Solução

- Chatbot que realiza elicitação de requisitos, ou seja, que conversa com o stakeholder para entender o problema e extrair os requisitos do sistema

Restrições necessárias do chatbot

- A ideia inicial é definir um domínio específico em qual o chat vai atuar, estou pensando em treinar para o contexto de e-commerce(mas poderíamos criar um contexto específico pare a saúde , para educação, ou qualquer outro), mas o importante é que nesses primeiros passos do chat ele seja voltado para um domínio específico para podermos explorar as outras funcionalidades mais importantes que falarei a seguir

- O segundo objetivo é fazer com que ele lide com apenas requisitos funcionais(e no futuro adaptarmos para os não funcionais também), tendo como output os requisitos no formato de User stories

- Outra limitação que iremos colocar é que estaremos sempre tratando de soluções em sistemas WEB

Estrutura do chat

- Modelos LLMs(aqui poderá variar , ainda faremos a decisão de qual modelo melhor poderá se desempenhar aqui)
    - Tamanho (quantidade de parâmetros) e capacidade de rodar localmente
    - Baixa taxa de alucinação.
    - Capacidade de manter contexto em interações multi-turn.
    - Open Source preferencialmente.
- langGraph/LangChain(ajudar a definir o fluxo da conversa e criar um sistema de feedback para o usuário ajudar a avaliar a resposta do modelo e assim ele conseguir treinar suas respostas a partir do uso)
    - Sistema de Feedbacks, para o desenvolvimento o sistema de feedbacks será baseado no estrutura de Util ou não util, ou seja, após gerar a user Storie tera dois emoticons um joinha positivo e um negativo e ao selecionar um deles enviaram um feedback para a máquina 👎 👍

- Fluxo Sugerido

Chatbot	Exemplo
Saudação + explicação da tarefa	"Olá! Estou aqui para te ajudar a definir as funcionalidades do seu sistema de e-commerce. Vamos começar?
Pergunta por uma funcionalidade esperada	Qual funcionalidade você gostaria que o seu site tivesse?"
Pergunta por objetivo	Legal. E para que serve essa funcionalidade?
Pergunta pelo usuário beneficiado	Quem vai usar essa funcionalidade? Um cliente, administrador...?
Geração de User Story	
Feedback	Avaliação de Feedback

- Interface do chat eu quero utilizar o Streamlit

- salvar as user stories em um excel ou docs

Id:	Titulo:	Role:	Eu quero:	Para:
US01	Adicionar produto ao carrinho	usuário	Adicionar um produto ao carrinho de compras	Possa finalizar minha compra

"id": "US01",
"titulo": "Adicionar produto ao carrinho",
"role": "usuário",
"Eu quero": "adicionar um produto ao carrinho de compras",
"Para": "possa finalizar minha compra posteriormente"

Cada coluna no excel / xlsx com um id, titulo etc…

- Em relação a escalabilidade ele vai salvar uma planilha para o MVP e em um futuro a gente realiza a integração com um banco mais complexo , seja um PostgreSQL ou algo assim

- Extra: eu quero adicionar a tecnologia speach-to-speach, após a realização do MVP e para isso pretendo utilizar alguma tecnologia S2S open source , porém este tipo de modelo não está tão preciso, caso necessário utilizaremos o da google cloud é pago , mas é mais robusto

Possiveis Problemas

- Falas ambíguas e utilizações de palavras genéricas(utilizaremos um fluxo para identificar algumas dessas situações e moldar o chat para contar isso, ex: o que você gostaria de falar com “produto bom”,)

1	Se o usuário disser “quero um sistema bom”, o chatbot pergunta:
2	O que você quis dizer com ‘bom’? Você está falando de facilidade de uso, segurança ou outra coisa?”
3	Banco de termos: Palavras como bom, rápido, intuitivo, melhor serão reconhecidas e o bot pedirá especificação.
4	O chat gera uma sugestão ao se deparar com isso
5	Quando você diz ‘um bom processo de pagamento’, você gostaria de algo como pagamento em um clique ou integração com várias bandeiras de cartão?”

