30/10/2025

Grupo: **MetaMind AI Agents**

Relatório do Projeto

Sumário

[Grupo e participantes 2](#_Toc211366503)

[PENDÊNCIAS – 09-10-2025 2](#_Toc211366504)

[Link de acesso ao Github 2](#_Toc211366505)

[Descrição do Tema Escolhido 2](#_Toc211366506)

[Público Alvo 3](#_Toc211366507)

[Justificativa do Tema Escolhido 3](#_Toc211366508)

[Detalhamento do Agente 3](#_Toc211366509)

[Elementos Adicionais: tabelas, gráficos, diagramas 10](#_Toc211366510)

[Anexos 11](#_Toc211366511)

## Grupo e participantes

Nome do Grupo: **MetaMind AI Agents**

**Participantes do Grupo:**

Henry Monteiro – [henry.monteiro@meta.com.br](mailto:henry.monteiro@meta.com.br)

Henrique Sardelli – [henrique.sardelli@meta.com.br](mailto:henrique.sardelli@meta.com.br)

Dario Volnei – [dario.yamacake@meta.com.br](mailto:dario.yamacake@meta.com.br)

Douglas Kondo – [douglas.kondo@meta.com.br](mailto:douglas.kondo@meta.com.br)

Felipe Santos – [felipe.rsantos@meta.com.br](mailto:felipe.rsantos@meta.com.br)

Lyndon Tavares - [lyndon.tavares@meta.com.br](mailto:lyndon.tavares@meta.com.br)

Elizabeth Marinho – [elizabeth.marinho@meta.com.br](mailto:elizabeth.marinho@meta.com.br)

Bruno Kenzo – [wakoteam@yahoo.com.br](mailto:wakoteam@yahoo.com.br)

## PENDÊNCIAS – 14-10-2025

Finalizar o relatório do projeto – CONCLUÍDO

Rever a apresentação do PPT – picth deck – CONCLUÍDO

Elaborar o vídeo de apresentação para a banca examinadora – duração máxima de 5min

Nome do Vídeo **I2A2\_Agentes\_Inteligentes\_Projeto\_Final\_MetaMind\_AI\_Agents**

Criar a pasta do projeto no nosso Git (**Projeto Final – Artefatos**) – CONCLUÍDO

O repositório no Git deve permitir acesso público e em seu readme.md deve-se destacar que o **projeto se encontra sob a licença MIT.** – CONCLUÍDO

## Link de acesso ao Github

<https://github.com/dmkondo/MetaMindAIAgents/tree/main/Projeto%20Final%20-%20Artefatos>

## Descrição do Tema Escolhido

O tema escolhido para o trabalho final foi **EXTRAÇÃO DE DADOS**. Esse tema indica algumas funcionalidades como:

1. Recuperar documentos fiscais em fontes conhecidas;
2. Utilizar OCR (Reconhecimento Óptico de Caracteres) em conjunto com NLP (Processamento de Linguagem Natural) para extrair dados relevantes dos documentos:
   1. Informações do emitente e destinatário;
   2. Itens da nota (descrição, quantidade, valor);
   3. Impostos (ICMS, IPI, PIS, COFINS);
   4. CFOP, CST e outros códigos fiscais.
3. Desafio:
   1. Como tornar o agente capaz de se adaptar a diferentes layouts e formatos de documentos;
   2. Como se adaptar às mudanças legais (ex. IVA)

O grupo desenvolveu, para este trabalho de conclusão conforme o tema escolhido e descrito acima, um agente de IA que realiza a extração de dados a partir da leitura de notas fiscais preenchendo automaticamente um formulário de reembolso com identificação automática do valor, data e tipo de despesa.

## Público Alvo

Colaboradores de empresas que necessitam realizar a prestação de contas para o registro de seus relatórios de reembolso.

