

# Watson AI Vehicle Financing Chatbot (PoC)

Murilo Zangari

## Watsonx: Plataformas Consideradas

- **Watson Assistant**  
Chatbot com fluxos estruturados, baseado em intents e entidades.
- **Watsonx.ai**  
Plataforma de IA generativa com foundation models e prompts.

# Objetivo

- Criar um chatbot simples sobre:
  - Financiamento de veículos
  - Parcelamento
  - Taxas de juros
  - Cálculos financeiros
- Responder perguntas **diversas e abertas**, com linguagem natural.








## Por que escolhi o Watsonx.ai?

- ✓ Maior flexibilidade com linguagem natural
- ✓ Mais alinhado com tendências de IA generativa
- ✓ Experiência prévia com Ollama e LLMs
- ✓ Diversos modelos para uso gratuito
- ✓ Integração direta via SDK (sem necessidade de endpoint).

## Modelo usado inicialmente: **granite-3-3-8b-instruct**

- Modelo de 8B parâmetros da família Granite, otimizado para seguir instruções e responder perguntas com clareza e precisão.
- Suporte a **português** e mais de 12 idiomas, ideal para o público-alvo.
- Capacidade de manter **conversas com histórico extenso** (até 131.072 tokens).
- Projetado para raciocínio lógico, explicações estruturadas e **aplicações em finanças, atendimento e educação**.
- Licença Apache 2.0 e código aberto via Hugging Face, permitindo transparência e futura evolução do PoC.

## Passo a passo do desenvolvimento

1.  Criei conta na IBM Cloud e acessei o Watsonx.ai Studio
2.  Explorei o Prompt Lab com o modelo gratuito `granite-3-3-8b-instruct`
3.  Validei um prompt base para perguntas sobre financiamento de veículos
4.  Salvei esse prompt como um **ativo tipo "Modelo de Prompt"** no projeto
5.  Criei um projeto local em Python
6.  Configurei variáveis de ambiente com `.env`
7.  Integrei o Watsonx via **SDK oficial** `ibm-watsonx-ai` (sem criar endpoint)

## Observação sobre o Ativo Criado no Prompt Lab

- O ativo salvo ( `Modelo de Prompt` ) no Watsonx Studio **não é um modelo funcional nem um serviço.**
- Ele funciona como um **repositório de referência**, útil para testes no Prompt Lab.

# Retrieval-Augmented Generation (RAG)

## Usado para:

- Para evitar alucinações e garantir respostas baseadas em fatos reais
- O modelo prioriza esse contexto ao gerar respostas, mesmo sem acesso à internet
- Melhora a precisão e confiabilidade do chatbot



## Problemas com Cálculos Iniciais

Durante os testes com o modelo `granite-3-3-8b-instruct`, observamos inconsistências em cálculos financeiros simples, como:

Parcela aproximada: R\$ 2.435,42 ✗  
Valor esperado: R\$ 1.807,62

- Isso indicava que o modelo aplicava a fórmula incorretamente ou cometia erros aritméticos.

## Troca pelo **mistral-medium-2025**

- ◆ Maior precisão matemática, especialmente com juros compostos.
- ◆ Melhor adesão a instruções específicas, como “calcule como uma calculadora Python”.
- ◆ Redução drástica nas alucinações numéricas.



## Ensinar o modelo a calcular

Atualizamos o `base_prompt.txt` com:

- Fórmula detalhada da parcela com juros compostos
- Instruções para exibir **todas as etapas**
- Exemplo explícito com resultado validado em Python:

```
(1 + 0.015) ** -36 = 0.586550  
1 - 0.586550 = 0.413450  
750 / 0.413450 = 1.815,03 ✓
```