FOLKS

Captação Inteligente

Case para vaga de estágio em Engenharia de Software - FOLKS

Março de 2025

FOLKS

Introdução

A **FOLKS** está assessorando um grande grupo hospitalar, com diversas unidades em todo o país. Este grupo enfrenta um desafio significativo: a perda de pacientes para a concorrência na realização de **exames de imagem** (como ressonâncias magnéticas, tomografias, ultrasonografias, etc.).

Após as consultas médicas em que esses exames são solicitados, muitos pacientes acabam optando por realizar os procedimentos em outras instituições. Essa situação tem um impacto direto na receita do grupo hospitalar e dificulta a fidelização dos pacientes, que poderiam ser melhor assistidos dentro da própria rede.

O objetivo deste case é desenvolver um sistema inteligente e proativo, que seja capaz de identificar automaticamente os pacientes com solicitações de exames de imagem pendentes. O sistema deverá incentivar esses pacientes, através do envio de mensagens personalizadas via WhatsApp, a agendar seus exames dentro da rede do grupo hospitalar. Com isso, buscamos:

- **Aumentar a taxa de conversão:** Transformar um maior número de solicitações de exames em agendamentos efetivos.
- **Otimizar a utilização dos recursos:** Garantir que os equipamentos e profissionais do grupo hospitalar sejam utilizados em sua capacidade máxima.
- **Fortalecer o relacionamento com os pacientes:** Oferecer um serviço mais conveniente e personalizado, aumentando a satisfação e a fidelidade.

Dados de Entrada

Para este teste, você receberá dados simulados, representando as informações que estariam disponíveis no sistema do hospital. Esses dados possuem duas classificações:

Estruturados: dados normalizados e com código de identificação para cada solicitação (Código TUSS)

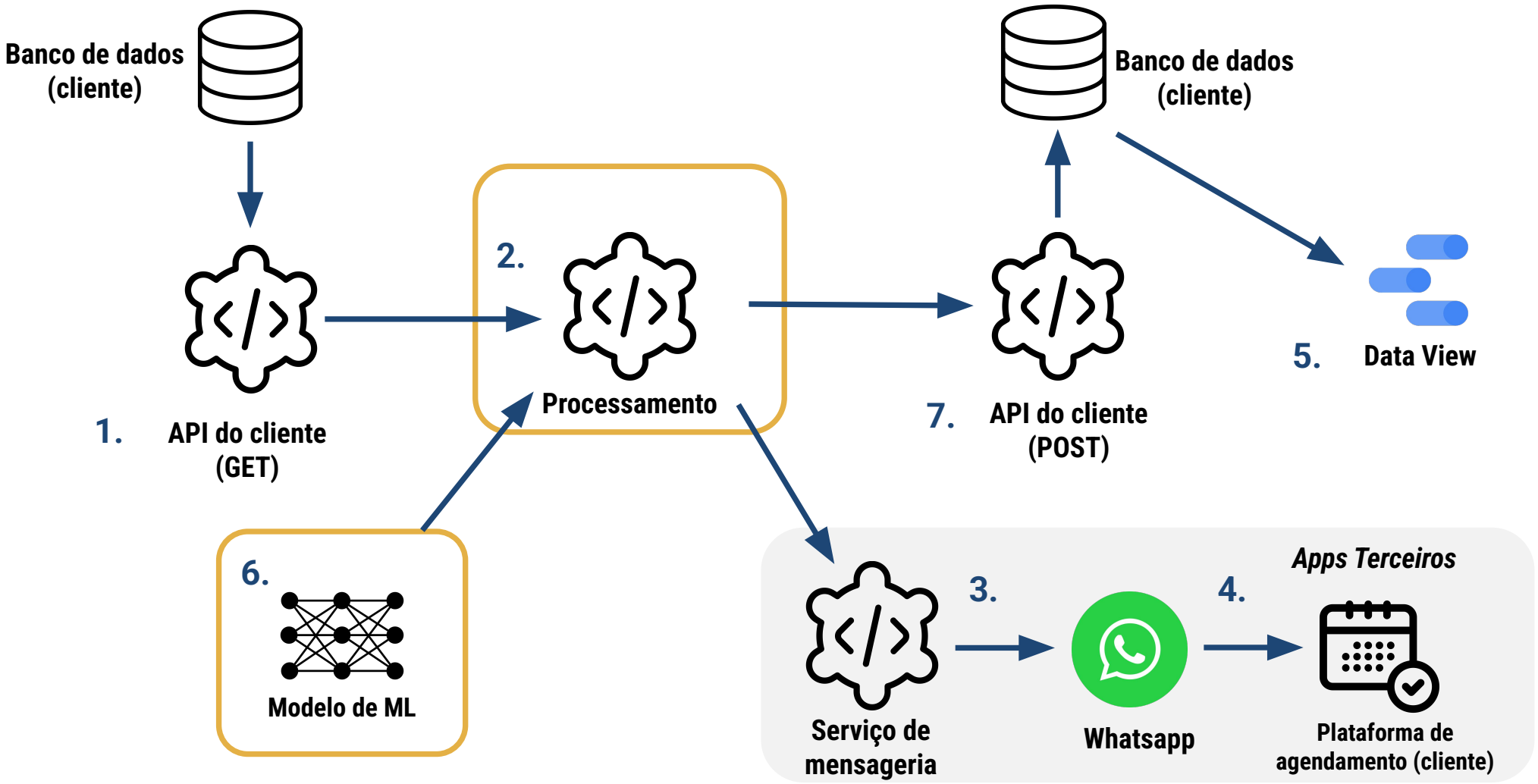
Não Estruturados: dados não normalizados nem classificados, basicamente texto livre (maior complexidade)