## Justificativa do Tema Escolhido

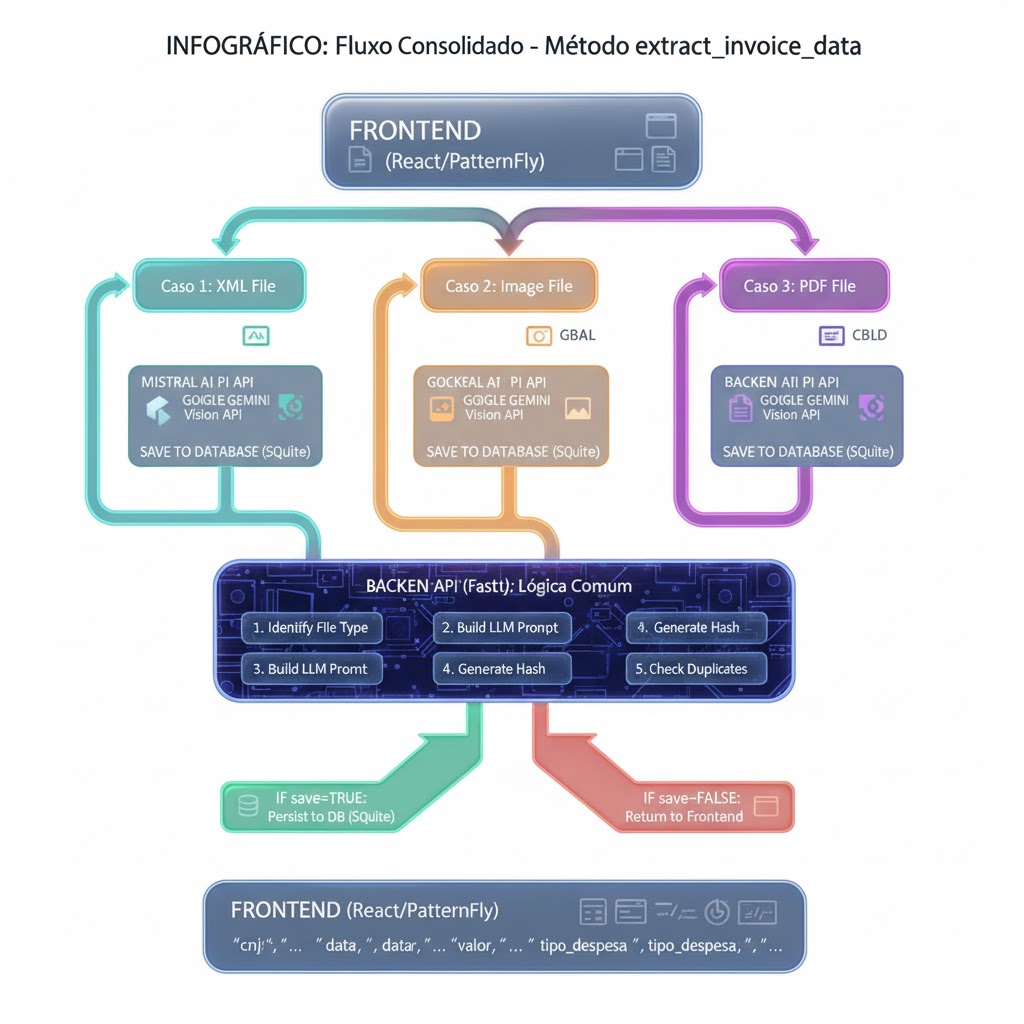
Otimização do tempo e redução de erros em operações manuais tornando-as automatizadas através da leitura de documentos fiscais e inserção de dados com o uso de IA.

## Detalhamento do Agente

Desenvolver uma API (Python) integrada com IA tendo um endpoint que receberá a imagem de uma nota fiscal e acionará um agente de IA para a extração e identificação dos dados persistindo-os em uma base de dados, juntamente com um segundo endpoint para consulta das mesmas.