- Fluxo conversacional quebrado A IA pode pular etapas, repetir perguntas ou sair do contexto Isso pode confundir ou cansar o usuário(feedbacks do usuário ajudarão a resolver esse problema)

- Alucinações, é muito comum este tipo de solução alucinar bastante, por isso as restrições indicadas no começo do documento são muito importantes para esse início de desenvolvimento , para limitarmos o escopo e irmos adaptando com o passar do tempo reduzindo assim as alucinações
- Limitações do Speach-to-Speach, reconhecimento semântico e de voz, e gastos com modelos que exigem a licença de uso , tendo isso em vista pretendo trabalhar nesse quesito após o MVP estiver concluído

Diferenciais de outros modelos

- diferente de outros LLMs , o chat seguirá um fluxo de conversa o que guiará o usuário, sem deixá-lo perdido

- as conversas serão personalizadas e adaptadas para este tipo de tarefa,(elicitação de requisitos, gerando user stories)

- Domínio especializado, fará perguntas guiadas baseado no domínio tratado(a priori e-commerce)

- Sua forma de aprender com os erros é mais eficaz, pois ele utiliza as interações multi-turn

- Geração automática de artefatos(documentos com as user stories)

- Integração com voz (Implementação após o MVP estiver pronto)

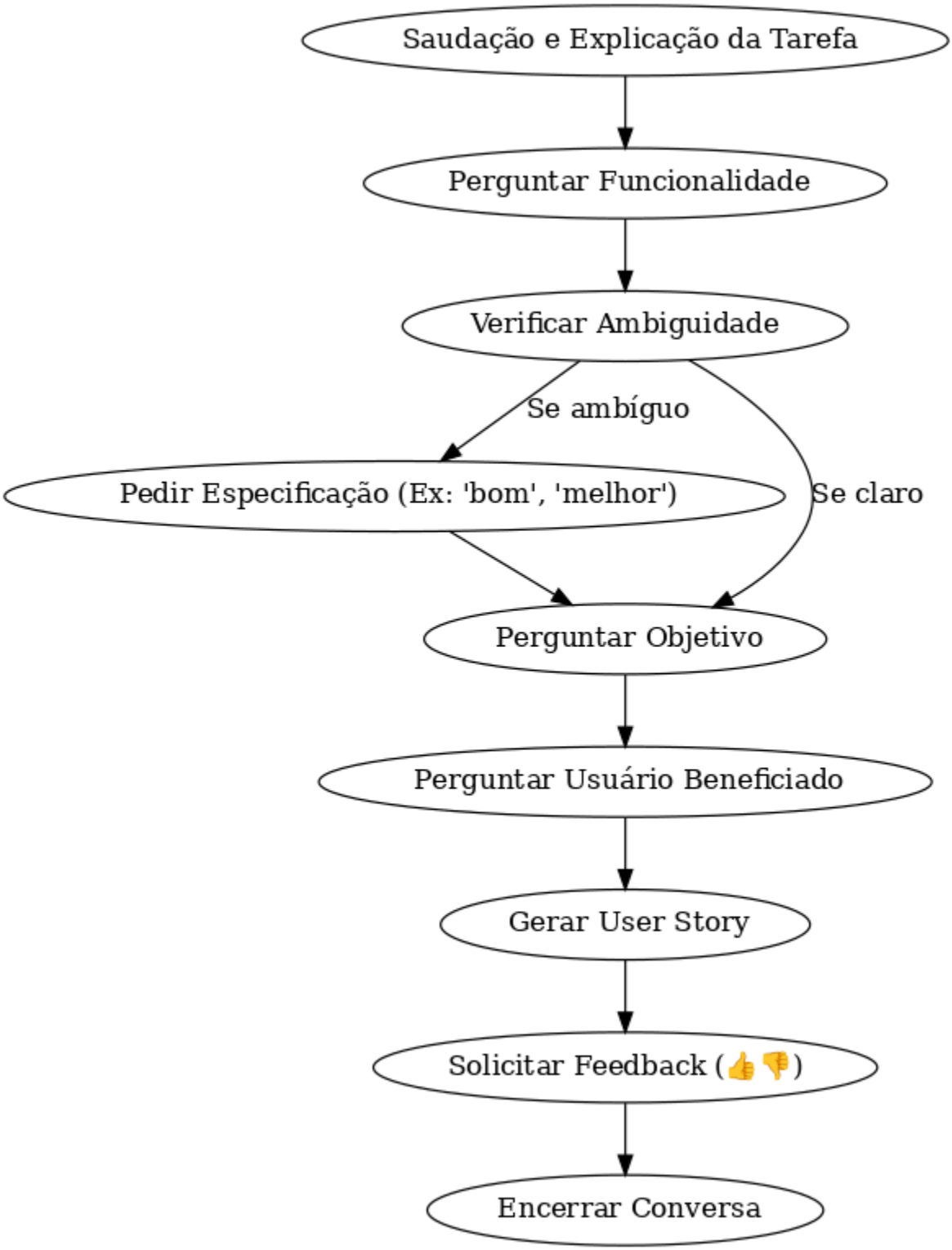
- Adaptável por domínio e empresa

MVP: Criar um chatbot simples que:

- Faça perguntas sobre requisitos funcionais de um sistema de e-commerce.