Arquitetura da Solução (Referência)

Apresentamos, apenas como referência, um diagrama que ilustra uma possível arquitetura para a solução.

Este diagrama descreve as funções de cada componente, mas você é totalmente livre para propor a arquitetura que considerar mais adequada, escolhendo as tecnologias que preferir e justificando suas decisões.

Arquitetura (Referência)



Descrição Funcional dos Componentes (Exemplo):

1. **Entrada de Dados (API):**
   - Responsável por receber os dados do sistema do hospital (tanto os dados estruturados quanto os não estruturados).
   - Deve ser um ponto de entrada bem definido e acessível.
2. **Processamento:**
   - Este é o "cérebro" da solução.
   - Recebe os dados da API de entrada.
   - Aplica as regras de negócio para determinar quais pacientes são elegíveis para receber mensagens.
   - Processa os dados não estruturados, extraindo informações relevantes (como o tipo de exame).
   - Constrói a mensagem personalizada a ser enviada
3. **Comunicação (Mensageria):**
   - Responsável por enviar as mensagens via WhatsApp para os pacientes.
   - Idealmente, deve ser capaz de lidar com um grande volume de mensagens.
4. **Interface de Agendamento (Fora do Escopo):**
   - É a plataforma de agendamento online do hospital.
   - O link na mensagem enviada ao paciente deve direcionar para essa plataforma.
5. **Visualização de Dados:**
   - Permite criar painéis (dashboards) interativos para visualizar os dados.
   - Facilita o acompanhamento das métricas de desempenho da solução.
   - Ajuda a identificar tendências e oportunidades de melhoria.
6. **Armazenamento de Modelo (Opcional):**
   - Se você optar por usar um modelo de Machine Learning para processar os dados não estruturados, este componente será responsável por armazenar e versionar esse modelo.
7. **Saída de Dados (API):**
   - Fornece um mecanismo para retornar informações sobre o status do envio das mensagens para o sistema do hospital (opcional, para integração).

Observação: Esta arquitetura é **APENAS UMA SUGESTÃO**. Você deve apresentar a arquitetura que considera mais apropriada para o problema, detalhando os componentes, as tecnologias que utilizaria e, *fundamentalmente*, justificando suas escolhas.

Tarefas

1. **Análise dos dados**
2. Modelagem (opcional, mas recomendado)
3. Desenvolvimento
4. Arquitetura e Escalabilidade
5. Melhorias e próximos passos
6. Documentação

- Explore cuidadosamente os dados de entrada fornecidos (tanto os estruturados quanto os não estruturados).
- Identifique quais campos são relevantes para a solução.
- Certifique-se de compreender completamente as regras de negócio.

Tarefas

1. Análise dos dados
2. **Modelagem (opcional, mas recomendado)**
3. Desenvolvimento
4. Arquitetura e Escalabilidade
5. Melhorias e próximos passos
6. Documentação

- Se você tiver familiaridade com técnicas de Processamento de Linguagem Natural (PLN) e Machine Learning, sugerimos que proponha um modelo para classificar e extrair informações dos dados não estruturados (por exemplo, identificar o tipo de exame solicitado).
- Não é necessário implementar o modelo completamente, mas descreva a abordagem que você utilizaria:
  - Tipo de modelo (ex: modelo baseado em regras, modelo de classificação, modelo de extração de entidades, etc.).
  - Quais features (características) você usaria para treinar o modelo.
  - Como você avaliaria o desempenho do modelo?
- Se preferir usar algum modelo pré treinado com fine tuning, indique como faria e leve em consideração os custos envolvidos.