Este documento técnico apresenta a arquitetura e os principais fluxos da API METAMIND, uma solução robusta para extração inteligente de dados de notas fiscais (imagens, PDFs e XMLs) utilizando modelos de linguagem (LLMs) avançados, como Google Gemini e Mistral AI. A API é a espinha dorsal de um sistema que visa automatizar e otimizar o processamento de documentos fiscais, fornecendo dados estruturados para análise e auditoria.

O frontend, desenvolvido em **React** e estilizado com **PatternFly**, oferece uma interface de usuário intuitiva para interagir com a API, visualizar dados extraídos e gerenciar configurações.



**Arquitetura da Solução**

A solução é composta por um conjunto de serviços interconectados, projetados para escalabilidade, desempenho e inteligência.

**Componentes Principais:**

* **Frontend (React + PatternFly):**
  + **Tecnologia:** React JS.
  + **Biblioteca UI:** PatternFly (design system baseado em React Components e CSS).
  + **Função:** Interface de usuário para upload de documentos, visualização de dados extraídos, edição e gerenciamento de configurações.
* **Backend (FastAPI):**
  + **Tecnologia:** Python com framework FastAPI.
  + **Função:** Gateway de API, orquestração de processos de extração, interação com LLMs, persistência de dados.
  + **Middleware:** CORS configurado para permitir comunicação com o frontend.
* **Modelos de Linguagem (LLMs):**
  + **Google Gemini (Flash/Pro Vision):** Utilizado para processamento multi-modal (imagem e PDF) e extração de dados complexos de documentos.
  + **Mistral AI (Medium):** Utilizado para chat geral e extração de dados baseada em texto (via OCR ou XML).
* **Base de Dados (SQLite):**
  + **Tecnologia:** SQLite (para demonstração/desenvolvimento; pode ser migrado para um banco de dados mais robusto em produção).
  + **Função:** Armazenamento dos dados de notas fiscais extraídos e configurações da aplicação.
* **Processamento de Imagem/OCR:**
  + **Pillow (PIL):** Manipulação de imagens.
  + **Pytesseract:** Execução de OCR em imagens quando necessário (especialmente para extract\_invoice\_data\_with\_mistral).

**Função Extract\_invoice\_data**

A função assíncrona extract\_invoice\_data é a principal responsável por processar e extrair dados de notas fiscais digitais, utilizando modelos de Linguagem Grande Multimodais (LLMs) do Google Gemini para realizar OCR (Reconhecimento Óptico de Caracteres) e classificação.

### 1. Visão Geral e Parâmetros

|  |  |  |
| --- | --- | --- |
| Parâmetro | Tipo | Descrição |
| file | UploadFile | O arquivo da nota fiscal (Imagem, XML ou PDF) enviado pelo usuário. |
| save | bool | Flag booleana que determina a finalidade: True para salvar no banco de dados (status=PENDENTE), ou False para apenas checar/extrair dados iniciais (status=CHECKING) e verificar duplicidade. |
| session | Session | A sessão do banco de dados (SQLAlchemy) para consultas e persistência. |

Exportar para as Planilhas

**Objetivo:** Extrair CNPJ, data de emissão, valor total e classificar o tipo\_despesa da nota fiscal, verificar duplicidade e, opcionalmente, salvar no banco de dados.

### 2. Extração de Dados e Classificação (LLM - Gemini)

A função utiliza a biblioteca genai e o modelo **GEMINI\_PRO\_VISION\_MODEL** para o processamento, dividindo a lógica em três casos principais baseados no content\_type do arquivo:

### **✅ XML (text/xml, application/xml)**

* Lê o conteúdo do XML como texto (xml\_text).
* O LLM é chamado para analisar o **texto XML puro** e extrair os campos necessários, seguindo o prompt de JSON estrito.
* Usa o modelo de texto multimodal, que é eficiente para estruturar dados a partir de texto.

