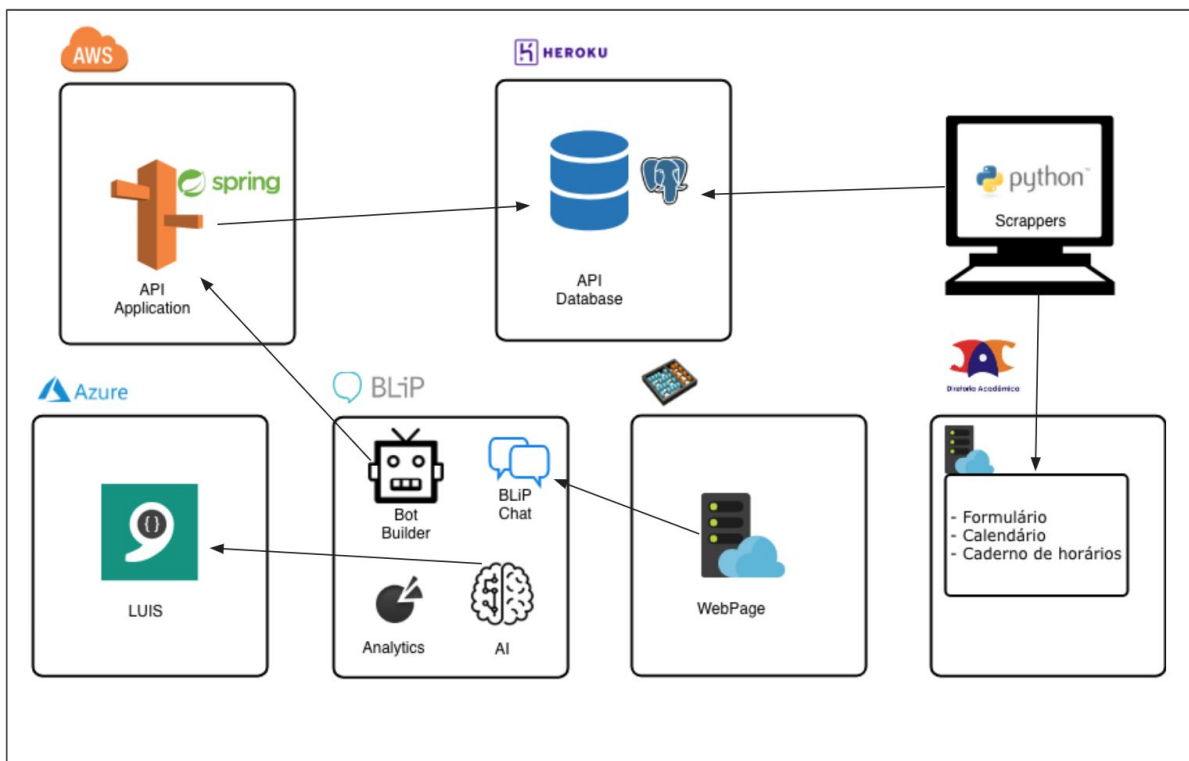


DACbot

Documentação e manual



1. Intenções, o que são:

O modelo de IA (inteligência artificial) pode ser composto por intenções e entidades. Basicamente, esses termos definem como a análise de uma frase (input do usuário) será feita.

Intenção, é aquilo que o chatbot identifica como sendo o desejo do usuário. Já a entidade representa parâmetros relativos a essa intenção identificada. Durante nossa análise do projeto proposto pela DAC, percebemos que, para o modelo de negócio em questão, não faria sentido definirmos nosso modelo de IA dessa forma.

Em virtude disso, optamos por fazer nosso modelo de IA (inteligência artificial) baseado apenas na definição de intenções, isto é, criamos intenções para todos os tópicos que o chatbot fosse abordar.

Para que nosso chatbot funcione, usamos como provedor de IA o LUIS, ele usa os exemplos que criamos para cada intenção e só consegue identificar as intenções que estão criadas e treinadas no BLiP.

2. Sobre a Ferramenta BLiP:

Segundo o site oficial da Take, que a criou, temos o resumo: “A Take criou a plataforma BLiP, que reúne diferentes funcionalidades e recursos para facilitar a construção, gestão e evolução de chatbots.

O BLiP foi pensado para unir o que há de melhor em experiências conversacionais, e todas as suas ferramentas seguem esse propósito. Da construção de chatbots sem código à publicação com um clique e à evolução ágil, com o BLiP é possível ter um contato inteligente de verdade para o seu negócio.”

Para além disso, o BLiP faz o intermédio dos seguintes elementos numa interface acessível sem a necessidade de escrever código(em negrito os elementos que serão mencionados neste documento e que são essenciais):

- As inteligências artificiais que ficam hospedadas na nuvem como o **LUIS** da Microsoft, o Watson da IBM e o Dialog Flow do Google. A escolha do LUIS para este chatbot foi econômica, por termos acesso a conta de estudante da Microsoft, pode ser mudado para outra IA.
- O **Builder** integra o fluxo do funcionamento responsável pelo tratamento de exceções, apresentação, despedida etc,
- Além disso o Builder cuida da integração de cada nó da árvore do fluxo com algo a se fazer ou mostrar, quando a intenção é reconhecida com sucesso. Isso inclui a integração com a API, os scripts, o BD, etc, caso seja necessário para a intenção em questão.
- Os **canais** que são aqueles escolhidos para ser a interface do bot com o usuário
- Os relatórios que são o **Analytics** do uso pelos usuários
- Além de gerir o **treinamento**, as **intenções**, fazer o aprimoramento e a **publicação** dessa IA.

3. Importação e Exportação no BLiP:

No BLiP existe a funcionalidade de salvar as configurações do fluxo no Builder. Após fechar uma nova versão, para exportar toda a configuração é só acessar a Configuração e Versão como nas imagens abaixo. Há opção de exportar o fluxo, o que gera um arquivo JSON contendo os dados dos blocos de intenções e como elas se relacionam. Esse arquivo deve ser armazenado em local seguro. Para importar as configurações que foram salvas, basta clicar em Importar Fluxo e selecionar no computador o arquivo de exportação JSON que foi gerado anteriormente. Automaticamente o fluxo será construído novamente, limpando qualquer configuração que estava vigente. Essa funcionalidade permite facilmente o versionamento do fluxo.

