**Visão Geral**

O projeto é um pipeline de automação dividido em três componentes independentes que trabalham em sequência: um **ETL** para buscar os dados, uma **API** para gerenciá-los e um **Robô (RPA)** para inseri-los no sistema de destino.

**Etapa 1: ETL RedCap (1\_etl\_redcap\_sinan)**

Este componente é o **Coletor de Dados**. Sua única função é buscar as informações mais recentes do RedCap e prepará-las para a próxima etapa.

**Como funciona:**

1. **Execução:** É um script Python (redcap.py) que você roda manualmente ou de forma agendada.
2. **Conexão:** Ele lê o arquivo .env para obter a URL e o token da API do RedCap, além das credenciais do banco de dados PostgreSQL.
3. **Extração:** O script faz uma chamada para a API do RedCap e baixa todos os registros de notificações.
4. **Transformação:** Utilizando a biblioteca Pandas, ele carrega esses dados, limpa-os e renomeia as colunas para um formato padronizado.
5. **Carga:** Ele se conecta ao banco de dados PostgreSQL (que pode ser iniciado com o docker-compose.yml do projeto), **apaga todos os dados da tabela de notificações** (TRUNCATE) e insere a nova carga de dados recém-baixada.

**Resultado:** Ao final desta etapa, o banco de dados contém uma cópia limpa e atualizada de todas as notificações do RedCap, pronta para ser consumida.

**Etapa 2: API (2\_digitador\_sinan\_api)**

Este componente é o **Gerente de Tarefas** ou a "ponte" entre os dados e o robô. Ele controla o fluxo de trabalho, entregando as notificações pendentes e registrando o que já foi feito.

**Como funciona:**

1. **Execução:** É uma API web construída com FastAPI, projetada para rodar continuamente. O método ideal de execução é através do docker-compose.yml, que sobe tanto a API quanto seu próprio banco de dados de forma isolada.
2. **Conexão:** A API se conecta ao banco de dados que foi populado pela Etapa 1.
3. **Endpoints:** Ela expõe URLs (endpoints) que o robô pode chamar:
   * Um endpoint para **buscar a lista** de todas as notificações que ainda não foram processadas.
   * Um endpoint para **atualizar o status** de uma notificação para "processado" depois que o robô termina seu trabalho com sucesso.
4. **Persistência:** Ele usa o Alembic para gerenciar as versões do banco de dados, permitindo que a estrutura das tabelas seja atualizada de forma controlada.

**Resultado:** Uma API robusta que serve como um ponto central de controle, garantindo que o robô saiba exatamente o que fazer e que nenhuma notificação seja processada duas vezes.

**Etapa 3: Robô RPA (3\_sinan\_rpa)**

Este componente é o **Trabalhador Digital**. Ele simula as ações de um ser humano para inserir os dados no sistema SINAN, que não possui uma API.

**Como funciona:**

1. **Execução:** É um script Python (main.py) que você roda na máquina que tem acesso ao sistema SINAN.
2. **Busca de Tarefas:** Ao iniciar, o primeiro passo do robô é chamar a API da Etapa 2 para pedir a lista de notificações pendentes.
3. **Automação de Tela:** Para cada notificação recebida, o robô:
   * Abre o sistema SINAN.
   * Navega pelos menus até chegar ao formulário correto.
   * Utiliza a biblioteca pyautogui para encontrar os campos e botões na tela. Ele faz isso comparando pequenas imagens (ex: salvar.png, ok.png) salvas na pasta imagens/ com o que está visível no monitor.
   * A lógica de busca e clique fica na função find\_and\_click do arquivo utils.py.
   * Ele "digita" as informações do paciente em cada campo do formulário. A lógica específica do preenchimento fica em scripts como agravosscripts/violencia.py.
4. **Feedback:** Após preencher e salvar um formulário com sucesso no SINAN, o robô faz uma segunda chamada à API da Etapa 2 para marcar aquela notificação como "processada".
5. **Logging:** Todas as ações importantes do robô (início, sucesso, erros) são gravadas em um arquivo de log (rpa\_log.txt) para fácil depuração, conforme configurado no logger.py.

**Resultado:** A automação completa do processo de digitação, eliminando o trabalho manual, reduzindo erros e liberando tempo da equipe.

**GITHUB**

**1. O tipo (Obrigatório)**

É uma palavra curta que descreve a categoria da mudança. Os tipos mais comuns são:

* **feat**: Para uma nova funcionalidade (um "feature").
* **fix**: Para a correção de um bug.
* **docs**: Para mudanças apenas na documentação.
* **style**: Para mudanças de formatação que não alteram o código (espaços, ponto e vírgula, etc.).
* **refactor**: Para uma refatoração do código que não adiciona funcionalidade nem corrige um bug.
* **perf**: Para uma mudança que melhora a performance.
* **test**: Para adicionar ou corrigir testes.
* **build**: Para mudanças que afetam o sistema de build ou dependências externas (ex: requirements.txt, Dockerfile).
* **ci**: Para mudanças nos arquivos de configuração de Integração Contínua (CI).
* **chore**: Para outras mudanças que não modificam o código-fonte ou os testes (ex: atualizar o .gitignore).

**2. O escopo (Opcional)**

Uma palavra entre parênteses que especifica o local/módulo do código que foi alterado. Para o nosso projeto, bons escopos seriam: (etl), (api), (rpa), (database).

**3. A descrição (Obrigatória)**

Uma descrição curta, no tempo verbal imperativo (ex: "corrige", "adiciona", "altera"), sobre o que a mudança faz.

* Começa com letra minúscula.
* Não termina com ponto final.

**4. O corpo (Opcional)**

Um parágrafo mais longo que explica o "porquê" da mudança, fornecendo mais contexto.

**5. O rodapé (Opcional)**

Usado principalmente para duas coisas:

* **BREAKING CHANGE:**: Para descrever uma mudança que quebra a compatibilidade com versões anteriores.
* **Referências a Issues:** Para fechar issues de um sistema como o GitHub (ex: Closes #45).

**Exemplos Práticos para o Nosso Projeto**

* **Commit Simples (Correção de Bug):** fix(rpa): corrige erro ao clicar no botão salvar em telas de alta resolução
* **Commit de Nova Funcionalidade com Escopo:** feat(api): adiciona endpoint para buscar pacientes por CPF
* **Commit de Documentação:** docs: atualiza o README com as novas variáveis de ambiente
* **Commit com corpo para explicar o contexto:**