### **✅ Imagem (image/\*)**

* Lê o conteúdo binário da imagem (image\_data).
* O LLM é chamado com o **conteúdo binário da imagem** (image\_parts).
* O modelo utiliza suas capacidades de Visão para realizar **OCR** na imagem e, em seguida, extrair e classificar os dados.

### **✅ PDF (application/pdf)**

* Lê o conteúdo binário do PDF (pdf\_data).
* O LLM é chamado com o **conteúdo binário do PDF** (pdf\_parts).
* O modelo Multimodal realiza a leitura e extração de dados diretamente do PDF, atuando como um **OCR avançado e extrator de estrutura**.

### 3. Processamento do Resultado e Hash

Para todos os formatos (XML, Imagem, PDF):

1. **Geração de Prompt:** O prompt é dinamicamente carregado da tabela de Configurations do banco de dados (se existir) ou usa um *fallback* padrão. O prompt exige que o LLM responda **SOMENTE em JSON estrito** com os campos: cnpj, data (DD/MM/AAAA), valor, tipo\_despesa e explicacao.
2. **Decodificação de JSON:** A resposta crua do LLM (raw\_response) é tratada para remover blocos de código (````json) e, em seguida, decodificada usando json.loads()`.
3. **Conversão de Valor:** O campo valor é convertido de forma segura para um tipo float.
4. **Geração de Hash:** Um hash\_value (identificador único do arquivo) é gerado a partir do conteúdo binário do arquivo usando a função gerar\_hash\_imagem.

### 4. Persistência e Checagem de Duplicidade

Após a extração bem-sucedida, a função aplica a lógica de persistência:

1. **Checagem de Duplicidade:** O banco de dados é consultado usando o imagem\_hash gerado para verificar se a nota fiscal já existe (existente).
2. **Lógica Condicional com save:**
   * **Se save=True (Tentativa de salvar):** Se o arquivo for um duplicado (existente), uma HTTPException(400) é levantada (erro, arquivo já cadastrado). Caso contrário, a nova fatura é salva com status="PENDENTE".
   * **Se save=False (Modo de checagem/preview):**
     + Se o arquivo for um duplicado, o objeto existente é retornado imediatamente.
     + Se for um arquivo novo, um novo objeto Invoice é criado, mas **não é persistido**, recebendo o status="CHECKING" e é retornado para a pré-visualização no frontend.

Este design garante que o frontend receba dados pré-extraídos e validados, independentemente de ser um arquivo novo ou um duplicado, otimizando o fluxo de trabalho do usuário.

**3. Endpoints da API METAMIND**

Esta tabela detalha os endpoints disponíveis na API, agrupados por suas tags funcionais.

