**Separar Claramente Backend e Frontend**

* **Rest APIs com Flask/FastAPI**: Mantenha o backend em Python separado da interface, criando APIs REST para manipulação dos dados. A interface pode ser desenvolvida com qualquer framework moderno como **Bootstrap** para o front-end (HTML, CSS e JavaScript). Isso facilita na manutenção e evolução do projeto e evita que as partes se tornem dependentes.
* **Microfrontends**: Se precisar de algo mais modular, pode dividir a interface em pequenos módulos independentes que interagem com o backend via APIs.

**1. Frontend: Linguagens e Frameworks**

**A. JavaScript (para interatividade e dinamismo)**

* **React.js**:
  + Uma das melhores opções para criar interfaces modernas e interativas.
  + Componentes reutilizáveis que facilitam a construção e manutenção.
  + Grande ecossistema, com bibliotecas como **Material-UI** ou **Ant Design** para criar interfaces atraentes e responsivas com pouco esforço.
  + Possibilidade de usar **React Router** para criar navegação fluida entre as páginas da aplicação.
  + Ótima integração com APIs RESTful, ideal para o backend que você já está construindo com **Flask** ou **FastAPI**.
* **Vue.js**:
  + Simples e rápido para prototipar interfaces.
  + Fácil de integrar em projetos existentes ou novos.
  + Oferece componentes reativos e integração rápida com bibliotecas de UI como **Vuetify** ou **ElementUI** para criar designs limpos e responsivos.
* **Svelte**:
  + Um framework JavaScript emergente, que foca na performance e facilidade de uso.
  + Mais eficiente do que React e Vue em termos de manipulação de DOM, tornando a interação mais rápida e a interface mais fluida.

**B. HTML5 e CSS3 (estrutura e estilo)**

* **Tailwind CSS**:
  + Uma biblioteca CSS utilitária que permite criar interfaces altamente customizáveis sem escrever CSS puro.
  + Permite a criação de designs responsivos e flexíveis, otimizando a experiência em diferentes dispositivos.
* **Bootstrap**:
  + Um framework CSS popular para construir interfaces responsivas rapidamente.
  + Oferece um conjunto de componentes prontos (botões, formulários, tabelas) que podem ser estilizados de forma customizável.
* **SCSS/SASS** (pré-processadores CSS):
  + Aumenta a flexibilidade e permite modularizar o código CSS para projetos grandes.

**2. Backend: Linguagens e Frameworks**

**A. Python com Flask ou FastAPI**

* Como você já está utilizando Python, manter Flask ou FastAPI como backend faz sentido, pois você já tem funções criadas para processamento de dados.
* **FastAPI** é uma ótima escolha para criar APIs performáticas e responsivas.
  + **Vantagens**:
    - Rápido e leve, ideal para aplicações modernas.
    - Facilita a comunicação com o frontend via APIs REST.
    - Possui ótima documentação e integração fácil com sistemas de autenticação, como OAuth2.
* **Django (com Django REST Framework)**:
  + Se você precisa de um sistema robusto e completo com autenticação, administração, e APIs REST prontas para consumo.
  + Ideal para sistemas com muitos relacionamentos entre entidades, onde você precisará de um banco de dados estruturado.

**3. Backend e Frontend Juntos: Full-Stack Frameworks**

**A. Next.js (para React)**

* Um framework full-stack que utiliza React para o frontend, mas também oferece um backend básico (Node.js). Ele facilita a criação de rotas dinâmicas e páginas otimizadas para SEO.
* Permite criar interfaces ricas com a lógica backend integrada diretamente ao frontend, economizando tempo de desenvolvimento.

**B. Nuxt.js (para Vue)**

* Um framework similar ao Next.js, mas para o Vue.js. Ele oferece vantagens similares para renderização no lado do servidor (SSR), roteamento automático e otimização de desempenho.

**C. Blazor (para C#)**

* Se você preferir uma abordagem de código mais próxima de C# e .NET, **Blazor** é uma alternativa interessante. Ele permite criar interfaces dinâmicas como React e Vue, mas usando C# no backend e frontend.

**4. UI Kits para Interface Moderna**

Além das linguagens, utilizar **UI Kits** prontos pode acelerar o desenvolvimento da interface e garantir um design moderno e consistente:

**A. Material UI (React)**

* Um kit de componentes prontos e estilizados, inspirado no design do **Google Material**.
* Funciona muito bem com React e Vue.js, permitindo criar interfaces profissionais e responsivas.

