Challenge QUOD - Sprint 2

Desenvolvimento de um backend para biometria e validação de documentos

1 - Integrantes do grupo:

RM553090 Diego Brasileiro Vilela Dias

RM553011 Gabriel Araújo Ferraz de Melo

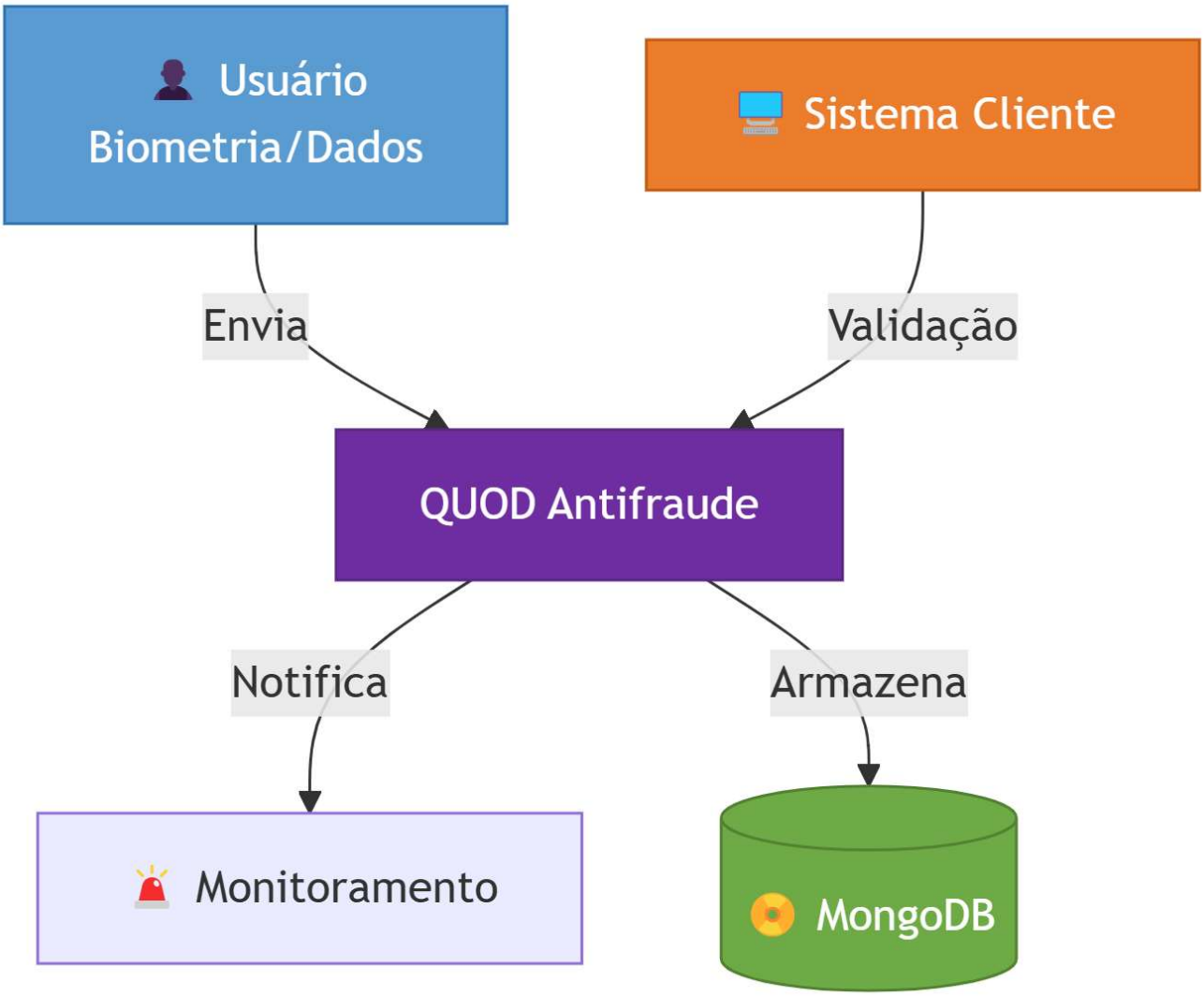
RM552951 Jonas Alves Moreira

RM554002 Paulo Cauê Krüger Costa

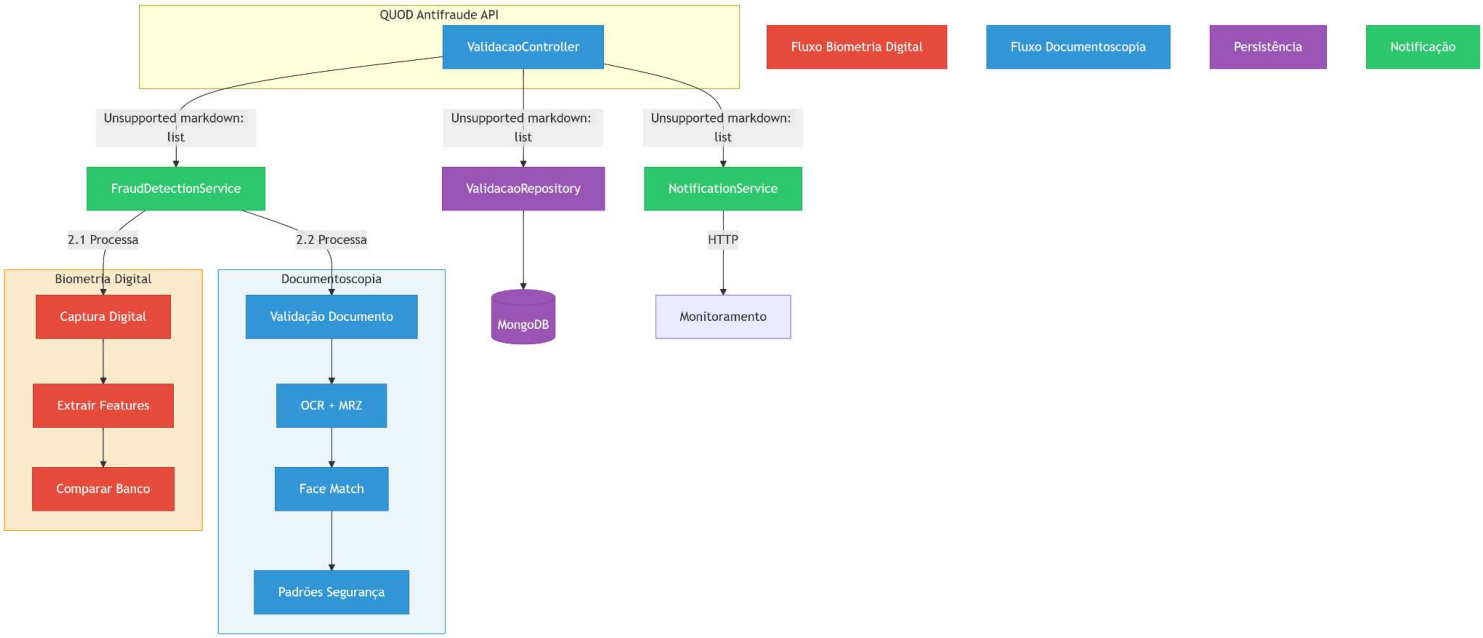
RM553807 Gabriel Paulucci

2 - Desenho de arquitetura (Modelo C4):

