TESTE DE ENTRADA PARA ESTAGIÁRIOS v2.0

Intuitive Care - Healthtech de SaaS Vertical

Este teste foi elaborado para identificar candidatos alinhados com a cultura "mão na massa" da Intuitive Care. O teste é prático e conectado com o dia a dia da empresa.

Importante: Você não precisa fazer tudo perfeitamente ou conhecer todas as tecnologias. O objetivo é avaliar sua capacidade de resolver problemas práticos, tomar decisões técnicas fundamentadas e documentar suas escolhas. Quanto mais você conseguir implementar e justificar adequadamente suas decisões, melhor será sua avaliação. Priorize qualidade sobre quantidade - é melhor fazer menos itens bem feitos e bem justificados do que muitos itens sem fundamentação.

Prazo de entrega: 1 semana

Instruções gerais:

- Escolha entre Python ou Java para as tarefas de programação
- Para SQL, use MySQL 8.0 ou PostgreSQL >10.0
- Organize seu código de forma clara e documentada
- Inclua um arquivo README explicando como executar cada parte do teste
- Compacte todos os arquivos em um único ZIP nomeado como `Teste_{seu_nome}.zip`

Sobre os trade-offs técnicos:
Ao longo do teste, você encontrará situações que apresentam diferentes opções de implementação (marcadas como "Trade-off técnico"). As opções listadas são exemplos comuns, mas **não são as únicas possíveis**. Se você conhecer ou preferir outra abordagem técnica, sinta-se livre para utilizá-la - desde que você **documente e justifique adequadamente** sua escolha no README, explicando os prós, contras e o contexto que levou à sua decisão.

---

1. TESTE DE INTEGRAÇÃO COM API PÚBLICA

Este teste deve ser realizado em Python ou Java. O código deverá executar:

1.1. Acesso à API de Dados Abertos da ANS

- Acesse a API REST disponível em: `https://dadosabertos.ans.gov.br/FTP/PDA/`
- Identifique e baixe os arquivos de **Demonstrações Contábeis** dos últimos 3 trimestres disponíveis
- Os arquivos estão organizados por ano/trimestre no formato: `YYYY/QQ/`
- **Observação:** Alguns trimestres podem ter múltiplos arquivos ou estruturas de diretório diferentes. Seu código deve ser resiliente a essas variações.

1.2. Processamento de Arquivos

- Baixe todos os arquivos ZIP dos trimestres identificados
- Extraia os arquivos ZIP automaticamente
- Identifique e processe apenas os arquivos que contenham dados de **Despesas com Eventos/Sinistros**
- **Desafio:** Os arquivos podem ter formatos diferentes (CSV, TXT, XLSX) e estruturas de colunas variadas. Você deve identificar automaticamente a estrutura e normalizar os dados.

> - **Trade-off técnico:** Decida entre processar todos os arquivos em memória de uma vez ou processar incrementalmente. Documente sua escolha e justifique considerando o volume de dados.

1.3. Consolidação e Análise de Inconsistências

- Consolide os dados dos 3 trimestres em um único arquivo CSV
- O CSV deve conter as colunas: `CNPJ`, `RazaoSocial`, `Trimestre`, `Ano`, `ValorDespesas`
- **Análise crítica:** Durante a consolidação, você encontrará:
  - CNPJs duplicados com razões sociais diferentes
  - Valores zerados ou negativos
  - Trimestres com formatos de data inconsistentes
- Documente como tratou cada tipo de inconsistência e **justifique sua abordagem** (ignorar, corrigir, marcar como suspeito, etc.)
- Compacte o CSV final em um arquivo ZIP nomeado `consolidado_despesas.zip`

---

2. TESTE DE TRANSFORMAÇÃO E VALIDAÇÃO DE DADOS

Crie um código em Python ou Java que execute as seguintes tarefas:

2.1. Validação de Dados com Estratégias Diferentes

- Utilize o CSV consolidado do teste 1.3
- Implemente validações para:
    - CNPJ válido (formato e dígitos verificadores)
    - Valores numéricos positivos
    - Razão Social não vazia

> - **Trade-off técnico:** Para CNPJs inválidos, você precisará decidir como tratá-los. Considere diferentes estratégias e escolha a que fizer mais sentido. Escolha uma abordagem, implemente-a e **documente no README** sua escolha e os prós/contras considerados.

