Frontend

Para o desenvolvimento do frontend, começamos com o template Next.js da Vercel, pré-configurado com Tailwind para estilos, e utilizando o App Router para navegação e roteamento.

Para as bibliotecas externas, optamos por usar PNPM para instalar e gerenciar as dependências. As principais bibliotecas utilizadas incluem:

Axios: para comunicação via HTTP com as Functions.
Firebase / Firebase Tools: para comunicação e gestão dos serviços do Firebase.
As páginas da aplicação foram divididas em dois escopos: "no-auth" (usuário não autenticado), que inclui login e cadastro, e "auth" (usuário autenticado), que representa a plataforma em si.

Para a prova de conceito (POC), as partes da lógica do sistema implementadas no frontend incluem o fluxo de autenticação do usuário, o cadastro de um arquivo PDF de edital (ou cadastro manual do conteúdo) e uma versão inicial do fluxo de pagamento.

Para o fluxo de pagamento, criamos uma conta na Stripe e configuramos um processo básico que redireciona o usuário para a plataforma da Stripe, onde ele pode assinar um plano. Toda a gestão da assinatura e pagamentos é feita através da Stripe. Escolhemos a Stripe pela facilidade de implementação e gestão dos pagamentos, o que nos permite dedicar mais tempo e flexibilidade aos outros aspectos da aplicação.

Decidimos manter a lógica principal da aplicação dentro das Functions do Firebase, o que nos proporciona maior liberdade, pois é um projeto separado, mas a base de código está integrada com o frontend.

Dentro das Functions, desenvolvemos o processamento do conteúdo do edital, que envolve receber um arquivo PDF, filtrar o conteúdo programático (matérias e conteúdos da prova), enviar esse conteúdo para o Gemini (que extrai e

retorna um JSON a partir dessas informações) e, por fim, retornar essas informações formatadas para o frontend.

Na parte de internacionalização usamos uma biblioteca, a i18next (conhecida também como i18n) e juntamente com um plugin, o i18next browser language detector, para configurar o i18n é necessário criarmos arquivos JSON que vão conter as traduções de cada linguagem, seguindo as mesmas keys, e carregá-los em objetos no código, assim podemos inicializar o i18n, nesse momento o plugin foi usado para inicializar a língua padrão do usuário baseada na linguagem do browser em que a plataforma foi aberta, e adicionado configurações e uma hierarquia para a troca de linguagem no sistema como querystring, local storage, cookies, etc.

Layout

No layout utilizamos o Figma para fazer toda a estilização do site. O Design no figma foi pensado para criamos a identidade visual do site antes da implementação, testar a visibilidade e o fluxo da jornada do usuário dentro da nossa aplicação.

Criamos 5 telas iniciais, passando pelo login, Dashboard principal e visualização do cronograma do usuário.

Plano de testes

Para o plano de testes foi utilizado um modelo disponibilizado pelo professor. Fizemos o plano baseado nos requisitos da aplicação que foram desenvolvidos na POC, afim de garantir a qualidade da aplicação e evitar possíveis erros e falhas durante o desenvolvimento do projeto.