Aplicações orientadas a hipermídia como solução para a Web moderna

Aluno: Fernando Henrique Junqueira Muniz Barbi Silva
Supervisor: Prof. Dr. Paulo Roberto Miranda Meirelles

INTRODUÇÃO

Atualmente, o desenvolvimento de software para a Web é caracterizado pela adoção das bibliotecas de *Single-Page Application* (SPA) como uma solução universal para a construção da interface do usuário.

A maioria da Web é estática, e não necessita do alto nível de abstração inerente à essas bibliotecas. **O modelo SPA implica em restrições na arquitetura do projeto que não são necessárias para a maioria dos sistemas**, e, portanto, não se beneficiam deste estilo.

Propomos fazer uma **análise crítica do modelo SPA** e a oferecer as aplicações orientadas à hipermídia (HDA) como alternativa viável para interfaces de usuário modernas.

REST

Cliente-servidor: os requerentes dos serviços são separados do fornecedor de um recurso ou serviço através de uma interface bem definida.

Ausência de estado: cada requisição feita por um cliente específico deve ser independente e conter todas as informações necessárias para ser inteiramente processada.

Interface uniforme: é necessário uma interface padronizada e consistente entre o cliente e o servidor que respeite as sub-restrições:

> **Identificação de recursos:** identificadores de recursos unívocos.
>
> **Manipulação de recursos por meio de representações:** o cliente manipula os recursos através do envio e recebimento de representações desses recursos.
>
> **Mensagens autodescritivas:** toda informação necessária para exibir e realizar operações em cima dos recursos devem estar presentes na resposta.
>
> **Hipermídia como motor do estado da aplicação (HATEOAS):** o cliente deve receber o estado do servidor em hipermídia pois não deve necessitar de saber previamente a estrutura do servidor ou quais recursos ele pode acessar.

Sistema em camadas: a arquitetura deve ser restringida em camadas hierárquicas, de modo que um componente não possa ver além da camada imediata com a qual está interagindo.

Código sob demanda (opcional): pode permitir que a funcionalidade do cliente seja estendida através do download e execução de código sob demanda, na forma de applets ou scripts.

CALAMIDADES DO SINGLE-PAGE APPLICATION

- A API JSON não é uma API de hipermídia, mas uma API de dados somente. Portanto, **não é RESTful** pois não respeita a restrição de interface uniforme.
- É necessário **duplicar e sincronizar o estado do servidor no cliente**.
- Há um **acoplamento forte entre cliente e servidor** pois, se o formato da resposta da API mudar, o cliente também deve mudar.
- O **processo de inicialização é inchado** devido ao processo de hidratação, e torna-se mais lento conforme o tamanho total da aplicação aumenta.
- É necessário um **processo de compilação** para a inicialização ser viável.
- **Alto nível de abstração** para ocultar a complexidade do funcionamento interno.
- A **curva de aprendizado é muito maior** para novos programadores;
- Geram enormes **cadeias de dependências.**
- Adotar **JavaScript como linguagem de programação do servidor** é necessário para evitar duplicação de código;
- JavaScript é **tratado como uma linguagem de propósito geral**, e não como *scripting*;
- Permite **sintaxe inválida** de HTML, por exemplo:
;
- O HTML é **tratado como uma linguagem de descrição visual**, e não como uma hipermídia;
- O navegador é **tratado como um ambiente de execução de JavaScript**, e não como um cliente de hipermídia;

BIBLIOGRAFIA

Roy Thomas Fielding. *"Architectural Styles and the Design of Network-based Software Architectures".*
Tese de dout. Irvine: University of California, 2000.
url: https://ics.uci.edu/~fielding/pubs/dissertation/top.htm (acesso em 06/04/2024)

Carson Gross, Adam Stepinski e Deniz Akşimşek. *Hypermedia Systems.*
Publicação independente, jul. de 2023.
url: https://hypermedia.systems/book/contents/ (acesso em 25/10/2024)

Roy Thomas Fielding. *REST APIs must be hypertext-driven.*
2008. url: https://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven (acesso em 13/11/2024)

