Advanced token security via token rotation

Recommended for developers building on or for security-minded organizations – opting into token rotation allows app tokens to automatically expire after they’re issued within your app code. [View documentation.](#)

⚠ At least one redirect URL needs to be set below before this app can be opted into token rotation

Opt In

OAuth Tokens

These tokens were automatically generated when you installed the app to Centralit teste. You can use these to authenticate your app. [Learn more.](#)

User OAuth Token

xoxp-7359811137200-7330390533814-7348296575906-73378dd1befc5f6c

Copy

Access Level: Workspace

Bot User OAuth Token

xoxb-7359811137200-7337603598002-g7CtIgf4p96Uu7eia1QSpLu

Copy

Access Level: Workspace

Reinstall to Centralit teste

5.5. Facebook Messenger

A criação de um webhook para o Messenger do Facebook exige que se tenha um portfólio empresarial e uma página relacionada a este portfólio.

Feito isso, segue o padrão de cadastrar webhook para recebimento de mensagens e receber um token para envio de mensagens. O identificador principal do facebook também será necessário para cadastro do Broker.

Nota: É obrigatório que primeiro cadastre os dados na AURA BLOCKS para que depois configure o webhook do passo 6.

1. Criar uma conta

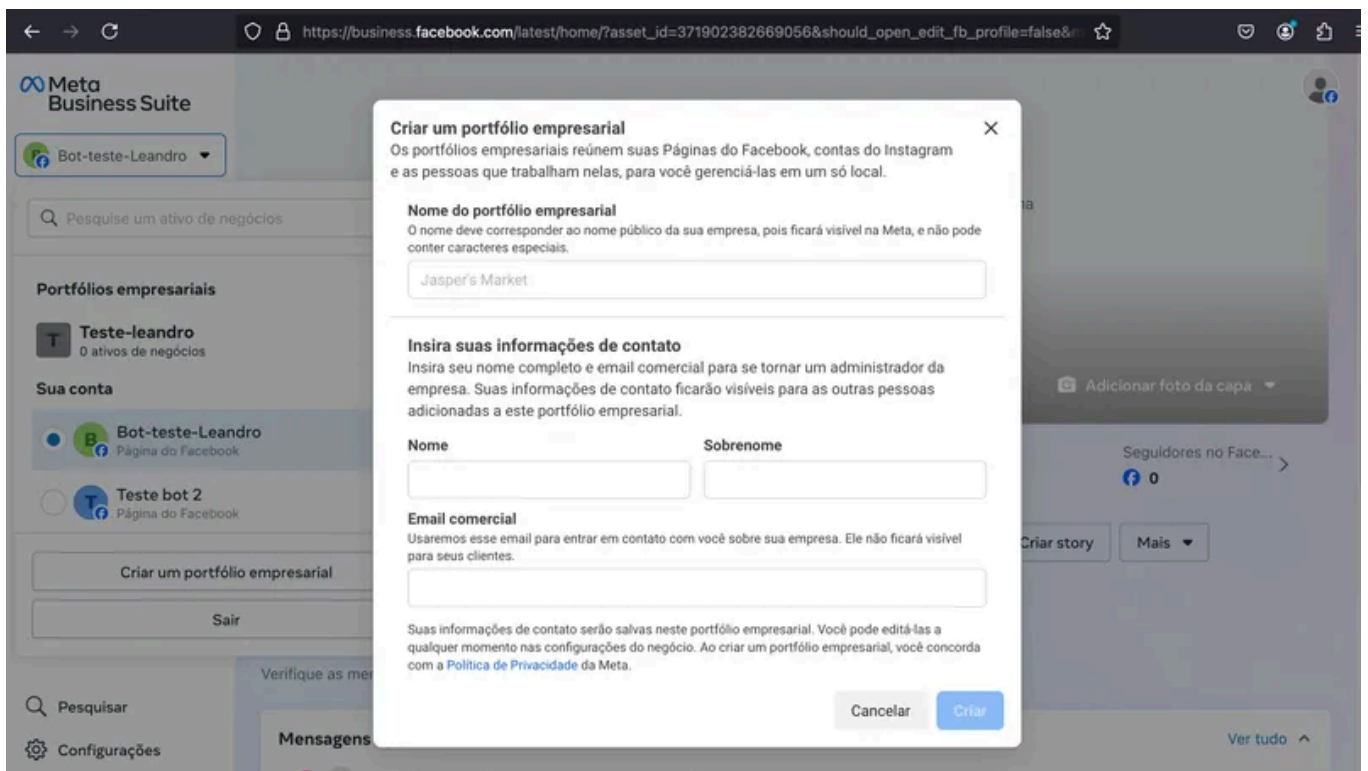
Através do link abaixo, se inscreva no Facebook pessoal.

link de acesso

2. Criar um portfólio empresarial

facebook business

No facebook business, crie um portfólio empresarial.

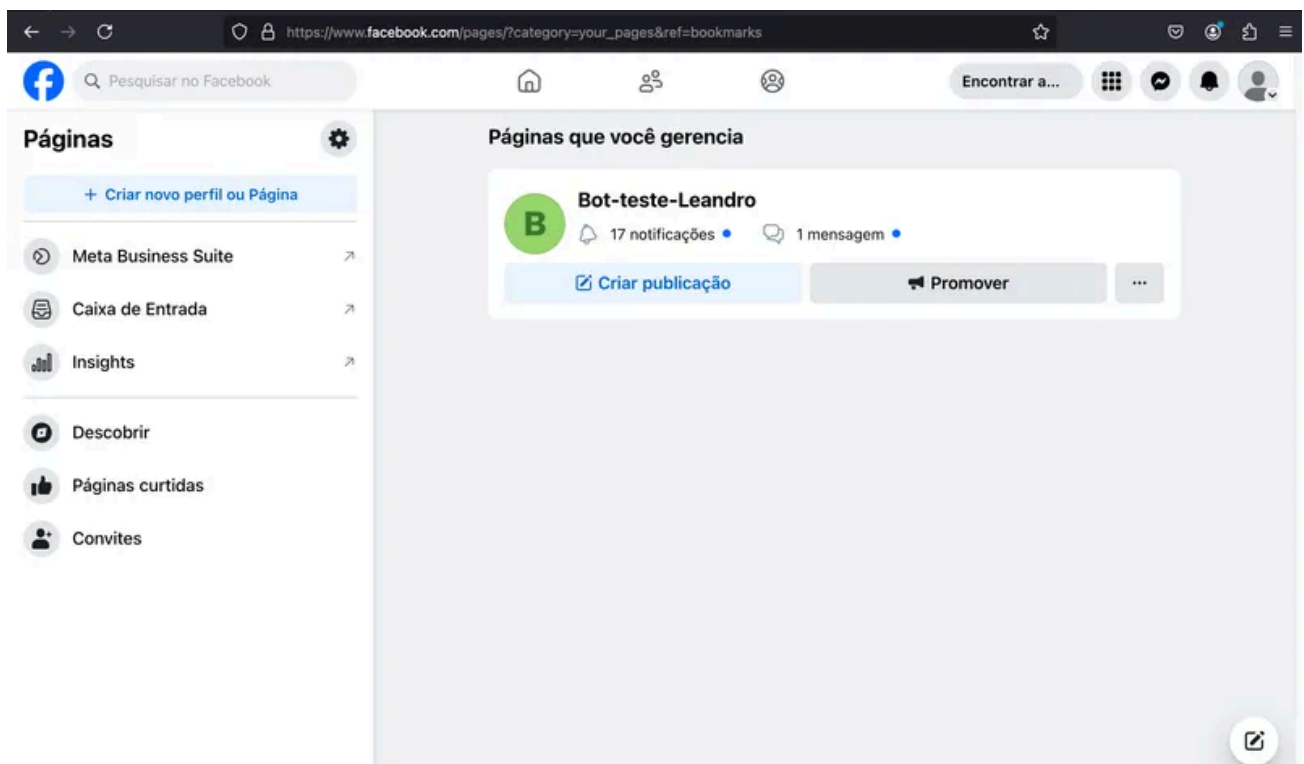


3. Criar uma página

link página

Dentro do Facebook crie uma página, é por lá que usuários externos poderão mandar mensagem para o bot.

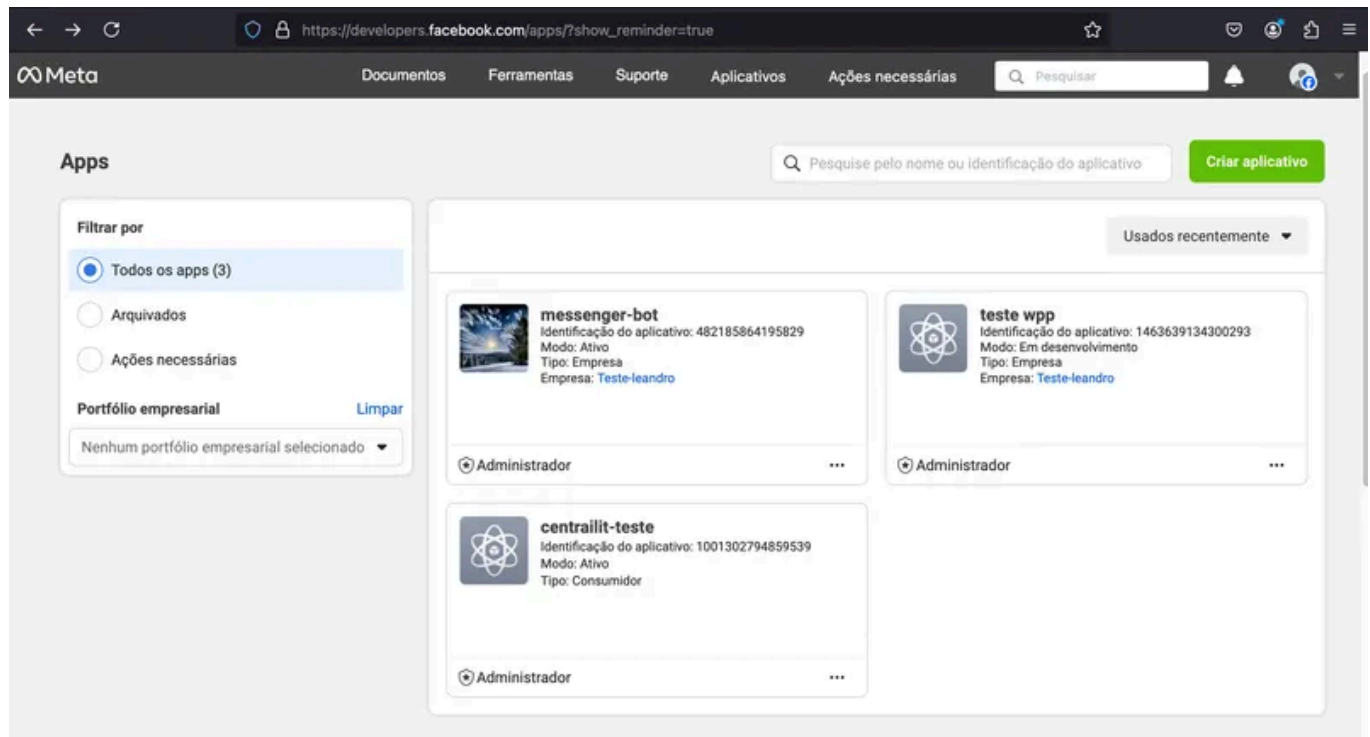
Volte ao Facebook Business para vincular a página criada.



4. Criar aplicativo no Facebook Developer

link developers

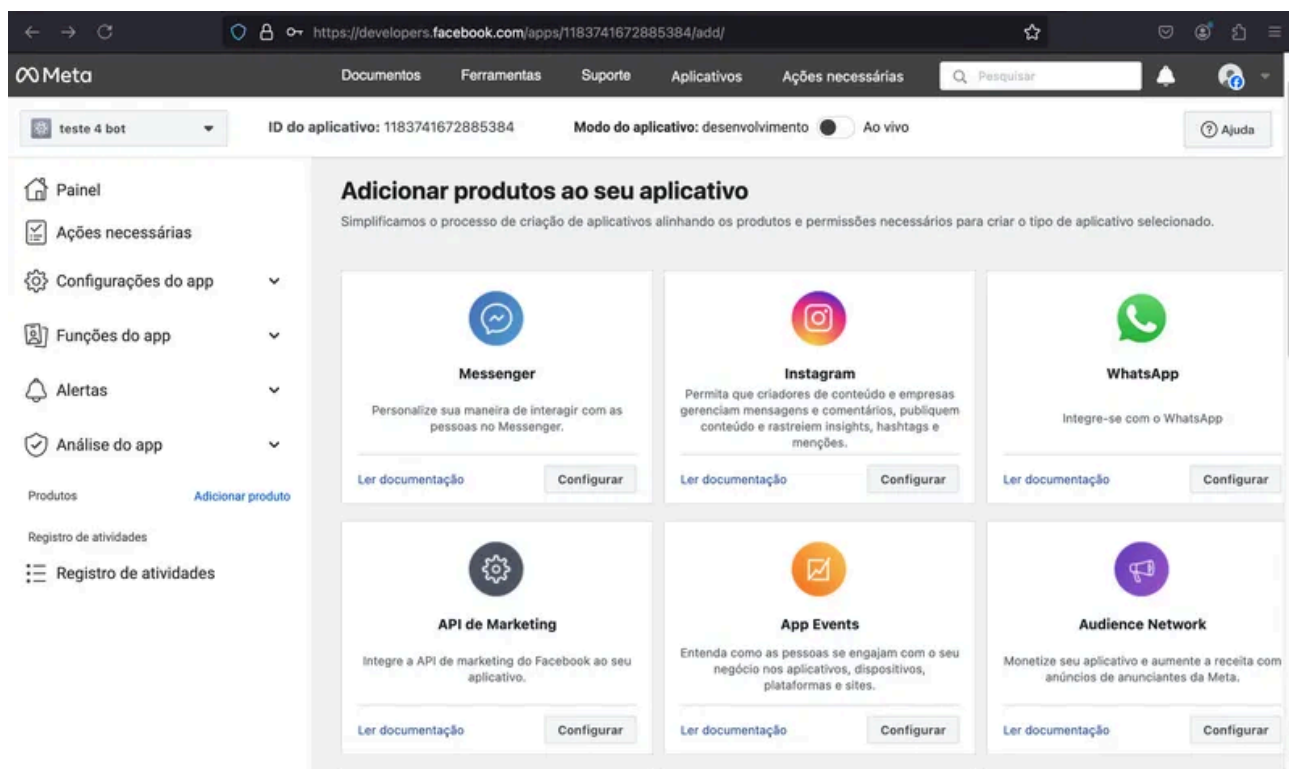
O aplicativo serve para usar e configurar ferramentas da API do Facebook. No processo de criação, o primeiro passo é escolher o portfólio empresarial criado, isso garante que mais recursos da api do facebook estarão disponíveis; no segundo passo, sobre função do aplicativo, escolher pela opção "Outro"; por último, o tipo de aplicativo é "Empresa".



