

Projeto Final

Sistema Multi-Agente Autônomo para Automação e Orquestração do Fluxo de Aprovação de Notas Fiscais de Serviço

Nome do Grupo: Agregadores I2A2

EQUIPE

Beatriz Marques Guerra	bmarquesguerra@gmail.com
José Carlos dos Passos	jcdpassos@hotmail.com
Guilherme Silveira Coelho	gscocoelho@gmail.com
Edvander Maroto	edvanderma@gmail.com
Leny Reiff	leny.reiff@gmail.com
Adriano Valadares	adrianovaladares74@hotmail.com
Rafael Castino	rafacastino12@gmail.com
Expedito Hebert	shinigamecontato@gmail.com

Sumário

1.	Descrição do Tema Escolhido.....	3
2.	Público-alvo.....	3
3.	Justificativa do Tema Escolhido	3
4.	Detalhamento do que foi Desenvolvido.....	4
4.1.	Funções e Módulos Principais	4
4.2.	Elementos Adicionais	4
5.	Melhorias Futuras	6
6.	Link para o Repositório do Github.....	6

1. Descrição do Tema Escolhido

O projeto final consiste em um **Sistema Multi-Agente Autônomo para Automação e Orquestração do Fluxo de Aprovação de Notas Fiscais de Serviço**.

A solução implementa um ecossistema de *scripts* Python que interagem para executar um processo de negócios completo, iniciado pelo *upload* de uma Nota Fiscal (PDF) em uma interface web construída em **Streamlit**. O sistema utiliza múltiplos agentes de *software* especializados:

1. **Agente Extrator (IA Generativa)**: O módulo pdf_processor.py usa **PyMuPDF (fitz)** para extrair o texto bruto do PDF. Este texto é então analisado pela IA **Google Gemini**, que atua como um agente de Extração de Entidade Nomeada (NER) para identificar e estruturar dados-chave em JSON, incluindo o **Número do Pedido (PC)**.
2. **Agente de Enriquecimento (Banco de Dados)**: O módulo db_manager.py usa o Nº do PC para consultar um banco de dados **SQLITE** (simulando o SAP da empresa), que enriquece a transação com os dados do **Solicitante** e **Centro de Custo**.
3. **Agente Orquestrador (Lógica de Estado)**: O workflow_manager.py é o núcleo lógico. Ele gerencia o estado da transação na tabela ProcessamentoNF (definindo o status como 'PENDING_VALIDATION') e gera um *token* de validação único.
4. **Agente de Comunicação (SMTP)**: O email_manager.py usa smtplib para enviar um e-mail de ação ao Solicitante, contendo *links* de webhook (Aprovar/Rejeitar) que apontam de volta para a aplicação Streamlit.
5. **Agente Monitor (Assíncrono)**: Um processo separado, scheduler.py, utiliza **APScheduler** para monitorar o banco de dados. Este agente verifica independentemente se alguma NF pendente excedeu o *timeout* de 48 horas, atualizando o status para 'TIMEOUT' e notificando o financeiro.

2. Público-alvo

O público-alvo primário é a **equipe de Engenharia/Projetos de uma empresa química**, especificamente a colaboradora (e integrante deste grupo) que atualmente executa esta tarefa de forma manual.

Secundariamente, o **departamento Financeiro** é beneficiado, pois o *input* para pagamento chega validado, estruturado e com o Centro de Custo correto, eliminando o retrabalho.

3. Justificativa do Tema Escolhido

A escolha do tema foi baseada em um problema de negócios real, de alto atrito e validado internamente, enfrentado por um dos membros do grupo. O processo manual de validação de Notas Fiscais consome, em média, **11% do tempo produtivo (três dias por mês)** da colaboradora da Engenharia, com picos de até 26 horas.

A automação deste fluxo não é apenas um ganho de eficiência, mas uma realocação de capital intelectual. A implementação deste agente agrega valor ao:

- **Resolver uma Dor Própria**: O profundo entendimento do problema permitiu uma modelagem precisa do *workflow* e das regras de negócio (como a trava de 48h).
- **Otimizar o Tempo**: Libera mais de um mês por ano em horas úteis da colaboradora para foco em atividades de engenharia.
- **Aumentar a Precisão (GenAI)**: O uso do **Gemini** para extrair o Nº do Pedido de um texto não estruturado (descrição de serviços) é mais robusto que Regex e elimina erros de digitação.
- **Garantir Compliance (Agente Monitor)**: O **APScheduler** garante que a regra de negócio de 48h seja aplicada de forma assíncrona, criando um processo auditável e seguro.

4. Detalhamento do que foi Desenvolvido

A solução foi desenvolvida em Python 3.9+, com módulos distintos para cada função do agente, e gerenciamento de segredos via python-dotenv.

