Documento de Arquitetura do Frontend

1. Visão Geral

O frontend é uma aplicação web desenvolvida em **Angular 17**, projetada para permitir o upload de arquivos Excel contendo indicadores ANS, exibir os dados em uma tabela paginada, e possibilitar o download de um arquivo JSON (`ans_records.json`). A interface é responsiva, centralizada, e inclui uma toolbar fixa no topo para identificação do sistema.

2. Tecnologias Utilizadas

- **Angular 17**: Framework para construção da aplicação SPA.
- **TypeScript**: Linguagem para lógica do frontend.
- **HTML5/CSS3**: Estrutura e estilização da interface.
- **RxJS**: Gerenciamento de chamadas assíncronas para comunicação com o backend.
- **Bootstrap** (opcional): Para estilos adicionais, se integrado.
- **Bibliotecas**:
  - `@angular/forms`: Para manipulação de inputs de arquivo.
  - `@angular/common/http`: Para chamadas HTTP ao backend.

3. Estrutura do Projeto

A aplicação segue a estrutura padrão do Angular:

```
src/
├── app/
│   ├── file-upload/
│   │   ├── file-upload.component.ts
│   │   ├── file-upload.component.html
│   │   ├── file-upload.component.css
│   │   ├── file-upload.service.ts
│   │   ├── ans-record.model.ts
│   ├── app.component.ts
│   ├── app.module.ts
├── assets/
├── styles.css
├── index.html
```

3.1. Componentes

- **FileUploadComponent**:
  - **Função**: Interface principal para upload de arquivos Excel, exibição de dados em tabela, e download de JSON.
  - **Funcionalidades**:
    - Upload de arquivos `.xlsx` via input de arquivo.
    - Exibição de tabela com 5 colunas (`ID Indicador`, `Superintendência`, `Indicador`, `Meta 2024`, `Real 2024`).

        - Paginação (10 registros por página) com botões "Anterior" e "Próximo".
        - Toolbar fixa no topo com título "ANS - Acordos de Níveis de Serviços".
    - **Propriedades**:
        - `selectedFile`: Armazena o arquivo selecionado.
        - `records`: Lista de registros ANS carregados.
        - `currentPage` e `pageSize`: Controle de paginação.
    - **Métodos**:
        - `onFileSelected()`: Manipula a seleção do arquivo.
        - `upload()`: Envia o arquivo ao backend e processa a resposta.
        - `previousPage()` e `nextPage()`: Navegação na tabela paginada.

3.2. Serviços

- **FileUploadService**:
    - **Função**: Comunicação com o backend via HTTP.
    - **Métodos**:
        - `uploadFile(file: File)`: Envia o arquivo Excel para o endpoint `/upload` e retorna os registros processados como JSON.

3.3. Modelos

- **AnsRecord**:

Interface TypeScript que define a estrutura dos registros ANS:

```
export interface AnsRecord {
  indicadorId: number;
  superintendencia: string;
  indicador: string;
  meta2024: number;
  real2024: number;
  // Outros campos, se necessário
}
```

    -

4. Fluxo de Dados

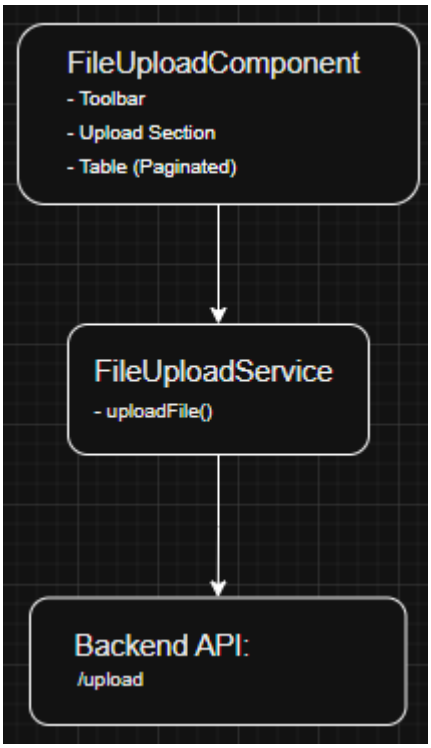
1. O usuário seleciona um arquivo Excel (`.xlsx`) no `FileUploadComponent`.
2. O método `onFileSelected()` armazena o arquivo em `selectedFile`.
3. O usuário clica em "Enviar", acionando `upload()`.
4. O `FileUploadService` envia o arquivo para o backend via POST (`/upload`).
5. O backend processa o arquivo e retorna um JSON com os registros.
6. O frontend armazena os registros em `records` e exibe na tabela paginada.
7. O JSON retornado é baixado automaticamente como `ans_records.json`.

5. Interface do Usuário

- **Toolbar**: Fixa no topo, com fundo azul (`#007bff`) e texto branco ("ANS - Acordos de Níveis de Serviços").
- **Seção de Upload**:
    - Input de arquivo (aceita `.xlsx`).

    - Botão "Enviar" (desabilitado até selecionar um arquivo).
    - Mensagem de status (e.g., "Arquivo JSON gerado com sucesso!").
- **Tabela**:
    - 5 colunas: `ID Indicador` (120px), `Superintendência` (350px), `Indicador` (300px), `Meta 2024` (80px), `Real 2024` (80px).
    - Paginação com 10 registros por página.
    - Centralizada, com fonte menor (12px) e linhas alternadas.
- **Estilização**:
    - CSS customizado (`file-upload.component.css`).
    - Toolbar com `position: fixed` e `z-index: 1000`.
    - Tabela com `max-width: 950px`, centralizada via `flex`.

6. Diagrama de Componentes



7. Considerações

- **Escalabilidade**: O componente pode ser estendido para suportar mais colunas ou filtros na tabela.
- **Manutenção**: Código modular, com serviços separados para facilitar atualizações.
- **Responsividade**: Ajustes no CSS podem ser feitos para telas menores.
- **Segurança**: Validar o tipo e tamanho do arquivo no frontend antes do upload.

8. Próximos Passos

- Integrar filtros na tabela (e.g., por `Superintendência`).
- Adicionar validação de dados no frontend antes do upload.
- Implementar um dashboard com gráficos (e.g., Power BI) para visualização dos indicadores.