5. Acessar a configuração do Messenger

Na imagem abaixo, acesse para configurar o Messenger.

Na aba de Configurações do app > Básico, encontra-se a Chave Secreta do Aplicativo, esta é **Chave Secreta**.











6. Configurar webhook e eventos

Guarde o **Facebook ID** que se encontra na página, no caso seria o 371902382669056 conforme mostrado na imagem.

Nessa página de configuração, primeiro insira a url do webhook com endereço **`https://{endereço_do_lunaris}/all/messenger/webhook?facebook_id={id_facebook}`**, o id_facebook é o Facebook ID. Nesta página, pede um Token de Webhook, este tem que ser o igual cadastrado na plataforma AURA BLOCKS chamado de **Código de Segurança**.

Depois gere Token de Acesso pela página criada e também assine o evento de "messages". Esse token será utilizado para enviar mensagens do broker para o usuário. O link de envio tem uma parte fixa: **`https://graph.facebook.com/v12.0/me/messages?access_token=`** seguida do **Token** obtido. Nessa parte, assine alguns eventos do webhook, por exemplo 'messages'.

Na mesma página, peça solicitação do 'pages_messaging'. A resposta por parte do facebook pode durar mais de um dia.




-  Painel
-  Ações necessárias
-  Configurações do app 
-  Funções do app 
-  Alertas 
-  Análise do app 
- Produtos  [Adicionar produto](#)
- Webhooks
- Messenger 
- [Configurações da API do Messeng...](#)

Mostrar erros recentes



2. Gere tokens de acesso

Conecte Páginas do Facebook para gerar tokens de acesso e configurar assinaturas de webhooks.

Nome da Página	Assinatura do webhook	Token
 Bot-teste-Leandro 371902382669056	messages, messaging_postbacks, message_deliveries,...  Ver 5 campos completos	Gerar 
Adicionar Página		



3. Faça a análise do app

Para acessar dados em tempo real com o app usando essa permissão ou recurso, é necessário fazer a análise do app.

5.6. Twilio

Para cadastrar o broker Twilio é necessário o número de telefone do enviador, SID e token da conta. No caso de teste, o Twilio oferece um número de teste pelo Sandbox, para usar um número real é preciso fazer um upgrade da conta.

1. Criar uma conta

Através do link abaixo, se inscreva no Twilio

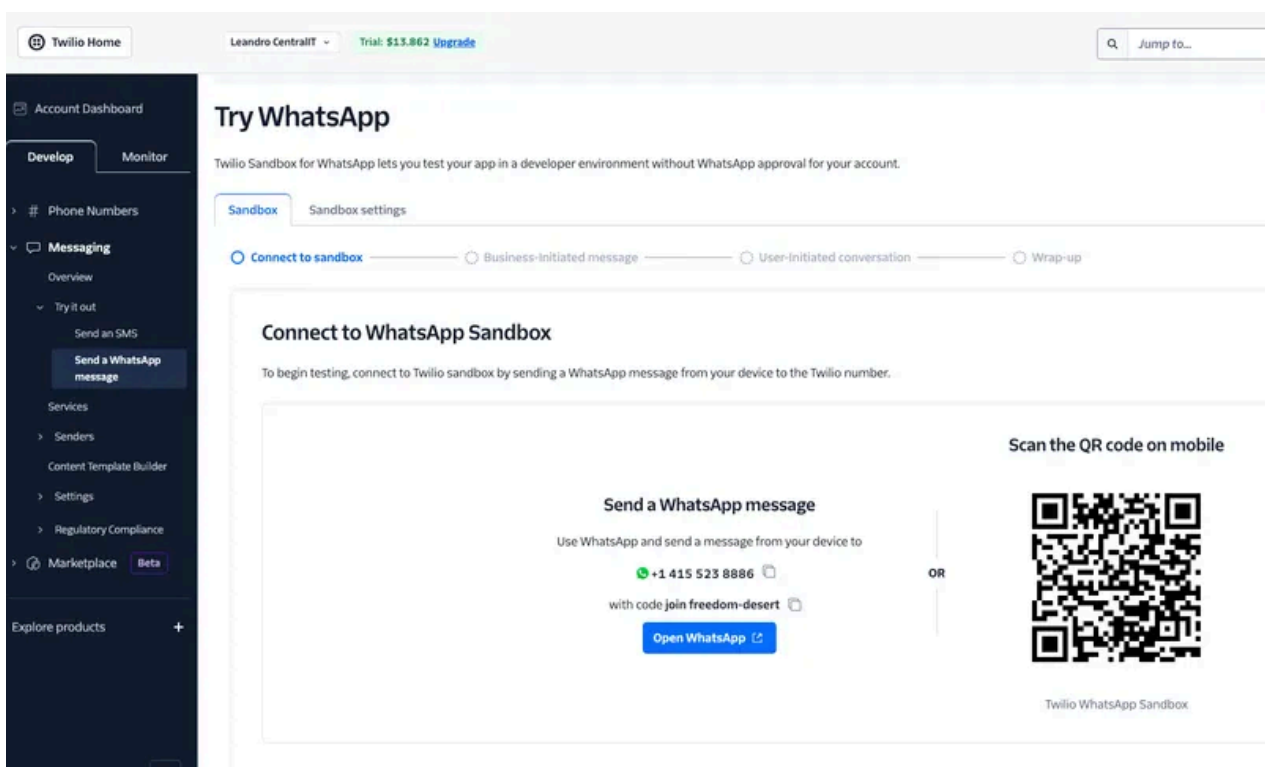
[link de acesso](#)

2. Enviar e Receber Mensagem no SandBox (para teste)

Para se conectar com o número oferecido, envie a mensagem para o número informado, no caso da imagem abaixo seria a mensagem "join freedom-desert".

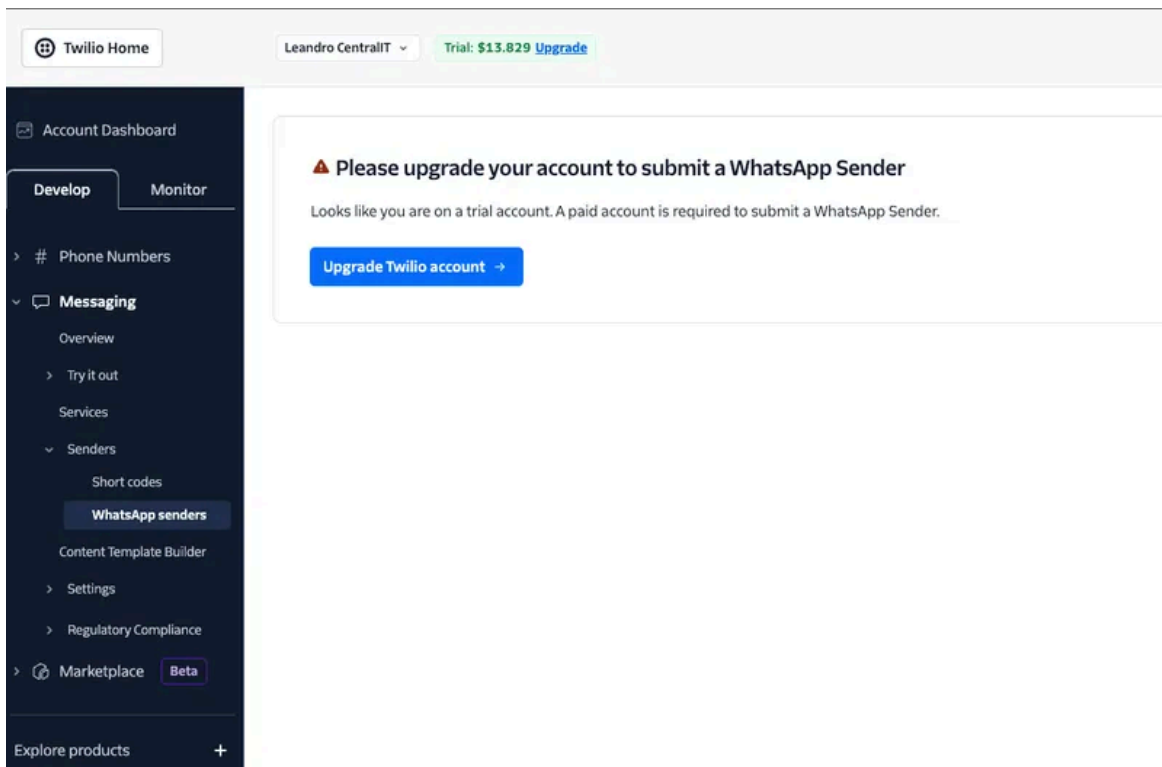
Para ser informado de novas mensagens pela Engine, cadastre o webhook na aba ao lado "SandBox settings" com endereço **https://{endereço_do_lunaris}/post/twilio/webhook**.

O contato desta conta é **14155238886**, como mostrado na imagem.



3. Cadastrar Webhook da Conta

No caso de um número real, faça upgrade da conta como na figura abaixo. Depois, pode cadastrar o link do webhook. No caso de número fictício ou real, este será o cadastrado nas configurações do Broker.



4. Obter SID e Token da conta

link de acesso

Na parte de configurações, em Keys & Credentials, na parte de Live credentials consegue-se obter o SID da conta, o Auth Token.

← Back

Manage account

General settings

Manage users

Subaccounts

Notification preferences

Audit events

Keys & Credentials

API keys & tokens

Credentials

Connected apps

<<

Leandro Rodrigues

leandro.rodrigues@centralit.c...

User Settings

Logout

Auth Tokens - United States (US1)

Auth tokens can be used to authenticate while making API requests. You will need to use HTTP Basic Authentication with `user=ACCOUNT_SID` and `password=AuthToken`. Auth tokens are specific to your account and can be used to access all APIs for the account. Please keep the auth tokens in a secure place and rotate them periodically. [Learn more about auth tokens](#).

Live credentials

Account SID- used to exercise the REST API
ACc301f5f33a0005cf3d8231c8abc1dd15

Auth token [Request a secondary token](#)

⚠ Sensitive information. Store your token securely to protect your account. [Learn more](#)

•••••

Keep this somewhere safe and secure

Test credentials

Account SID- used to exercise the REST API
ACf945d4392d71cb79fd9eae877f9379b

Test Auth token

•••••

Keep this somewhere safe and secure

5.7. Telegram

A especificação a seguir atua como um guia detalhado para utilização do broker **Telegram**, fornecendo todas as instruções e orientações necessárias para sua configuração.

Configuração

Para realizar o cadastro do broker, é imprescindível fornecer o token do bot e o identificador do chat. Todos esses dados podem ser facilmente encontrados após a criação do bot no **BotFather**.

O que é o BotFather?

O **BotFather** é o bot oficial do Telegram usado para gerenciar e criar novos bots. Com ele, você pode registrar um novo bot, definir o nome, a descrição, o avatar e até gerenciar as permissões do seu bot.

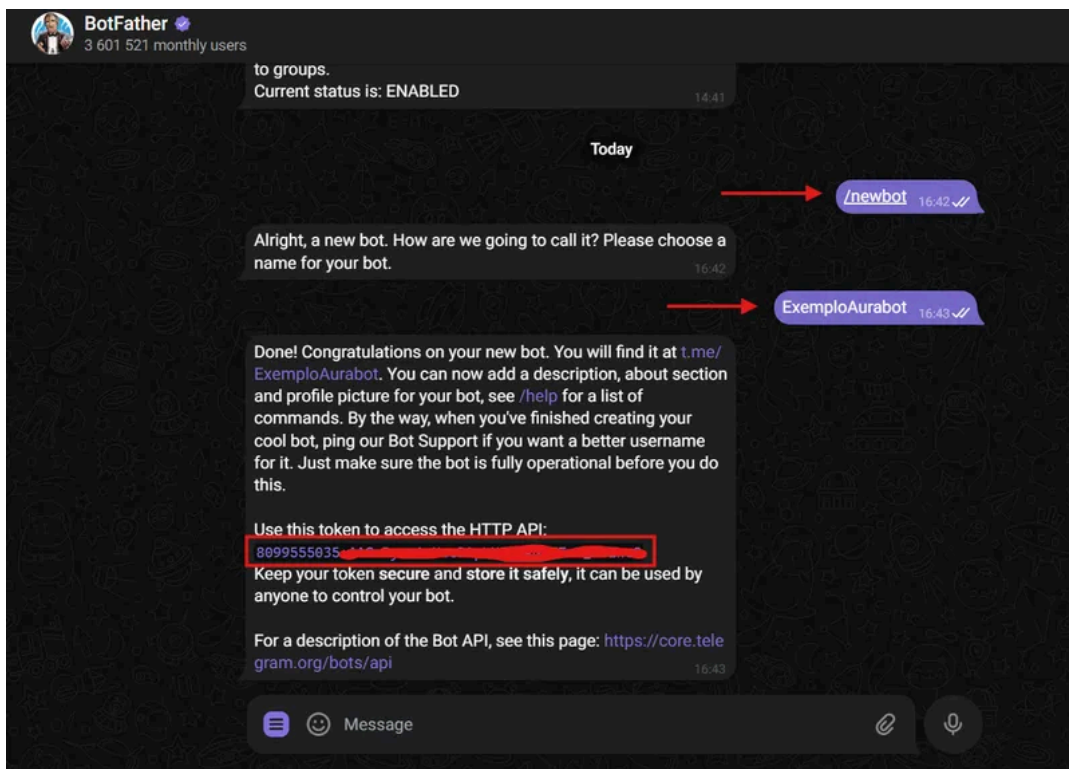
Criar um Bot

Siga os passos abaixo para criar um bot no Telegram e integrá-lo à plataforma low-code.