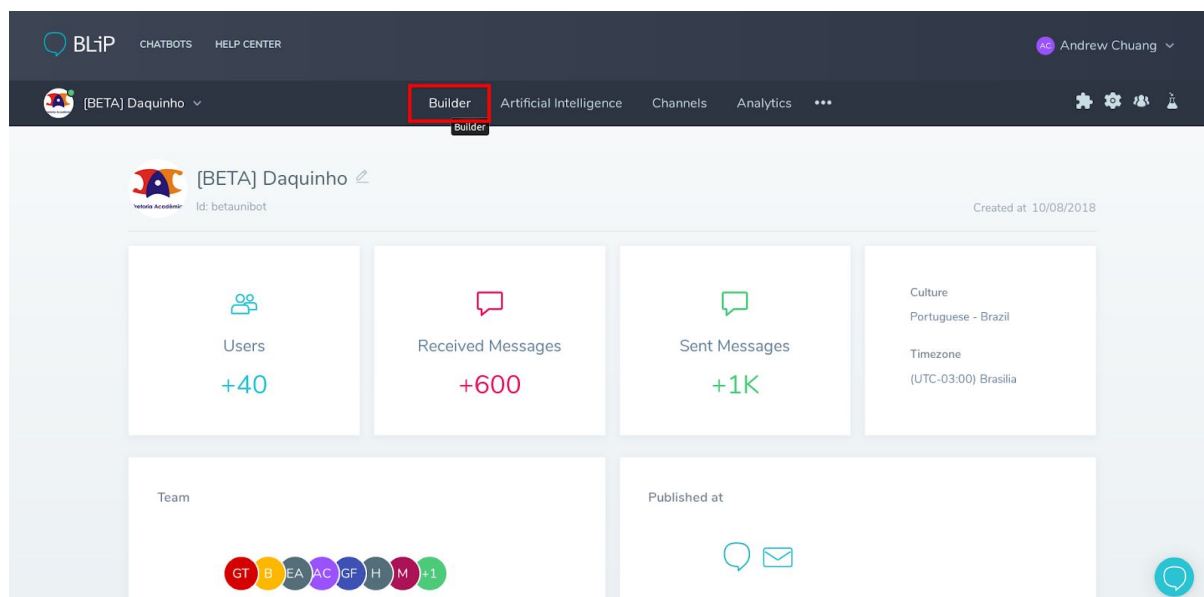


Figura 3.1 Acesso a importação e exportação de fluxo

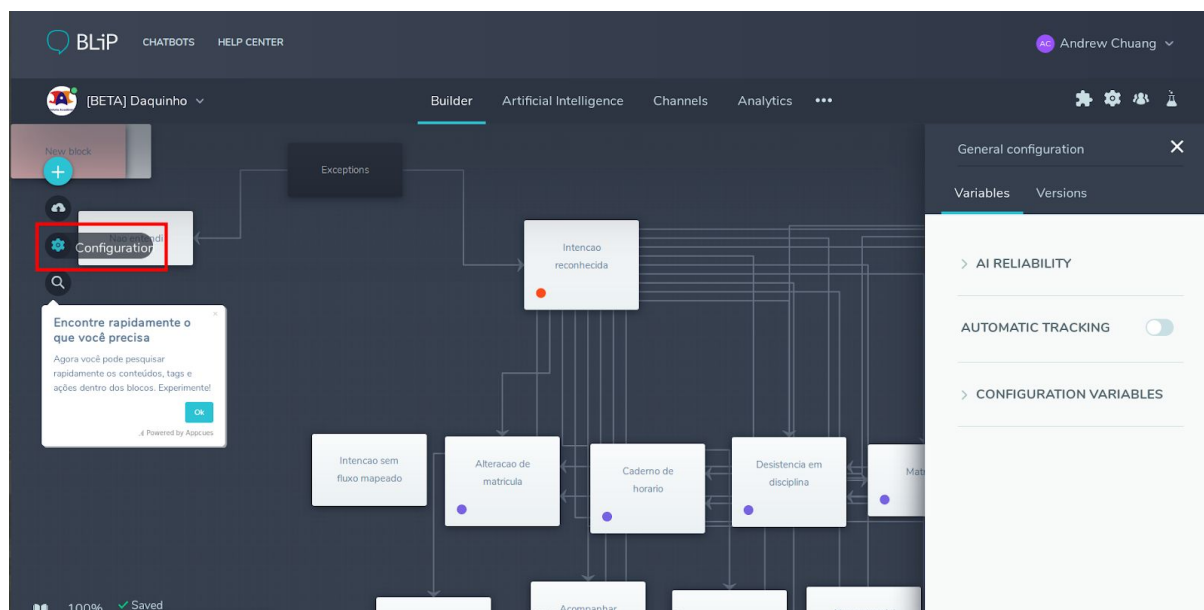


Figura 3.2 Acesso a importação e exportação de fluxo

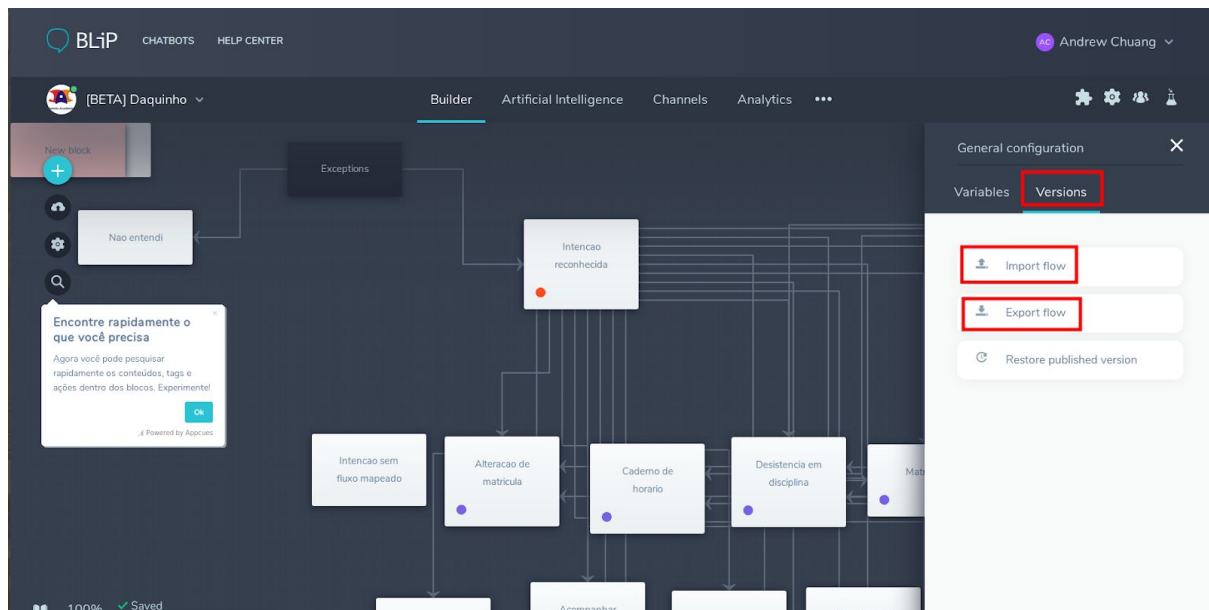


Figura 3.3 Acesso a importação e exportação de fluxo

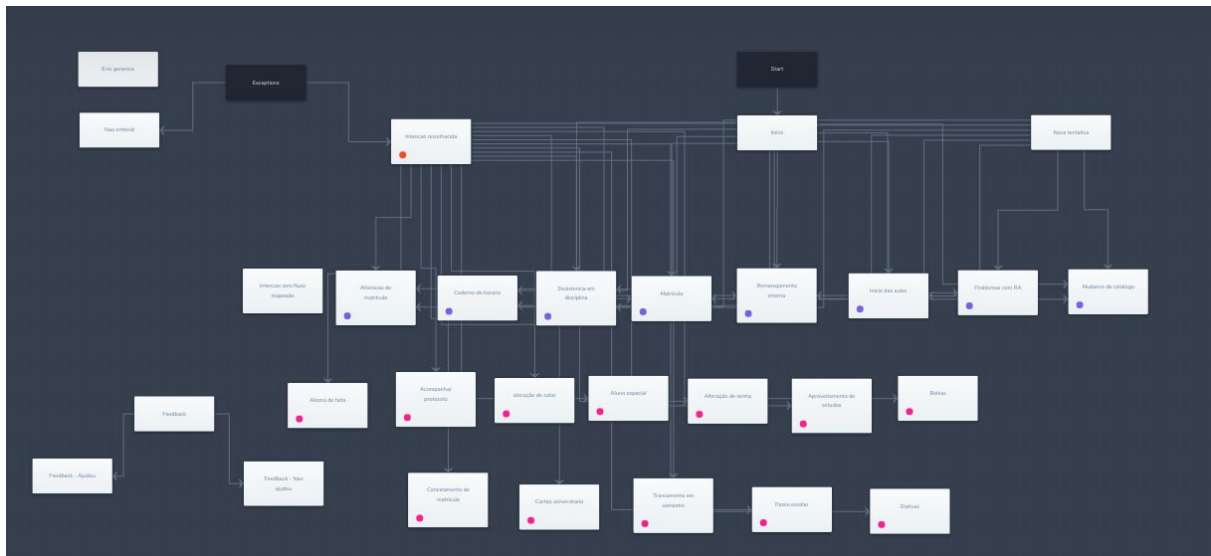
4. Links interessantes:

- Integrações da plataforma: <https://blip.ai/integracoes/>
- Suporte: <https://help.blip.ai/hc/pt-br>
- Documentações da API: <https://docs.blip.ai/#introduction>
- Fórum: <http://forum.blip.ai/>
- Visão geral do Builder:
<https://help.blip.ai/hc/pt-br/articles/360000677132-Visão-geral-do-Builder>
- Como criar um bot:
<https://help.blip.ai/hc/pt-br/articles/360001094371-Como-criar-um-bot-com-o-Builder-do-BLiP>
- Visão geral da Plataforma:
<https://help.blip.ai/hc/pt-br/articles/360000912352-Visão-geral-da-plataforma>
- Dúvidas sobre o Builder:
<https://help.blip.ai/hc/pt-br/sections/360000157291-Builder>
- Dúvidas sobre IA:
<https://help.blip.ai/hc/pt-br/sections/360000152772-Inteligência-Artificial>
- Portal: <https://portal.blip.ai/#/application>
- Preço: <https://blip.ai/precos/>

5. Fluxo, um tutorial:

O fluxo do blip está localizado na aba "Builder" como mostrado na figura 3.1

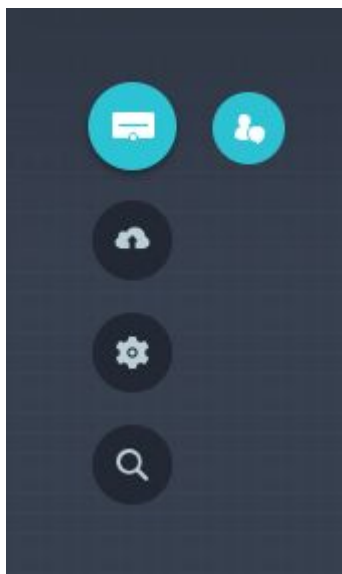
Atualmente o fluxo se encontra assim



O fluxo está dividido em 4 partes:

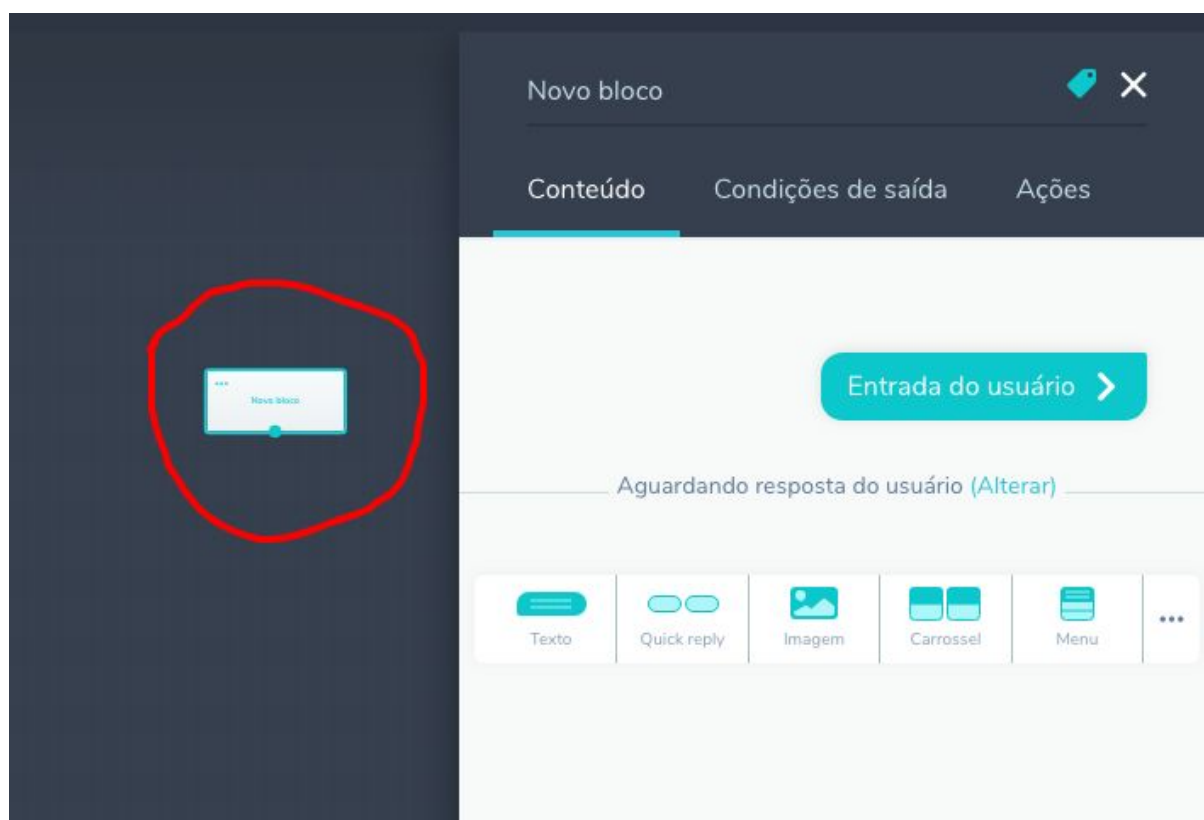
1. Blocos de botão rápido (roxo)
2. Blocos respostas de intenções (rosa)
3. Feedback (inferior na parte esquerda)
4. Entendimento das intenções (superior na parte esquerda)

Os blocos marcados em roxo são os botões rápidos vindo do bloco "início", enquanto os que estão marcados em rosa são as respostas vindo após a identificação da intenção. Nem todas as intenções estão mapeadas no fluxo, pois os blocos estão sendo usados apenas para respostas mais complexas.

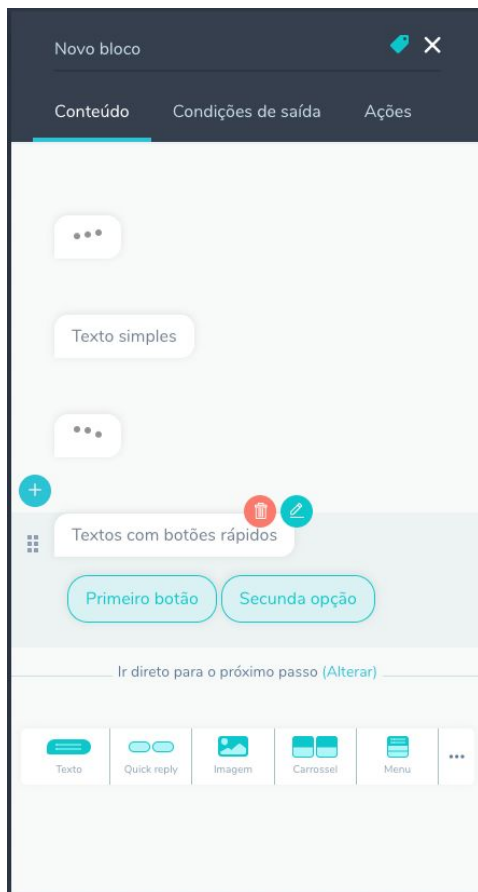


Na imagem, o botão azul claro selecionado na esquerda é para criar um novo bloco, enquanto o da direita é para a criação de um canal direto (conversar com alguém)

Ao criar um novo Bloco temos o seguinte:



O novo bloco está destacado em vermelho, enquanto as opções de configuração do bloco está no lado direito.



O nome do bloco está "Novo bloco" e do lado tem o botão para poder escolher as cores das marcações.

No conteúdo é possível configurar como vai ser a resposta do bot ao chegar nesse bloco. Nesse caso ao chegar no bloco o bot irá esperar uma entrada do usuário para depois fazer algo.

Para apagar a espera pela entrada do usuário é só colocar o mouse sobre o balão e irá aparecer um botão vermelho. As opções de resposta estão na parte inferior da imagem. E mais opções podem ser acessadas nas reticências do canto inferior direito.

Já em condições de saída estão as opções para que esse bloco vá para outro bloco. No nosso fluxo temos o exemplos de ir para outro bloco sem nenhuma condição, já no bloco "Intenção reconhecida" temos um exemplo onde colocamos todas as condições de saída dependem da intenção que foi reconhecida.

Intencao reconhecida

Important x

Adicionar tag...

Conteúdo

Condições de saída

Ações

CONDIÇÕES DE SAÍDA ⓘ

Se

Intenção identificad:

Condição

Igual a

Remanejamento Interno x

Valores

+

Ir para

Remanejamento interno

Se

Intenção identificad:

Condição

Igual a

Alteração de Matrícula x

Valores

+

Ir para

Alteracao de matricula

Se

Intenção identificad:

Condição

Igual a

Ao estabelecer uma condição de saída o bloco primeiro executa o que está em "Conteúdo" e depois testa as "Condições de saída" e na primeira condição que ele entrar ele vai para o próximo bloco, que é configurado em "Ir para" na parte inferior de cada nova condição de saída.

Caso nenhuma condição seja satisfeita, o bloco segue para a "Saída padrão" que é configurada na abaixo de todas as condições de saída.

Se

Intenção identificad:

Condição

Igual a

Eletivas x

Valores

+

Ir para

Eletivas

+ Adicionar condição de saída

SAÍDA PADRÃO ⓘ

Ir para

Intencao sem fluxo mapeado

A seta que liga os blocos não será exibida

6. Treinamento do Bot:

Agora vamos ver como fazer o modelo de IA, criando intenções, treinando-as, aprimorando-as e publicando o nosso modelo de IA.

Toda essa parte é feita no portal do BLiP, no Menu "Inteligência Artificial":

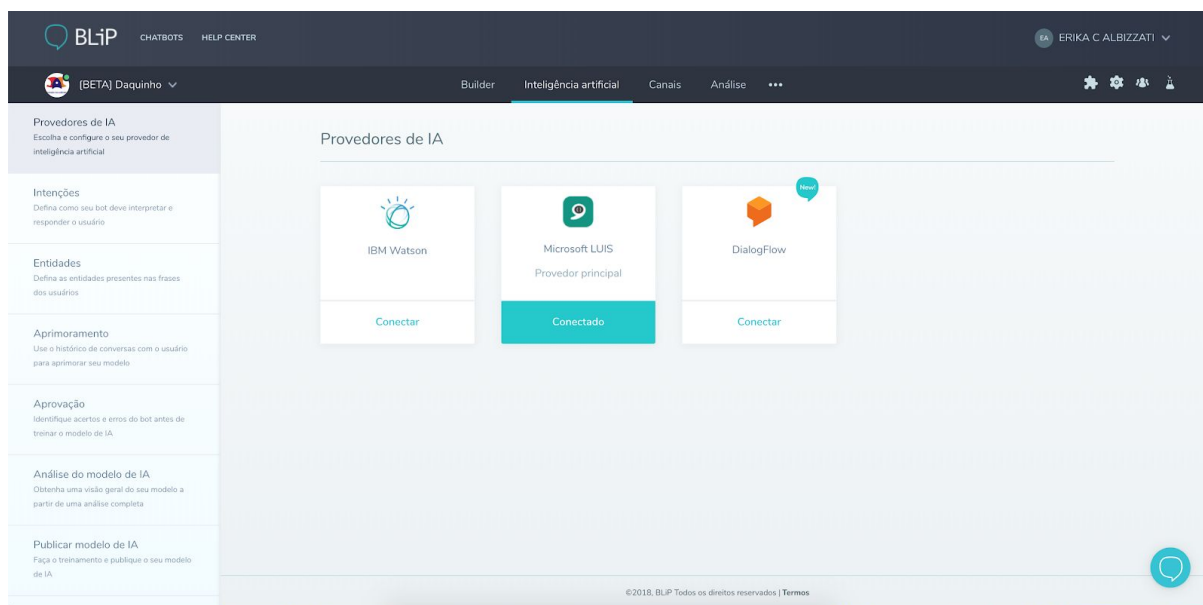


Figura 6.1: Inteligência Artificial. Já abre diretamente a parte de Provedores de IA. No nosso caso usamos o LUIS.

