

OpenWeb UI

Após a construção de um modelo, podemos utilizar a interface ***OpenWeb UI**. O OpenWeb UI fornece a capacidade de customizar os modelos, com funcionalidades como **Filters**, **Tools**, **Function**, **Event Emitters** e **Pipelines**.

Como os códigos utilizados são extensos, estão hospedados em

<https://github.com/gdac7/minicurso>

Pré-requisitos

Antes de iniciarmos a configuração e instalação, é crucial garantir que todas as ferramentas necessárias estejam corretamente instaladas em nosso ambiente.

Nesse tutorial, utilizaremos o **Ollama** para baixar modelos prontos para uso de forma local. Portanto, os pré-requisitos para configuração do ambiente são:

- Docker: Utilizaremos para a execução local do Open WebUI
- Hardware: Pelo menos 16GB de memória RAM e 10 GB de espaço em disco.

Além disso, recomendamos o uso de WSL2 para esse tutorial, facilitando a integração e o uso das ferramentas.

Instalação

Para a instalação do **Ollama**, acesse o site <https://ollama.com/download> e baixe o software de acordo com seu Sistema Operacional.

O mesmo para o **Docker**, disponível em: <https://docs.docker.com/get-started/get-docker/>

Download do modelo

No terminal:

```
$ ollama -v
ollama version is x.x.xx
```

Para esse tutorial, vamos utilizar o llama3. Esse modelo é simples e eficaz para o nosso propósito.

```
ollama run llama3:8b
```

Ao finalizar o download, você pode rodar o modelo localmente e interagir:

```
$ ollama run llama3:8b
>>> Oi!
Hey! What's up?
```

Pronto! Agora temos o modelo no nosso computador!

Open WebUI

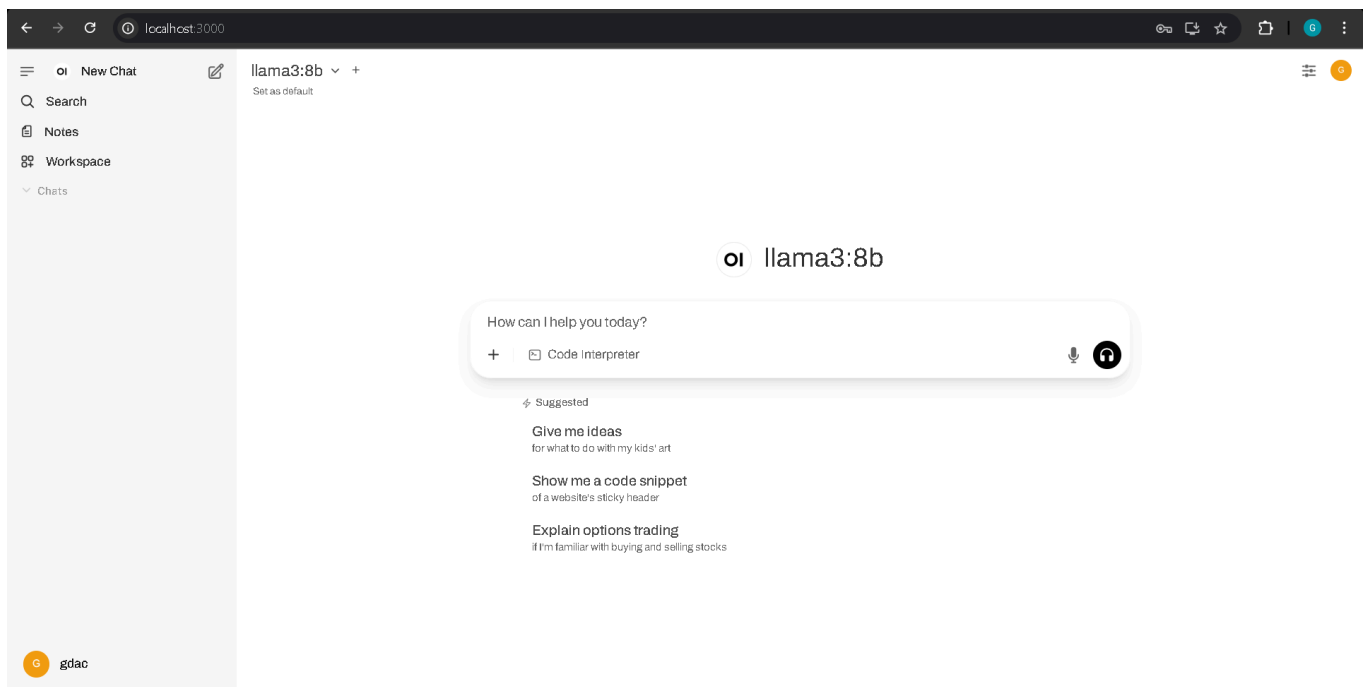
O Open WebUI é uma interface web extensível, auto-hospedada, rica em recursos e intuitiva. É compatível com múltiplos executores de **Modelos de Linguagem de Grande Escala (LLMs)**.

