Sumário

---

1. Atividade Prática – Frontend Frameworks

1.1. App React + Vite consumindo API Harry Potter (Vercel)

Disciplina: **Frontend Frameworks**

Professor: **João Ferreira**

1.2. Objetivo da atividade

Desenvolver uma aplicação **web** utilizando **React** com **Vite**, consumindo uma **API REST** pública de livros de Harry Potter, com foco em:

- Criação de projeto com **Vite** e React;
- Consumo de API REST utilizando **fetch** ou **axios**;
- Organização de componentes e rotas em uma SPA;
- Manipulação de estado e passagem de dados entre telas;
- Persistência local (**localStorage** ou solução baseada em **SQLite no navegador**);
- Publicação da aplicação no **Vercel**;
- Versionamento de código com **Git** e **GitHub**.

> ⚠ **Atenção**: O **não seguimento de todas as regras e requisitos listados abaixo irá inabilitar a correção do exercício.**

1.3. Ambiente de desenvolvimento obrigatório

Você deverá utilizar **obrigatoriamente** o seguinte ambiente de desenvolvimento:

- **Node.js**: versão **20.19.***
- **Vite**: versão **7.*** (template React)
- **Framework**: **React** (React + Vite)
- **IDE / Editor**: **Visual Studio Code (VSCode)**

Outras ferramentas auxiliares (como Git, extensões do VSCode, eslint, etc.) podem ser utilizadas livremente, desde que não violem as regras da atividade.

1.4. API a ser utilizada

A aplicação deverá consumir a API pública disponível em:

- Endpoint de livro aleatório (obrigatório): `https://potterapi-fedeperin.vercel.app/en/books/random`
- Documentação da API: `https://github.com/fedeperin/potterapi`

Você deverá consultar a documentação para entender o formato do JSON retornado e os campos disponíveis. Em especial, serão utilizados os seguintes parâmetros do objeto de resposta do livro:

- `cover` – URL da imagem de capa do livro
- `number` – número do livro na série
- `originalTitle` – título original do livro em inglês
- `releaseDate` – data de publicação
- `pages` – número de páginas
- `description` – descrição do livro

1.5. Estrutura da aplicação

1.5.1. Nome do projeto React

O nome da aplicação React **deve ser exatamente**:

```
react_vercel_app
```

> **!** O nome do projeto e do repositório é **obrigatório**. Não altere.

Sugestão de criação do projeto (exemplo):

```
npm create vite@latest react_vercel_app -- --template react
```

ou

```
pnpm create vite@latest react_vercel_app --template react
```

(Verifique a sintaxe de acordo com a versão do Vite e do gerenciador de pacotes.)

1.5.2. Telas obrigatórias

A aplicação deverá conter **3 telas/páginas principais**:

1. **Tela Inicial (Home)**
2. **Tela de Detalhes do Livro**
3. **Tela de Favoritos**

A navegação entre as telas deve ser feita utilizando alguma solução de roteamento para React, por exemplo **React Router**.

1.6. Especificação das telas

1.6.1. Tela 1 – Página Inicial (Home)

Ao abrir a aplicação, a **Página Inicial** deverá:

1. **Fazer uma requisição REST** ao endpoint: `https://potterapi-fedeperin.vercel.app/en/books/random`
2. Exibir na tela os seguintes elementos:
   - **Imagem de capa** do livro, utilizando o campo `cover` retornado pela API.
   - **Texto abaixo da capa** no seguinte formato: `Livro # - TITULO` Onde:
     - `#` é o valor de `number` retornado pela API;
     - `TITULO` é o conteúdo de `originalTitle` (título original em inglês).
3. **Comportamento ao clicar na imagem da capa**:
   - Ao clicar/tocar sobre a **imagem de capa**, o usuário deverá ser redirecionado para a **Tela de Detalhes do Livro** (Tela 2), passando todos os dados necessários (via rota com parâmetros, estado de navegação ou contexto).
4. **Atualização ao voltar da tela de detalhes**:
   - Sempre que o usuário voltar para a **Tela Inicial**, a aplicação deverá **buscar um novo livro aleatório** na API.

> Dica: utilize `useEffect` para disparar a requisição à API quando a tela inicial for carregada ou recarregada.