Na Figura 6.1 podemos ver os Provedores disponíveis para fazer conexão com o BLiP, indicando, se houver um, qual está conectado.

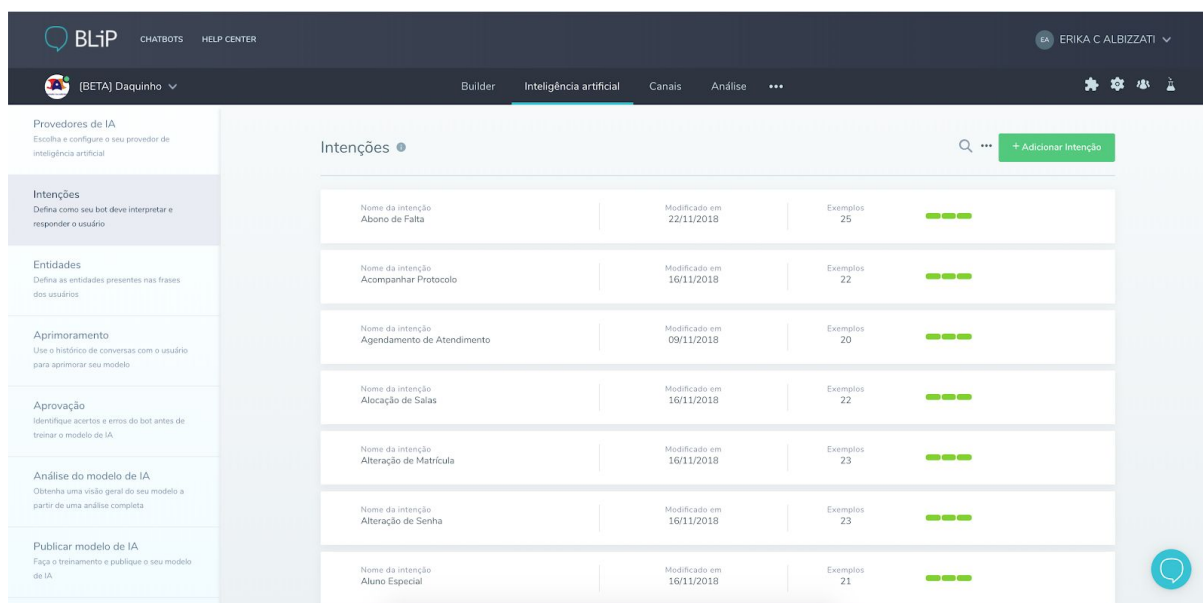


Figura 6.2: Menu esquerdo "Intenções"

Na Figura 6.2 podemos ver a lista de intenções já cadastradas, se houver alguma intenção que está precisando de melhorias (inclusão de mais exemplos), essas intenções ficarão no início da lista.

Podemos criar uma intenção pelo botão verde "+ Adicionar Intenção". Também podemos buscar uma intenção (é útil quando temos uma lista de intenções muito grande) pelo ícone de lupa, como é possível verificar na Figura 6.3.

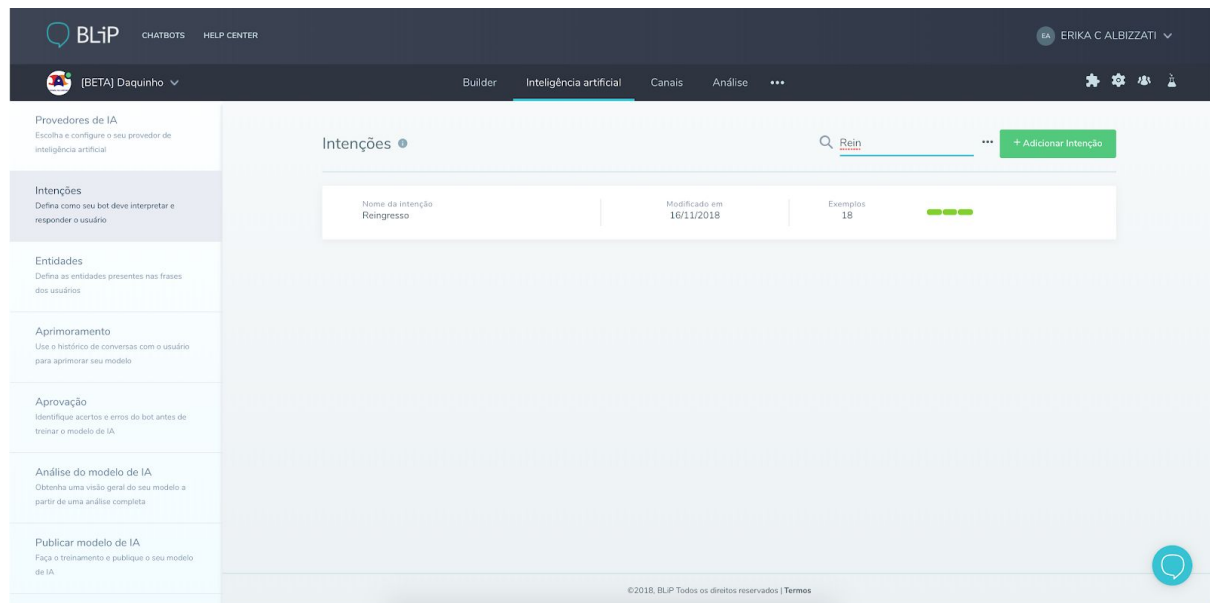


Figura 6.3: Busca textual por intenção.

Ao selecionar alguma intenção, somos direcionados para a tela de edição da intenção, onde podemos adicionar, editar ou remover exemplos e respostas. Podemos ver essa tela na Figura 6.4.

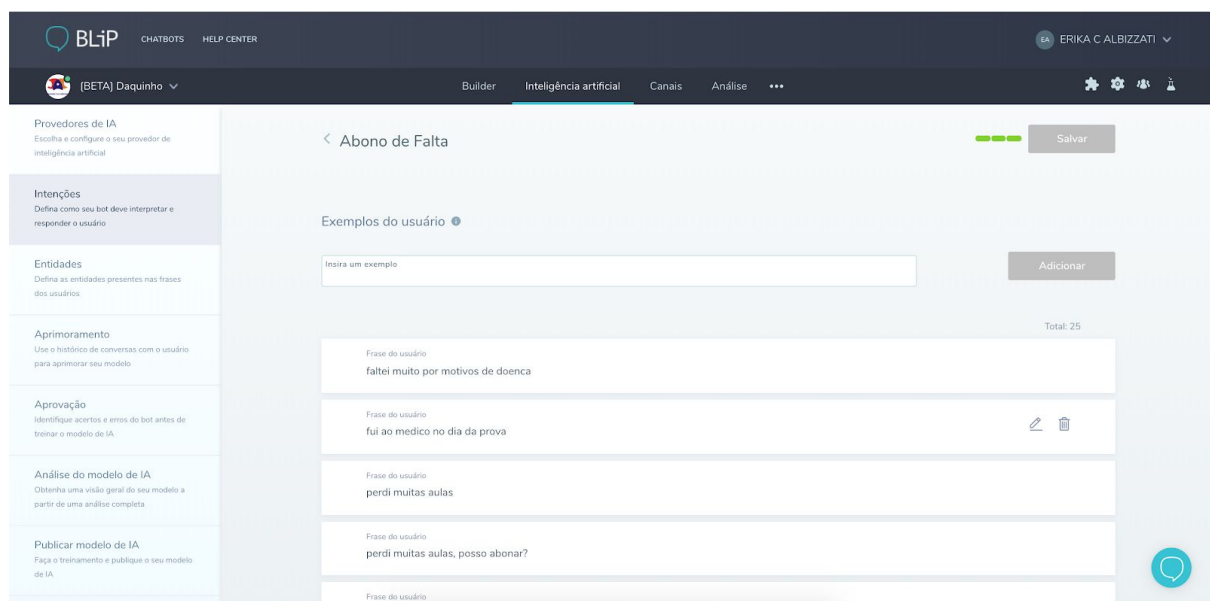


Figura 6.4: Detalhes de uma Intenção

Já o menu de Aprimoramento da Figura 6.5, nos permite avaliar a classificação feita pelo bot de todas as entradas de usuários. Podemos filtrar por período, confiabilidade e intenção.