1. Abra o aplicativo Telegram e inicie uma conversa com o **BotFather**.
2. Digite o comando `/newbot` e pressione enter.

O BotFather irá pedir que você escolha um nome para o bot. Esse será o nome que aparecerá nos chats. O nome deve terminar com a palavra `bot`. Por exemplo: `Meubot`.

1. Após escolher o nome, o BotFather irá fornecer um **token**, que será utilizado para o cadastro dentro da plataforma low-code.



1. Entre no chat com seu bot, que pode ser acessado no link `t.me/<nome_do_bot>`, e pressione o botão "start".
2. Em seguida, acesse a URL

`https://api.telegram.org/bot<token_do_bot>/getUpdates`

, e um JSON será retornado. Nele, procure a seção abaixo e copie o **identificador (id)**

do chat para utilizá-lo dentro do low-code.

```
JSON

{
  chat: {
    id: 7288249035,
    first_name: "Nome",
    last_name: "Sobrenome",
    type: "private"
  }
}
```

3. Para garantir a segurança da comunicação, você precisa criar um segredo que será usado para validar as mensagens entre o bot e a plataforma. Esse segredo pode ser

qualquer texto com 1 a 256 caracteres, mas recomenda-se utilizar uma chave aleatória gerada com os comandos abaixo:

É possível gerar chaves aleatórias com os seguintes comandos:

PowerShell



```
$chars = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ012345  
$key = -join ((1..256) | ForEach-Object { $chars[(Get-Random -Maximu  
Write-Output $key
```

Bash



```
< /dev/urandom tr -dc 'A-Za-z0-9' | head -c 256
```

Bash



```
LC_ALL=C < /dev/urandom tr -dc 'A-Za-z0-9' | head -c 256
```

Todos devem gerar uma chave similar a essa:

```
FJHv5Us4M1lS6EKK603PQEs3TzLJUQRjPp5LoDk1lAs3GCZTkUiYMPINKPIreUX7cEhW6zd15
```

y...

Configuração

Normalmente, a configuração é feita automaticamente pelo próprio plugin, mas, em alguns casos, pode falhar. Use esta seção caso o bot não responda após a primeira interação.

ATENÇÃO: Basta copiar e colar as URLs a seguir na barra de pesquisa de qualquer navegador para acionar a API do Telegram.

Após configurar o seu bot, é necessário configurar o webhook. Você pode fazer isso utilizando a seguinte URL:

```
https://api.telegram.org/bot<TOKEN_DO_BOT>/setWebhook?url=https://{seu-  
dominio}/{lunaris_ambiente}/post/telegram/webhook
```

Se tudo ocorrer corretamente, a mensagem a seguir será exibida, confirmando que o webhook foi configurado.

JSON



```
{  
  "ok":true,  
  "result":true,  
  "description":"Webhook was set"  
}
```

5.8. Microsoft Teams

A especificação a seguir atua como um guia detalhado para utilização do broker **Microsoft Teams**, fornecendo todas as instruções e orientações necessárias para sua configuração.

Atualmente, o teams não está suportando alguns tipos de arquivos. Imagens estão funcionando normalmente. [Confira mais](#)

Configuração

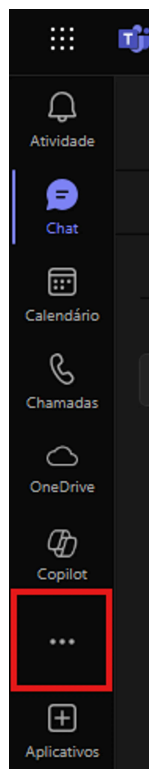
Para realizar o cadastro do broker, é imprescindível fornecer o **nome**, **identificador (ID)**, **chave secreta (segredo)** e **locatário (tenant)**.

Este manual está dividido em seções para facilitar a localização das informações. As seções são:

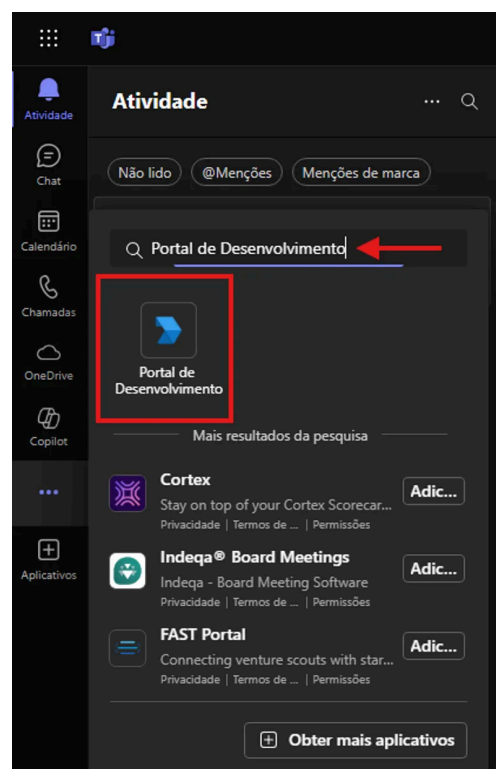
[Criar Bot](#) □ [Adquirir Configuração](#) □ [Implantar Bot](#) □ [Importar e Exportar](#)

Criar Bot

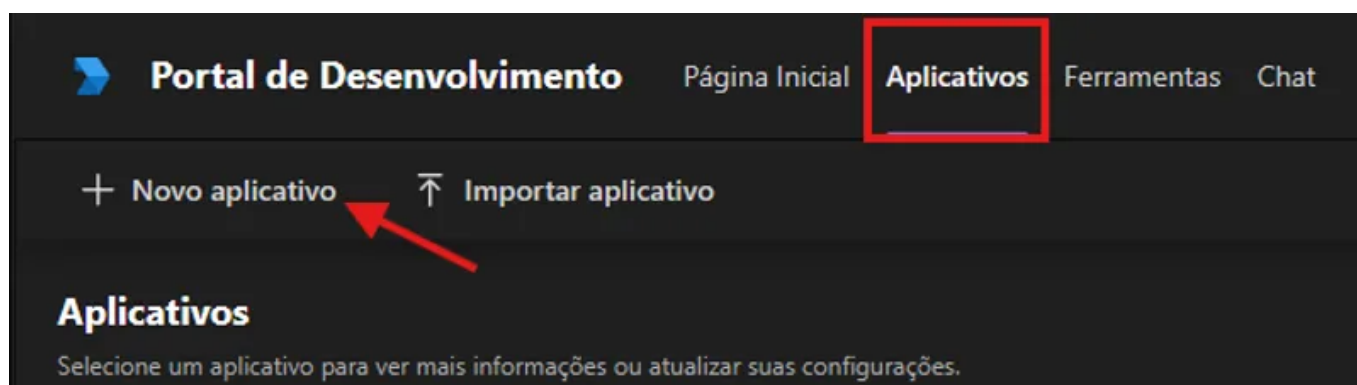
1. Na barra lateral, clique no ícone de **reticências**.



1. No campo de pesquisa, digite **Portal de Desenvolvimento** e selecione-o na lista de opções.



1. Usando o menu, aperte em **Aplicativos > Novo aplicativo**



1. Insira o nome desejado para o seu aplicativo e clique em **Adicione**.

1. No menu, selecione a opção **Informação básica** e preencha **todos os campos obrigatórios** conforme indicado nas imagens e, ao final, clique no botão Salvar.

Portal de Desenvolvimento

Página Inicial

Aplicativos

Ferramentas

Chat

< Aplicativos

Salvar

Reverter

Exemplo

Visão geral

Configurar

Avançado

Painel

Análise

Informação básica

Marca

Recursos do aplicativo

Editor de pacote de aplicativos

Permissões

Logon único

Idiomas

Domínios

Proprietários

Conteúdo do aplicativo

Ambientes

Planos e preços

Configurações de administração

Site (deve ser um válido HTTPS URL)*

https://exemplo.com.br

URLs dos aplicativos

Você deve fornecer links para sua política de privacidade e termos de uso. Saiba mais sobre as práticas recomendadas para links.

Política de privacidade*

https://exemplo.com.br/privacy

Termos de Uso*

https://exemplo.com.br/terms

ID do Aplicativo (cliente) *

Specify the app ID assigned when you registered your app with Azure Active Directory.

XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

ID de Parceiro da Microsoft:

Se você fizer parte do Microsoft AI Cloud Partner Program, especifique seu ID de Parceiro. Saiba mais sobre a parceria com a Microsoft.

XXXXXXXX

Salvar

Reverter

Portal de Desenvolvimento

Página Inicial

Aplicativos

Ferramentas

Chat

< Aplicativos

Salvar

Reverter

Exemplo

Visão geral

Configurar

Avançado

Painel

Análise

Informação básica

Marca

Recursos do aplicativo

Editor de pacote de aplicativos

Permissões

Logon único

Idiomas

Domínios

Proprietários

Conteúdo do aplicativo

Ambientes

Planos e preços

Configurações de administração

Versão

1.0.0

Informações do desenvolvedor

Nome do desenvolvedor ou empresa*

Nome Exemplo da Empresa

Site (deve ser um válido HTTPS URL)*

https://exemplo.com.br

URLs dos aplicativos

Você deve fornecer links para sua política de privacidade e termos de uso. Saiba mais sobre as práticas recomendadas para links.

Política de privacidade*

https://exemplo.com.br/privacy

Termos de Uso*

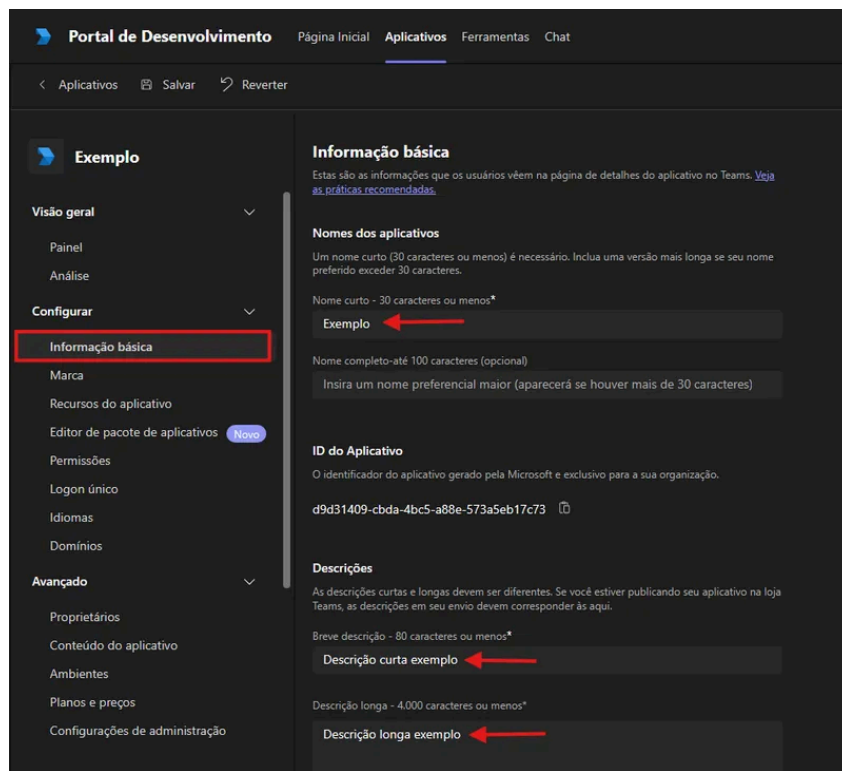
https://exemplo.com.br/terms

ID do Aplicativo (cliente) *

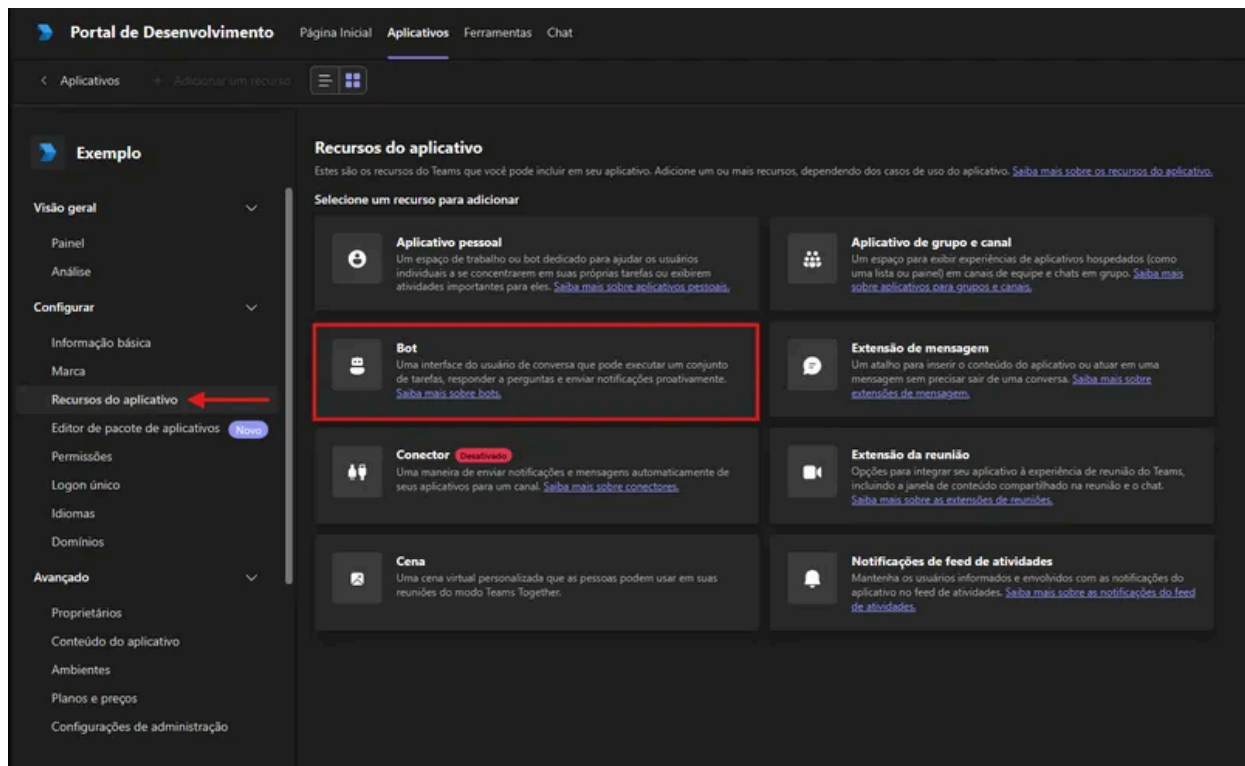
Specify the app ID assigned when you registered your app with Azure Active Directory.

XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

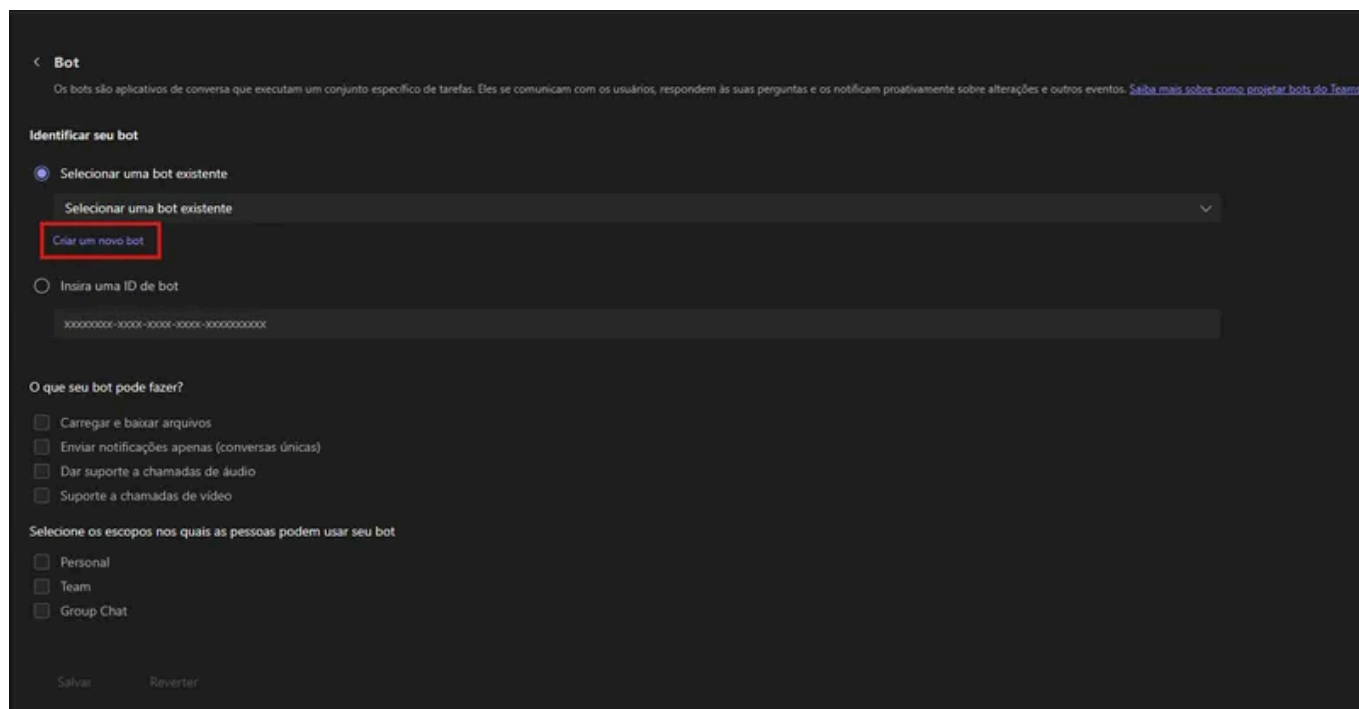
ID de Parceiro da Microsoft:



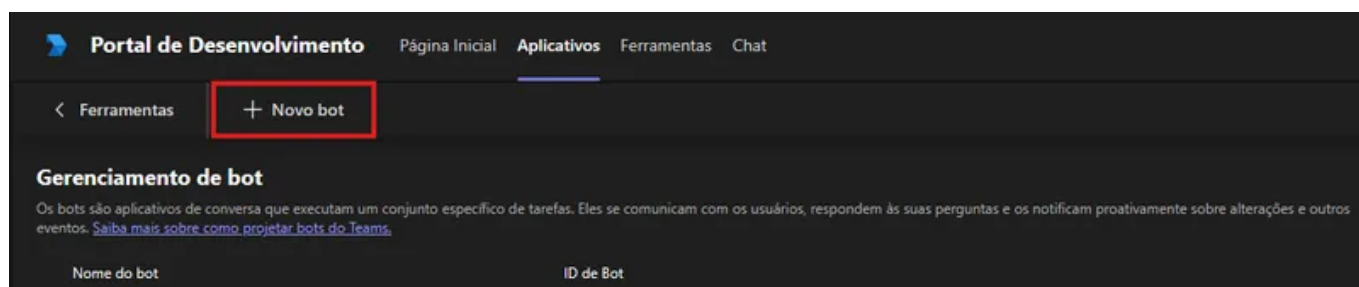
1. Selecione em **Recurso do Aplicativo > Bot**



1. Aperte em **Criar um novo bot**



1. Usando o menu, aperte em **Novo bot**



1. Insira o nome desejado para o seu bot e clique em **Adicione**.

Adicionar bot

Nome do bot*

ExemploBot

Cancelar

Adicione

1. Na seção **Configurar**, insira o ponto de extremidade desejado e clique em **Salvar**.

Os endpoints são formados por: `https://{dominio}/{caminho-teams-server}/webhook`

The screenshot shows the 'Portal de Desenvolvimento' (Development Portal) interface. At the top, there's a navigation bar with 'Portal de Desenvolvimento' and links for 'Página Inicial', 'Aplicativos', 'Ferramentas', and 'Chat'. Below this, a sidebar on the left shows a list of bots, with 'ExemploBot' selected and its 'Configurar' (Configure) option highlighted with a red box. The main area on the right is titled 'Configurar' and contains a link to the 'Portal do Bot Framework' for bot update icons. Under the 'Endereço do ponto de extremidade' (Endpoint address) section, there's a text input field containing the URL 'https://seu-dominio/caminho-teams-server/webhook', which is also highlighted with a red box and a red arrow. At the bottom of this section are two buttons: 'Salvar' (Save) and 'Reverter' (Revert).

1. Volte para **Aplicativos > {SeuBot} > Recursos do Aplicativo > Bot**. Marque a opção **Selecionar bot existente** e escolha o bot que você criou na caixa de seleção. Depois,

selecione **Personal** na aba de escolha de escopo e pressione **Salvar**.

< Bot

Os bots são aplicativos de conversa que executam um conjunto específico de tarefas. Eles se comunicam com os usuários, respondem às suas perguntas e os notificam proativamente sobre alterações e outros eventos. [Saiba mais sobre como projetar bots do Teams](#).

Identificar seu bot

☒ Selecionar uma bot existente

ExemploBot (8579cc8f-d3d8-4a4d-835d-d6ed3d3728a9)

ExemploBot (8579cc8f-d3d8-4a4d-835d-d6ed3d3728a9) ←

teste (ac1f341a-633e-4dd7-9095-f5c693331336)

☐ Criar um novo bot

8579cc8f-d3d8-4a4d-835d-d6ed3d3728a9

O que seu bot pode fazer?

☒ Carregar e baixar arquivos ←

☐ Enviar notificações apenas (conversas únicas)

☐ Dar suporte a chamadas de áudio

☐ Suporte a chamadas de vídeo

Selecione os escopos nos quais as pessoas podem usar seu bot

☒ Personal ←

☐ Team

☐ Group Chat

Salvar Reverter

Portal de Desenvolvimento

Página Inicial Aplicativos Ferramentas Chat

< Aplicativos + Adicionar um recurso

Exemplo

Visão geral

Painel

Análise

Configurar

Informação básica

Marca

Recursos do aplicativo ←

Editor de pacote de aplicativos

Permissões

Logon único

Idiomas

Domínios

Avançado

Proprietários

Conteúdo do aplicativo

Ambientes

Planos e preços

Configurações de administração

Recursos do aplicativo

Estes são os recursos do Teams que você pode incluir em seu aplicativo. Adicione um ou mais recursos, dependendo dos casos de uso do aplicativo. [Saiba mais sobre os recursos do aplicativo](#).

Selecione um recurso para adicionar

Aplicativo pessoal

Um espaço de trabalho ou bot dedicado para ajudar os usuários individuais a se concentrarem em suas próprias tarefas ou exibirem atividades importantes para eles. [Saiba mais sobre aplicativos pessoais](#).

Aplicativo de grupo e canal

Um espaço para exibir experiências de aplicativos hospedados (como uma lista ou painel) em canais de equipe e chats em grupo. [Saiba mais sobre aplicativos para grupos e canais](#).

Bot

Uma interface do usuário de conversa que pode executar um conjunto de tarefas, responder a perguntas e enviar notificações proativamente. [Saiba mais sobre bots](#).

Extensão de mensagem

Um atalho para inserir o conteúdo do aplicativo ou atuar em uma mensagem sem precisar sair de uma conversa. [Saiba mais sobre extensões de mensagem](#).

Conector

Uma maneira de enviar notificações e mensagens automaticamente de seus aplicativos para um canal. [Saiba mais sobre conectores](#).

Extensão da reunião

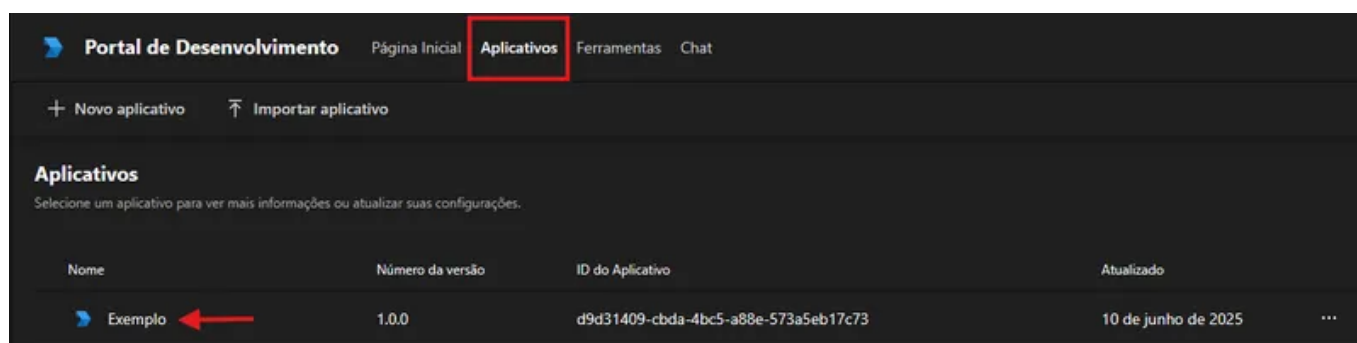
Opções para integrar seu aplicativo à experiência de reunião do Teams, incluindo a janela de conteúdo compartilhado na reunião e o chat. [Saiba mais sobre as extensões de reuniões](#).

Cena

Uma cena virtual personalizada que as pessoas podem usar em suas reuniões do modo Teams Together.

Notificações de feed de atividades

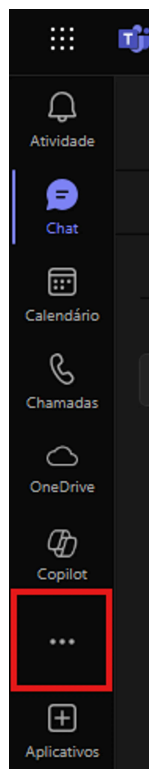
Mantenha os usuários informados e envolvidos com as notificações do aplicativo no feed de atividades. [Saiba mais sobre as notificações do feed de atividades](#).



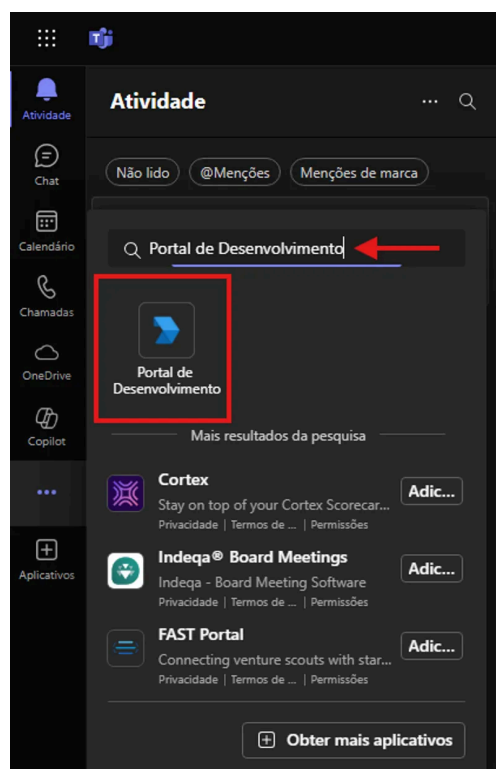
Recomenda-se seguir as instruções das imagens acima, pois elas contêm o exemplo utilizado pela equipe Citsmart Aura e funcionam de forma confiável. Opções adicionais, como "team" ou "suporte para chamadas de vídeo", podem ajudar a personalizar a experiência do bot para diferentes cenários de uso no Teams, mas não há garantia de que funcionarão corretamente após a configuração.

