uvicorn main:app --reload

uvicorn extrair\_docentes\_discentes:app –reload

<http://127.0.0.1:8000/docs>

"LIMA, VICTOR M.E.; ESTOLANO, AMANDA M.L.; ALMEIDA, YEDA M.B.; HENRIQUE, MARIANA A.; ARAÚJO, FERNANDA W.C.; Antônio Acacio de Melo Neto"

Com base em tudo o que você já explicou e nas suas necessidades, aqui estão as linguagens e frameworks mais adequados para construir uma interface moderna, visualmente atraente e com fácil interação para o usuário final, viáveis para atingir os seus objetivos:

### \*\*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.

De acordo com o backend em fastAPI que desenvolvi, você indica qual? Saiba algumas das funcionalidades que vou querer : - Campos editáveis com sua alteração implicando no recalculo em tempo real ou atraves de um botão - Vou querer uma tabela com tudo que o meu backend retornou e outra abaixo com os calculos das pontuações de acordo com formulas que vou fornecer - Vou querer exportar os resultados

Ok, vamos fazer agora a Tabela 1 - Classificação Qualis, posicionada antes da Tabela 2, mas como o mesmo estilo. Vai ter um cabeçalho com as seguintes colunas: Ano A1 A2 A3 A4 B1 B2 B3 B4 até C e N (não tem classificação Qualis). Depois teremos mais 3 colunas #A (total no estrato A), #B (total no estrato B) e #Ano (total no ano).

Depois as linhas serão por ano. A ultima linha é o total. Quero essa tabela dinâmica,

uvicorn main:app –reload

<http://127.0.0.1:8000/docs>

npm start

http://localhost:3000