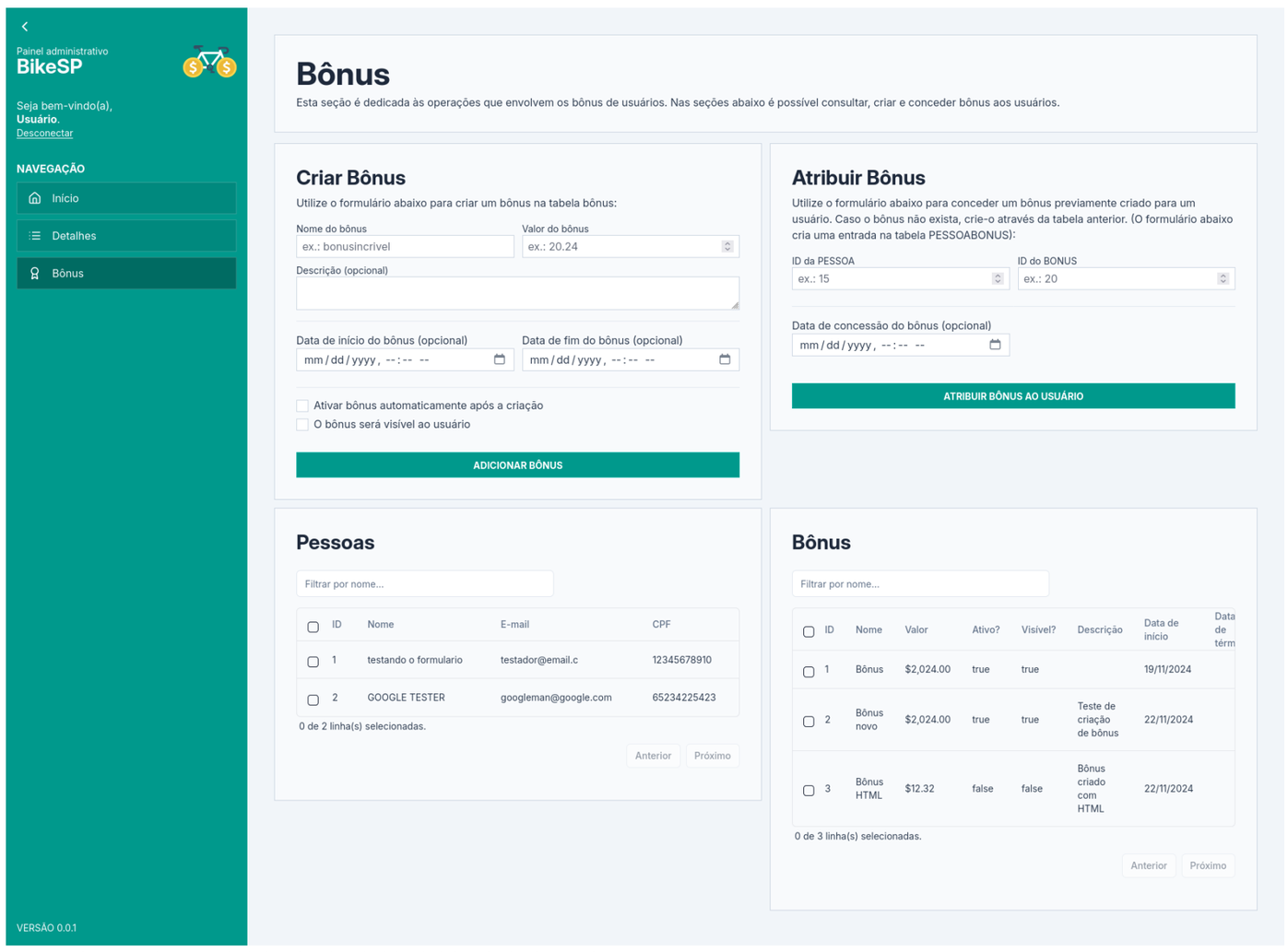

Figura 1: *Tela capturada da interface do painel de administrador implementado em React.*