2.2. Enriquecimento de Dados com Tratamento de Falhas

- Baixe o arquivo CSV de **Dados Cadastrais das Operadoras Ativas** em: `https://dadosabertos.ans.gov.br/FTP/PDA/operadoras_de_plano_de_saude_ativas/`
- Faça um join entre o CSV consolidado e os dados cadastrais usando o CNPJ como chave
- Adicione as colunas: `RegistroANS`, `Modalidade`, `UF` ao CSV final
- **Análise crítica:** Você encontrará CNPJs no arquivo consolidado que não existem no cadastro (ou vice-versa). Decida como tratar:
    - Registros sem match no cadastro
    - CNPJs que aparecem múltiplas vezes no cadastro com dados diferentes

> - **Trade-off técnico:** Para o join, você precisará decidir como processar os dados. Considere diferentes estratégias de processamento e escolha a que fizer mais sentido para o seu contexto. Documente sua escolha e justifique baseado no tamanho estimado dos dados e nas características do problema.

2.3. Agregação com Múltiplas Estratégias

- Agrupe os dados por `RazaoSocial` e `UF`
- Calcule o total de despesas por operadora/UF
- **Desafio adicional:** Calcule também:
    - Média de despesas por trimestre para cada operadora/UF
    - Desvio padrão das despesas (para identificar operadoras com valores muito variáveis)

> - **Trade-off técnico:** Para ordenação, você precisará escolher uma estratégia considerando o volume de dados e os recursos disponíveis. Justifique sua escolha no README.

- Ordene por valor total (maior para menor)
- Salve o resultado em um novo CSV nomeado `despesas_agregadas.csv`
- Compacte o arquivo em `Teste_{seu_nome}.zip`

---

3. TESTE DE BANCO DE DADOS E ANÁLISE

Crie scripts SQL (compatíveis com MySQL 8.0 ou PostgreSQL >10.0) que executem as tarefas abaixo.

Tarefas de Preparação:

3.1. Utilize os arquivos CSV gerados nos testes anteriores:

- `consolidado_despesas.csv` (teste 1.3)
- `despesas_agregadas.csv` (teste 2.3)
- CSV de dados cadastrais das operadoras (teste 2.2)

Tarefas de Código:

3.2. Crie queries DDL para estruturar as tabelas necessárias:

- Tabela para dados consolidados de despesas
- Tabela para dados cadastrais das operadoras
- Tabela para dados agregados
- Defina chaves primárias e índices apropriados
- **Trade-off técnico - Normalização:**
  - **Opção A:** Tabela desnormalizada com todos os dados
  - **Opção B:** Tabelas normalizadas separadas

  Escolha uma abordagem e **justifique** considerando:
  - Volume de dados esperado
  - Frequência de atualizações
  - Complexidade das queries analíticas
- **Trade-off técnico - Tipos de dados:**
  - Para valores monetários: DECIMAL vs FLOAT vs INTEGER (centavos)
  - Para datas: DATE vs VARCHAR vs TIMESTAMP

  Justifique suas escolhas considerando precisão, performance e uso futuro.

3.3. Elabore queries para importar o conteúdo dos arquivos CSV:

- Atente para o encoding correto (UTF-8)
- Trate possíveis inconsistências nos dados
- **Análise crítica:** Durante a importação, você encontrará:
  - Valores NULL em campos obrigatórios
  - Strings em campos numéricos
  - Datas em formatos inconsistentes
- Documente como tratou cada caso e **justifique** sua abordagem (rejeitar, usar valor padrão, tentar conversão, etc.)

3.4. Desenvolva queries analíticas para responder:

- **Query 1:** Quais as 5 operadoras com maior crescimento percentual de despesas entre o primeiro e o último trimestre analisado?
    - **Desafio:** Considere operadoras que podem não ter dados em todos os trimestres. Como tratar? Justifique.
- **Query 2:** Qual a distribuição de despesas por UF? Liste os 5 estados com maiores despesas totais.
    - **Desafio adicional:** Calcule também a média de despesas por operadora em cada UF (não apenas o total).
- **Query 3:** Quantas operadoras tiveram despesas acima da média geral em pelo menos 2 dos 3 trimestres analisados?
    > - **Trade-off técnico:** Esta query pode ser resolvida com diferentes abordagens. Escolha uma e justifique considerando performance, manutenibilidade e legibilidade do código.