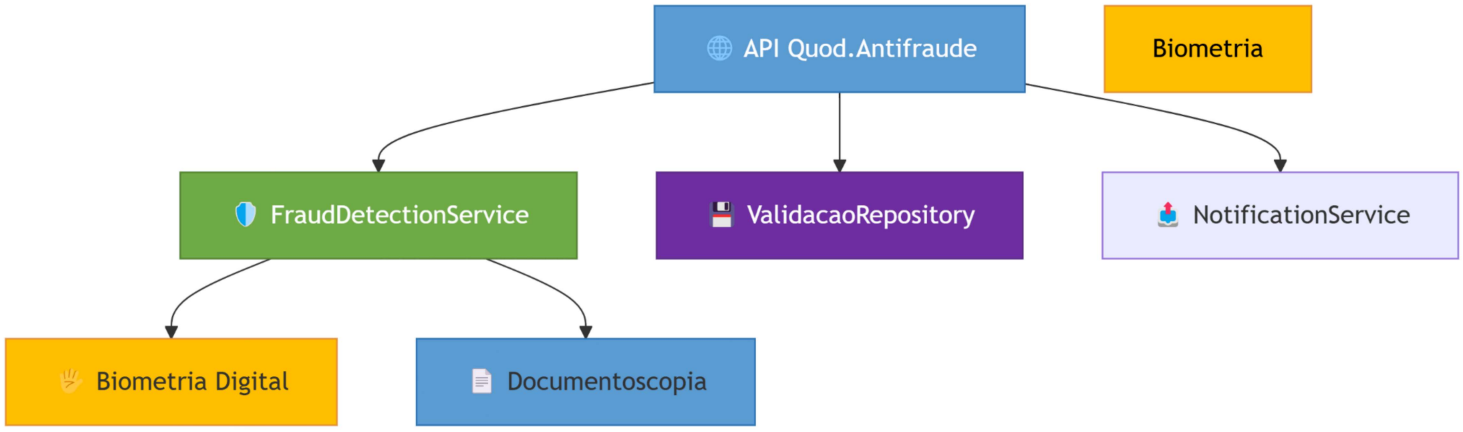
2.1 - Diagrama de Contexto



2.2 - Diagrama de Contêiner



2.3 - Diagrama de Componentes



2.4 - Diagrama de Sequência

