KENZIE

🧠 Desafio Frontend

Tópicos do conteúdo

Introdução

Por favor, leia este documento do começo ao fim, com muita atenção. O intuito deste teste é avaliar seus conhecimentos técnicos com o front-end.

Instruções de entrega do desafio

1. Primeiro, crie um repositório público no Github;
2. Em seguida, implemente o projeto tal qual descrito abaixo, em seu clone local;
3. Faça o push do seu projeto local para um repositório público no Github;
4. Por fim, envie no canvas o link do repositório.

Objetivo

O objetivo do desafio é validar seus conhecimentos nos seguintes tópicos:

1. **JavaScript:** aproveite o desafio para mostrar tudo o que sabe sobre as novas features da linguagem.
2. **React:** siga boas práticas e mantenha o código idiomático. Busque utilizar features recentes e se mantenha atento a problemas comuns como re-renders desnecessários.
3. **EXTRA - TypeScript:** caso opte por usá-lo, mostre todo o seu domínio.
4. **Componentização**

5. **CSS:** vanilla
6. **EXTRA - Sass ou Scss:** pré-processadores.
7. **EXTRA - Testes unitários:**
8. **EXTRA - Testes end-to-end:** caso opte por usá-lo, mostre todo o seu domínio.

Analisaremos seu teste com base nos critérios acima, então dê um show para ficarmos impressionados.

Restrições

1. **Não é permitido:** utilizar frameworks e/ou bibliotecas de UI que não seja o React (como Vue.js ou Angular).
2. **São permitidas:** ferramentas modernas de desenvolvimento como TypeScript, Babel, eslint, webpack, assim como o uso de polyfills (e outras ferramentas para melhorar o suporte a browsers, como Modernizr) e/ou bibliotecas para testes.
3. **São permitidos:** São permitidos pré-processadores de CSS e/ou ferramentas CSS-in-JS.
4. Não é uma regra, mas evite usar lodash, underscore, ramda e similares.

Sobre o desafio

Hoje nossos clientes precisam saber quanto custa antecipar uma transação, e para isso, precisamos desenvolver uma calculadora de antecipação para que os mesmos consigam saber quais valores receberão caso optem por antecipar o recebimento.

Você deverá desenvolver o teste seguindo os requisitos abaixo.

Requisitos

- Use componentização.

- Os períodos de recebimento devem ser configuráveis já que a API pode receber uma lista de periódos para realizar os cálculos.

Extra

Lembrando que extra não é obrigatório, mas seria um diferencial a implementação.

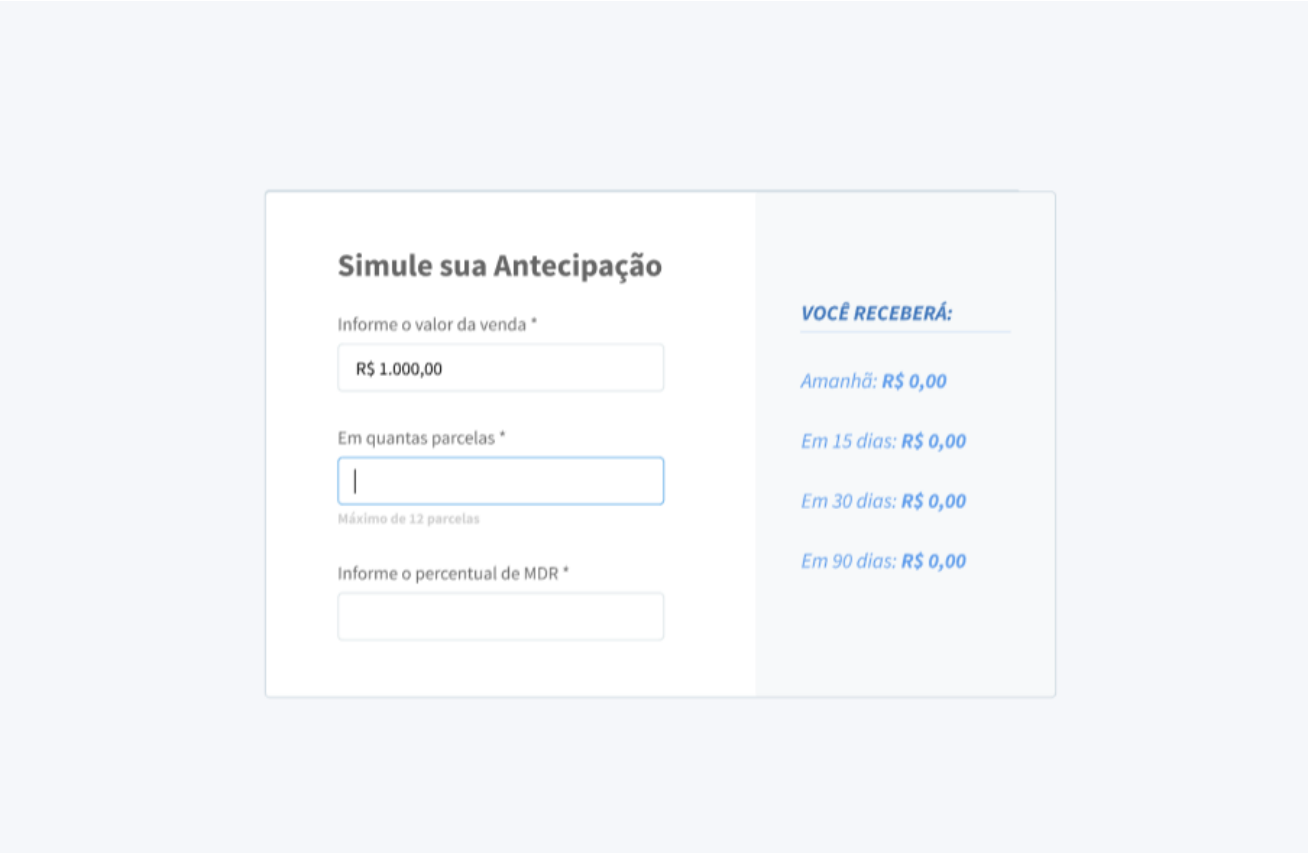
- Faça testes unitários e/ou de ponta-a-ponta (end-to-end)