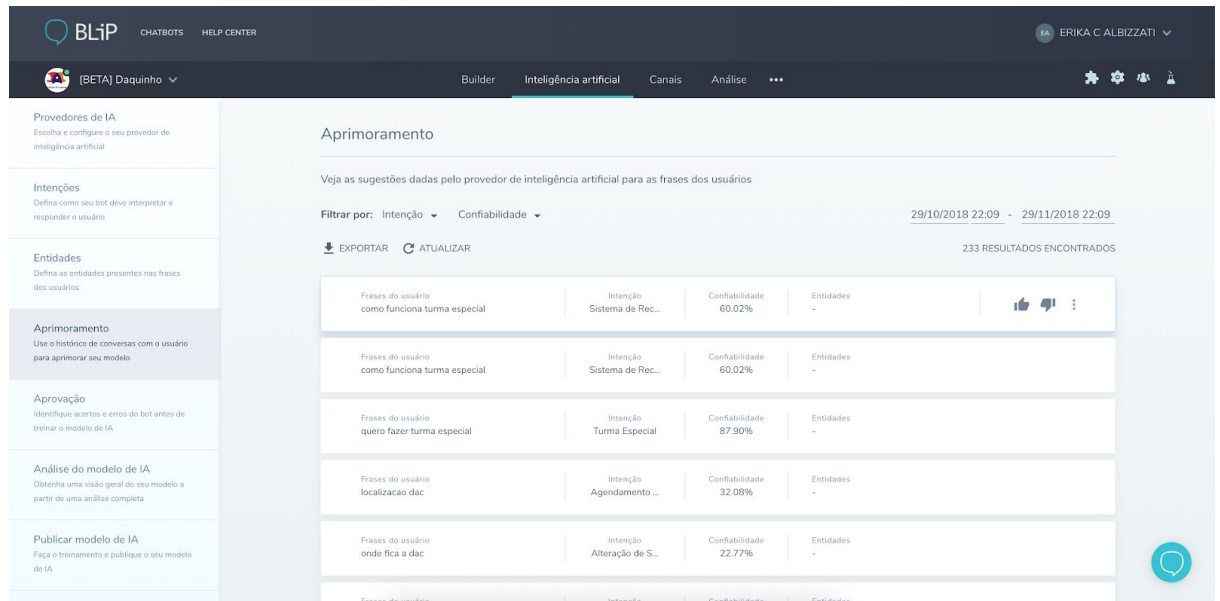


Figura 6.5: Menu de Aprimoramento.

Nessa tela, também é possível "concordar" (like) ou "discordar" (dislike) da classificação feita pelo LUIS. Se discordarmos, devemos classificar com a intenção correta, como vemos na Figura 6.6



Figura 6.6: Reclassificação da intenção ao discordarmos da intenção identificada pelo LUIS.

Quando discordamos de uma classificação, essa "correção" fica pendente de aprovação no menu "Aprovação", apresentado na Figura 6.7.

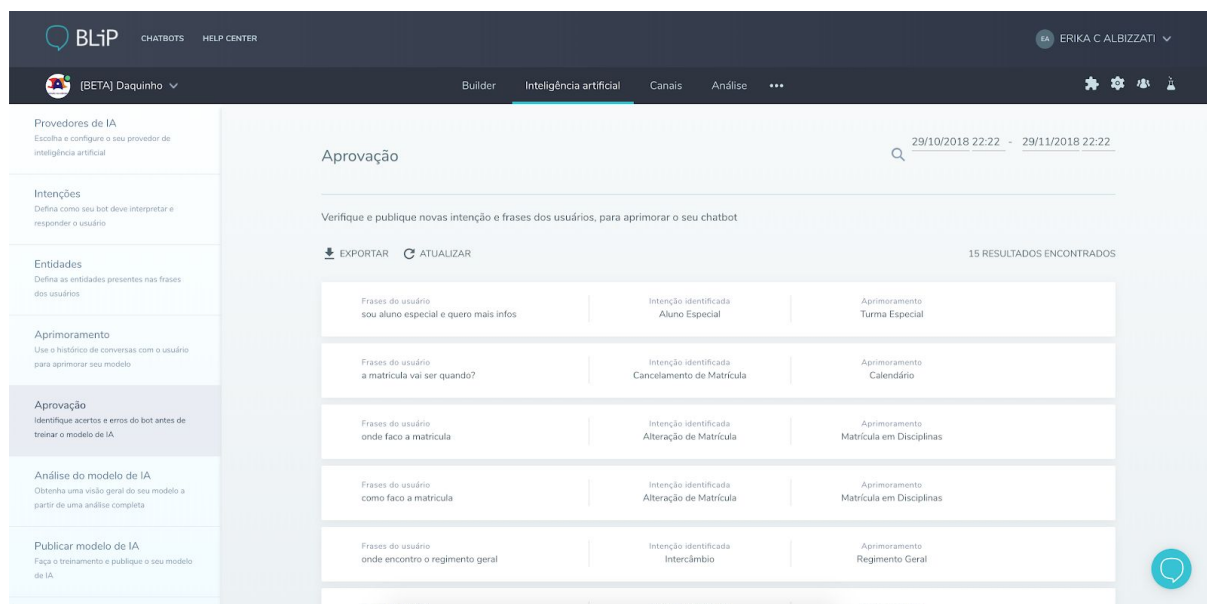


Figura 6.7: Menu de Aprovação

Nessa tela, assim como na tela de aprimoramento, é possível filtrar por período, exportar e atualizar o conteúdo.

Na Figura 6.8, é possível ver as ações que podem ser tomadas na tela de aprovação. Essas ações são para aprovar ou rejeitar.

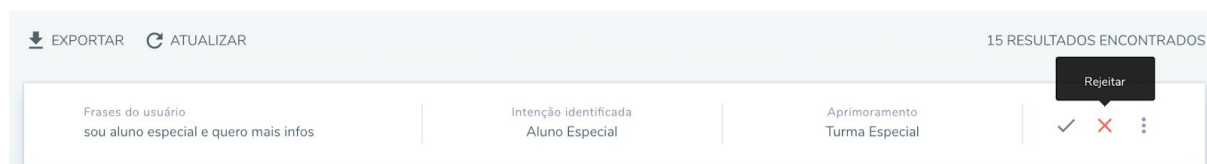


Figura 6.8: Ações no menu Aprovação.

No menu "Publicar Modelo de IA" da Figura 6.9, é onde toda a mágica acontece. É aqui que é possível ver o checklist do modelo atual, com insights sobre o balanceamento das questões. Ali ele sugere quantos exemplos devem conter cada intenção, quantas intenções estão abaixo, acima ou na média, além de total de intenções.

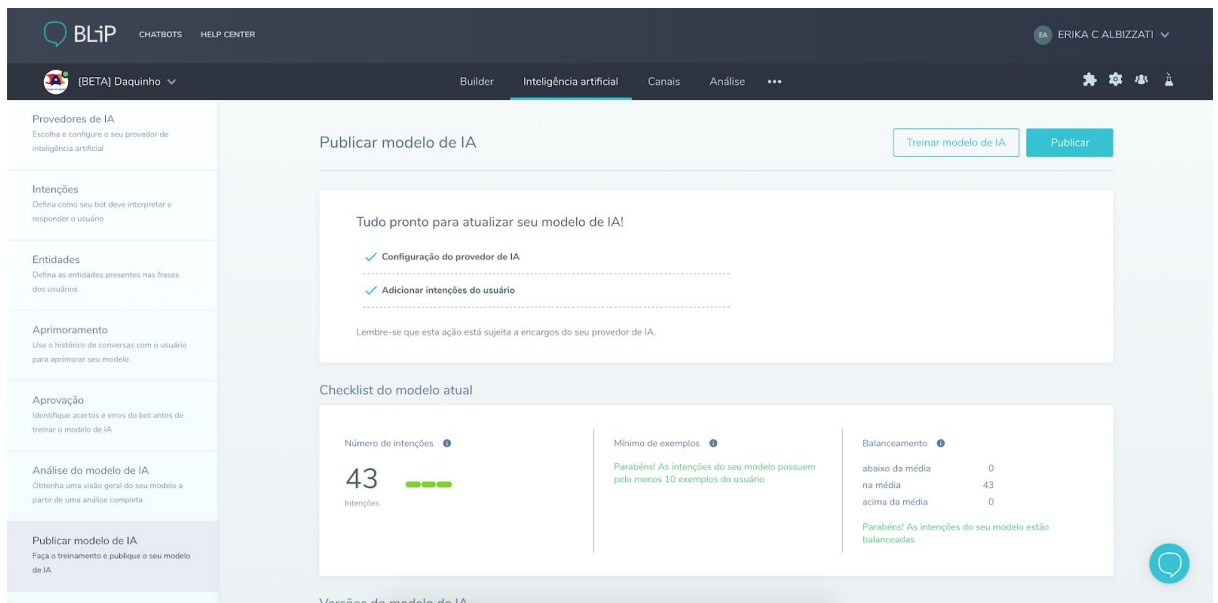


Figura 6.9: Menu "Publicar modelo de IA"

Para podermos perceber os efeitos gerados pelas alterações feitas nos menus anteriores, é necessário usar o botão "Treinar modelo de IA" e quando o treinamento terminar (pode demorar alguns minutos) usar o botão "Publicar". Pronto. Agora basta você testar suas alterações.

7. Usando o Analytics do BLiP:

Acessando a página de **Analytics** do BLiP (figura 7.1), podemos notar na barra lateral dois tipos de relatórios (figura 7.2): um dashboard nativo do BLiP e outra opção personalizável.

