



# Hiring Challenge (Soft. Engineers Fullstack) - Live or Take Home version

Desafio técnico para a posição de Software Engineering full stack.



## Sumário

Sumário

Preparação

Desafio

[Frontend](#)

[Backend](#)

[Arquivo CSV do desafio](#)

[Boilerplate base para desenvolvimento do projeto](#)

[Informação importante:](#)

---



## Preparação

Nessa etapa, enviamos um documento para o candidato se preparar para o *live test ou para fazer em casa (take home)*.

### Preparação

Você tem 2 opções para realizar nosso teste técnico: ~~tomar a pílula vermelha e encarar a verdade ou a pílula azul e viver na ilusão.~~

1. *Take home*: implementar o teste no conforto da sua casa e utilizar o tempo de sessão de teste técnico para fazer code review com alguns membros de nosso time, ou;
2. *Live code*: fazer a implementação ao vivo durante a sessão de teste técnico. Afinal quem sabe faz ao vivo não é mesmo?

## Optou por fazer em casa?

Toda informação necessária está no documento que descreve o escopo do teste.

Mesmo fazendo em casa, recomendamos a utilização de nosso *boilerplate* para se basear e implementar o desafio.

Lembre-se de analisar as decisões tomadas, documentar e claro: **divirta-se!**



## Caso opte por *LIVE CODING*: Por que *Live Coding*?

Na Kanastra 🏔, resolvemos adotar como fase de **teste técnico** do processo seletivo, uma etapa de *Live Coding*. Temos total noção de como uma etapa de *Live Coding* pode trazer uma sensação de pressão e nervosismo, mas gostaríamos deixar claro que nosso propósito é **totalmente contrário a isso!**

Adotamos a etapa de *Live Coding* com o intuito de ter uma experiência de **como seria resolver um problema em par com você** (acreditamos e utilizamos muito do *pair programming* em nosso dia a dia na empresa) e também **reduzir o tempo total de duração do processo**, diminuindo a fase de teste prático de

dias (que era um *case* técnico maior para o candidato realizá-lo e enviar a solução para revisarmos) para apenas 1 dia, numa etapa *live*.

Por isso **relaxe, pense junto e divirta-se!** 😊

## Sobre o teste

Nesse teste, esperamos que ele tenha duração de no máximo 1:30h e iremos simular os requisitos técnicos para a construção de uma *feature* que temos na plataforma da Kanastra. Essa funcionalidade envolve *upload* de arquivos, através de um *frontend* que consequentemente comunica-se com um *backend* e dessa forma buscamos validar os conhecimentos básicos de um engenheiro de software fullstack para Kanastra 🏔.

No momento do teste, você receberá todos os detalhes e requisitos necessários.

## Instruções iniciais

- Para ser mais produtivo, em caso de *live coding*, você deve chegar com seu ambiente de desenvolvimento preparado. Tanto para começar algum desenvolvimento quanto para compartilhar a tela;
- Para testar as habilidades em **fronted**, teremos um *boilerplate* em React, onde você precisará entender os componentes do projeto e completar os casos de uso que serão explicados;
- Para **backend**, a **linguagem** e **IDE** que irá utilizar é uma decisão completamente **pessoal** do candidato, aqui na Kanastra nós amamos as linguagens PHP, TypeScript e Python!
- Recomendamos utilizar frameworks de desenvolvimento. Aqui na Kanastra nós utilizamos alguns como: Laravel, Django e FastAPI.



Se você é um Software Engineer pleno, Sr. Software Engineer, ou tem senioridade ainda maior, **Unit and integration tests** são mandatórios nos desafios para casa. Não aceitaremos cases sem eles.

## Boilerplate *frontend*

Segue *link* do repositório de um *boilerplate* para *frontend* em React que deverá ser utilizado no *live test*: <https://github.com/Kanastra-Tech/kanastra-challenge-boilerplate>

## Pontos de atenção e lembretes

- Lembre-se dos princípios *S.O.L.I.D*;
- Lembre-se de Unit and integration tests (testes **no backend, no frontend**, testes sempre!);
- Lembre-se de *Error Handling*;
- Não será como uma prova de escola. Você poderá e provavelmente precisará consultar o google;
- Pode manter os dois projetos (*frontend* e *backend*) em um só repositório. Facilitará nossa revisão e testes;
- Possivelmente iremos intervir em alguns pontos pra introduzir mudanças de escopo durante problema.

### Documento para enviar ao candidato:



#### Preparação



## Desafio

Você faz parte de um *tech squad* da Kanastra e recebeu o desafio de construir um sistema de cobranças na plataforma. Esse sistema precisa ser capaz de cumprir os seguintes requisitos:

### Frontend

- **Receber um arquivo .csv através de uma interface de formulário.**
- Além disso, criar e manter uma listagem atualizada que funcionará como histórico de arquivos recebidos na interface. **Pode ou não** usar um endpoint da API que você criará como provedor da listagem.

- A rota de upload deve ser diferente da rota de listagem e, além disso, ao subir um novo arquivo, a listagem deve ser atualizada automaticamente.
- Gostaríamos de ver a utilização de *context* do **React**, e a atualização de estados desse *context* por meio de *actions* dentro de um *reducer*.
- Utilize os componentes fornecidos e melhore-os a fim de criar uma boa experiência no frontend.
- Fluidez, performance, re-renders, uso excessivo de hooks como *useEffect/useState/useMemo*; tipagem; *promise, loading* e *error handling*; *context API*; Todos esses são pontos que serão avaliados pela nossa equipe.

## Backend

- O formulário deverá **usar um *endpoint*** da API que você criará para processar o arquivo;
  - Na Kanastra, processamos muitos registros e precisamos de escala. Por esse motivo, o processamento deve ser feito em menos de **60s**;
- Baseado *input* recebido, o sistema precisa regularmente gerar os **boleto**s para cobrança e **disparar** mensagens para os e-mails da lista;
- É necessário configurar um arquivo docker-compose para rodar o projeto dentro de contêineres.

Esse arquivo .csv terá as seguintes colunas:

1. **name** → nome
2. **governmentId** → número do documento
3. **email** → email do sacado
4. **debtAmount** → valor
5. **debtDueDate** → Data para ser paga
6. **debtID** → uuid para o débito

Exemplo do conteúdo do arquivo:

```
name,governmentId,email,debtAmount,debtDueDate,debtId  
John Doe,1111111111,johndoe@kanastra.com.br,1000000.00,2022-10-01,1
```

## Arquivo CSV do desafio

[input.csv](#)

## Boilerplate base para desenvolvimento do projeto

<https://github.com/Kanastra-Tech/kanastra-challenge-boilerplate.git>

## Informação importante:

Entregue o projeto com testes (unitários, integração). Este requisito é eliminatório.