Documento de Arquitetura do Backend

1. Visão Geral

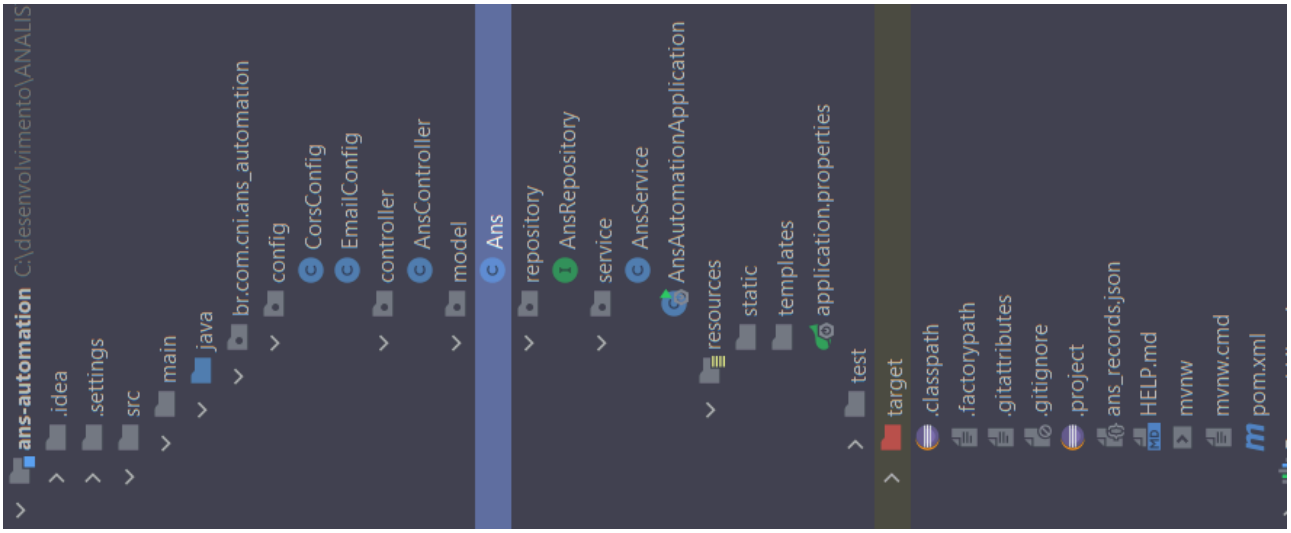
O backend é uma API REST desenvolvida em **Java** com **Spring Boot**, responsável por receber arquivos Excel (`.xlsx`) enviados pelo frontend, processar os dados usando a biblioteca **Apache POI**, e retornar um JSON com os indicadores ANS. O arquivo JSON processado (`ans_records.json`) é salvo no servidor e retornado ao frontend para download.

2. Tecnologias Utilizadas

- **Java 17**: Linguagem principal para o backend.
- **Spring Boot 3**: Framework para construção da API REST.
- **Spring Web**: Para endpoints REST e upload de arquivos.
- **Apache POI**: Biblioteca para leitura e manipulação de arquivos Excel (`.xlsx`).
- **Jackson**: Para serialização/deserialização de JSON.
- **Spring Boot Starter Validation**: Para validação de entradas.
- **Maven**: Gerenciador de dependências.
- **Outras dependências**:
  - `spring-boot-starter-web`: Para suporte a REST.
  - `org.apache.poi:poi` e `org.apache.poi:poi-ooxml`: Para manipulação de Excel.
  - `lombok`: Para redução de código boilerplate (opcional).

3. Estrutura do Projeto

A aplicação segue a estrutura padrão do Spring Boot:



3.1. Componentes

- **Controladores** (`AnsController.java`):
  - Define o endpoint `/upload` (POST) para receber arquivos Excel.
  - Valida o arquivo e delega o processamento ao serviço.
- **Serviços** (`AnsService.java`):

        - Processa o arquivo Excel usando `Apache POI`.
        - Extrai os dados, converte para uma lista de `AnsRecord`, e gera `ans_records.json`.
        - Salva o JSON no diretório `output/`.
- **Modelos** (`AnsRecord.java`):

Classe que representa um registro ANS:
```
public class AnsRecord {
    private Long indicadorId;
    private String superintendencia;
    private String indicador;
    private Double meta2024;
    private Double real2024;
    // Getters e setters
}
```

        -

- **Diretórios**:
    - `uploads/`: Armazena temporariamente os arquivos enviados.
    - `output/`: Armazena os arquivos JSON gerados (há ser desenvolvido como melhoria).

4. Endpoints

- **POST /upload**:
    - **Descrição**: Recebe um arquivo Excel, processa os dados, e retorna um JSON com os indicadores.
    - **Parâmetros**:
        - `file`: Arquivo Excel (`MultipartFile`, via multipart/form-data).
    - **Resposta**:
        - Sucesso: `200 OK` com JSON dos registros (`List<AnsRecord>`).
        - Erro: `400 Bad Request` (arquivo inválido) ou `500 Internal Server Error` (falha no processamento).