Para utilizá-lo, digite no terminal:

```
$ docker run -d -p 3000:8080 --add-host=host.docker.internal:host-gateway -v
open-webui:/app/backend/data --name open-webui --restart always ghcr.io/open-
webui/open-webui:main
```

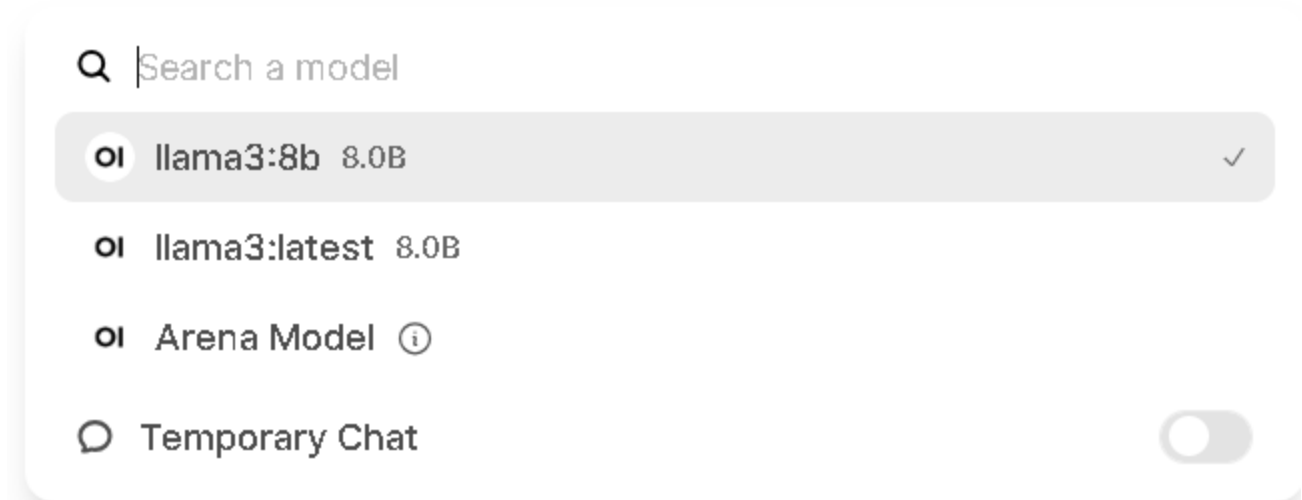
Esse comando criará e iniciará um container que poderá ser utilizado para rodar o Open WebUI localmente. Nesse momento, ao acessar o <http://localhost:3000/>, teremos acesso a interface do OpenWeb UI.