1.6.2. Tela 2 – Página de Detalhes do Livro

Ao acessar esta tela (vinda da Tela Inicial), deverão ser exibidos:

1. O texto **acima** da capa, no seguinte formato: `Livro # - TITULO` Utilizando os mesmos campos:
    - `number` → número do livro;
    - `originalTitle` → título original em inglês.
2. A **imagem de capa** do livro (campo `cover`) logo **abaixo** do texto.
3. Logo **abaixo da capa**, exibir os seguintes dados, cada um com seu rótulo em português:
    - `Data de publicação:` → valor de `releaseDate`
    - `Páginas:` → valor de `pages`
    - `Descrição:` → valor de `description`
4. A tela deverá conter **dois botões** claramente visíveis:
    - **Botão 1 – Voltar à Página Inicial**
        - Ao clicar, o usuário deverá ser levado de volta à **Tela Inicial**.
        - Ao retornar, a **Tela Inicial deve buscar um novo livro aleatório** na API, não reaproveitando os dados anteriores (ou seja, um novo livro deverá ser exibido).
    - **Botão 2 – Adicionar aos Favoritos**
        - Ao clicar, o livro exibido na tela deverá ser **adicionado à lista de favoritos**.
        - Após salvar, deverá ser exibido um **alerta** informando algo como:

            > `Livro adicionado aos favoritos!`

            Esse alerta pode ser implementado com:
            - `window.alert`, ou
            - um componente visual próprio (modal, toast, etc.).
    - A implementação da lista de favoritos deve utilizar **obrigatoriamente** uma das opções a seguir:
        - `localStorage`;
        - **ou** uma solução baseada em **SQLite** para uso no navegador (por exemplo, bibliotecas que persistem dados em IndexedDB/SQLite em ambiente web).

> A escolha entre `localStorage` ou uma solução baseada em SQLite é livre, mas **uma das duas deve ser implementada** corretamente.

1.6.3. Tela 3 – Página de Favoritos

A **Página de Favoritos** deverá:

1. Recuperar os livros que foram salvos anteriormente via **localStorage** ou solução baseada em **SQLite**;

2. Exibir os livros em uma **lista** (por exemplo, um `<ul>` com `<li>` ou um componente de lista estilizada);

3. Para cada livro na lista de favoritos, exibir pelo menos:

    - O **título do livro** (`originalTitle`)
    - Opcionalmente, podem ser exibidos também número do livro, data, ou uma miniatura da capa.

4. A página deve ser acessível a partir da navegação da aplicação, por exemplo:

    - Um link na Home (ex.: “Ver Favoritos”);
    - Um item em um menu de navegação ou header;
    - Um botão em alguma das telas.

1.7. Persistência dos dados (Favoritos)

Você deverá implementar o armazenamento dos livros favoritos utilizando um dos mecanismos abaixo:

- **Opção 1 – localStorage**

    - Salvar a lista de livros em uma chave específica (por exemplo, `favoritesBooks`);
    - Converter os dados para JSON (`JSON.stringify`) ao salvar;
    - Converter de volta para objeto/array (`JSON.parse`) ao ler;
    - Garantir que não haja erro caso não existam favoritos ainda (tratar `null`).

- **Opção 2 – Solução baseada em SQLite**

    - Utilizar alguma biblioteca que permita utilizar SQLite/IndexedDB no navegador;
    - Criar uma estrutura de tabela/coleção para armazenar os dados principais do livro;
    - Implementar as operações de inserção e leitura;
    - Garantir que os dados persistam entre recarregamentos da página.

> Independente da opção escolhida, os dados devem ser persistidos de forma que, ao recarregar a aplicação no navegador, a lista de favoritos continue disponível.

1.8. Publicação no Vercel

A aplicação deverá ser **buildada e publicada** no **Vercel**.

- A URL final deverá apontar para a aplicação funcionando, gerada a partir do build do Vite;
- A aplicação deve estar **acessível publicamente** (sem senha/login).

Você deverá pesquisar os passos necessários para:

1. Gerar o build de uma aplicação React + Vite para produção, normalmente com:

```
npm run build
```

2. Configurar o projeto para ser servido no Vercel, incluindo:

- pasta de saída (`dist`);
- configuração de build no painel do Vercel, se necessário;
- ajustes para roteamento em SPA, se aplicável (por exemplo, `redirects`/`rewrites`).