Tarefas

1. Análise dos dados
2. Modelagem (opcional, mas recomendado)
3. **Desenvolvimento**
4. Arquitetura e Escalabilidade
5. Melhorias e próximos passos
6. Documentação

- Crie um script (preferencialmente em Python). O script deve:
  - Ler os dados de entrada (simulando o recebimento de dados de uma API). Você pode usar arquivos CSV ou JSON para representar os dados.
  - Aplicar as regras de negócio para filtrar os pacientes que devem receber mensagens.
  - Processar os dados não estruturados (usando expressões regulares, bibliotecas de PLN ou a lógica do modelo de ML que você propôs).
  - Processar os dados estruturados (use algum dicionário para identificar cada exame)
  - Construir a mensagem personalizada para cada paciente.
  - Simular o envio da mensagem (você pode simplesmente imprimir a mensagem no console, ou usar uma ferramenta de simulação de envio, se preferir).
  - Registrar o envio da mensagem (pode ser em um arquivo de log simples, em um banco de dados SQLite, ou em outra solução de armazenamento de sua escolha).

Tarefas

1. Análise dos dados
2. Modelagem (opcional, mas recomendado)
3. **Desenvolvimento**
4. Arquitetura e Escalabilidade
5. Melhorias e próximos passos
6. Documentação

- Crie uma visualização em dashboard
  - Use Looker Studio ou a ferramenta que preferir

Tarefas

1. Análise dos dados
2. Modelagem (opcional, mas recomendado)
3. Desenvolvimento
4. **Arquitetura e Escalabilidade**
5. Melhorias e próximos passos
6. Documentação

- **Proponha uma Arquitetura Detalhada:**
  - Descreva os componentes da sua solução (quais partes farão o quê).
  - Especifique as tecnologias que você usaria para cada componente (linguagens de programação, bancos de dados, serviços de mensageria, frameworks, etc.).
  - Explique como esses componentes se comunicariam entre si (APIs, filas de mensagens, etc.).
  - Desenhe um diagrama da sua arquitetura (use a ferramenta que preferir).
- **Justifique Suas Escolhas:**
  - Para cada tecnologia escolhida, explique por que você a selecionou. Quais as vantagens e desvantagens em relação a outras opções?
- **Escalabilidade:**
  - Como a sua solução lidaria com um aumento significativo no volume de dados (por exemplo, milhares de solicitações de exames por dia)?
  - Quais componentes precisariam ser escalados? Como você faria isso?

Tarefas

1. Análise dos dados
2. Modelagem (opcional, mas recomendado)
3. Desenvolvimento
4. **Arquitetura e Escalabilidade**
5. Melhorias e próximos passos
6. Documentação

- **Segurança:**
  - Como você garantiria a segurança e a privacidade dos dados dos pacientes, em conformidade com a Lei Geral de Proteção de Dados (LGPD)?
  - Quais medidas de segurança você implementaria?
- **Custos:**
  - Faça uma estimativa muito geral dos custos envolvidos na execução da sua solução. Considere os custos de armazenamento, processamento, envio de mensagens, etc. (Não precisa ser um cálculo preciso, mas uma estimativa aproximada).

Tarefas

1. Análise dos dados
2. Modelagem (opcional, mas recomendado)
3. Desenvolvimento
4. Arquitetura e Escalabilidade
5. **Melhorias e próximos passos**
6. Documentação

- **Melhorias:**
  - Sugira pelo menos três melhorias que poderiam ser implementadas na solução no futuro.
  - **Exemplos:**
    - Testes A/B para otimizar o conteúdo das mensagens.
    - Integração com outros sistemas do hospital (prontuário eletrônico, sistema de agendamento).
    - Uso de Machine Learning para prever a probabilidade de agendamento e personalizar ainda mais a abordagem.
    - Implementação de um canal de comunicação bidirecional (permitir que os pacientes respondam às mensagens).