Adquirindo configuração

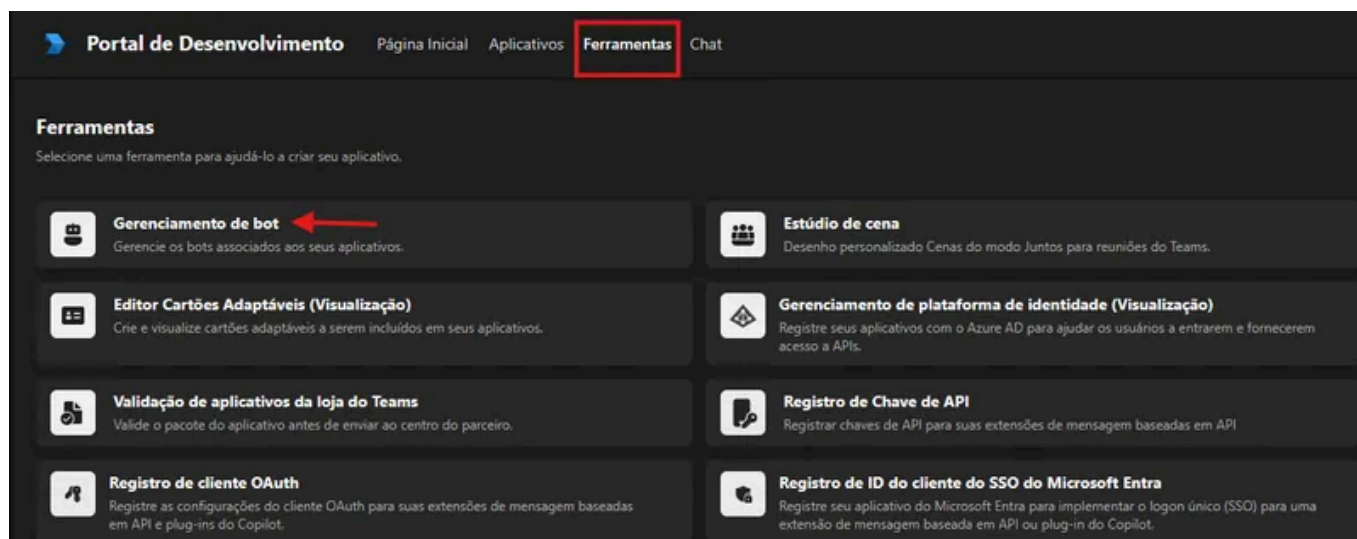
1. Na barra lateral, clique no ícone de **reticências**.



1. No campo de pesquisa, digite **Portal de Desenvolvimento** e selecione-o na lista de opções.

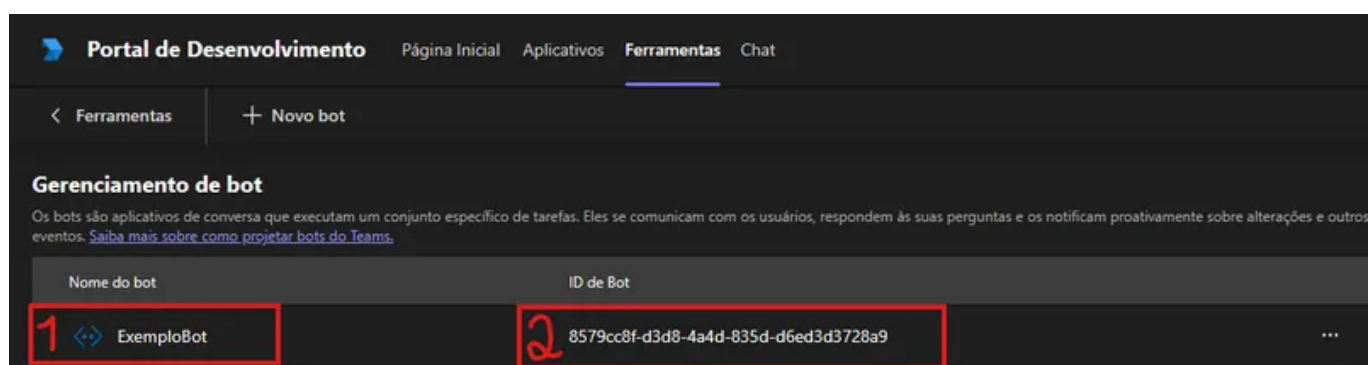


1. Usando o menu, aperte em **Ferramentas > Gerenciament de Bot**



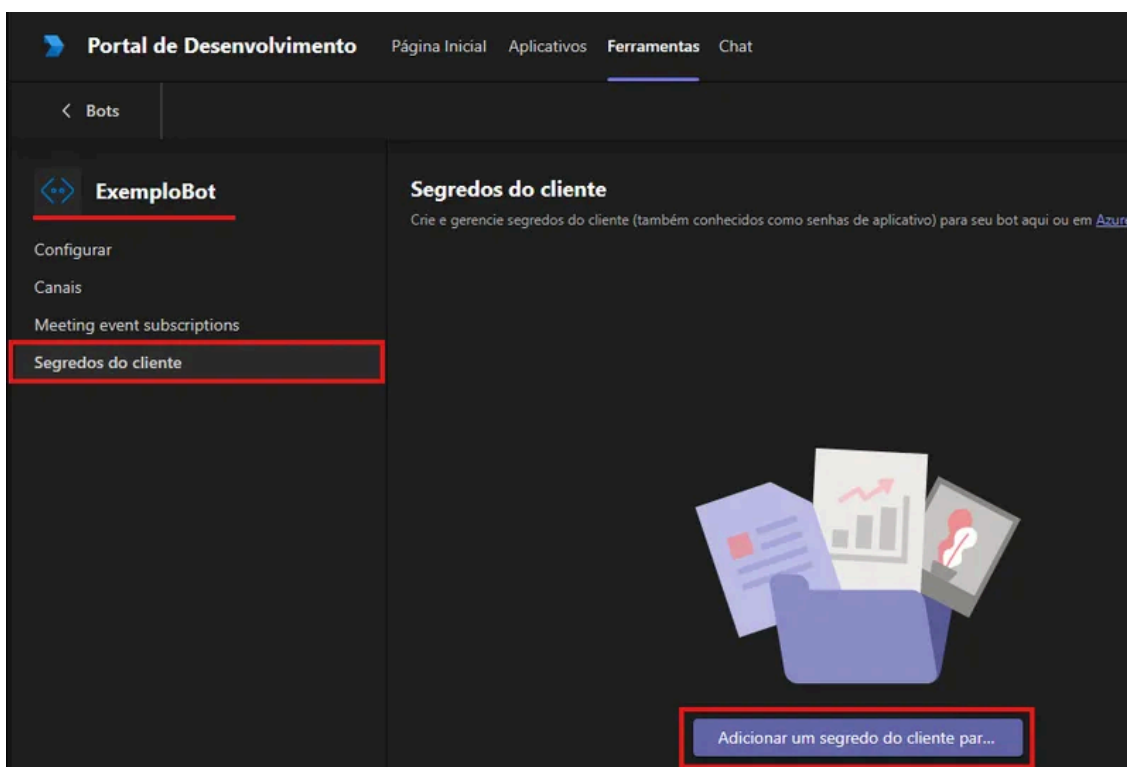
1. Com isso, você poderá localizar e anotar:

- **1: Nome do bot (Nome de exibição)**
- **2: Identificador (ID do aplicativo)**



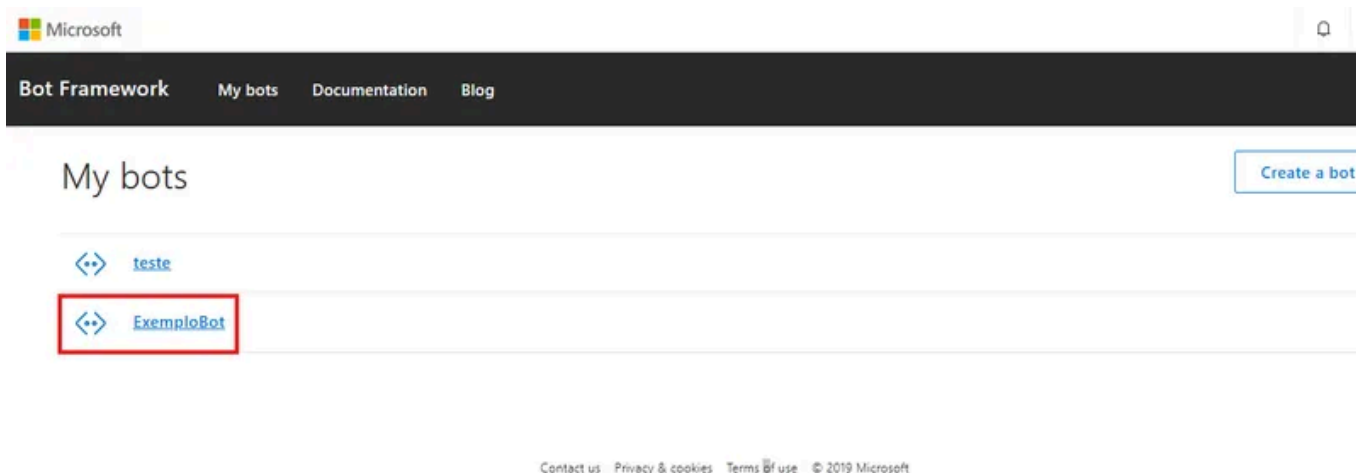
1. Para encontrar o segredo, basta selecionar **{Seu Bot} > Segredos do cliente** e clicar em **Adicionar um segredo do cliente....**

O segredo fica disponível apenas no momento da criação. Portanto, caso sejam necessárias múltiplas configurações, será preciso criar mais de um segredo ou anotá-lo em um local seguro.



O locatário do app é localizável apenas dentro do Portal do Botframework

1. Primeiro, acesse o **Portal do Botframework** e realize o login.
2. Selecione o bot que você criou e, no menu, clique em **Settings**.






1. Clique no botão **Manage Microsoft App ID and password**



Register your bot with Microsoft to generate a new App ID and password


Manage Microsoft App ID and password


1. Localize o ID do locatário


Página inicial >


 **ExemploBot**  


 **Visão geral**




 Início rápido

 Assistente de integração

 Diagnosticar e resolver problemas


▼ Gerenciar


 Identidade visual e Propriedades

 Excluir  Pontos de extremidade  Versões prévias dos recursos

^ Fundamentos

Nome de exibição : [ExemploBot](#)

ID do aplicativo (cliente) : 

ID do Objeto : 

ID do diretório (locatário) : d483e5ff-b6da-4938-b7c9-0facee9e6746

Tipos de conta com supo... : [Várias organizações](#)

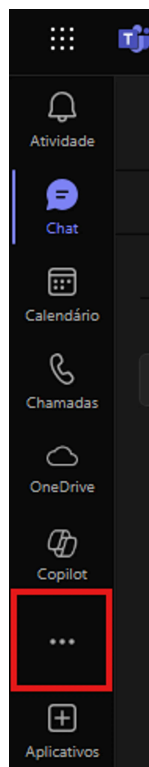
Com isso, é possível adquirir todos os campos necessários para registrar dentro da plataforma:

- **Nome do Chatbot** > Nome de exibição

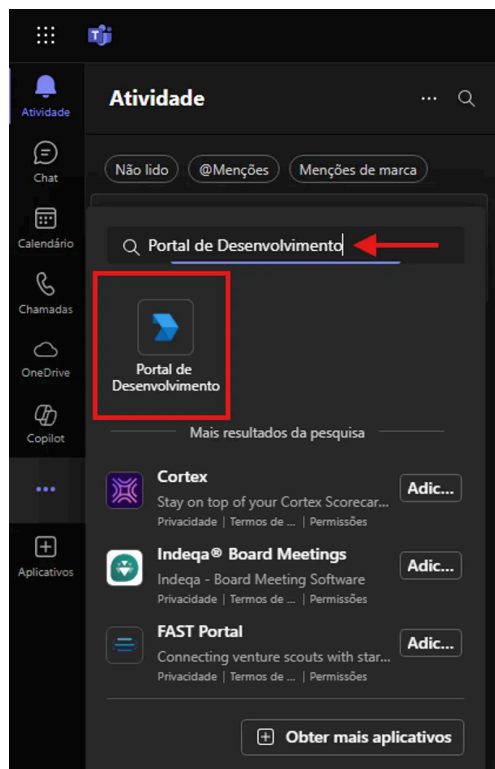
- **ID do App** > ID do aplicativo (cliente)
- **ID do Locatário** > ID do diretório (locatário)
- **Segredo do App** > Segredo

Implantar Bot

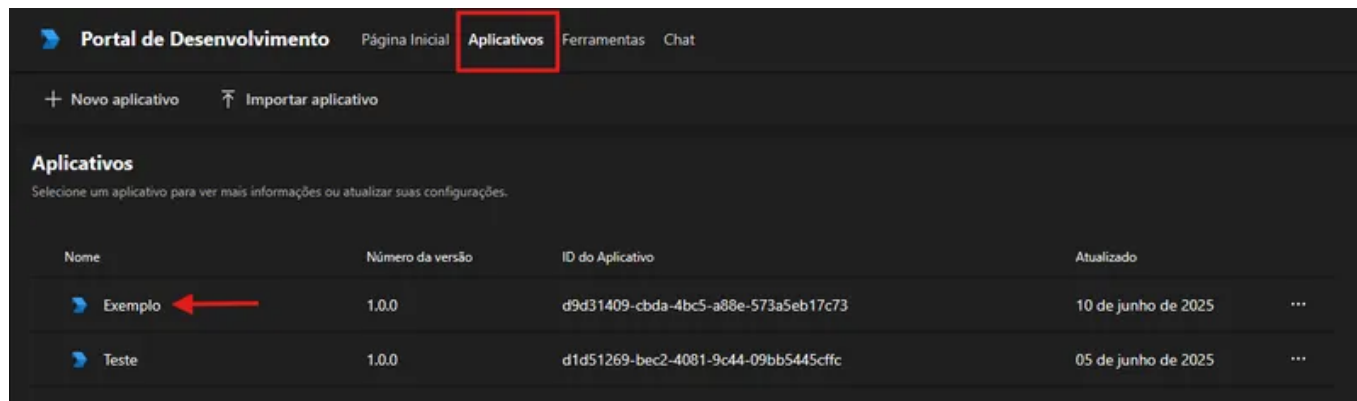
1. Na barra lateral, clique no ícone de **reticências**.