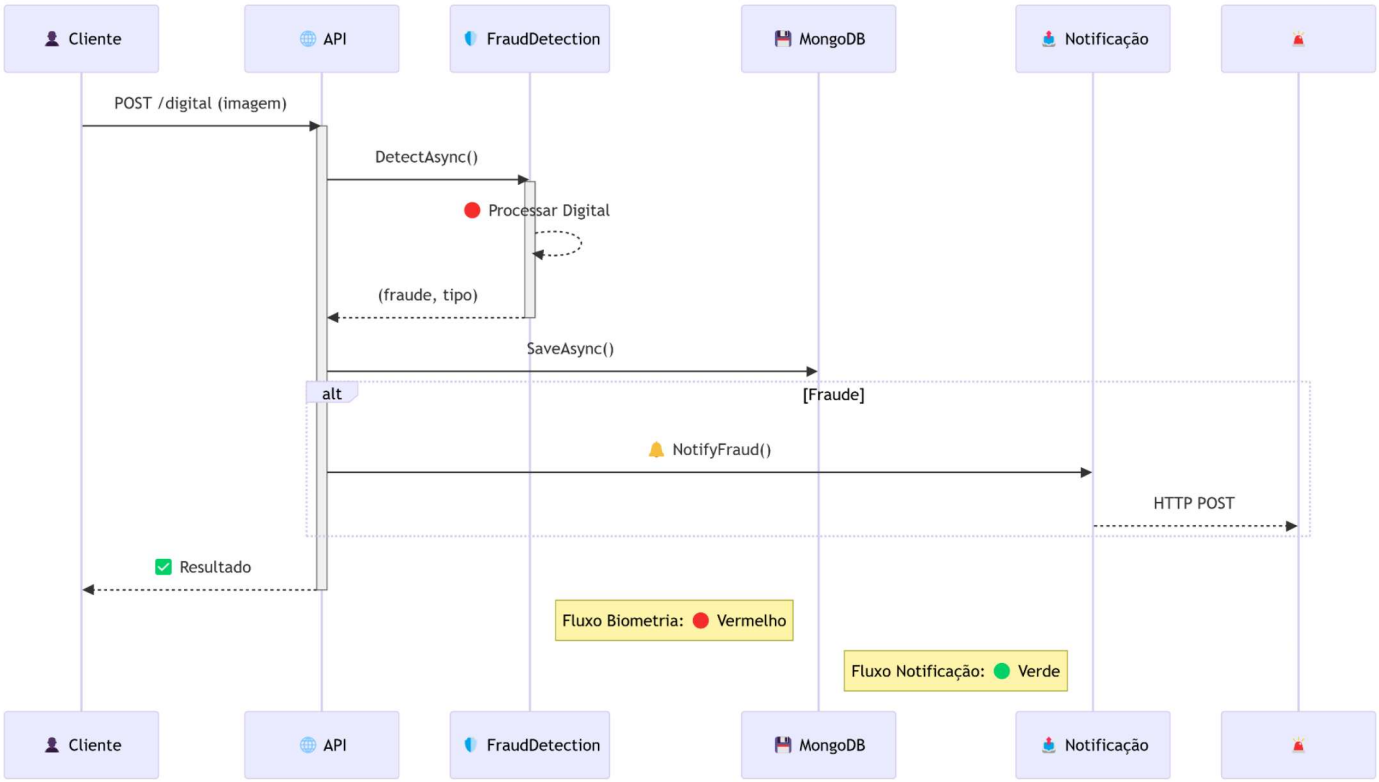


Diagrama de Containers do Sistema Antifraude

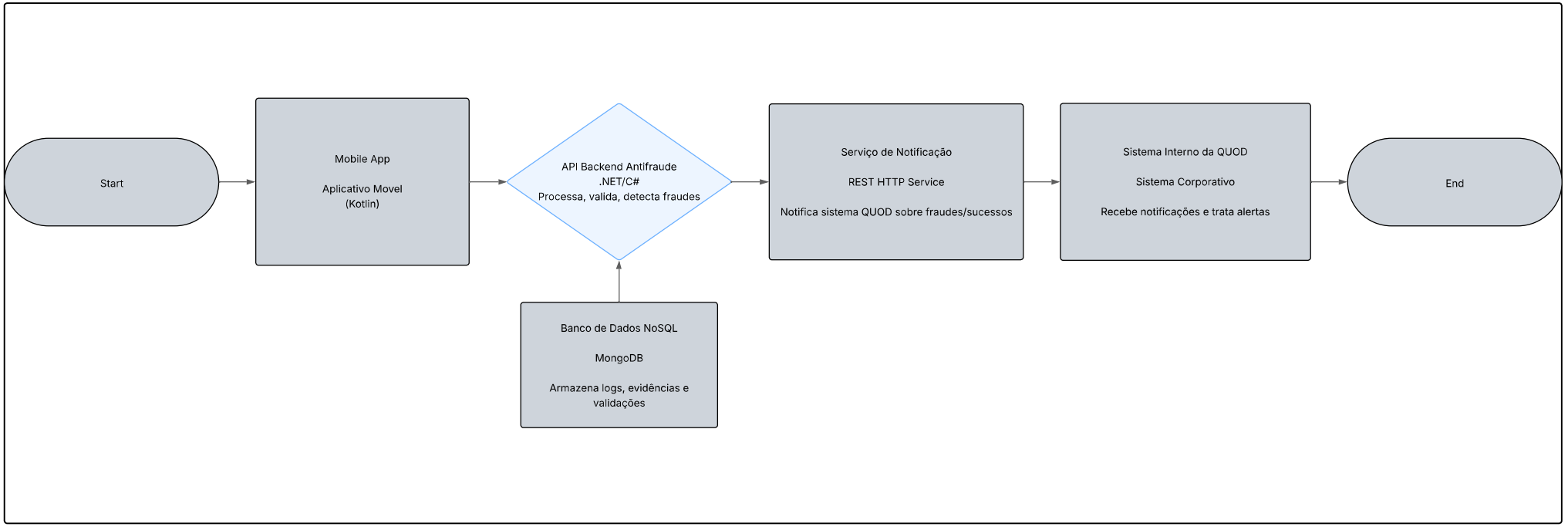


Diagrama de Contexto

O sistema é dividido em containers como:

Aplicativo Mobile (front-end)

Backend Antifraude (.NET ou Spring Boot)

Banco NoSQL

Serviço de Notificação

Sistema interno da QUOD (externo)

Diagrama de Contexto

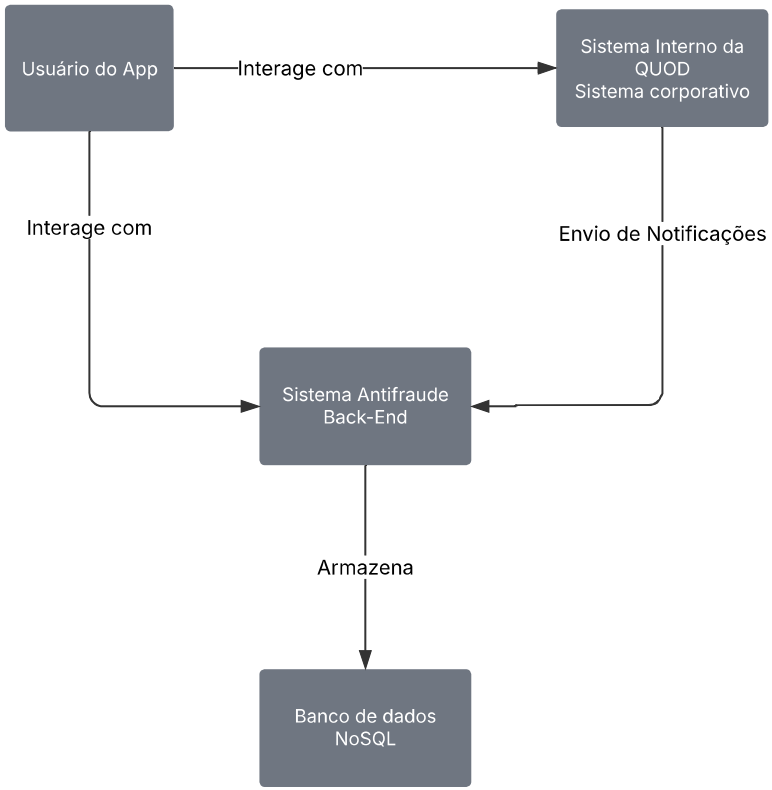


Diagrama de Componentes – Back-End Antifraude

Camadas e Componentes:
1. Controller (Camada de entrada – API REST):

- **Responsabilidade:** Receber requisições externas (do app ou outros serviços) e encaminhá-las aos serviços apropriados.
- **Componentes:**
  - BiometriaController: Expõe endpoints para validação de biometria facial e digital.
  - DocumentoController: Expõe endpoints para análise de documentos (documentoscopia).

2. Service (Camada de negócio):

- **Responsabilidade:** Implementar a lógica de validação, análise e comunicação.
- **Componentes:**
  - ValidadorImagemService: Executa validações básicas das imagens (formato, qualidade, metadados).
  - FraudeDetectionService: Realiza validações avançadas, como detecção de deepfake, uso de máscara ou foto de foto.
  - NotificacaoService: Realiza a integração com o sistema de notificação da QUOD, enviando os dados quando necessário.