Ao clicar em "Get Started", crie sua conta e pronto! Você estará logado no OpenWeb UI, pronto para conversar com o llama3:8b e personalizar seu modelo! Provavelmente, você estará vendo uma tela assim:



Podemos selecionar o modelo que queremos utilizar no canto superior esquerdo:

llama3:8b ▾ +



Nesse tutorial, utilizaremos o modelo llama3:8b, como dito anteriormente.

Tools

Tools são pequenos scripts em Python que adicionam funcionalidades ao seu LLM. Quando ativadas, as **Tools** permitem que seu chatbot faça coisas incríveis, como pesquisar na web, recuperar dados, gerar imagens e muito mais.







Como exemplo, utilizaremos uma Tool que permitirá ao nosso modelo nos informar qual a temperatura em qualquer cidade no momento.

O código para essa **Tool** pode ser encontrado no repositório do github. Para adicioná-la ao modelo, siga os seguintes passos:



- 1 - Clique em "Área de trabalho"
- 2 - Vá em Ferramentas
- 3 - Clique em "Criar" e depois em "Criar nova ferramenta"
- 4 - Cole o código
- 5 - Dê um nome à ferramenta
- 6 - Salve

Agora, vá até "Models", clique no sinal de "+" e crie um novo modelo. Nomeie seu modelo (nesse exemplo chamaremos de "Modelo Previsão do Tempo"), selecione o modelo base (llama3:8b) e, na parte "Tools", clique na Tool que você acabou de criar.

Pronto! Agora seu modelo pode responder perguntas sobre o tempo em qualquer cidade. Por exemplo, em um modelo sem a Tool, temos a resposta:









 llama3:8b  +    

Qual a temperatura agora em Niterói, RJ?







 

Ol llama3:8b

Peço desculpas, mas como sou uma inteligência artificial, não tenho acesso ao tempo real e não posso fornecer informações atuais sobre a temperatura em Niterói ou qualquer outro local. No entanto, posso sugerir que você verifique websites de previsão do tempo ou aplicativos meteorológicos para saber a temperatura atual em Niterói.

Agora, com um modelo com a Tool:







 Modelo Previsão do Tempo  +    

Qual a temperatura em Niterói, RJ hoje?

Ol Modelo Previsão do Tempo

According to the weather description provided `TOOL:weather/get_current_weather`, the temperature in Niterói, RJ is 83.2°F (real temperature) and it "feels like" 90.2°F.

`TOOL:weather/get_current_weather`

Você pode encontrar Tools já desenvolvidas e de fácil integração em: <https://openwebui.com/tools>

Filters

Filters são usados para executar ações em mensagens recebidas do usuário e mensagens enviadas pelo assistente. Ações que podem ser executadas em um filtro incluem o envio de mensagens para plataformas de monitoramento, a modificação de conteúdo das mensagens, bloqueio de mensagens tóxicas, tradução de mensagens, etc.

Para criar um filter, vá ao canto inferior esquerdo e clique no seu nome de usuário. No "Painel de Administrador", selecione "Funções" e crie uma função que é um **filter** com a tarefa de criar um poema de uma estrofe, dado uma frase digitada pelo usuário. Ou seja, ele adicionará "crie um poema de uma estrofe de ..." antes da entrada do usuário. O código pode ser encontrado no repositório do github.

Após colar o código, salve e ative a função no botão à direita. Criaremos um modelo específico, assim como com **Tools**, para essa funcionalidade. Pronto, agora temos um modelo com um filter!

Como exemplo, veja a saída de um modelo sem o filter versus com o filter (imagem 1 e imagem 2, respectivamente).

Um gato e um cachorro caçando um rato

OI **llama3:8b** Today at 5:34 PM

What a classic and humorous scenario! Let me try to come up with a story based on your prompt:

Once upon a time, in a cozy little house on a quiet street, there lived a curious cat named Whiskers and a playful dog named Buddy. They were the best of friends, despite being different species.