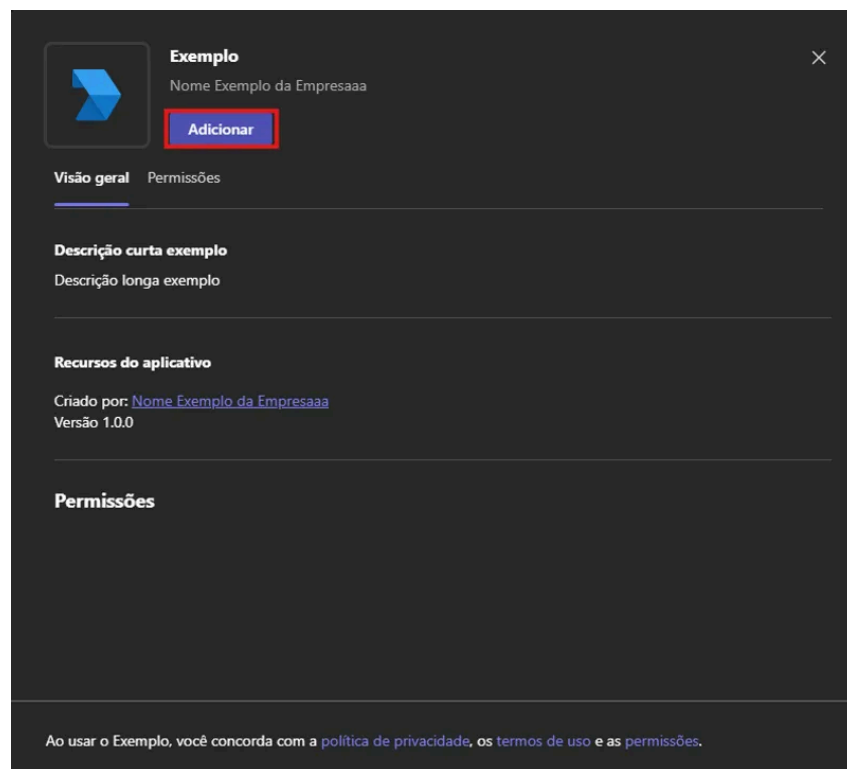
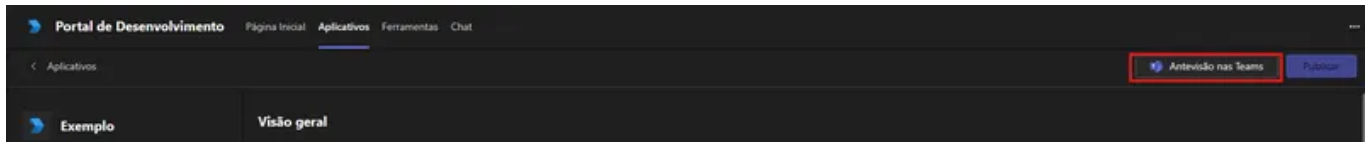
1. No campo de pesquisa, digite **Portal de Desenvolvimento** e selecione-o na lista de opções.



1. Usando o menu, aperte em **Aplicativos** > {Seu Aplicativo}



1. Clique em **Antevisão nas Teams** e selecione adicionar.

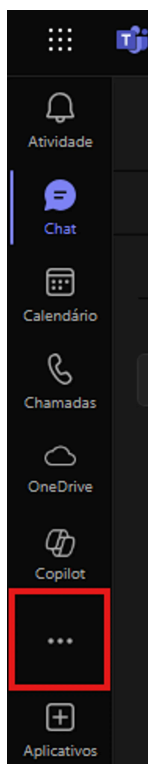


Exportar e Importar Bots

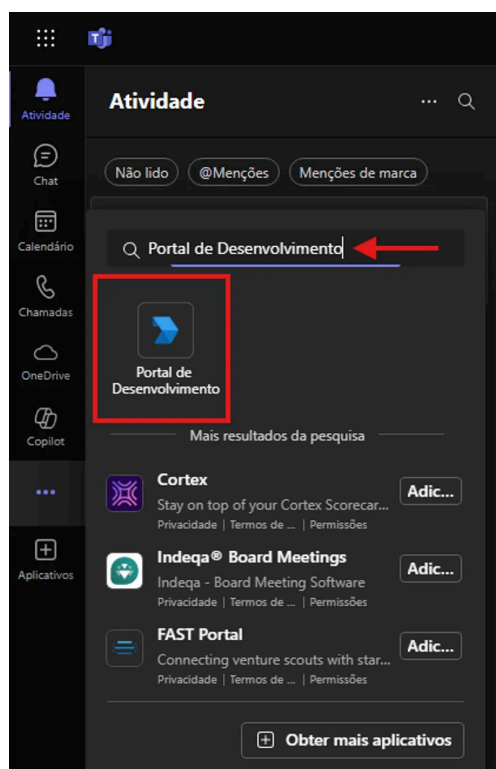
[Importar](#) | [Exportar](#)

Exportar

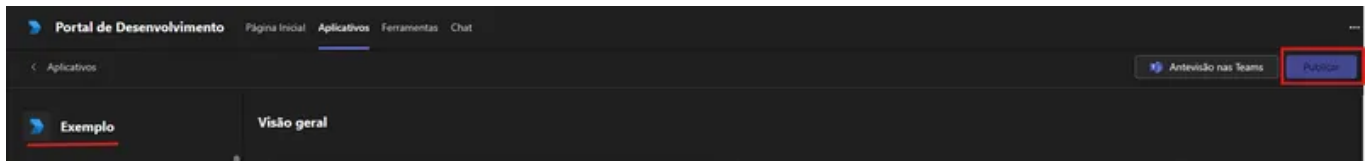
1. Na barra lateral, clique no ícone de **reticências**.



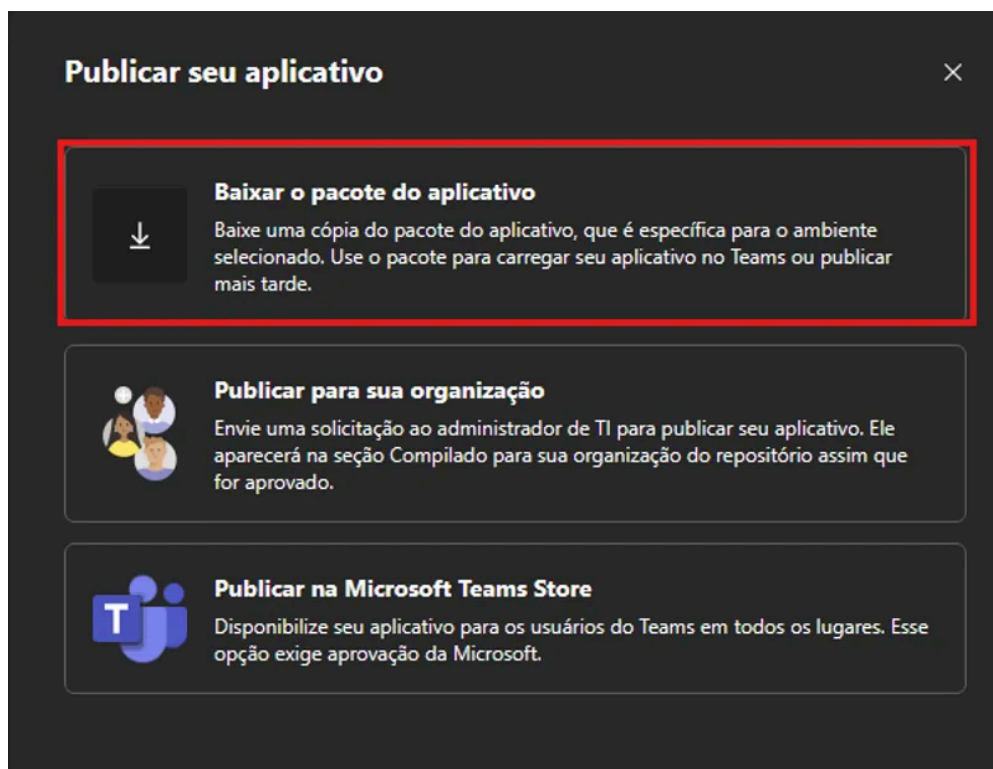
1. No campo de pesquisa, digite **Portal de Desenvolvimento** e selecione-o na lista de opções.



1. Vá para **Aplicativos > {SeuBot}** e clique em **Publicar**

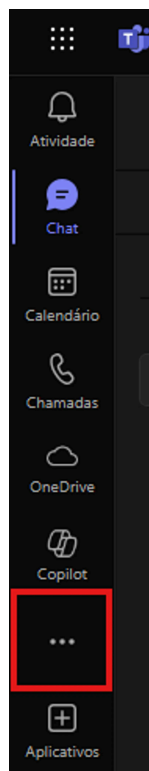


1. Selecione **Baixar o pacote do aplicativo**

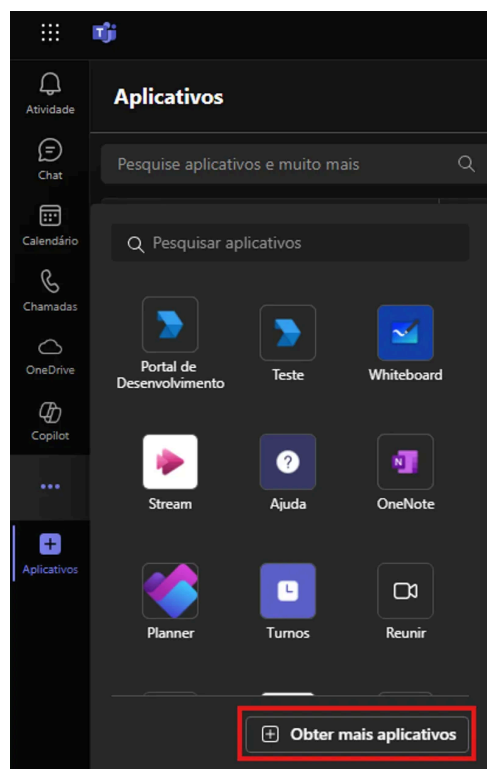


Importar

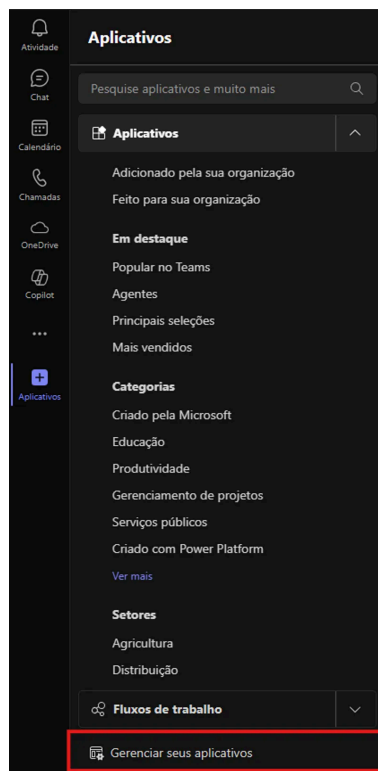
1. Na barra lateral, clique no ícone de **reticências**.



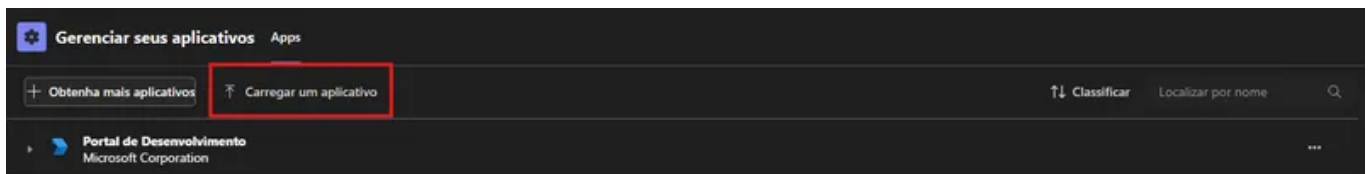
1. Clique em **Obter mais aplicativos**



1. Selecione **Gerenciar seus aplicativos**



1. Aperte em **Carregar um aplicativo**



1. Escolha a opção de envio que melhor se adequa à sua necessidade e selecione o arquivo compactado (.zip).

Carregar um aplicativo



Fazer o upload de um aplicativo personalizado

Carregue um pacote de aplicativo (arquivo .zip) para você ou para uma equipe. Essa é uma ótima maneira de testar um aplicativo à medida que ele é desenvolvido.




Envie um aplicativo para sua organização

Envie um aplicativo ao seu administrador de TI para aprovação e verifique o status de seus envios.

Caso o envio tenha sido feito por "Enviar um aplicativo à sua organização", será necessário que um administrador aceite o pedido. Após isso, o aplicativo será adicionado à organização inteira. Caso o envio tenha sido feito por "Fazer o upload de um aplicativo personalizado", siga a imagem abaixo.

1. Clique em **Adicionar**



Exemplo
Nome Exemplo da Empresa

Adicionar

Visão geral

Permissões

Descrição curta exemplo

Descrição longa exemplo

Recursos do aplicativo

Criado por: [Nome Exemplo da Empresa](#)
Versão 1.0.0

Permissões

Ao usar o Exemplo, você concorda com a [política de privacidade](#), os [termos de uso](#) e as [permissões](#).

5.9. Widget Customizável

A plataforma permite que desenvolvedores criem **widgets personalizados**, com total liberdade para escrever código.

Customizando

É possível alterar a configuração padrão do widget utilizando a ferramenta de edição de código, permitindo ao desenvolvedor personalizar diretamente o CSS do widget.