Os possíveis cenários devem ser cobertos e terem soluções implementadas. Não foi desenvolvido layout para isso, pois queremos observar como você lidará com eles:

- Demora de respostas da API
- Timeout da API
- Conexão lenta
- Usuário estar offline

Front

O layout proposto para essa calculadora pode ser visto na imagem abaixo.



Layout Calculadora

API

Você consumirá uma API já existente no endereço abaixo. Em seguida há uma especificação simplificada dela.

`https://frontend-challenge-7bu3nxh76a-uc.a.run.app`

Post

Parâmetro	Obrigatório?	Tipo	Descrição
`amount`	Sim	`number`	Valor total da transação em centavos
`installments`	Sim	`number`	Número de parcelas
`mdr`	Sim	`number`	É a taxa cobrada pelas adquirentes sobre cada transação de cartão de crédito ou débito
`days`	Não	`Array<number>`	Uma lista com os dias a serem calculadas as antecipações

Layout Calculadora

Exemplo

```
$ curl --request POST \
       --url https://frontend-challenge-7bu3nxh76a-uc.a.run.app \
       --header 'content-type: application/json' \
       --data '{"amount": 15000,
       "installments": 3,
       "mdr": 4
      }'

      {"1":13267,"15":13536,"30":13824,"90":14400}
```

Exemplo informando períodos

```
$ curl --request POST \
       --url https://frontend-challenge-7bu3nxh76a-uc.a.run.app \
       --header 'content-type: application/json' \
       --data '{"amount": 15000,
       "installments": 3,
       "mdr": 4,
       "days": [30, 60, 90]
      }'

      {"30":13824,"60":14208,"90":14400}
```

Simulando Timeout, Internal Server Error e Delay de resposta

Para **Timeout** basta executar a request post passando `timeout` através da query string, exemplo: `https://frontend-challenge-7bu3nxh76a-uc.a.run.app?timeout`

Para **Internal Server Error** basta executar a request post passando `internalError` através da query string, exemplo: `https://frontend-challenge-7bu3nxh76a-uc.a.run.app?`

`internalError`

Para **Delay** de resposta, que pode ser usado como simulador de conexão lenta, basta executar a request post passando `delay`, e informando o tempo do delay em milissegundos, exemplo: `https://frontend-challenge-7bu3nxh76a-uc.a.run.app?delay=tempoEmMilissegundos`

Avaliação

Sua performance será avaliada com base nos seguintes pontos:

1. A aplicação funciona conforme o esperado.
2. Os problemas foram resolvidos com eficiência.
3. A aplicação é fornecida com comandos de instalação e execução para ambientes de desenvolvimento e de testes.
4. Você demonstra conhecimento de como testar as partes críticas da aplicação. Não exigimos 100% de cobertura.
5. A aplicação tem uma estrutura lógica e bem organizada.
6. O teste acompanha documentação com o raciocínio sobre as decisões tomadas.
7. O código está documentado e/ou é de fácil leitura.
8. Segue algum guia de estilo de código padronizado.
9. Possui um histórico do git (mesmo que breve) com mensagens claras e concisas.

Boa sorte!

Queremos sempre evoluir juntos com você

Gostaria de deixar seu feedback?

Separamos uma área que você pode contribuir para uma melhor experiência ao consumir nosso conteúdo

Abrir formulário de feedback