📄 Desafio Técnico: Agente Bancário Inteligente

🏦 Cenário

Você está desenvolvendo um sistema de atendimento ao cliente para um banco digital fictício, o Banco Ágil. O atendimento é realizado por agentes de IA especializados, cada um com escopo de atuação e responsabilidades bem definidas.

👥 Agentes disponíveis no sistema:

- **Agente de Triagem**: Autentica o cliente e direciona para o agente apropriado.
- **Agente de Crédito**: Informa sobre limites de crédito e processa solicitações de aumento de limite.
- **Agente de Entrevista de Crédito**: Conduz uma entrevista financeira para atualizar o score de crédito.
- **Agente de Câmbio**: Realiza consulta de cotação de moedas.

🤖 Agente de Triagem

🎯 **Objetivo** Atuar como porta de entrada no atendimento, recepcionando o cliente, **coletando CPF e data de nascimento para autenticação contra uma base de dados (clientes.csv)**, e direcionando para o agente mais apropriado, conforme a necessidade identificada, **somente após a autenticação bem-sucedida**.

🧩 **Fluxo de atendimento:**

1. Saudação inicial.
2. Coleta do **CPF**.
3. Coleta da **data de nascimento**.
4. **Validação dos dados do cliente contra uma base (clientes.csv) para autenticação.**
5. **Se autenticado:**
    - Identificação do assunto da solicitação.
    - Redirecionamento para o agente adequado.
6. **Se não autenticado:**
    - Informar sobre a falha na autenticação. Permitir até 2 (duas) novas tentativas. Após a terceira falha consecutiva, o agente deve informar de maneira agradável que não foi possível autenticar e encerrar o atendimento.

💳 Agente de Crédito

🎯 **Objetivo** Auxiliar na **consulta de limite de crédito** e permitir que o cliente **solicite um aumento de crédito**.

🧩 **Responsabilidades:**

1. **Consulta de limite de crédito** disponível (após autenticação pelo Agente de Triagem).
2. **Tópico para solicitar aumento de crédito:**
    - Cliente informa o **novo limite de crédito desejado**.
    - O sistema deve gerar um pedido formal dessa solicitação, registrando-o em um arquivo CSV nomeado `solicitacoes_aumento_limite.csv` com as seguintes colunas: `cpf_cliente` (string), `data_hora_solicitacao` (timestamp ISO 8601), `limite_atual` (float), `novo_limite_solicitado` (float), `status_pedido` (string, e.g., 'pendente', 'aprovado', 'rejeitado').
    - Com o pedido montado é feito a checagem do score com base na tabela score_limite.csv se o valor solicitado é permitido para o

score atual do cliente. Caso o score seja suficiente é realizada a aprovação da solicitação que caminhará para o status ‘aprovado’, caso contrário caminhará para ‘rejeitado’.

3. Se, após a solicitação de aumento de limite, o status da solicitação for ‘reprovado’ o Agente de Crédito deve informar o cliente sobre a possibilidade de oferecer o redirecionamento para o Agente de Entrevista de Crédito para tentar reajustar seu score. Se o cliente não desejar a entrevista, o agente deve caminhar para encerramento da conversa ou demais redirecionamentos que façam sentido .

🗣️ Agente de Entrevista de Crédito

🎯 **Objetivo** Realizar uma entrevista conversacional estruturada com o cliente para coletar dados financeiros e recalcular seu score de crédito com base em uma fórmula ponderada.

🧩 **Responsabilidades:**

1. Conduzir perguntas sobre:
    - Renda mensal;
    - Tipo de emprego (formal, autônomo, desempregado);
    - Despesas fixas mensais;
    - Número de dependentes;
    - Existência de dívidas ativas.
2. Calcular um novo score de crédito (0 a 1000).
3. Atualizar o score do cliente na base de dados (clientes.csv).
4. Redirecionar o cliente de volta ao Agente de Crédito para nova análise.

📐 **Fórmula de Score (exemplo ajustável):**

```
None
    score = (
      (renda_mensal / (despesas + 1)) * peso_renda +
      peso_emprego[tipo_emprego] +
```

```
    peso_dependentes[num_dependentes] +
    peso_dividas[tem_dividas]
  )
```

Pesos sugeridos:

```
None
  peso_renda = 30
  peso_emprego = {
    "formal": 300,
    "autônomo": 200,
    "desempregado": 0
  }
  peso_dependentes = {
    0: 100,
    1: 80,
    2: 60,
    "3+": 30
  }
  peso_dividas = {
  "sim": -100,
  "não": 100
  }
```

💱 Agente de Câmbio

🎯 **Objetivo** Permitir ao cliente **consultar a cotação de moedas** em tempo real.

🧩 **Responsabilidades:**

1. Buscar a **cotação atual do dólar** (ou outra moeda solicitada) por meio de uma API externa (ex: Tavily, SerpAPI, ou outra de sua escolha).
2. Apresentar a cotação atual ao cliente.
3. Encerrar o atendimento específico de cotação com uma mensagem amigável.

📌 Regras Gerais (para todos os agentes)

- A qualquer momento, se o usuário solicitar o fim da conversa, o agente deve chamar a ferramenta de encerramento para finalizar o loop de execução.
- Os agentes devem manter um tom respeitoso e objetivo, evitando repetições desnecessárias.
- Nenhum agente pode atuar fora do seu escopo definido.
- Os redirecionamentos entre agentes devem ser realizados de maneira implícita, de modo que o cliente não perceba a transição. Ou seja, para o cliente ele está conversando com um único agente com habilidades diferentes.
- Sempre que possível, utilizar ferramentas apropriadas para acessar APIs, ler/escrever em arquivos CSV/planilhas, ou realizar cálculos.
    - **Tratamento de Erros e Exceções:** Cada agente deve ser capaz de lidar com erros esperados (ex: falha na leitura de CSV, API indisponível, entrada inválida do usuário) de forma controlada, informando o cliente sobre o problema de maneira clara e, se possível, oferecendo alternativas ou registrando o erro para análise técnica posterior sem interromper abruptamente a interação.

📦 Entrega e Requisitos Técnicos

🧾 Entrega Esperada

O candidato deverá entregar:

- **Repositório público no GitHub** contendo:
    - Código-fonte completo da solução.
    - Um arquivo `README.md` com as seguintes seções obrigatórias:
        - Visão Geral do projeto.
        - Arquitetura do sistema, explicando os agentes, fluxos e como os dados são manipulados.
        - Funcionalidades implementadas.

        - Desafios enfrentados e como foram resolvidos.
        - Escolhas técnicas e justificativas.
        - Tutorial de execução e testes.
    - Estrutura organizada do código (divisão clara por módulos e responsabilidades dos agentes).

🖥️ Interface do Usuário

- A aplicação deve incluir uma **UI simples** para testes, construída com ferramentas como:
    - Streamlit

Essa interface deve permitir que um usuário interaja com os agentes simulando um atendimento bancário completo.

⚙️ Tecnologias e Ferramentas Sugeridas

Você é livre para escolher a stack, mas aqui estão sugestões de frameworks e bibliotecas úteis para este desafio, especialmente no contexto de agentes e LLMs:

🧠 Frameworks para Agentes e LLMs

- Google ADK (Agent Developer Kit)
- CrewAI
- LangChain
- LangGraph
- LlamaIndex

🗝️ APIs com LLMs – Free Tier Disponíveis

- 🔑 Gemini API - Google
- 🔑 Groq API (com suporte a Mixtral e LLaMA)
- 🔑 OpenAI API
- 🔑 TogetherAI

- 🔑 OpenRouter