**B. Vuetify (Vue.js)**

* Uma biblioteca de componentes UI baseada no Material Design, específica para o Vue.js.
* Facilita o desenvolvimento de layouts responsivos e sistemas interativos.

**C. Ant Design**

* Um kit de UI focado em projetos empresariais, altamente customizável, com uma grande variedade de componentes prontos.
* Compatível com **React** e **Vue.js**.

**5. Ferramentas de Visualização de Dados (para os gráficos e tabelas)**

Já que o seu projeto envolve análise de dados (artigos, cálculos de pontuação, etc.), integrar bibliotecas de visualização de dados ajudará a gerar gráficos interativos e dinâmicos:

**A. Chart.js** (para gráficos simples):

* Biblioteca leve e fácil de usar com Vue ou React, ideal para gráficos de barra, linha, e radar.

**B. D3.js** (para visualizações mais customizadas):

* Uma biblioteca poderosa, porém mais complexa, que permite criar gráficos e visualizações de dados customizadas de alto nível.

**C. Plotly**:

* Ideal para criar gráficos interativos, tanto no backend (Python) quanto no frontend (JavaScript).
* Integração com React e Vue.js é possível para gráficos mais complexos (scatter, box plots, 3D).

**6. Ferramentas de Desenvolvimento e Integração**

**A. Docker**

* Para facilitar o desenvolvimento e a implantação de todo o sistema, utilizando containers para o backend e frontend.
* Docker permite isolar o backend Python (FastAPI/Flask) do frontend (React/Vue), facilitando o deploy em produção.

**B. Git e GitHub/GitLab**

* Controle de versão e colaboração.
* Use **GitHub Actions** ou **GitLab CI** para automação e deploy contínuo (CI/CD).

**Conclusão: Melhor Linguagem e Framework para seus Objetivos**

Com base no que você já construiu e nos seus objetivos de criar uma interface moderna e funcional, a melhor abordagem seria:

1. **Frontend**:
   * **React.js** (com **Material UI** ou **Ant Design** para uma interface moderna).
   * **Vue.js** (com **Vuetify**) se você preferir algo mais simples e leve.
2. **Backend**:
   * **FastAPI** (pela facilidade de integração com o frontend e pela performance).
   * **Flask** se você preferir manter o que já começou, mas considere a transição para FastAPI.
3. **Visualização de Dados**:
   * **Chart.js** para gráficos simples e interativos.
   * **Plotly** se precisar de gráficos mais detalhados.
4. **Infraestrutura**:
   * **Docker** para organizar o desenvolvimento e o deploy.
   * **Git** para controle de versão e integração contínua.

Isso dará flexibilidade, performance e uma interface moderna e interativa para seus usuários. Se precisar de ajuda para começar com qualquer um desses pontos, podemos detalhar o setup inicial em um novo chat.

Vamos criar o o segundo FastAPI, por exemplo, xlsx\_csv\_api.py. Quero extrair os nomes do docentes, sua categoria (Docente Permanente - DP ou Docente Colaborador), os nomes dos discentes, a classificação qualis, SJR,SJR Best Quartile, etc.

Para cada arquivo tem uma configuração diferente para a extração, vamos 1 por 1:

- C:\Users\Decivil\Dropbox\PPGEC\Indicadores\chatgpt\FastAPI\Docentes PPGEC.xlsx

Nesse arquivo, a coluna A (NOME) tem o nome do docente e a coluna B (CATEGORIA) tem a categoria do docente que pode ser Permanente (DP) ou Colaborador (DC)

- C:\Users\Decivil\Dropbox\PPGEC\Indicadores\chatgpt\FastAPI\Todos\_Discentes\_2021\_2024.json - Estão todos os nomes de todos os discentes entre 2021 e 2024

C:\Users\Decivil\Dropbox\PPGEC\Indicadores\chatgpt\FastAPI\Classificação Qualis 2017\_2020.xlsx

- Neste arquivo estão os qualis dos periódicos, com o ISSN extraído no primeiro FastAPI, podemos correlacionar com ISSN extraído desse arquivo que está na coluna A (ISSN) e retornar a classificação Qualis que está na coluna D (Estrato)

C:\Users\Decivil\Dropbox\PPGEC\Indicadores\chatgpt\FastAPI\ scimagojr 2023.csv – Nesse arquivo podemos correlacionar com ISSN extraído no primeiro FastAPI e retornar o SJR (no arquivo em questão seria o “SJR”) e o melhor quartil (no arquivo em questão seria o “SJR Best Quartile”) desse periódico.