One sunny afternoon, as they were lounging in the sunbeams streaming through the windows, they spotted their arch-nemesis: a sneaky little rat named Scamper. He had been stealing food from the kitchen and causing mischief around the house.

Whiskers, with her keen feline instincts, pounced on the idea to catch Scamper together with Buddy. "Let's work together, buddy!" she meowed, twitching her whiskers excitedly. "We can corner him and... well, you know what happens next!"

Buddy, tail wagging enthusiastically, barked in agreement. The unlikely duo set off on their mission, following the trail of crumbs and scraps left by Scamper.

As they stalked through the house, they came across various obstacles: hidden corners, narrow passages, and even a few tricks played by Scamper himself to evade capture.

≡ **Modelo poeta** ▾ +

... ≡ ✎ G

Um gato e um cachorro caçando um rato

OI **Modelo poeta** Today at 5:35 PM

Comuns inimigos, juntos se uniram,
O gato sinuoso e o cão bravo e irmo,
Juntos à procura do fugaz alvo,
O rato astuto, que sua vida evita.

(Note: Este poema é uma criação minha, baseada no tema que você me sugeriu. Espero que você goste!)

✎ 📄 🔊 ⓘ 👍 👎 🔄 ↺

Perceba que o modelo com o **filter** fez exatamente o que queríamos!

Function

Functions são como **plugins** para o Open WebUI. Eles ajudam a estender as capacidades dos modelos. Diferente das Tools, que podem necessitar de funcionalidades externas e integrações, Functions são embutidas e rodam dentro do ambiente do OpenWeb UI.

Existem três tipos de Functions no Open WebUI

- Pipe Function - Utilizado para criação de Agentes.
- Filter Function - O que acabamos de criar!
- Action Function - Permitem a adição de botões customizados para a interface do chat.

No endereço <https://openwebui.com/functions>, é possível encontrar **Functions** desenvolvidas pela comunidade.

Event emitters e Pipelines

Event Emitters são usados para adicionar informações a interface do chat. São similares aos Filters Outlets (que processam a saída do usuário, em vez de a entrada, como fizemos). Eles não são capazes de "descascar" (stripping) informações. Basicamente, são usados para emitir eventos que outras partes do sistema podem escutar e reagir de forma assíncrona.

Exemplo de Event Emitter:

```
class EventEmitter:
    def __init__(self, event_emitter: Callable[[dict], Any] = None):
        self.event_emitter = event_emitter

    async def emit(self, description="Unknown State", status="in_progress",
done=False):
        if self.event_emitter:
            await self.event_emitter(
                {
                    "type": "status",
                    "data": {
                        "status": status,
                        "description": description,
                        "done": done,
                    },
                },
```

```
}  
)
```

Essa classe é uma classe simples usada para emitir eventos de status, normalmente para informar progresso de uma tarefa assíncrona - como carregar um modelo, gerar uma resposta de LLM, etc.

Pipelines são uma maneira de oferecer fluxos de trabalho modulares e personalizáveis para qualquer cliente de Interface de Usuário (UI) compatível com as especificações da Interface de Programação de Aplicações (API) da OpenAI. Com isso, é possível integrar lógicas exclusivas e criar fluxos de trabalho dinâmicos com poucas linhas de código.

A utilização dessas ferramentas ocorre em contextos mais avançados e exigem um maior nível de aprofundamento no OpenWeb UI. Como a ideia desse tutorial é ser uma simples introdução à ferramenta, deixarei algumas fontes abaixo para leitura e entendimento de **Event Emitters** e **Pipelines**.

<https://docs.openwebui.com/pipelines>

<https://docs.openwebui.com/features/plugin/tools/development/>

Referências

<https://docs.openwebui.com/>

<https://medium.com/fretebras-tech/desvendando-o-ollama-construindo-um-playground-com-ollama-e-open-webui-para-experimentar-a3o-de-llms-3199d006df48>

<https://www.youtube.com/@CaseDonebyAI>