

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):

- Saudação**: acolhimento e orientação do que o assistente consegue fazer.
- Emergência**: triagem básica (perguntas-chave) e instrução de ação imediata quando aplicável.
- 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 **MOCK (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=mock``

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
```

```
cd backend
```

```
pip install -r requirements.txt
```

```
python app.py
```

```
```
```

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

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

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

Fase 2 (IA & NLP / triagem e risco)

A Fase 5 reutiliza o mesmo objetivo central: entender linguagem natural e organizar sinais/sintomas para triagem inicial.

Evidências: ``FASES ANTERIORES/Fase2/README.md`` e ``FASES ANTERIORES/REPORT-DE-AVANÇO.MD``.

****Fase 3 (IoT & monitoramento contínuo)****

A Fase 5 assume o contexto de monitoramento e alerta: o assistente conversa com o paciente *como porta de entrada* para sinais vitais e acompanhamento.

Evidências: `FASES ANTERIORES/FASE3/README.md` e relatórios em `FASES ANTERIORES/FASE3/reports/`.

****Fase 4 (Visão computacional por imagem)****

A Fase 5 complementa a experiência com orientação e encaminhamento: o paciente inicia pelo chat e pode ser direcionado ao módulo de imagem quando aplicável.

Evidências: `FASES ANTERIORES/FASE4/README.md` e `FASES ANTERIORES/FASE4/reports/REPORT_PHASE4.md`.

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.