> A aplicação **deve estar funcional** no Vercel no momento da correção. Se o link não abrir ou retornar erro, o exercício será considerado **não entregue**.

1.9. Repositório GitHub

O código da aplicação deverá estar versionado em um repositório público no GitHub, obedecendo às seguintes regras obrigatórias:

1. **Nome do repositório**:

   ```
   react_vercel_app
   ```

2. **Visibilidade**: o repositório deve ser **PÚBLICO**.

3. **Licença**:

   - O repositório deverá conter uma licença do tipo **Creative Commons** (por exemplo, um arquivo `LICENSE` apropriado ou indicação clara no README).
   - Não utilizar licenças proprietárias ou incoerentes com o modelo Creative Commons.

4. **Arquivo README.md** O repositório deverá conter obrigatoriamente um arquivo `README.md` com, no mínimo:

   - Descrição do projeto (objetivo da aplicação);
   - Tecnologias utilizadas (React, Vite, Node, etc.);
   - Instruções básicas de instalação e execução em ambiente local (clonar, instalar dependências, rodar `npm run dev`, etc.);
   - **Pelo menos 3 *prints de tela*** (screenshots) da aplicação em funcionamento, cobrindo:
     - Tela Inicial com a capa do livro;
     - Tela de Detalhes;
     - Tela de Favoritos.

   As imagens podem estar na pasta `public/`, `src/assets/` ou em uma pasta própria (por exemplo, `docs/` ou `screenshots/`) e devem ser referenciadas no README.

1.10. Forma de entrega da atividade

A entrega será feita **exclusivamente** através do **Microsoft Teams**, na atividade correspondente.

Você deverá enviar **apenas dois links**:

1. **Link do repositório público no GitHub**

- Exemplo (apenas ilustrativo): `https://github.com/seu-usuario/react_vercel_app`

2. **Link da aplicação publicada no Vercel**
   - Exemplo (apenas ilustrativo): `https://react_vercel_app-seuusuario.vercel.app`

> ✖ **Não envie**: documentos, arquivos compactados (.zip, .rar), imagens, PDFs ou qualquer outro tipo de anexo. ✔ Somente os **dois links** (GitHub e Vercel) devem ser enviados no campo de resposta da atividade.

1.11. Regras de individualidade da atividade

Esta atividade é **estritamente pessoal**:

- Não deve ser feita em grupo;
- Não deve ser publicada em repositórios compartilhados de grupo;
- Não deve reutilizar código idêntico de colegas;
- Não deve ser desenvolvida colaborativamente.

Qualquer indício de cópia, clonagem direta de repositórios de colegas ou plágio poderá resultar em **anulação da atividade** para todos os envolvidos, conforme regras acadêmicas da instituição.

1.12. Critérios gerais de avaliação

Serão considerados na avaliação:

1. **Atendimento integral a todos os requisitos obrigatórios** desta especificação;
2. Funcionamento correto das **3 telas** e navegação entre elas;
3. Consumo correto da API e exibição dos dados;
4. Implementação válida e funcional da **lista de favoritos** com persistência local;
5. Publicação correta e funcional no **Vercel**;
6. Organização do código, estrutura de pastas e clareza;
7. Qualidade mínima da interface (layout organizado, textos legíveis, responsividade básica);
8. Presença de **README.md completo** com prints e instruções;
9. Presença de **licença Creative Commons** no repositório.

> ⚠ **Importante**: Caso qualquer uma das regras essenciais (nome do repositório, publicação no Vercel, uso da API correta, 3 telas, forma de entrega, etc.) não seja seguida, a atividade poderá **não ser corrigida** e receber **nota zero**.

1.13. Recomendações finais

- Faça commits frequentes no GitHub com mensagens claras;
- Teste a aplicação em ambiente de desenvolvimento (`npm run dev`) e em build de produção (`npm run build` + Vercel);
- Teste a navegação completa:
  - Tela Inicial → Tela de Detalhes → Favoritar → Voltar → Novo livro;

    - Acesso à Tela de Favoritos e exibição da lista persistida;
- Verifique cuidadosamente os links antes de enviar no Teams.

> Em caso de dúvidas conceituais (React, Vite, consumo de API, rotas, estado, etc.), revisite o material da disciplina e os exemplos vistos em aula.

Bom trabalho!
Prof. João Ferreira