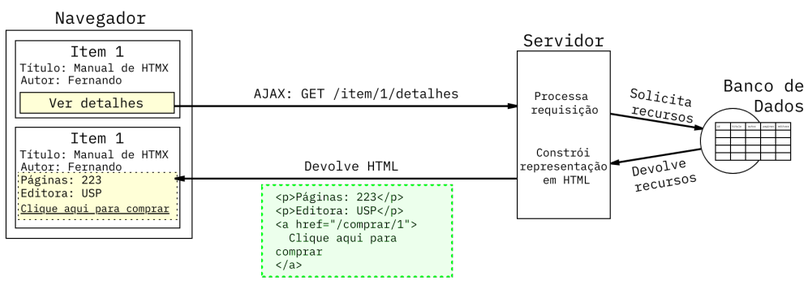

Figura 2a: *Diagrama do fluxo de uma requisição em HDA.*

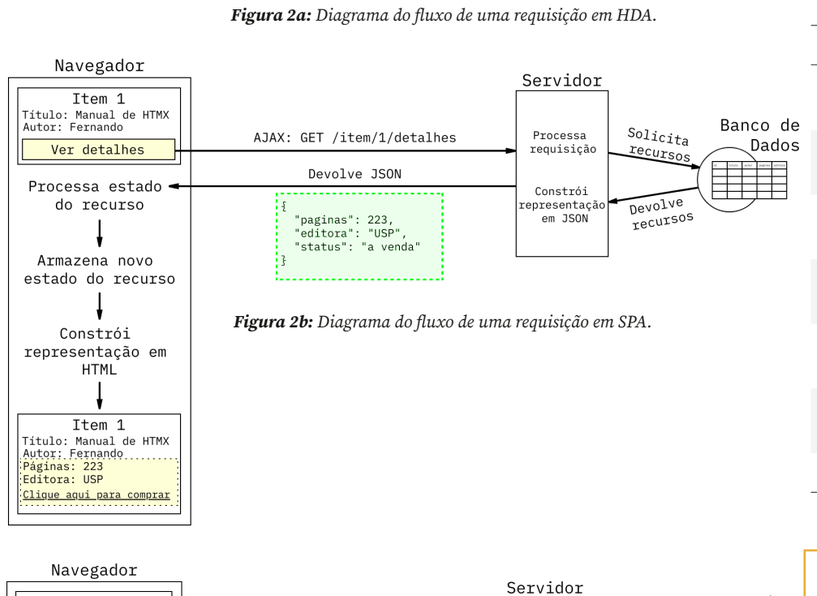

Figura 2b: *Diagrama do fluxo de uma requisição em SPA.*

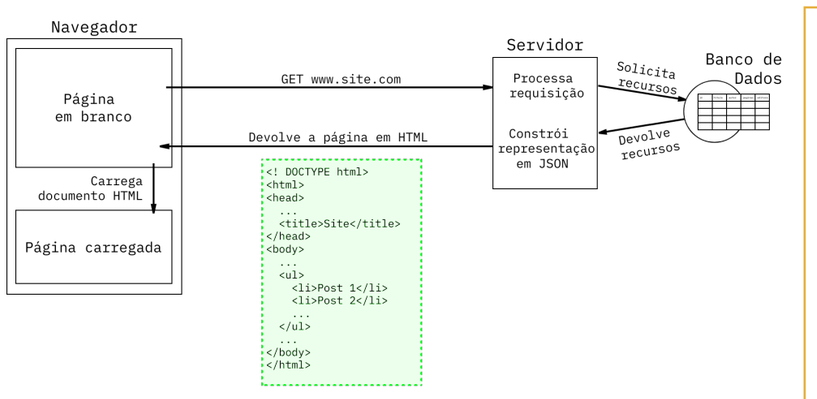

Figura 3a: *Diagrama do processo de inicialização do sistema em HDA.*

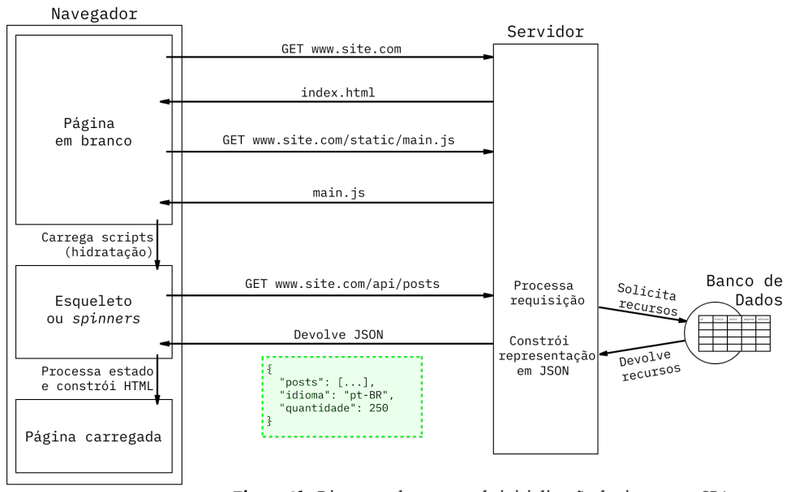

Figura 3b: *Diagrama do processo de inicialização do sistema em SPA.*

HDA (htmx)

Linguagem	Linhas de código
HTML	708
CSS	141

Tabela 1b: *linhas de código por linguagem em HDA.*

SPA (React)

Linguagem	Linhas de código
TypeScript	1262
JSON	131
JavaScript	57
CSS	13
HTML	13
SVG	1

Tabela 1a: *linhas de código por linguagem em SPA.*

	SPA (React)	HDA (htmx)
Número de dependências adicionadas ao projeto	33	3
Total de linhas de código do cliente (ignorando dependências)	1477	849
Total de linhas de código alteradas no servidor	113	304
Total de novos arquivos criados (excluindo dependências)	39	17
Total de dados carregados na inicialização no navegador	3,52MB	86,76kB
Total de scripts carregados na inicialização no navegador	3,43MB	53,67kB
Tempo médio de compilação	8,12s	0 (não há)

Tabela 2: *Algumas métricas de comparação entre as duas implementações.*

PROVA DE CONCEITO

Foram realizadas **duas implementações distintas** para um painel de administrador do projeto BikeSP — uma em SPA, usando um ecossistema popular em React, e outra em HDA, usando HTML estendido por htmx — para servir como estudo de caso da adoção do modelo HDA, com ênfase nos aspectos relacionados à experiência de desenvolvimento.

Houve uma **redução** muito relevante no número de dependências, de linguagens utilizadas, de linhas de código, quantidade de arquivos e, principalmente, no **volume de dados transacionados** entre cliente e servidor (tabelas 1a, 1b e 2).

A **inicialização do cliente foi simplificada**, de três requisições (figura 3b) para apenas uma (figura 3a).

Não há mais um processo complexo de compilação e substituição de todos os arquivos do servidor a cada *deploy*, mas **apenas a troca dos arquivos efetivamente alterados**.

O servidor em HDA constrói a representação HTML, cabe ao cliente apenas exibí-la (figura 2a). **Não foi preciso implementar nenhuma lógica do lado do cliente** para processar a resposta (figura 2b).