4.1. Funções e Módulos Principais

- **Interface e Webhook (app.py):** Desenvolvida em **Streamlit**. Possui uma dupla função:
 1. **UI de Upload:** Fornece o *front-end* para o usuário carregar o PDF.
 2. **Endpoint de Resposta:** Usa st.query_params para ler a action e o token dos links de e-mail, atuando como o *webhook* que processa a aprovação ou rejeição.
- **Banco de Dados (db_manager.py):** Utiliza **SQLite** como um banco de dados *serverless* para prototipagem rápida. A tabela ProcessamentoNF é o "cérebro" da memória de estado do agente, rastreando status, validation_token e timestamp_envio.
- **Processador de PDF e IA (pdf_processor.py):**
 1. **Extração de Texto:** Usa fitz (PyMuPDF) para extrair o texto bruto do PDF de forma eficiente.
 2. **Inteligência (Gemini):** Envia o texto bruto ao **Google Gemini** com um *prompt* de engenharia específico para estruturar os dados em JSON, focando no numero_pedido.
- **Orquestração (workflow_manager.py):** Conecta todos os módulos. handle_uploaded_invoice orquestra a extração, consulta ao DB e envio de e-mail. handle_validation_response processa o clique no *webhook* e notifica o financeiro.
- **Agente Monitor (scheduler.py):** Um *script* separado e essencial que usa **APScheduler** para rodar a função check_timeouts() em segundo plano (ex: a cada 1 hora). Este agente consulta o DB por NFs pendentes e rejeita automaticamente aquelas que excederam 48 horas, garantindo a lógica de negócio assíncrona.

4.2. Elementos Adicionais

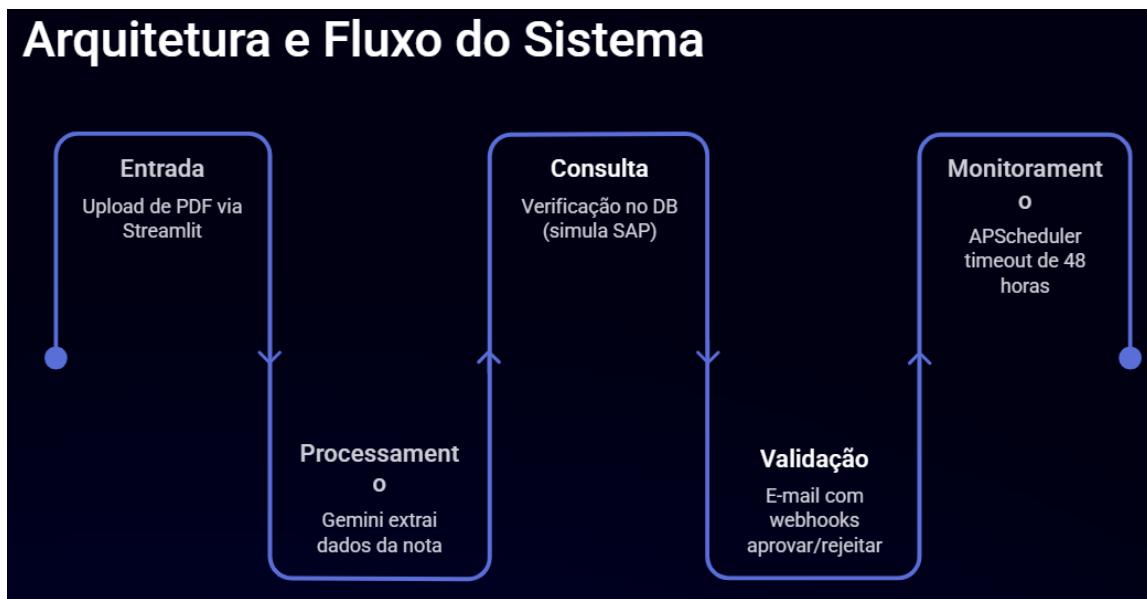


Figura 1. Arquitetura e Fluxo do Sistema

Tabela 1. Tabela de Tecnologias Utilizadas

Categoria	Ferramentas (Bibliotecas)
Interface Web e Webhook	Streamlit
IA Generativa (Extração)	google-generativeai (Gemini)
Leitura de PDF	PyMuPDF (fitz)
Banco de Dados (Estado)	SQLite (via sqlite3)
Comunicação	smtplib
Agendamento (Timeout)	APScheduler
Gestão de Segredos	python-dotenv, .gitignore

Validação de Nota Fiscal

Olá, Usuario Teste,

Uma Nota Fiscal foi recebida e associada a um pedido em seu nome. Por favor, revise os dados abaixo e confirme se as informações do pedido estão corretas para que o pagamento seja processado.

Detalhes da Nota Fiscal (Extraídos pela IA)

Número NF	00123
Fornecedor	Soluções em TI LTDA
Data	28/10/2025
Valor NF	R\$ 1.500,50

Detalhes do Pedido (do Banco de Dados)

Número Pedido	PED-1001-XYZ
Valor do Pedido	R\$ 1.500,50
Centro de Custos	TI-INFRA

As informações do pedido acima estão corretas?

SIM, APROVAR

NÃO, REJEITAR

Figura 2. Validação Teste de Nota Fiscal realizada por e-mail

5. Melhorias Futuras

1. **Integração Real com SAP:** Substituir o banco de dados SQLite por uma conexão direta ao SAP (via APIs/RFC), permitindo o enriquecimento de dados mais complexos.
2. **Automação de Lançamento (Smartsheet):** Em vez de apenas notificar o Financeiro por e-mail, o agente poderia usar a API do **Smartsheet** para preencher automaticamente o formulário de controle de pagamentos, que é o processo em implantação na empresa, fechando 100% do ciclo de automação.

6. Link para o Repositório do Github

<https://github.com/Agregadores-I2A2/ProjetoFinal-Artefatos>