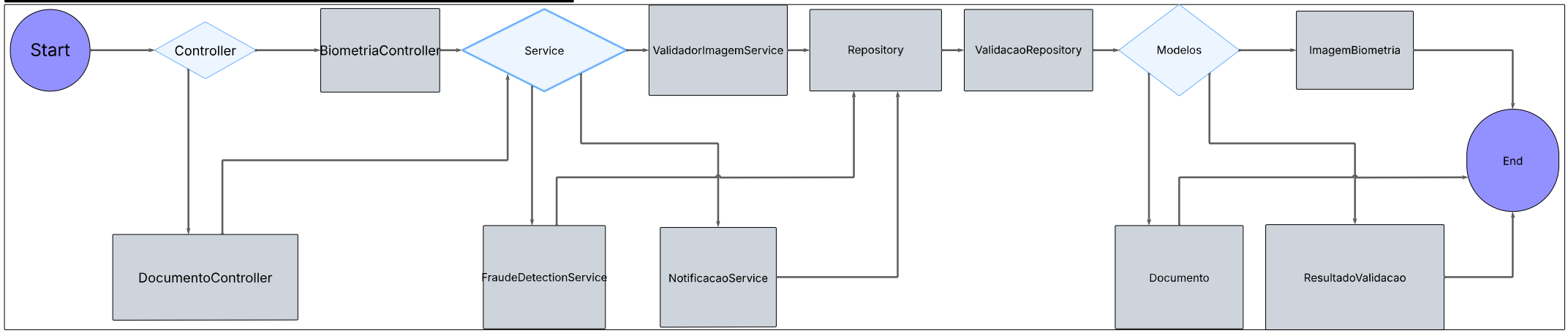
3. Repository (Camada de persistência):

- **Responsabilidade:** Gerenciar a gravação e recuperação dos dados no banco NoSQL.
- **Componentes:**
  - ValidacaoRepository: Grava e consulta os registros de validações e fraudes processadas.

4. Modelos (Entidades de dados):

- **Responsabilidade:** Representar os objetos manipulados pela aplicação.
- **Componentes:**
  - ImagemBiometria: Estrutura que armazena os dados da imagem facial ou digital.
  - Documento: Estrutura que representa os dados da imagem do documento capturado.
  - ResultadoValidacao: Representa o resultado final da validação, indicando sucesso ou fraude e contendo os metadados relevantes.

Diagrama de Componentes do Back-End API



3 - Detalhes das bibliotecas e referências utilizadas para desenvolvimento do aplicativo:

1. Microsoft.AspNetCore.Http.Abstractions
    - Descrição: Fornece interfaces e tipos para manipulação de requisições e respostas HTTP, como IFormFile para upload de arquivos.
    - Uso: Facilita a manipulação de uploads e dados HTTP em APIs.

2. Microsoft.Extensions.Options.ConfigurationExtensions
    - Descrição: Permite o uso de configurações fortemente tipadas via arquivos de configuração (appsettings.json, por exemplo).
    - Uso: Utilizada para injetar configurações como MongoSettings e NotificationSettings via DI.

3. SixLabors.ImageSharp
    - Descrição: Biblioteca moderna para processamento e manipulação de imagens em .NET.
    - Uso: Utilizada para operações de leitura, transformação e análise de imagens, especialmente em funcionalidades de documentoscopia.

4. Tesseract
    - Descrição: Wrapper .NET para o mecanismo OCR Tesseract.
    - Uso: Permite extração de texto de imagens, fundamental para leitura de documentos e validação de informações.

5. Microsoft.AspNetCore.App (FrameworkReference)
    - Descrição: Inclui todos os pacotes essenciais do ASP.NET Core, como MVC, DI, autenticação, etc.
    - Uso: Necessário para desenvolvimento de APIs modernas, controllers, middlewares e integração com o pipeline HTTP.

6. MongoDB.Driver, MongoDB.Bson
    - Descrição: Bibliotecas oficiais para acesso e manipulação de bancos de dados MongoDB em .NET.
    - Uso: Utilizadas para persistência e consulta de dados não relacionais, como documentos e logs de validação.

7. Swashbuckle.AspNetCore (Swagger)
    - Descrição: Gera documentação interativa (Swagger UI) para APIs RESTful.
    - Uso: Facilita testes e integração de endpoints durante o desenvolvimento.

Essas bibliotecas, combinadas com a arquitetura modular do projeto (Core, Infrastructure, Services), proporcionam uma base robusta para o desenvolvimento de APIs seguras, escaláveis e com recursos avançados de processamento de documentos e antifraude.