Exemplo:
```
curl -X POST -F "file=@avaliação_técnica.xlsx" http://localhost:8080/upload
```

    -

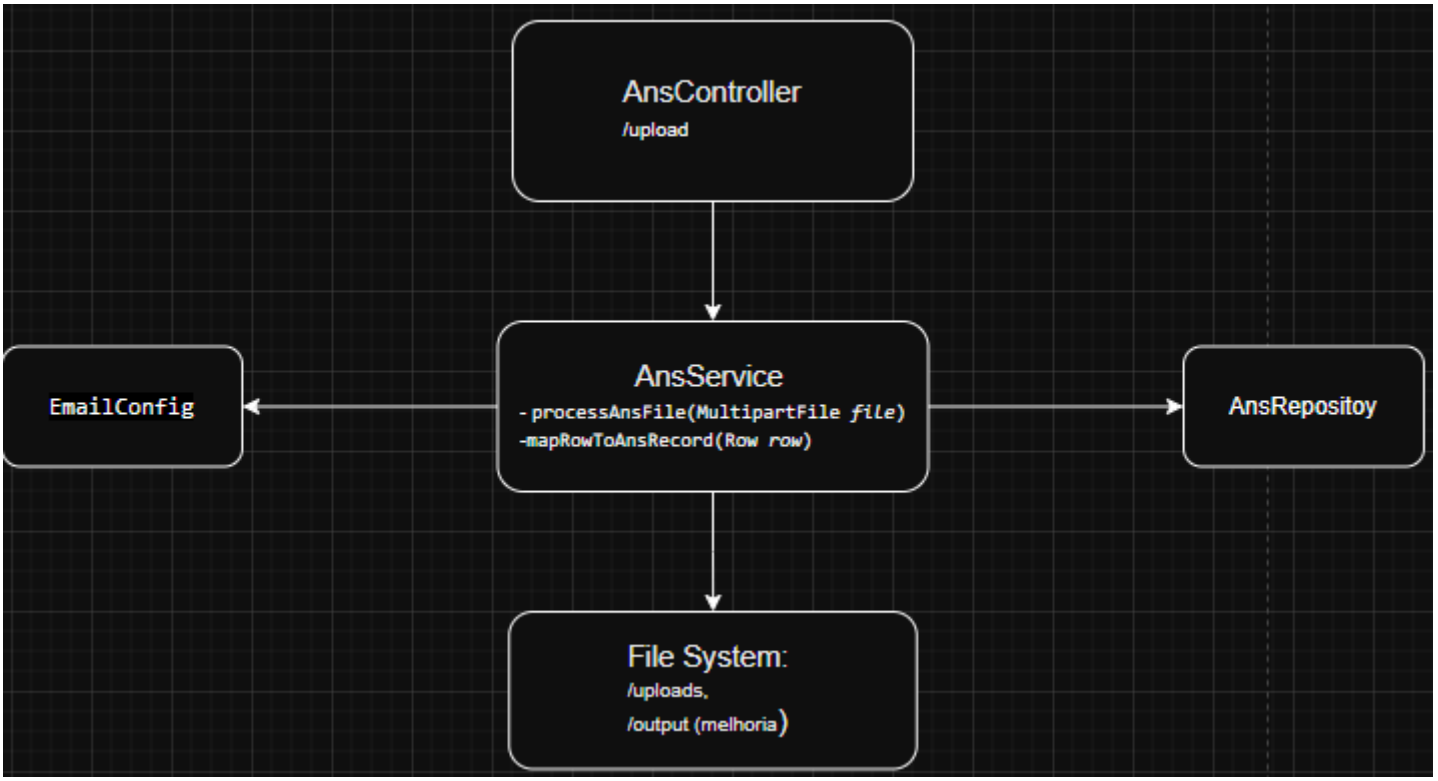
5. Fluxo de Processamento

1. O frontend envia o arquivo Excel via POST para `/upload`.
2. O `AnsController` recebe o arquivo como `MultipartFile` e valida o tipo (`.xlsx`) e tamanho.
3. O `AnsService` usa `Apache POI` para:
    - Ler o arquivo Excel.
    - Extrair os dados das linhas/colunas correspondentes a `indicadorId`, `superintendencia`, etc.
    - Mapear os dados para uma lista de `AnsRecord`.
4. O `AnsService` converte a lista de `AnsRecord` para JSON usando `Jackson`.
5. Caso seja necessário guardar informações no banco de dados, já esta configurado a classe `AnsRepository`.
6. O JSON é salvo em `output/ans_records.json`.
7. O backend retorna o JSON ao frontend, que o exibe na tabela e inicia o download.

6. Integração com o Frontend

- O `FileUploadService` do frontend faz uma requisição POST para `/upload` com o arquivo Excel.
- O backend retorna o JSON, que é exibido na tabela do `FileUploadComponent` e baixado como `ans_records.json`.
- Comunicação via HTTP com CORS habilitado (configurado em `application.properties` ou classe de configuração).

7. Diagrama de Componentes



8. Considerações

- **Escalabilidade**: O backend pode ser escalado com um banco de dados (e.g., PostgreSQL) para persistência dos registros.
- **Segurança**:
  - Validar o tipo e tamanho do arquivo no controlador.
  - Implementar autenticação (e.g., Spring Security com JWT) para proteger o endpoint.
- **Performance**: Cache de arquivos processados pode ser implementado para evitar reprocessamento.
- **Manutenção**: Estrutura modular (controladores, serviços, modelos) facilita adição de novos endpoints.
- **Dependências**:
  - Assegurar que `Apache POI` e `Jackson` estejam configurados corretamente no `pom.xml`.

9. Próximos Passos

- Implementar endpoint para gerar `ans_report.json` com análises estatísticas.
- Adicionar validação avançada de dados no Excel (e.g., checar valores nulos).
- Integrar com um banco de dados para persistência dos indicadores.
- Configurar testes unitários com JUnit e Mockito.
- Criação do endpoint `output/,` para armazenar em um diretório na rede, para expor o arquivo .json para o Power BI.