Relatório (Parte 1) - Assistente Conversacional com NLP (Watson)

Objetivo

Implementar um protótipo de assistente cardiológico conversacional capaz de realizar um atendimento inicial em saúde, com um fluxo coerente, respostas contextualizadas e tratamento básico de exceções, integrando o **IBM Watson Assistant** a um backend simples em **Flask**.

Visão Geral da Solução

A solução foi implementada em três camadas:

Frontend (Web - React): interface de chat para enviar mensagens e visualizar respostas.

Backend (Flask): API responsável por receber mensagens do usuário e repassá-las ao assistente.

Watson Assistant: lógica conversacional publicada no ambiente `live`.

Integração frontend/backend:

O backend expõe `POST /api/message` e retorna `{ response, intents, entities }`.

O frontend consome a API e renderiza a conversa.

Como evidência acadêmica do modelo solicitado no enunciado, incluímos um **export JSON** com **intents**, **entities** e **dialog nodes**.

Modelagem Conversacional (Intents / Entities / Dialog Nodes)

Evidência do modelo clássico (export)

Arquivo: `watson_skill_export.json`

Intents (exemplos):

`saudacao`

`agendar_consulta`

`dor_no_peito`

(demais intents no JSON)

Entities:

Exemplos de entidades para datas/sintomas (ver JSON)

Dialog nodes:

Nós cobrindo saudação, emergência e agendamento, incluindo respostas padrão (fallback).

Implementação em produção (Watson Assistant)

O assistente foi publicado no ambiente **live** e é acessado via **Assistant V2 API** (SDK oficial).

Fluxos principais (atendimento inicial):

1. **Saudação**: acolhimento e orientação do que o assistente consegue fazer.

2. **Emergência**: triagem básica (perguntas-chave) e instrução de ação imediata quando aplicável.

3. **Agendamento**: coleta data e confirma o pré-agendamento.

Tratamento de Exceções (básico)

Foram considerados cenários comuns:

Mensagem vazia (não vira erro: o backend orienta e sugere próximos passos)

Falha de credenciais / indisponibilidade do Watson

Usuário responde algo fora do esperado em etapas de confirmação/data

Para garantir que o avaliador consiga testar **mesmo sem credenciais**, existe um modo **LOCAL (offline)** que executa o fluxo conversacional localmente **com base no export JSON**:

`backend/mock_assistant.py` (lê `watson_skill_export.json`)

Ativação: `CARDIOIA_ASSISTANT_MODE=local`

Integração Backend <-> Watson

Código do backend:

`backend/app.py`: endpoints e gerenciamento simples de sessão

`backend/watson_service.py`: integração com SDK do Watson Assistant V2

Fluxo técnico:

1. Frontend envia `POST /api/message` com `{ message, user_id }`

2. Backend cria/recupera uma sessão (por usuário)

3. Backend chama o Watson Assistant via SDK e retorna `text`, `intents` e `entities`

Como Executar (resumo)

1. Configurar `.env` (ver `.env.example`)

2. Rodar:

```bash
pip install -r backend/requirements.txt
python run_server.py
```

3. Abrir `http://127.0.0.1:5000`

4. Na UI, usar as abas: **Conversa**, **Organizar relato**, **Monitoramento**, **Imagem (Fase 4)**.

Conexão com as Fases Anteriores (2, 3 e 4)

Esta fase **não é isolada**: ela consolida a evolução do CardioIA, transformando as peças técnicas anteriores em uma **experiência de atendimento ao paciente**.

Fase 2 (IA & NLP / triagem e risco)

Reuso direto no backend: `backend/phase2_triage.py` chama `FASES ANTERIORES/Fase2/src/diagnose.py` para gerar `risk` e `diagnosis`.

Isso aparece no protótipo via `POST /api/phase2/triage` e também dentro do "Organizar relato" (Ir Além 1) via `POST /api/clinical/extract`.

Fase 3 (IoT & monitoramento contínuo)

A regra de alerta (temperatura/bpm) foi reaproveitada e exposta em `POST /api/phase3/vitals` (ver `backend/phase3_vitals.py`).

Na UI, isso aparece como "Simulador de vitais" na aba **Monitoramento**.

Fase 4 (Visão computacional por imagem)

A Fase 5 integra de forma opcional um health-check do serviço da Fase 4: `GET /api/phase4/health` (ver `backend/phase4_cv.py`).

Quando `PHASE4_CV_URL` estiver configurada e o serviço estiver rodando, a UI mostra o status na aba **Imagem (Fase 4)**.

Em resumo, a Fase 5 atua como “camada de interface e orquestração” para o que foi desenvolvido/planejado nas fases anteriores.

Conclusão

O projeto entrega um assistente conversacional funcional, com modelagem de diálogo documentada, integração via API e interface de chat. A solução mantém a evidência do **modelo clássico** (intents/entities/dialog nodes) e também mantém evidências do **Watson publicado** para demonstração em vídeo.