Configuração de um Widget

Registro

Endpoint:

```
POST /widgets/save/{name}
```

Observações:

- O registro é obrigatório para utilizar qualquer funcionalidade.
- O registro deve ser feito utilizando o **LowCode** ou a **CLI Aura**.

Enviar Arquivo

Endpoint:

```
POST /send-file/{widget_name}/{session_id}
```

Parâmetros:

- `widget_name`: Nome do widget registrado.

- `session_id` : ID da sessão ativa.
-

Enviar Mensagem

Endpoint (WebSocket):

`/ws/{widget_name}/{session_id}`

Parâmetros:

- `widget_name` : Nome do widget registrado.
 - `session_id` : ID da sessão ativa.
-

Histórico de Mensagens

Endpoint:

`GET /history/{widget_name}/{session_id}`

Parâmetros:

- `widget_name` : Nome do widget registrado.
 - `session_id` : ID da sessão ativa.
-

Download de Anexos

Endpoint:

`GET /widgets/{widget_name}/download/{ano}/{mes}/{dia}/{numero}.{ext}`

Parâmetros:

- `widget_name` : Nome do widget registrado.
- `ano` , `mes` , `dia` : Data do envio.
- `numero` : Identificador do anexo.

- `ext` : Extensão do arquivo (ex: `.pdf` , `.png`).
-

Pré-visualização de Mensagens

Endpoint de referência:

`/receive-message`

Observações:

Toda resposta enviada do sistema segue o formato da mensagem do tipo `ReceiveMessageFromEngine` , exibindo uma **prévia (preview)** do conteúdo.

6. Extensões

6.1. Disparo

Esta funcionalidade está disponível apenas para o papel de **Supervisor**.

A ferramenta de disparo de mensagens foi desenvolvida para facilitar o envio de comunicações em massa pelo WhatsApp, utilizando templates previamente aprovados. Com ela, é possível organizar listas de contatos, realizar envios de forma rápida e acompanhar o status de cada mensagem, garantindo uma comunicação eficiente e padronizada. O sistema suporta o envio de mensagens do tipo texto, imagens e carrossel.

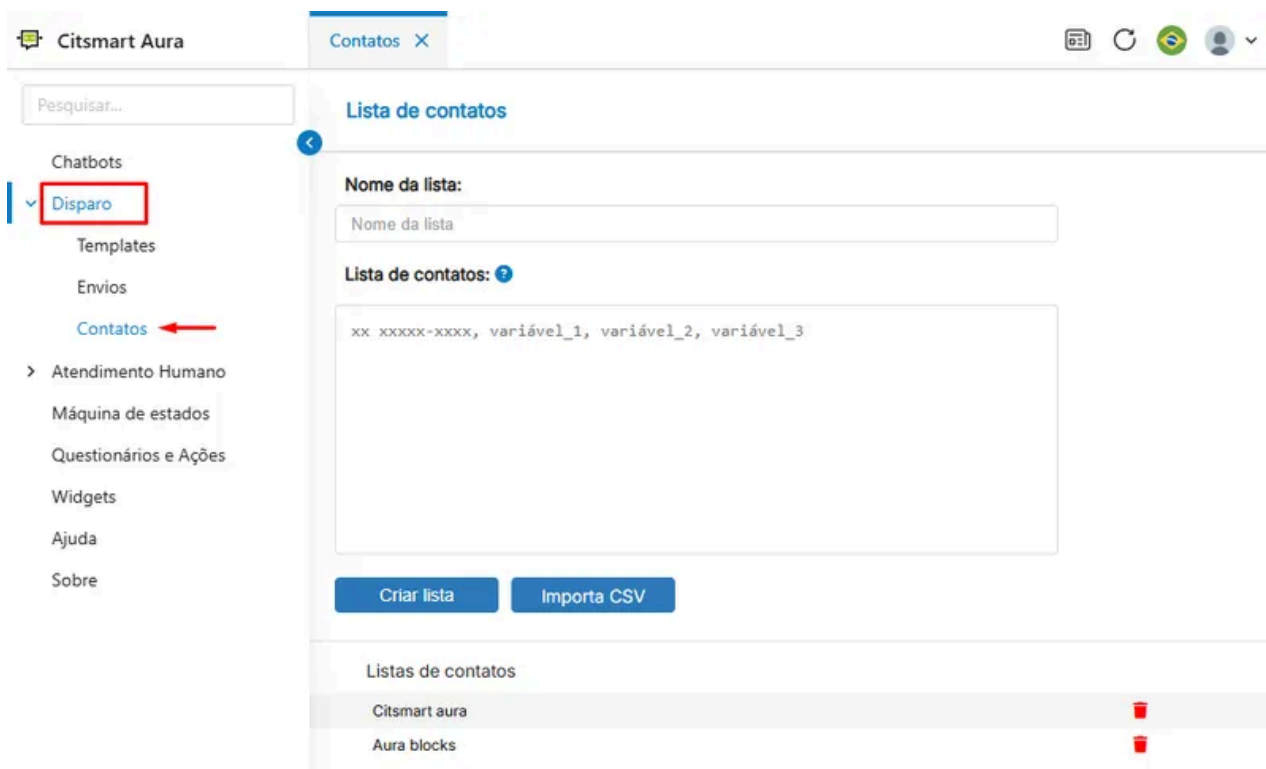
Contatos

A seção contatos permite a criação e o gerenciamento de listas de contatos que serão utilizadas nos envios de mensagens. O usuário pode adicionar contatos manualmente ou importar um arquivo CSV com os dados.

Para criar uma lista, siga os passos abaixo:

Acessando a tela de contatos

1. No menu lateral, clique em **Disparo > Contatos**.
2. O sistema exibirá a seguinte tela:



Realizando cadastro

1. Dê um nome para sua lista no campo **"Nome da lista"**.
2. No campo **"Lista de contatos"**, insira os contatos no seguinte formato:
 - `+55DDD988887777,variável_1,variável_2,variável_3,...`

⚠ Informação Importante ⚠

O número de telefone deve ser informado no formato internacional:

- **+55** → Código do Brasil.
- **61** → DDD (exemplo: Brasília).
- **988887777** → Número do telefone.

Variáveis (marcadores):

As variáveis inseridas após o telefone permitem personalizar cada mensagem de acordo com o template selecionado:

- Cada `variável_X` corresponde a um marcador `{{X}}` no corpo do template.
- A ordem das variáveis deve ser exatamente a ordem de substituição no texto.

Funcionamento:

- Ao disparar, cada **{{1}}**, **{{2}}**... é substituído pelo valor da sua respectiva **variável_1,variável_2...**
- Os marcadores são definidas na criação do template na **Gupshup**. Esses marcadores são substituídos ao informar as váriaves na criação da lista de contatos.

Exemplo:

Lista de contatos

Nome da lista:

Nome da lista

Lista de contatos: ?

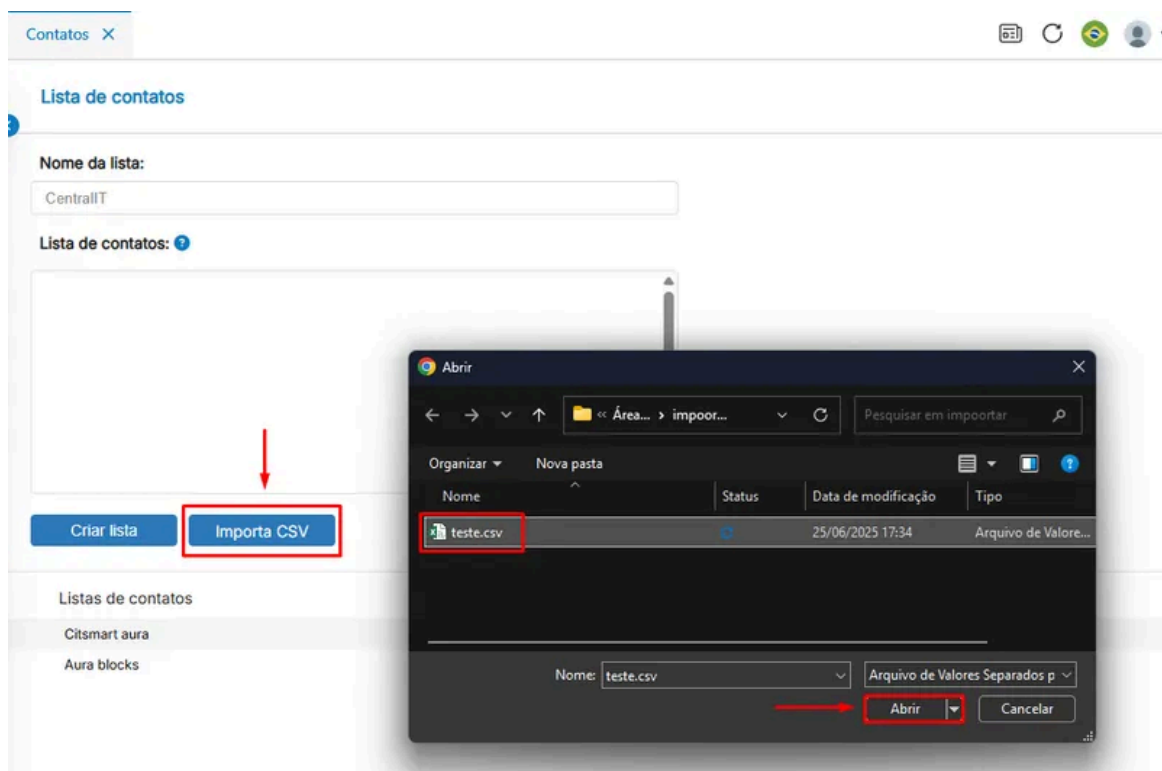
+5561999998888,Carlos,15-08-1986
+5574988887777,Pedro,24-07-2000
+5521998987878,Ricardo,25-02-1965
+5564982827878,Maria,210-09-1993

Criar lista Importa CSV

1. Após concluir a configuração, clique em **"Criar lista"** para finalizar.

Importar via arquivo CSV

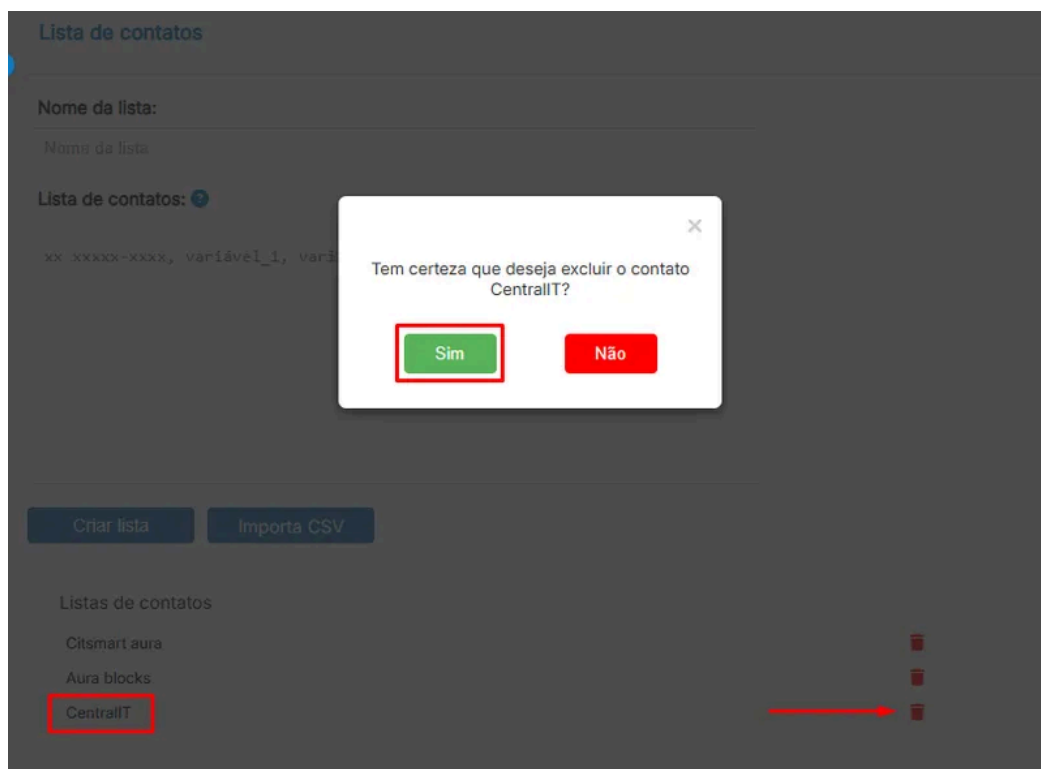
1. Dê um nome para sua lista no campo **"Nome da lista"**.
2. No campo **"Lista de contatos"**, selecione um arquivo **.csv**.
3. Clique em **Importa CSV** para selecionar o arquivo.



1. O sistema processará o arquivo e exibirá os contatos na **"lista de contatos"**.
2. Para finalizar a criação, clique em **"Criar lista"**.

Excluindo lista

Após a criação, as listas de contatos ficam visíveis na parte inferior da tela. Para excluir uma lista, clique no ícone da "lixeira" localizado ao lado do nome da lista desejada e confirme a ação quando solicitado. A exclusão é imediata e não pode ser desfeita.



Templates

Na opção templates, o sistema exibe todos os modelos de mensagens previamente aprovados na Gupshup e disponíveis para envio via WhatsApp. Através dessa tela, o usuário pode visualizar os templates cadastrados e iniciar um disparo de mensagens selecionando o modelo desejado.

Para realizar o cadastro de um template na Gupshup, siga as informações abaixo:

Criando template na Gupshup

1. Acesse o site da **Gupshup** e faça o login na plataforma.
2. Após o login, você será direcionado para a página de **Dashboard**.
3. Escolha a **API do WhatsApp** com a qual deseja trabalhar e clique na opção **"Overview"**.



Painel

Veja o desempenho de seus apps e gerencie-os diretamente através deste painel.