- Armazene as respostas do usuário.

- Gera um resumo básico ou um documento com os requisitos coletados.
- Receba feedback de cada resposta (útil ou não útil). 👎 👍



Critérios de Sucesso

A avaliação do desempenho do chatbot na elicitação de requisitos será feita com base em métricas claras e objetivas, considerando tanto a forma quanto a qualidade das *user stories* geradas.

1. Aderência ao Padrão de Formato

- Métrica: Percentual de *user stories* geradas no formato padrão:
  "Como [usuário], eu quero [objetivo] para [benefício]"
- EXEMPLO: Meta de sucesso: Alcançar ao menos 85% de *user stories* com estrutura válida, demonstrando entendimento correto do modelo esperado.

2. Avaliação com INVEST

- As *user stories* geradas serão analisadas conforme os critérios do framework INVEST:
  - Independente
  - Negociável
  - Valiosa
  - Estimável
  - Simples (Small)
  - Testável

- Métrica: Cada critério será avaliado de forma qualitativa por avaliadores humanos, ou semi automaticamente via checklist.
- EXEMPLO: Meta de sucesso: Pelo menos 80% das *user stories* devem atender a 5 ou mais critérios INVEST.

3. Avaliação com QUS Framework (Lucassen et al.)

- As *user stories* também serão analisadas com base nos 13 critérios de qualidade linguística definidos pelo QUS Framework, agrupados em três categorias:

SINTÁTICOS

- Bem-formada: Inclui papel e meio.
- Atômica: Refere-se a uma única funcionalidade.
- Minimalista: Contém apenas papel, meio e fim.

SEMÂNTICOS

- Conceitualmente coerente: Meio expressa funcionalidade e fim expressa justificativa.
- Orientada ao problema: Não apresenta soluções.
- Não ambígua: Livre de termos genéricos ou vagos.
- Livre de conflitos: Sem contradições com outras *user stories*.

PRAGMÁTICOS

- Frase completa: Estrutura gramatical correta.
- Estimável: Clara o suficiente para estimativas.

- Única: Sem repetições.
- Uniforme: Segue o mesmo modelo/template das demais.
- Independente: Sem dependências explícitas.
- Completa: Conjunto de histórias gera sistema funcional.
- Métrica: Percentual de *user stories* que atendem a ao menos 10 dos 13 critérios.
- EXEMPLO: Meta de sucesso: Atingir no mínimo 75% de conformidade com os critérios do QUS Framework.

4. Feedback do Usuário

- O chatbot deve coletar feedback após cada resposta, classificando-a como útil ou não útil.
- Métrica: Percentual de respostas avaliadas como úteis.
- EXEMPLO: Meta de sucesso: Obter mínimo de 80% de respostas úteis, indicando que o fluxo está adequado à compreensão do usuário.

[ ] definir escolha”e-commerce” ou escolher outro contexto :

Requisitos bem delimitados e padronizados

Sistemas de e-commerce apresentam um conjunto de requisitos funcionais recorrentes e bem definidos, o que reduz ambiguidades e facilita a implementação de estratégias de elicitação automatizada. Isso é essencial para um MVP que foca inicialmente em requisitos funcionais representados no formato de User Stories, como:

"Como cliente, eu quero adicionar produtos ao carrinho para finalizar minha compra posteriormente."

Alta aplicabilidade para testes e validação do chatbot

Por ser um domínio comum, é mais fácil recrutar possíveis stakeholders com conhecimento ou experiência em e-commerce para participar de sessões de teste do chatbot, contribuindo para uma validação das funcionalidades do sistema, qualidade das user stories e desempenho do modelo LLM no fluxo conversacional proposto.

Acessibilidade e familiaridade dos usuários

O comércio eletrônico é amplamente conhecido por usuários comuns, o que contribui para interações mais naturais com o chatbot durante o processo de elicitação. Esse fator facilita a coleta de requisitos claros e reduz o impacto de termos técnicos ou específicos.

[ ] Definição de critérios de sucesso: Faltam métricas claras e quantitativas para avaliar o chatbot na elicitação de requisitos.

[ ] Detalhamento do fluxo conversacional: Embora mencionado como diferencial, deverão ser definidos exemplos concretos de como será o fluxo de diálogo ou as estratégias para manter o contexto da conversa.

[ ] Critérios para seleção do LLM: O documento menciona que a escolha do modelo LLM ainda será feita, mas não estabelece os critérios que guiarão essa decisão.

