Atividade de Recuperação — Linguagem de Programação

Frontend em JavaScript + Backend em Haskell (JSON)
Tema: Cálculo de Juros Compostos

Prazo final: 16/11/2025 às 23:59 (horário de Brasília)
Envio por e-mail: `juliocesar@cotemig.com.br`

Contexto

Esta atividade substitui um trabalho não entregue no prazo. O objetivo é demonstrar domínio prático de integração **frontend (JS)** com **backend (Haskell)** por meio de uma API JSON simples, aplicando a fórmula de *juros compostos*.

Requisitos Funcionais

RF1 O sistema deve calcular o montante composto: $A = P \cdot (1 + \frac{r}{n})^{n \cdot t}$.

RF2 Entradas pelo **frontend**:

- `principal` (P, valor inicial, número real > 0)
- `rate` (r, taxa anual em decimal, e.g. 0.12 para 12%)
- `timesPerYear` (n, inteiro ≥ 1)
- `years` (t, número real > 0)

RF3 O **backend** expõe um endpoint `POST /api/compound` que recebe JSON com os campos acima e responde JSON.

RF4 **Validação**: rejeitar inputs ausentes/negativos/NaN; retornar `HTTP 400` com JSON de erro.

RF5 **Internacionalização mínima**: aceitar ponto decimal; exibir no frontend moeda BRL formatada.

Requisitos Não Funcionais

RNF1 **Hospedagem gratuita**: aplicação *online* acessível publicamente (por exemplo: frontend em Vercel/Netlify; backend em Render/Railway/Fly.io).[1]

RNF2 **CORS habilitado** no backend para o domínio do frontend.

RNF3 **Segurança** básica: sem chaves no cliente; tratar erros sem expor *stack traces*.

RNF4 **Reprodutibilidade**: `README.md` com passos de `build/run`, versões e variáveis de ambiente.

[1] (*Use qualquer equivalente gratuito de sua preferência.*)

Especificação Técnica Sugerida

Backend (Haskell)

Sugestão com `stack` + `scotty` + `aeson`.

Frontend (JavaScript)

HTML/JS simples usando `fetch`.

Deploy (sugestão)

1. **Backend**: preparar `Dockerfile` ou `start command`; configurar porta (ex.: 8080) e CORS.
2. **Frontend**: `index.html` estático em Vercel/Netlify; apontar a variável `api` para a URL do backend.
3. **Teste público**: validar `POST` com `curl`.

Repositório GitHub — *Checklist*

- Nome sugestivo (ex.: `lp-recuperacao-js-hs-compound`).
- **README.md** com: descrição, arquitetura, endpoints, exemplos de request/response, instruções de build e deploy, e **links de produção** (frontend & backend).
- **Tag/Release**: criar a tag `v1.0-RECUP-LP-2025`.
- **Identificação da atividade**: arquivo `RECUP_LP_2025.md` contendo seu nome completo, turma, e os links de produção.
- Estrutura mínima: `/frontend`, `/backend`, `LICENSE`, `.gitignore`.

Entrega (obrigatória)

E1 Até **16/11/2025 às 23:59 (BRT)** enviar e-mail para `juliocesar@cotemig.com.br` com assunto:
`[RECUP LP 2025] Nome Completo – Turma`.

E2 No corpo do e-mail, incluir dois **links públicos**: (i) **aplicação funcionando** (URL do frontend) e (ii) **repositório GitHub**.

E3 Não anexar código ao e-mail; somente links.

Critérios de Avaliação (100% dos pontos recuperados)

- **API Haskell correta e validando entradas** (25%)
- **Frontend funcional e usável** (20%)

- **Comunicação via JSON e CORS** (15%)
- **Deploy público (uptime e acessibilidade)** (15%)
- **Documentação (README, exemplos, passos)** (15%)
- **Organização do repo (tag, arquivos, estrutura)** (10%)

Penalidades: links quebrados (−20%), falta de validação (- 10%), $ausnciadetag/arquivodeidentificao$(- 5%).

Boas Práticas Exigidas

- Nomes claros para variáveis, commits descritivos.
- Tratamento de erros com mensagens amigáveis no cliente.
- Formatação monetária (`pt-BR`, BRL) no frontend.
- Separação de camadas (sem lógica de cálculo no DOM).

Observações Finais

- Trabalho **individual**. Plágio ou cópia resultarão em nota zero e medidas acadêmicas.
- Você pode usar qualquer biblioteca/stack equivalente, desde que respeite os requisitos e publique os links.