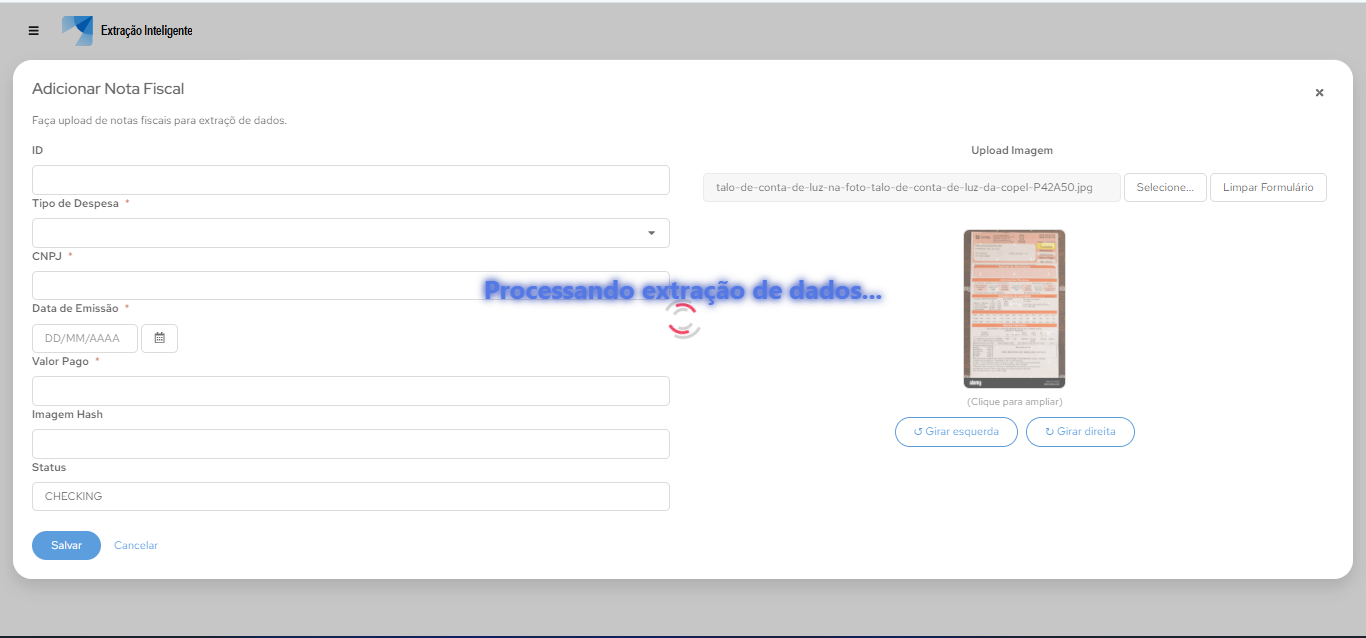
|  |  |  |  |
| --- | --- | --- | --- |
| Tag | Endpoint | Método HTTP | Descrição |
| Interação com LLM | /chat/mistral | POST | Recebe uma requisição de chat e a encaminha para a API do Mistral AI, retornando a resposta do modelo. Ideal para interações conversacionais ou geração de texto. |
| Interação com LLM | /invoices/extract/mistral | POST | Recebe uma imagem de nota fiscal, realiza OCR (via PyTesseract) e extrai CNPJ, data e valor total usando o modelo Mistral AI. Retorna os dados extraídos em formato JSON. (Não persiste no DB). |
| Interação com LLM | /invoices/extract/save | POST | Recebe um arquivo de nota fiscal (imagem, PDF ou XML), extrai CNPJ, data, valor total e tipo de despesa (ALIMENTACAO, VEICULO, ESCRITORIO) utilizando o Google Gemini Vision. Salva os dados extraídos no banco de dados. |
| Interação com LLM | /invoices/extract/check | POST | Recebe um arquivo de nota fiscal (imagem, PDF ou XML), extrai CNPJ, data, valor total e tipo de despesa (ALIMENTACAO, VEICULO, ESCRITORIO) utilizando o Google Gemini Vision. Não salva os dados no banco de dados, retornando-os para verificação. |
| Crud | /invoices | GET | Retorna uma lista de todas as notas fiscais (documentos) que foram extraídas e salvas no banco de dados. |
| Crud | /invoices/{id} | GET | Retorna os detalhes de um documento de nota fiscal específico, identificado pelo seu ID. |
| Crud | /invoices/add | POST | Adiciona um novo documento de nota fiscal manualmente ao banco de dados, utilizando os dados fornecidos no corpo da requisição. |
| Crud | /invoices/{id} | PUT | Atualiza parcialmente os dados de um documento de nota fiscal existente, identificado pelo seu ID. Permite modificar campos como CNPJ, tipo de despesa, data de emissão, valor total e status. |
| Crud | /invoices/{id} | DELETE | Exclui permanentemente um documento de nota fiscal do banco de dados, identificado pelo seu ID. |
| Configuração | /configuration | PUT | Atualiza o prompt de extração de dados padrão usado pelos modelos de LLM. Isso permite personalizar a forma como os dados são extraídos sem alterar o código da API. |
| Configuração | /configuration | GET | Retorna o prompt de extração de dados atualmente configurado na aplicação. |

