

## Ir Além 1 - IA Generativa e Extração de Informações Clínicas

### Objetivo

Expandir o projeto para interpretar conteúdo clínico **não estruturado** (texto livre) usando técnicas de **prompting** e **IA Generativa**, organizando o resultado em um formato estruturado (JSON).

### Implementação

Notebook: ``notebooks/genai_extraction.ipynb``

Integração com o protótipo (Fase 5):

Endpoint: ``POST /api/clinical/extract`` (backend Flask)

Interface: aba **"Organizar relato"** (frontend React)

Abordagem:

1. Recebemos um texto clínico simulado (ex: relato do paciente).
2. Aplicamos um prompt orientando o modelo a extrair campos relevantes.
3. A saída é estruturada (JSON) para facilitar armazenamento, validação e integração com outros sistemas.

Exemplo de Campos Extraídos (modelo)

Identificação: ``nome`` (quando presente), ``idade``

Sintomas: ``sintomas[]``

Sinais vitais (quando descritos): ``pressao_arterial``, ``frequencia_cardiaca``

Red flags: ``dor_no_peito``, ``falta_de_ar``, ``tontura``, etc.

Recomendações: ``orientacao_inicial``

Por que JSON?

JSON permite:

Persistência e indexação simples

Validação de schema

Integração direta com APIs e bancos NoSQL

Como Rodar

1. Configurar ``GEMINI_API_KEY`` no ``.env`` (ou usar variáveis de ambiente)
2. Abrir o notebook e executar as células:

```
```bash
```

```
jupyter notebook notebooks/genai_extraction.ipynb
```

```
```
```

Observações de Boas Práticas

Prompting: instruções claras, delimitação do texto e formato de saída explicitamente definido.

Governança: o protótipo simula um assistente e não substitui orientação médica.

Privacidade: o projeto usa dados fictícios/simulados.