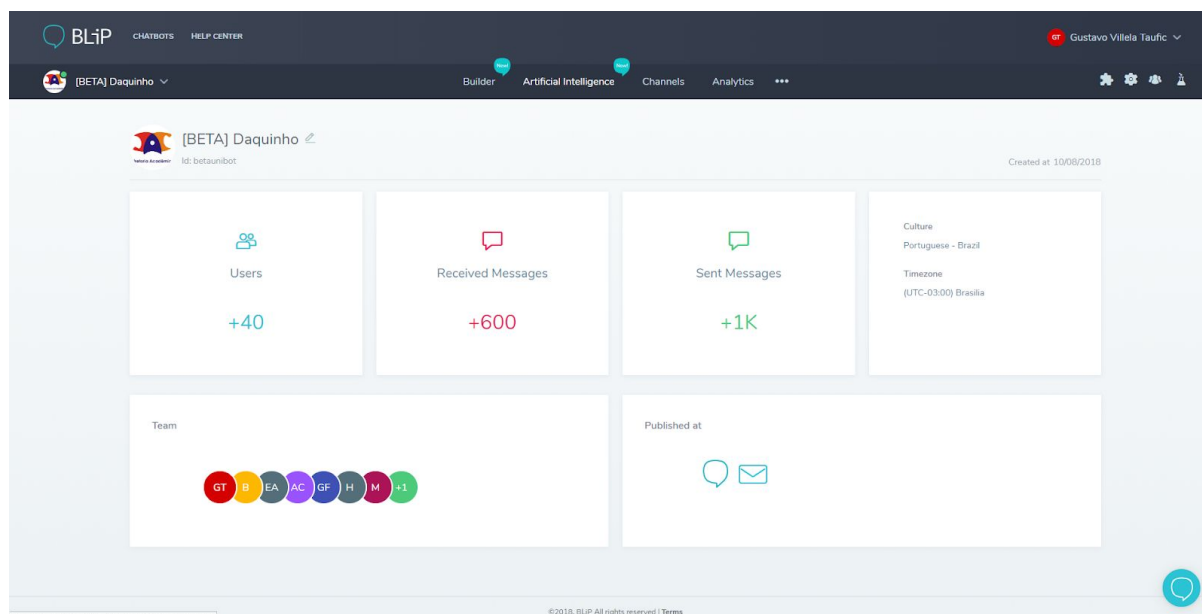


Figura 7.1: Como acessar a página de analytics.

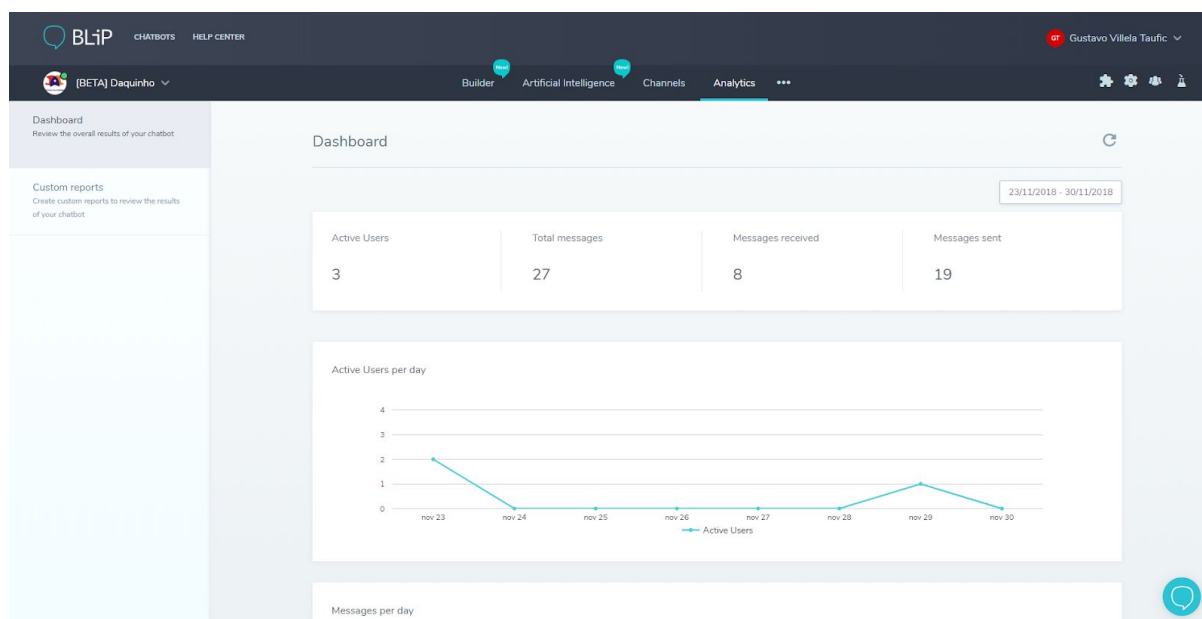


Figura 7.2: Tipos de Relatório.

Selecionando a opção personalizável (figura 7.3), podemos criar nossos próprios relatórios e métricas a serem mostradas. Assim, surge a possibilidade de mensurar

e acompanhar indicadores que sejam relevantes para evolução do chatbot ou aperfeiçoamento das regras de negócio deste.

No caso do projeto *Beta*, temos um primeiro relatório criado, expondo dados sobre assertividade de IA, atividade de usuários, acessos por pontos de fluxo e afins.

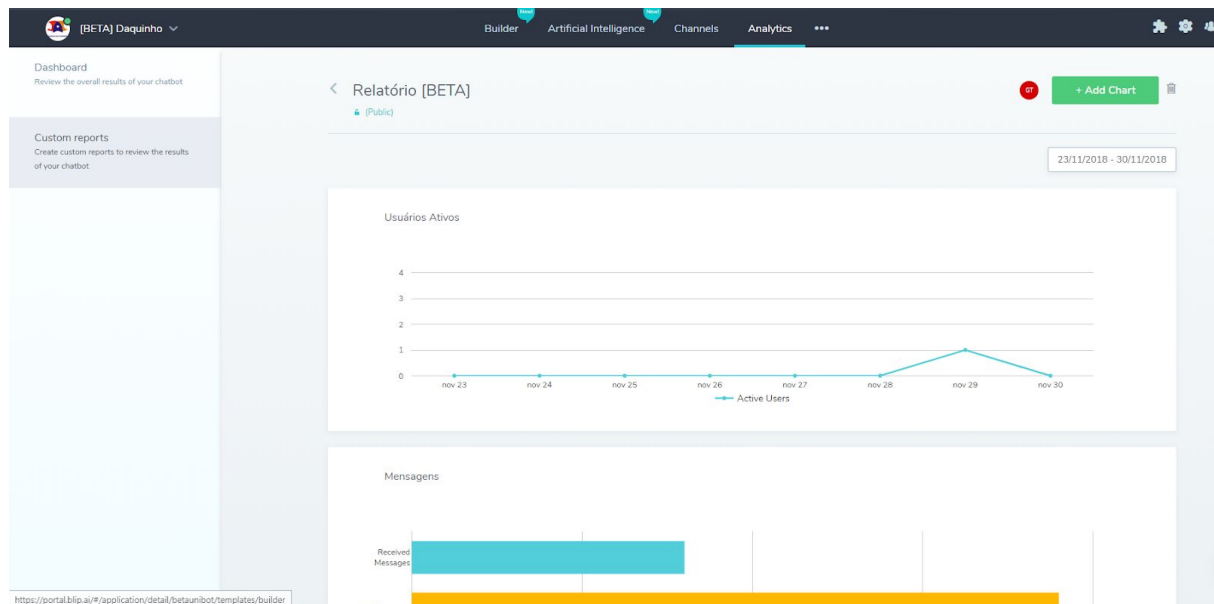


Figura 7.3: Relatório personalizado.

Acessando opção de adicionar gráfico, podemos selecionar qual tipo de visualização mais agrada o indicador a ser mostrado (figura 7.4). Feito isso, são configurados os nomes do gráfico e a grandeza a se acompanhar, podendo essa ser dados nativos do BLiP sobre mensagens, usuários ou os **eventos personalizados** (figura 7.5).

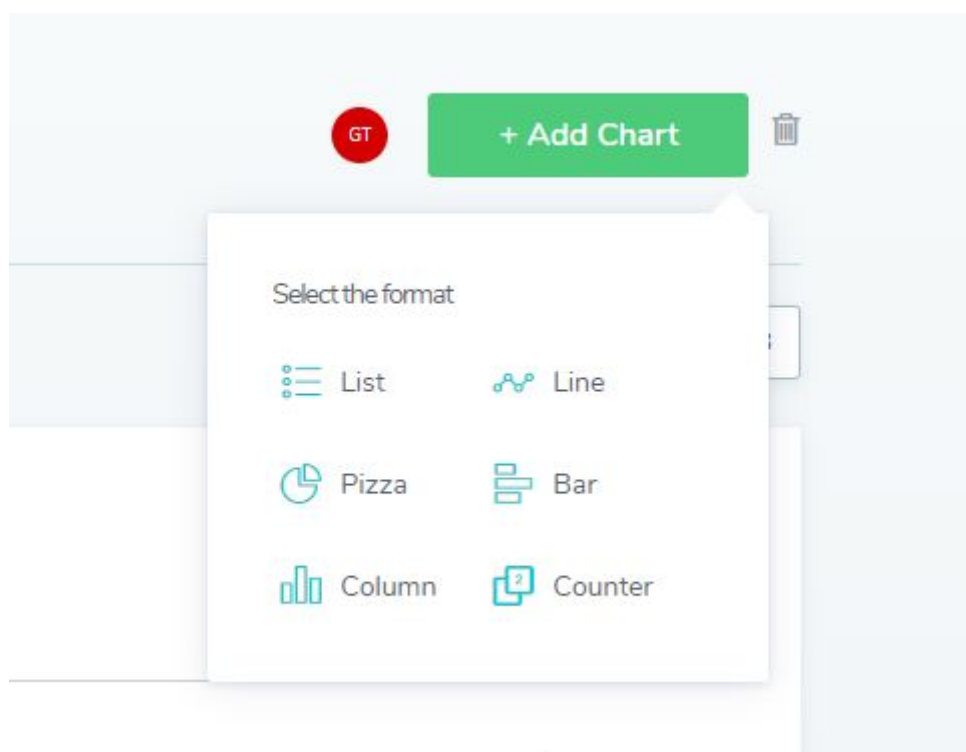


Figura 7.4: Tipos de representação gráfica.