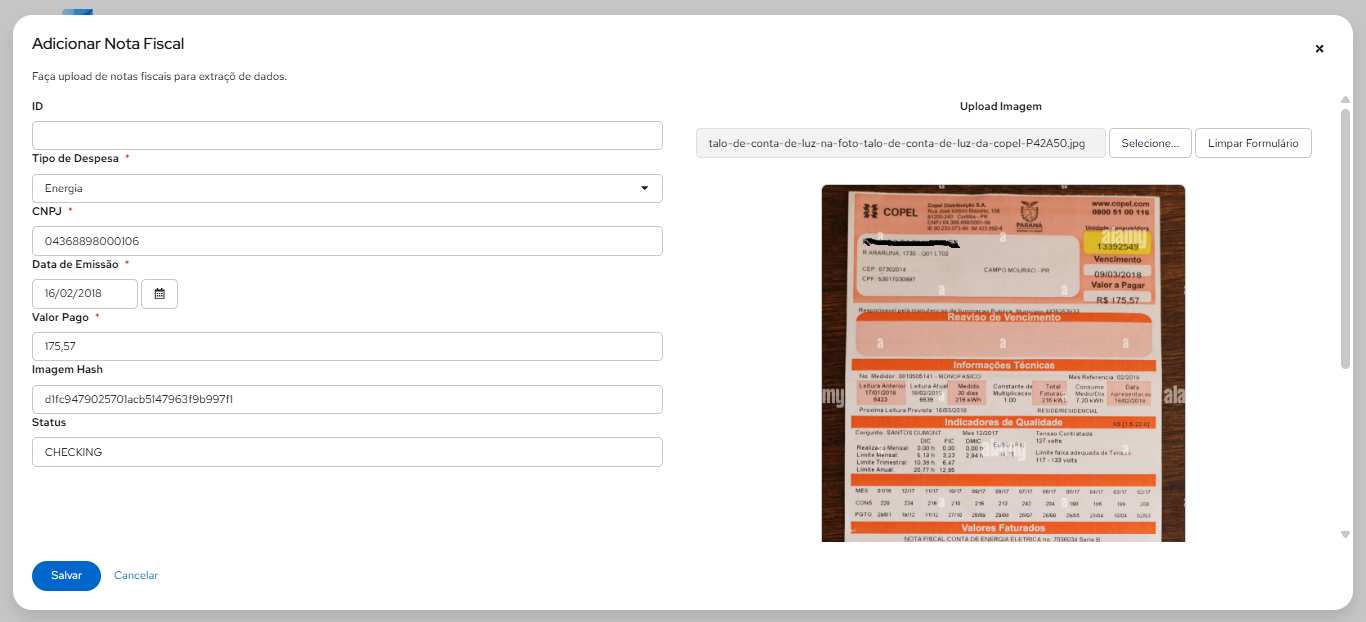
**4. Front-end**

Exemplo de front-end em React usando design system Patterfly.

**4.1 Adicionar Nota Fiscal**

Ao fazer upload da nota fiscal, é chamado endpoint de extração e checagem. O endpoint verifica o tipo de arquivo: pdf, xml ou imagem. Ao extrair os dados (data emissão, valor total, cnpj ), também faz a classificação da categoria de despesa.



****

## Elementos Adicionais: tabelas, gráficos, diagramas

Banco de dados: *SQLite*.

Utilizar um framework – *FastAPI*, para estabelecer um padrão para o *Swagger*, assim conseguimos ter uma *API*. A integração com o *agent*, será feita via *langchain,* quanto ao modelo estamos avaliando a melhor solução entre os modelos *Mitre* ou *Gemini* ou *ChatGP*. Para o OCR estamos estudando também usar a biblioteca *pytesseract*, que é uma biblioteca *free* que viabiliza diminuir o custo para o desenvolvimento do nosso *agent.*

## Anexos