Tarefas

1. Análise dos dados
2. Modelagem (opcional, mas recomendado)
3. Desenvolvimento
4. Arquitetura e Escalabilidade
5. **Melhorias e próximos passos**
6. Documentação

- **Desafios e Riscos:**
  - Identifique os principais desafios e riscos que você antecipa na implementação e operação da solução.
  - Como você mitigaria esses riscos?
- **Métricas de Sucesso:**
  - Quais métricas você usaria para acompanhar o desempenho da solução e medir seu sucesso?
- **Exemplos:**
  - Taxa de conversão (solicitações de exames que se transformam em agendamentos).
  - Taxa de abertura das mensagens.
  - Tempo médio entre a solicitação do exame e o agendamento.
  - Satisfação dos pacientes (medida por pesquisas, por exemplo).

Tarefas

1. Análise dos dados
2. Modelagem (opcional, mas recomendado)
3. Desenvolvimento
4. Arquitetura e Escalabilidade
5. Melhorias e próximos passos
6. **Documentação**

- **Código:**
  - Seu código deve ser bem documentado, com comentários explicando o que cada parte faz.
  - Use nomes de variáveis e funções significativos.
  - Siga boas práticas de programação.
- **Arquitetura:**
  - Documente a arquitetura da sua solução de forma clara e concisa, incluindo o diagrama e a descrição dos componentes.
- **Decisões e Justificativas**:
  - Explique o racional por trás de suas principais decisões de design e tecnologia.

Critérios de Avaliação

Seu trabalho será avaliado com base nos seguintes critérios:

- **Compreensão do Problema:** Você demonstrou um entendimento claro do desafio proposto e dos objetivos do case?
- **Qualidade do Código:** Seu código é limpo, organizado, bem comentado e segue boas práticas de programação?
- **Eficácia da Solução:** A solução que você propôs é eficaz em resolver o problema? Ela atende aos requisitos funcionais e não funcionais?
- **Escalabilidade da Solução:** A solução é capaz de lidar com um grande volume de dados e solicitações?
- **Justificativa das Escolhas:** Você justificou de forma convincente suas escolhas de tecnologia e arquitetura?
- **Pensamento Crítico e Análise:** Você demonstrou capacidade de analisar o problema, identificar desafios e propor soluções?
- **Proposição de Melhorias:** Você sugeriu melhorias relevantes e realistas para a solução?
- **Clareza na Comunicação:** Você conseguiu comunicar suas ideias de forma clara e concisa, tanto no código quanto na documentação?
- **Criatividade e Inovação:** Você apresentou soluções originais ou abordagens inovadoras para o problema?

Entregáveis

Ao final do teste, você deverá entregar:

- **Código Fonte**: O código completo do seu script (preferencialmente em um repositório GitHub).
- **Documentação**: Um documento (em formato Markdown, PDF, Google Docs, ou similar) contendo:
  - A **descrição detalhada** da sua arquitetura (**com diagrama**).
  - As **justificativas** para suas escolhas de tecnologia.
  - A explicação do seu código (se necessário, além dos comentários no próprio código).
  - Suas sugestões de melhorias, desafios, riscos e métricas.
  - Instruções claras sobre como executar seu código (se aplicável).
- **Apresentação:** Uma breve apresentação em slides resumindo sua solução (20 minutos no máximo, deve ser enviado o slide e a apresentação será feita remotamente em uma data a definir).

Dicas

Se usar **python**, use e abuse de bibliotecas como:

- **re**: para expressões de regex
- **nltk**: para NLP
- **pandas**, **polars** ou **fireducks**: para manipulação de dados
- **scikit** ou **pytorch**: caso queira treinar ou usar algum modelo pré treinado
    - para modelos pré treinados, recomendo usar modelos do **hugging face**
    - se for criar um modelo, leve em consideração que estamos falando basicamente de um classificador. Pense em classes como: Prescrição, Encaminhamento, Exames, Outros.

Ao pensar em uma arquitetura, leve em consideração a utilização de Cloud (AWS, GCP, Azure) e como pode-se desenvolver soluções com custo previsível e parametrizável.

Aqui na FOLKS usamos GCP, uma solução desta pode ser feita praticamente em todas as cotas gratuitas oferecidas lá.

Use e abuse de LLMs para te ajudar a ter insights (DeepSeek, GPT, Gemini, Claude, etc)

mas não deixe de lado seu senso crítico.

Considerações finais

Você terá **7 dias** a partir do recebimento deste documento para completar o teste e enviar os entregáveis.

Lembre-se: este é um case para **estágio**, não esperamos a melhor solução do mundo (“*antes feito do que perfeito*”).

Caso não consiga entregar algo, **NÃO DESISTA!**

Todo esforço será levado em consideração.

Em caso de dúvidas, entre em contato através do e-mail

arodrigues@folks.la

ou whatsapp

+5516997687272

Estamos à disposição para ajudar!