---

4. TESTE DE API E INTERFACE WEB

Desenvolva uma interface web usando Vue.js que interaja com um servidor em Python para realizar as tarefas abaixo.

Tarefas de Preparação:

4.1. Utilize os dados do banco de dados criado no teste 3 (ou CSVs se preferir trabalhar sem banco)

Tarefas de Código:

4.2. Crie um servidor em Python (Flask ou FastAPI) com as seguintes rotas:

- `GET /api/operadoras` - Lista todas as operadoras com paginação (parâmetros: `page`, `limit`)
- `GET /api/operadoras/{cnpj}` - Retorna detalhes de uma operadora específica
- `GET /api/operadoras/{cnpj}/despesas` - Retorna histórico de despesas da operadora
- `GET /api/estatisticas` - Retorna estatísticas agregadas (total de despesas, média, top 5 operadoras)

Trade-offs técnicos - Backend:

4.2.1. Escolha do Framework:

- **Opção A:** Flask
- **Opção B:** FastAPI

Escolha um e **justifique** considerando complexidade do projeto, performance esperada e facilidade de manutenção.

4.2.2. Estratégia de Paginação:

- **Opção A:** Offset-based
- **Opção B:** Cursor-based
- **Opção C:** Keyset pagination

Escolha uma abordagem e **justifique** considerando o volume de dados e frequência de atualizações.

4.2.3. Cache vs Queries Diretas:

- Para a rota `/api/estatisticas`, você pode:
    - **Opção A:** Calcular sempre na hora
    - **Opção B:** Cachear resultado por X minutos
    - **Opção C:** Pré-calcular e armazenar em tabela

Escolha uma abordagem e **justifique** considerando frequência de atualização dos dados e requisitos de consistência.

4.2.4. Estrutura de Resposta da API:

- Para paginação, você pode retornar:

- **Opção A:** Apenas os dados (`[{...}, {...}]`)
- **Opção B:** Dados + metadados (`{data: [...], total: 100, page: 1, limit: 10}`)

Escolha e **justifique** considerando facilidade de uso no frontend e necessidade de informações adicionais.

4.3. Elabore uma interface Vue.js que:

- Exiba uma tabela paginada com as operadoras
- Permita busca/filtro por razão social ou CNPJ
- Mostre um gráfico (use uma biblioteca como Chart.js ou similar) com a distribuição de despesas por UF
- Tenha uma página de detalhes da operadora mostrando o histórico de despesas

Trade-offs técnicos - Frontend:

4.3.1. Estratégia de Busca/Filtro:

- **Opção A:** Busca no servidor
- **Opção B:** Busca no cliente
- **Opção C:** Híbrido

Escolha uma abordagem e **justifique** considerando volume de dados e experiência do usuário.

4.3.2. Gerenciamento de Estado:

- Para gerenciar dados das operadoras, você pode:
  - **Opção A:** Props/Events simples
  - **Opção B:** Vuex/Pinia
  - **Opção C:** Composables (Vue 3) ou mixins (Vue 2)

Escolha e **justifique** considerando complexidade da aplicação e necessidade de compartilhamento de estado.

4.3.3. Performance da Tabela:

Para exibir muitas operadoras, considere diferentes estratégias de renderização. Justifique sua escolha considerando volume de dados e requisitos de UX.

4.3.4. Tratamento de Erros e Loading:

- Documente como você trata:
  - Erros de rede/API
  - Estados de loading
  - Dados vazios
- **Análise crítica:** Considere se você mostra mensagens genéricas ou específicas, e justifique sua escolha.

4.4. Documentação:

- Crie uma coleção no Postman para demonstrar todas as rotas da API
- Inclua exemplos de requisições e respostas esperadas

- **Importante:** No README, documente todos os trade-offs técnicos que você considerou e as justificativas para suas escolhas

---

CRITÉRIOS DE AVALIAÇÃO

O teste será avaliado considerando:

- **Funcionalidade:** O código executa e produz os resultados esperados
- **Qualidade do código:** Organização, legibilidade, tratamento de erros
- **Documentação:** README claro explicando como executar cada parte
- **Pensamento crítico:** Tratamento de casos extremos e validações adequadas
- **Praticidade:** Soluções simples e eficientes (KISS - Keep It Simple)