×

Add chart Bar

Set up your chart with usage metrics from your chatbot or custom events.

Chart title
Fluxo

Dimension
Custom Events

Category
Fluxo

Cancel

Add

Figura 7.5: Adicionar gráfico.

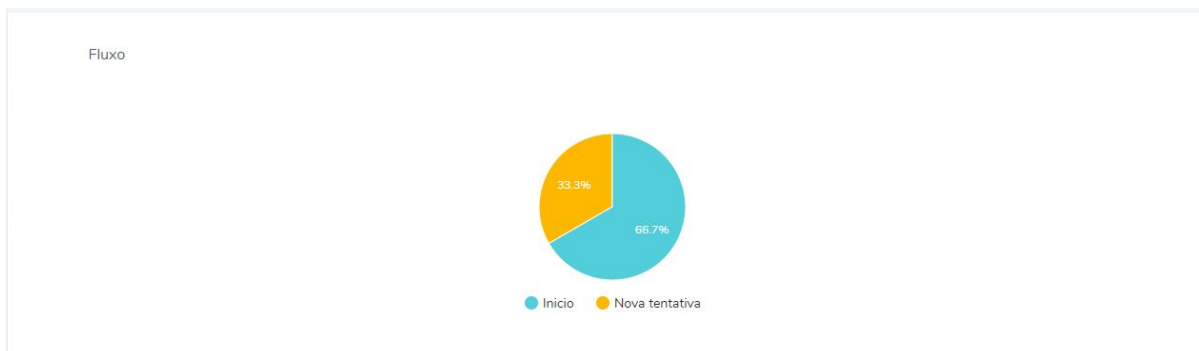


Figura 7.6: Gráfico adicionado.

Um evento personalizado é resultado da adição de um *tracking* no fluxo do builder, dentro das Ações de um ponto de fluxo (figura 7.7). Assim, que o ponto de fluxo que ele está colocado, foi acessado durante uma conversa com usuário, esse tracking será contabilizado.

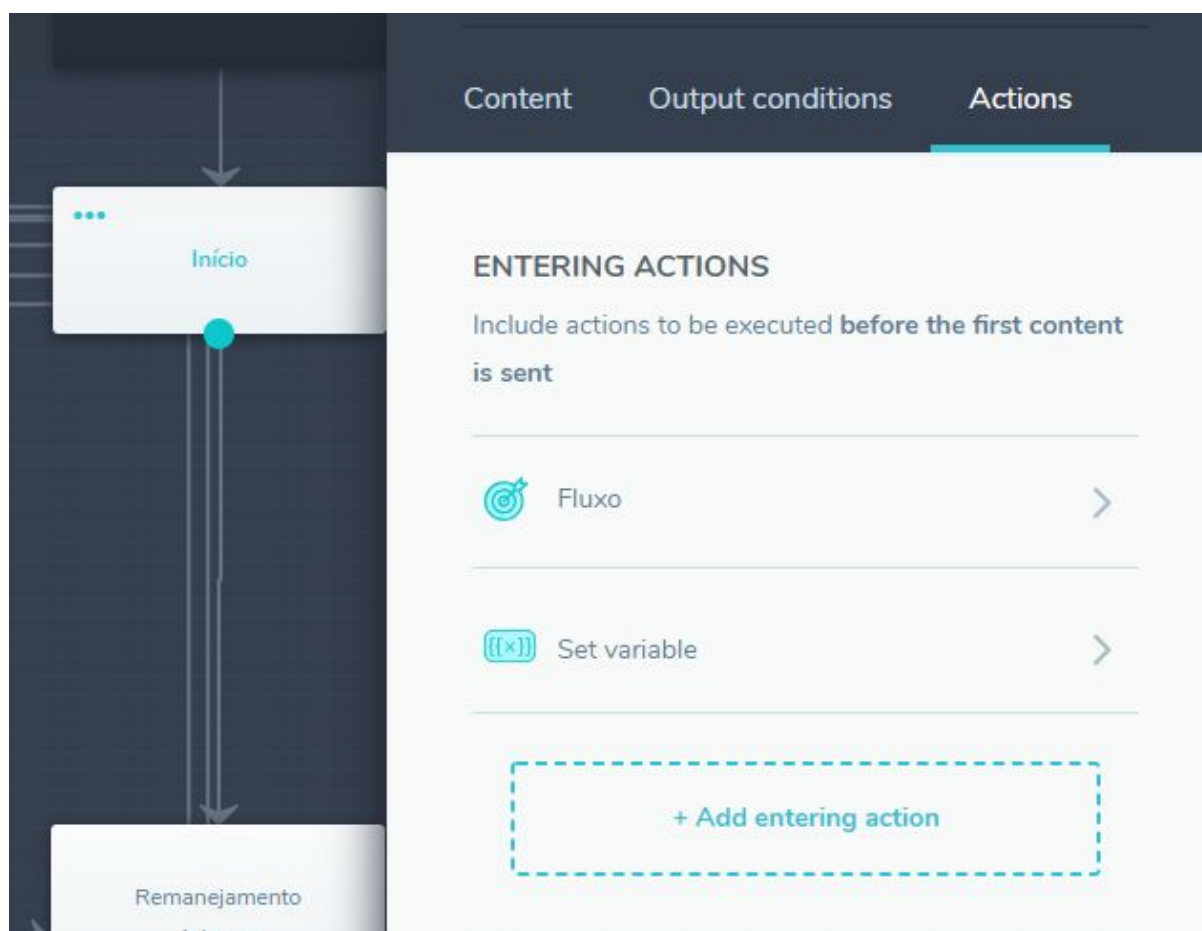


Figura 7.7: Como adicionar o tracking no fluxo.

Ao adicionar um *tracking*, este deve levar uma Categoria e uma Ação (Figura 7.8). Basicamente a categoria será um tabela de evento customizado e a ação será um

agregador para entender quantas vezes dentro de um evento, se deu tal ação. O resultado do exemplos ilustrado pode ser visto na **figura 7.6**.

Vale ressaltar que um evento personalizado só aparece para ser adicionado nos relatórios após esse estar incluso em um, ou mais, pontos de fluxo do chatbot e ter algum registro desde já feito.