[ ]Tratamento de ambiguidades: A proposta reconhece o problema das ambiguidades e palavras genéricas, mas não apresenta uma estratégia clara para resolvê-lo além de um "fluxo" não detalhado.

[ ] - Especificação dos formatos de saída: Embora mencione a geração de user stories e armazenamento em Excel/Docs, faltam detalhes sobre o formato, estrutura e processo de extração dessas informações a partir do diálogo.

[ ] - Esclarecimento sobre requisitos não funcionais: Há inconsistência sobre quando e como os requisitos não funcionais serão incorporados (inicialmente mencionados como futuros, depois incluídos no MVP). Os requisitos não funcionais não estarão no MVP, foi inicialmente colocado, mudamos de ideia e acabei esquecendo de tirar

[Fause] - Planejamento mais detalhado para integração de voz: A funcionalidade speech-to-speech é apresentada com incertezas ("talvez kkkk") e sem uma estratégia clara para implementação. É um diferencial , mas queremos focar nas estrutura básica primeiro o MVP, e explorar este diferencial como uma etapa complementar

4) Fragilidades

[ ] - Mitigação de alucinações: O documento reconhece o problema das alucinações em LLMs, mas depende principalmente da limitação de escopo como estratégia de mitigação, o que pode ser insuficiente para um sistema que precisa de alta precisão.

Validação e filtragem do output

Após a geração das user stories, um validador automático verifica se a estrutura segue o padrão correto (“Como [usuário], eu quero [ação] para [benefício]”) e se não contém termos

ambíguos ou genéricos (“produto bom”, “rápido”, “melhor”). Se detectar inconsistência, pode sugerir reescrita ou acionar nova rodada de perguntas ao usuário.

Use system prompts com instruções como:

"Se a informação não estiver disponível, diga que não sabe.

Restringir o contexto com base de dados confiável

Sistema de feedback

O próprio sistema já prevê o feedback 👍👎 — esse componente deve ser utilizado como sinal para treinar ou ajustar futuras respostas, com base em histórico de interações reais. Além disso, pode-se implementar um modo de revisão humana (moderador ou desenvolvedor) durante as primeiras iterações do MVP.

[ ]- Dependência da qualidade do treinamento inicial: O sucesso do chatbot está fortemente vinculado à qualidade e abrangência dos dados iniciais para o domínio específico escolhido.

[ ]- Ausência de estratégias concretas para resolver ambiguidades: Apesar da menção ao problema, faltam técnicas específicas para identificar e resolver termos ambíguos ou genéricos durante a conversa.

[ ] - Desafios com tecnologias speech-to-speech: A proposta reconhece limitações nas tecnologias open source para voz e menciona soluções pagas como alternativa, o que pode implicar em custos não previstos inicialmente.

[ ]- Escalabilidade do armazenamento de dados: O uso de Excel/Docs para armazenamento pode não ser a solução mais escalável à medida que o volume de dados aumenta.

[ ]- Complexidade do MVP: Mesmo sendo um MVP, a proposta inclui diversos componentes tecnológicos e desafios (LLM, LangChain, feedback, etc.) que podem tornar a implementação inicial mais complexa que o necessário.

[ ] - Afirmações sem validação empírica: Algumas alegações como "forma de aprender com os erros é mais eficaz" carecem de evidências ou métricas para suporte.

Artigo:
GORDIJN, Jaap; AKKERMANS, Hans; VAN VLIET, Hans. *Value based requirements creation for electronic commerce applications*. In: **Proceedings of the 32nd Annual Hawaii International Conference on System Sciences (HICSS)**. IEEE, 2000. Link: https://www.researchgate.net/publication/224071698.

Repos:
Spree Commerce (Ruby on Rails)
Plataforma modular de e-commerce com catálogo, checkout, carrinho e painel administrativo.
GitHub: https://github.com/spree/spree
Site oficial: https://spreecommerce.org

OpenCart (PHP)
Solução de carrinho de compras open-source com módulos de produtos, pedidos, envio e pagamento.
GitHub: https://github.com/opencart/opencart
Site oficial: https://www.opencart.com

Saleor (Python/Django)
Plataforma headless com API GraphQL, checkout avançado e gerenciamento de pedidos.
GitHub: https://github.com/saleor/saleor
Site oficial: https://saleor.io

WooCommerce (WordPress/PHP)
Plugin amplamente usado para WordPress com catálogo, carrinho, checkout, envio e relatórios.
GitHub: https://github.com/woocommerce/woocommerce
Site oficial: https://woocommerce.com
Guia de funcionalidades: https://litextension.com/blog/woocommerce-features/