[+ Criar app](#)

ID do cliente : 401111111

[Registro de alterações](#)[Documentos da API](#)[Ajuda](#)[Mostrar tudo](#)

API de acesso

CentralIT

Limite de mensagens : N/D

abr 02 2025 | 3:40 PM

Não publicado

[Publicar](#)

Comentários

1. Para criar um novo "Template", clique na opção "Criar modelo".

**Modelos**[Opções de optin](#)[Perfil](#)[Configurações](#)[Webhooks](#)[Sandbox](#)[Conta](#)

Lista de modelos

Veja a lista de modelos que você pode usar ao enviar mensagens de modelo.
São aprovados pelo WhatsApp para o seu app.

[+ Criar modelo](#)

Nome do aplicativo

Centralit

Namespace

7af7eb63_ea16_3501_9510_146a72bea81a

Tipo do app

ACP

Comentários

Seus modelos[Modelos pré-aprovados](#)[Biblioteca de modelos](#)

1. Preencher os campos conforme descrito abaixo:

- **Nome do template:** Escolha um nome descritivo para o seu template.
- **Categoria:** Selecione a categoria adequada para o template, de acordo com as diretrizes da Meta (**Marketing** ou **Utilidade**).
- **Idioma:** Selecione o idioma do template.

- **Rótulos de modelos:** Insira um rótulo que facilite a organização dos templates.
- **Cabeçalho (opcional):** Adicione um título no topo da mensagem, se necessário.
- **Corpo:** Insira o texto principal da mensagem. Caso tenha variáveis, utilize os **marcadores** na seguinte forma: `{{1}}` , `{{2}}` , `{{3}}` ...

Exemplo de corpo com marcadores:

Text 

Oi `{{1}}`, seu pedido `{{2}}` foi confirmado para `{{3}}` às `{{4}}`.

- **Variáveis (opcional):** Se você adicionou **marcadores** no campo **Corpo**, expanda a seção Variáveis e defina um nome para cada um deles.
 - A ordem dos nomes deve seguir o índice dos marcadores. O primeiro ao `{{1}}` , o segundo ao `{{2}}` e assim por diante.
 - Esses nomes facilitam a identificação das variáveis e, uma vez definidos, passam a ser campos obrigatórios no momento do envio.
 - Na nossa plataforma, ao cadastrar sua **lista de contatos**, você poderá informar, para cada contato, um valor diferente para cada variável configurada.

Carrossel (Categoria Marketing): Formato interativo com até 10 cards, com imagem, título, legenda e botão(s). Caso opte por esse formato, preencha as seguintes informações:

- **Tipo de mídia do cabeçalho:** Atualmente, apenas o formato **imagem** é suportado.
- **Número de botões:** Defina quantos botões deseja incluir.
- **Tipo de botão:**
 - **Resposta rápida:** Envia uma resposta predefinida ao clicar.
 - **URL:** Redireciona o usuário para um site específico.
- **Card:** Adicione a **imagem** (máx. 5 MB), insira a **descrição** e configure os **botões** conforme o tipo escolhido.
- **Rodapé (opcional):** Adicione um rodapé para a mensagem.
- **Botões (opcional):** Adicione botões de resposta rápida ou ações.

1. Após preencher todas as informações, clique em **"Visualizar e enviar"** para revisar o conteúdo. Se estiver tudo correto, finalize clicando em **"Enviar"**.

Visualizar



Visualize seu modelo antes de enviá-lo para o WhatsApp

teste_de_envio

Categoria: MARKETING

Idioma pt_BR, Tipo: TEXT

Cabeçalho:

Teste

Modelo:

Olá, essa é uma mensagem de teste. Obrigado.[Mensagem recebida]

Exemplo:

Teste

Olá, essa é uma mensagem de teste. Obrigado.[Mensagem recebida]

Preview

Teste

Olá, essa é uma mensagem de teste.

Obrigado.

10:26 AM

➔ Mensagem recebida

Cancelar

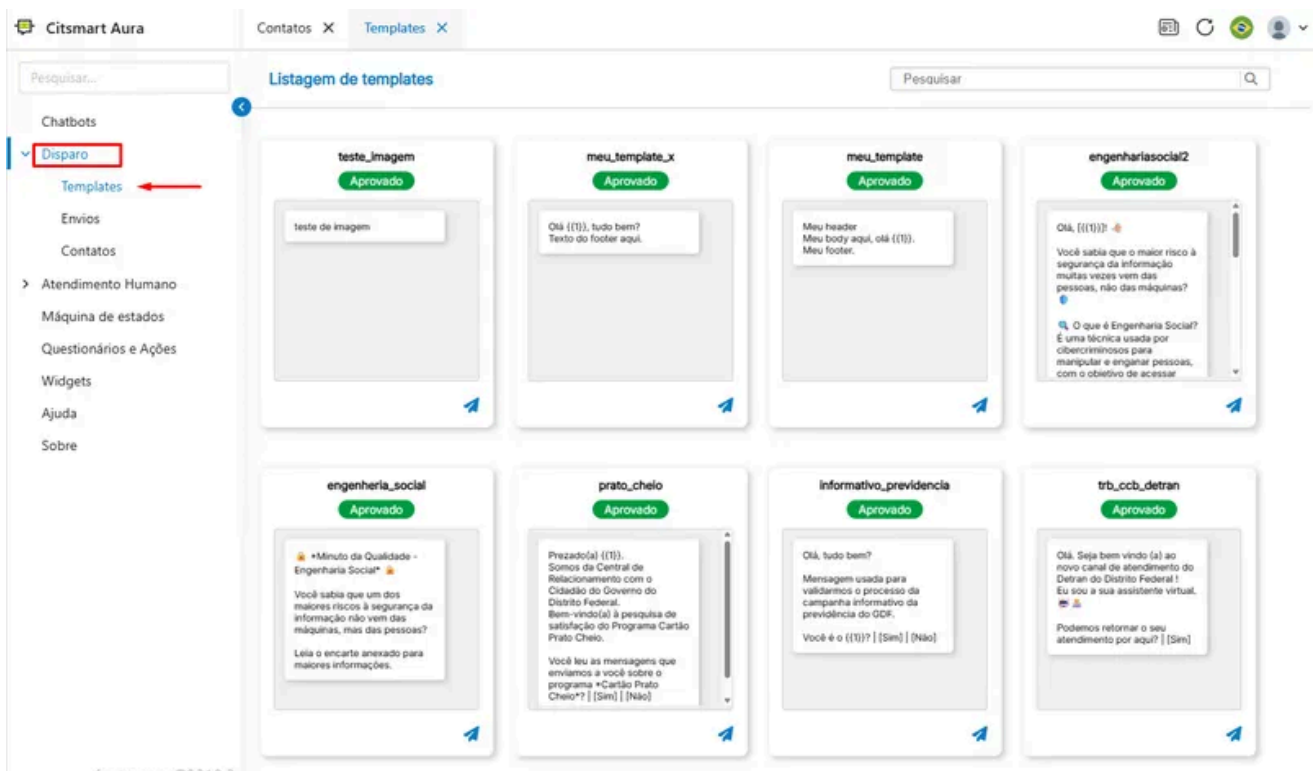
Enviar

Observações:

- As categorias de template seguem definições da Meta. Escolha a que melhor representa o propósito da mensagem.
- O conteúdo do template deve seguir as políticas e diretrizes da Meta para aprovação.
- A aprovação do template pode levar algum tempo.

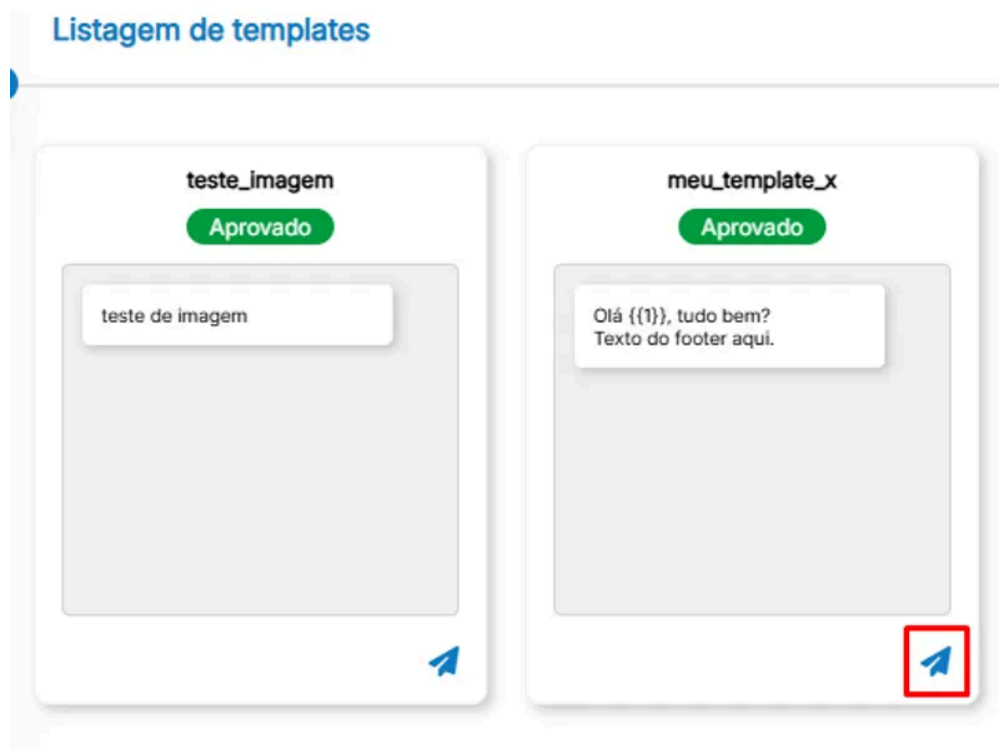
Citsmart Aura - Acessando menu "Template"

1. No menu lateral, clique em **Disparo > Templates**.
2. A tela exibirá todos os templates já aprovados.



Realizando o disparo

1. Localize o template desejado e clique no ícone azul, conforme imagem:



1. O sistema abrirá o seguinte formulário:

Contatos X Templates X

← Formulário de envio

Lista de contatos:

Citsmart aura

Horário de envio: 16:39

Data de envio: 30/06/2025

Criar disparo

Disparo criado com sucesso.
Disparo, será executado 30/06/2025, 16:39

1. Selecione a **lista de contatos** previamente criada.
2. Defina o **horário** e a **data de envio** desejados e clique em **Criar disparo**.

O sistema irá agendar o envio e exibirá uma confirmação. Você poderá acompanhar o status na aba **Envios**.

⚠ **Observação Importante** ⚠ Os templates utilizados nos envios são importados automaticamente da Gupshup. É necessário que o template seja criado, enviado e aprovado por lá antes de estar disponível na plataforma.

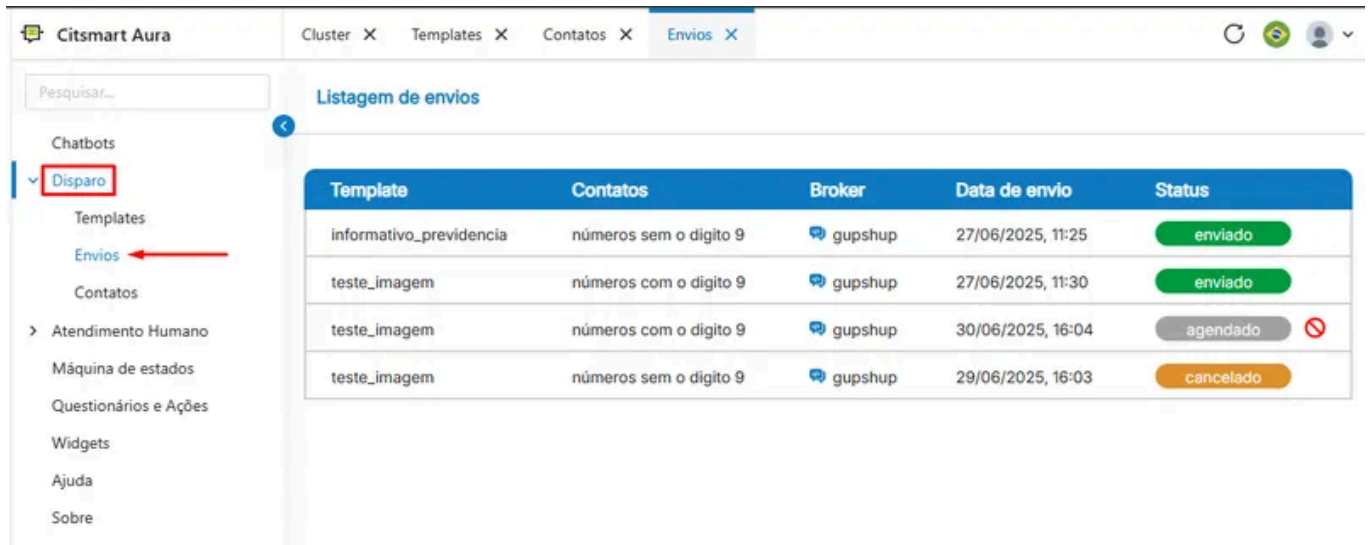
Envios

A tela de Envios permite acompanhar todos os disparos de mensagens realizados através da plataforma. Nela, o usuário pode visualizar o status de cada envio, identificando se a mensagem foi enviada com sucesso, cancelada, apresentou falha ou está agendada para um horário futuro.

Veja abaixo como acessar a funcionalidade:

Acessando os envios

1. No menu lateral, clique em **Disparo > Envios**.
2. A tela exibirá a listagem de envios agendados, concluídos ou com erro.




O sistema possui os seguintes status:

- **Agendado (cinza)** – Envio programado para data/hora futura.
- **Enviado (verde)** – Mensagem enviada com sucesso.
- **Falha (vermelho)** – Ocorreu um erro no envio.
- **Cancelado (laranja)** – Envio cancelado antes da execução.

Cancelar agendamento

Se o status estiver como **Agendado**, é possível cancelar o envio clicando no ícone de cancelamento, conforme imagem abaixo:

Data de envio	Status	
27/06/2025, 11:25	enviado	
27/06/2025, 11:30	enviado	
30/06/2025, 16:04	agendado	
29/06/2025, 16:03	cancelado	