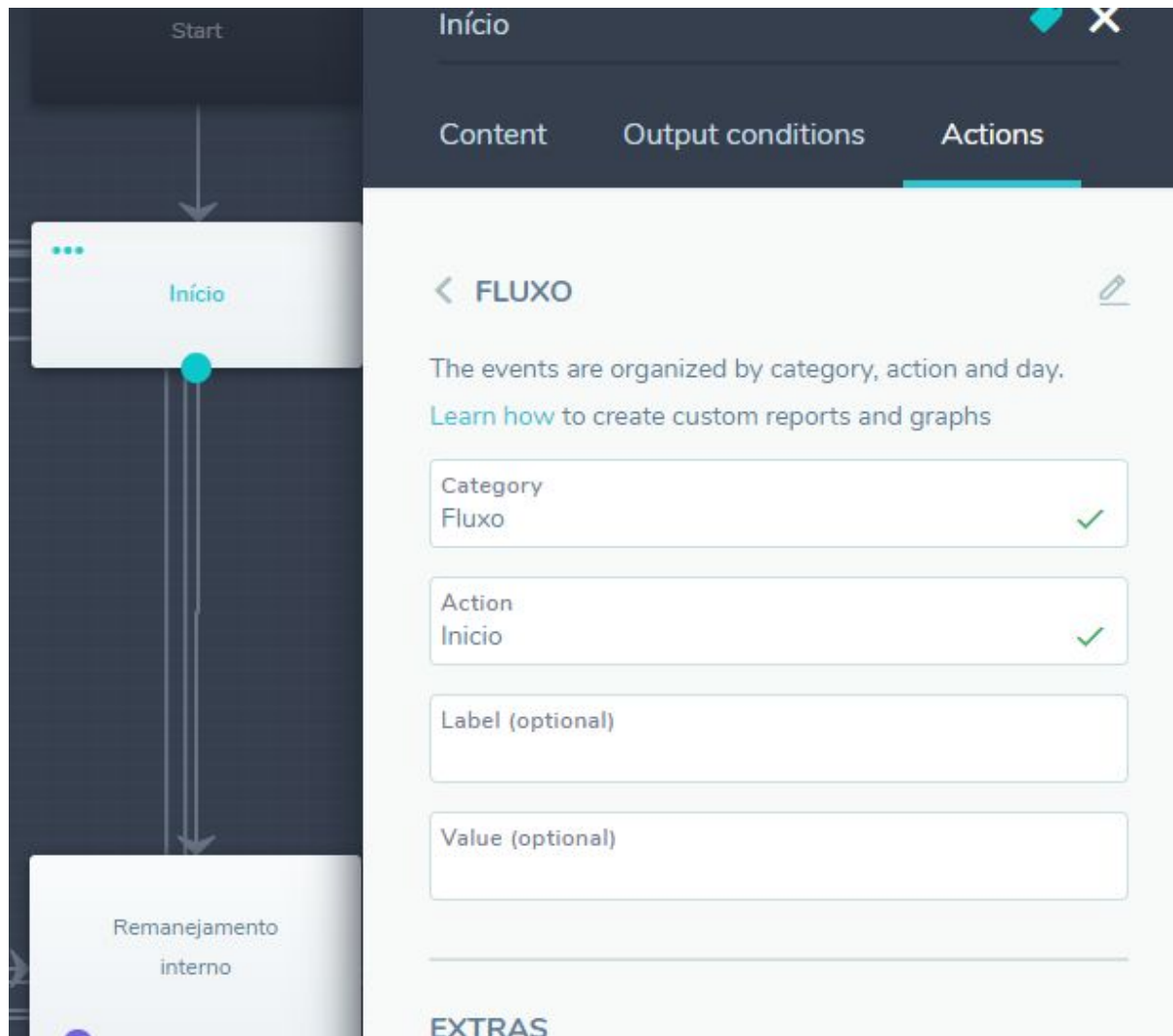


Figura 7.8: Escolhendo categoria e ação de um tracking.

8. Scrappers, BD e API:

Os scripts foram feitos usando:

- Python 3
- Beautiful Soup
- Requests

Foram usados os seguintes links para extrair as informações:

- Calendário: <https://www.dac.unicamp.br/portal/calendario/2018/graduacao>
- Caderno de Horários: <https://www.dac.unicamp.br/portal/caderno-de-horarios/>
- DAC FAQ: <http://www.dac.unicamp.br/portal/duvidas-frequentes/duvidas-frequentes-graduacao>
<http://www.dac.unicamp.br/portal/duvidas-frequentes/duvidas-frequentes-pos-graduacao>
- DAC Formulários: <https://www.dac.unicamp.br/portal/formularios>
- Regimento Geral Graduação: <https://www.dac.unicamp.br/portal/graduacao/regimento-geral>

Em geral, é necessário primeiro olhar o HTML do site, para então ver onde as informações estão localizadas, e como elas são referenciadas, por classe, código, alguma informação única.

O requests é encarregado de fazer a requisição e receber o HTML, depois usamos um parser para que o BeautifulSoup possa ser usado na procura e extração da página. O BeautifulSoup já tem funções para encontrar determinados elementos do HTML facilitando a extração de textos e outras informações como links, imagens.

Banco de Dados:

O PostgreSQL armazena as informações sobre as datas de atividades do Calendário. O Scrapper do Calendário também cria um script .sql que é usado para popular o banco de dados. Assim que as informações são carregadas nas variáveis, deve se escrever no arquivo as linhas de criação de registros no seguinte formato, exemplificado na linguagem Python:

```
f.write("INSERT INTO CALENDAR (YEAR_INTEGER, SEMESTER_INTEGER,
ENTITY_STR, DESCRIPTION_STR, INITIAL_DATE, END_DATE, URI_STR)
VALUES (%s, %s, '%s', '%s', '%s', '%s', '%s');\n" % (dataAno,
semestre, entidade, info, dataInicio, dataFim, url))
```

o que resulta na seguinte linha, por exemplo:

```
INSERT INTO CALENDAR (YEAR_INTEGER, SEMESTER_INTEGER, ENTITY_STR,
DESCRIPTION_STR, INITIAL_DATE, END_DATE, URI_STR) VALUES (2018, 2,
'vestibular_unicamp', 'Provas da 2ª fase do Vestibular Unicamp
2018.', '01/14/2018', '01/16/2018',
'https://www.dac.unicamp.br/portal/calendario/2018/graduacao');
```

Então é só executar o script SQL gerado no banco de dados para poder ser usado pela API.

O ideal é hospedar o banco de dados em alguma nuvem, como o Heroku Cloud. O mesmo permite acesso remoto gratuitamente.

API:

Diante da necessidade de se disponibilizar dados dinâmicos e o requisito de se ter um formato amigável para uso no chatbot, feito datas do calendário processadas pelo scrapper, surgiu a ideia de se ter uma aplicação voltada para tal.

Para buscar e fornecer tais informações, a API foi desenvolvida como mais um componente separado do Chatbot. Sua integração com o restante do sistema distribuindo se deu através da exposição serviços acessíveis através de requests HTTP e busca de informações no banco de dados suportado pela cloud Heroku.

No ambiente de "Beta", a aplicação esteve hospedada na AWS, através do serviço Elastic Beanstalk (Amazon Linux VM). Já o desenvolvimento se deu utilizando a linguagem Java e Spring Framework, um dois mais populares para desenvolvimento de aplicações web.

Respeitando as premissas do projeto, a utilização do serviço da AWS permite alta **escalabilidade** e **disponibilidade** de aplicações. Podendo assim, facilmente se criar novas instâncias desta ou aumentar a capacidade da máquina virtual que a hospeda.

Além disso, o serviço de cloud utilizado possibilita a configuração de diversas camadas de segurança, ambientes, alertas de funcionamento, análise de dados e muito mais.

Seguem alguns materiais explicando mais sobre:

- Vídeo introdutório sobre o EB:
<https://www.youtube.com/watch?v=SrwxAScdyT0>
- Recursos AWS EB: <https://aws.amazon.com/pt/elasticbeanstalk/details/>
- Preço EB: <https://aws.amazon.com/pt/elasticbeanstalk/pricing/>
- Criando e fazendo deploy de aplicações Java no EB AWS:
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create_deploy_Java.html

Para não gerar custos desnecessários, a aplicação só esteve hospedada na *cloud* durante alguns dias, em período de testes, apresentação e provas de conceito do projeto. No entanto, após ter acesso ao código fonte e ter o banco de dados configurado, esta pode ser executada localmente com as seguintes configurações e serem feitos testes com os serviços expostos:

- **Dependências:**

- Java 8.X
- Spring Framework 2.1.0
- Gradle 5.0
- Eclipse
- **Comandos** para configurações da API:
 - Após navegar até a raiz do projeto com o terminal, executar o seguinte comando para o Gradle Wrapper preparar o projeto:

```
./gradlew eclipse
```

- *Build*:

```
./gradlew clean build
```

- Obs.: Para definir o perfil de desenvolvimento *local*, incluir o parâmetro: `-Dspring.active.profile=local`

- **Exemplos dos serviços:**

- Busca de registro do calendário, por ID no Banco de Dados:

- Endpoint: `GET /calendar/{id}`

- Exemplo (requisição e resposta):

- Requisição:

```
GET http://dacbot.sa-east-1.elasticbeanstalk.com/calendar/5
```

- Resposta:

```
{
  "id": 5,
  "entity": "ferias_verao",
  "year": 2018,
  "semester": 1,
  "initDate": "2018-01-04",
  "endDate": "2018-01-04",
  "uri": "https://www.dac.unicamp.br/portal/calendario/2018/graduacao",
  "description": "DAC divulga na WEB relatórios de matrículas do período de Férias de Verão - 2018."
}
```

- Busca de registro do calendário, por nome da entidade no Banco de


Dados:

- Endpoint: `GET /calendar?entity=ferias_verao`
- Exemplo (requisição e resposta):
 - Requisição:

```
GET http://dacbot.sa-east-1.elasticbeanstalk.com/calendar?entity=ferias_verao
```

- Resposta:

```
[
  {
    "id": 2,
    "entity": "ferias_verao",
    "year": 2018,
    "semester": 1,
    "initDate": "2018-01-02",
    "endDate": "2018-01-03",
    "uri": "https://www.dac.unicamp.br/portal/calendario/2018/graduacao",
    "description": "Coordenadorias de Cursos adequam matrículas do período de Férias de Verão - 2018."
  },
  {
    "id": 5,
    "entity": "ferias_verao",
    "year": 2018,
    "semester": 1,
    "initDate": "2018-01-04",
    "endDate": "2018-01-04",
    "uri": "https://www.dac.unicamp.br/portal/calendario/2018/graduacao",
    "description": "DAC divulga na WEB relatórios de matrículas do período de Férias de Verão - 2018."
  },
  {
    "id": 6,
    "entity": "ferias_verao",
    "year": 2018,
    "semester": 1,
    "initDate": "2018-01-04",
    "endDate": "2018-02-21",
    "uri": "https://www.dac.unicamp.br/portal/calendario/2018/graduacao",
    "description": "Prazo para entrada de Médias e Frequências do período de Férias de Verão - 2018, na WEB."
  },
  {...}
]
```



Vale ressaltar que hoje, o conjunto API, Scrapper e BD é uma solução para a indisponibilidade de serviços nativos de busca de horários. Acreditamos que para uma evolução do chatbot seja interessante ter um serviço com conteúdo diretamente ligado à fonte de dados do portal da DAC e responsáveis pela gerência do calendário universitário.

Mesmo assim, a aplicação tem alguns pontos e serviços que servem como uma primeira estrutura do que o chatbot precisa para se manter através dos períodos letivos.