Relatório Técnico: Integração Vue.js e Django do Projeto Metflix

Integrantes:

- Bárbara Ferreira Rodrigues
- Bernardo Rasteldi Angelo
- Otávio Henrique Gonçalves Ribeiro

Objetivo

Este relatório tem como objetivo apresentar as diferenças técnicas entre o uso dos **templates tradicionais do Django** e a nova abordagem adotada com **Vue.js** no projeto Metflix, destacando o que foi feito, suas vantagens, limitações e o impacto na estrutura do sistema.

Abordagem Utilizada

- O sistema utiliza **Django como backend principal**, com views tradicionais (`render(request, ...)`) e rotas do `urls.py`.
- A parte visual foi modernizada com o uso de **Vue.js 3** e **Vite** para empacotamento dos arquivos.
- **Não foi utilizada nenhuma API ou comunicação via fetch/axios**. O Vue foi incorporado diretamente ao HTML.
- Os arquivos gerados pelo build do Vite (`index.html`, `.js`, `.css`) são colocados dentro de `static/vue` e carregados nos templates Django via `{% static %}`.

Comparativo Técnico

Item	Templates Django Tradicionais	Vue.js Integrado ao Django
Renderização	HTML montado no backend	HTML básico com Vue montado no frontend
Interatividade	Limitada devido a distribuição do Django	Dinâmica com reatividade do Vue
Reutilização de componentes	Baixa	Modularização por componentes Vue
Estilo e organização visual	Inline ou estático nos arquivos `.html`	Separação por arquivos `.vue` no projeto Vue
Atualização de layout	Requer edição direta nos templates	A maior parte é mantida no projeto Vue
Carga dos dados	Passados via contexto nas views	Também via contexto

---

Vantagens Observadas

- **Melhoria estrutural**: a interface ficou mais moderna.
- **Componentização**: a estrutura Vue permitiu separar melhor a lógica visual.
- **Facilidade de estilização**: com uso de CSS moderno e escopo específico.

- **Manutenção centralizada do front-end**: concentrando o comportamento visual no projeto Vue.

Limitações Encontradas

- É necessário rodar `npm run build` a cada alteração para refletir no Django.
- Como o Vue não está desacoplado (sem API), seu uso é mais visual do que funcional.
- A estrutura de rotas e carregamento de dados continua sendo dependente do Django, apesar de já utilizar Vue Router.

Oportunidades Futuras com Vue.js

Apesar de estar sendo usado apenas para modernizar a camada visual, o Vue.js oferece diversas possibilidades que ainda podem ser exploradas no projeto:

- **Transformar o sistema em uma SPA (Single Page Application)**
  Com o uso completo do Vue Router, é possível deixar a navegação mais fluida, sem recarregamento de páginas.
- **Integração com Django REST Framework**
  Separando o backend em uma API e utilizando chamadas HTTP no Vue (via `axios` ou `fetch`), o sistema ganha em desacoplamento, escalabilidade e flexibilidade.
- **Autenticação via token (JWT ou Session)**
  Ao usar Django REST + Vue, é possível implementar autenticação mais moderna e segura, facilitando o consumo em múltiplas plataformas.

Essas melhorias são compatíveis com o que já foi implementado e podem ser introduzidas gradualmente.

Diferenças no Código

- Template (html + tags do Django):

Temos uma estrutura HTML sendo dinamicamente montada utilizando as tags disponibilizadas pelo Django.

- View (html + vuejs):

DashboardView.vue M

metflix-project > vue_frontend > src > views > DashboardView.vue > {} style scoped > body

You, 5 seconds ago | 1 author (You)

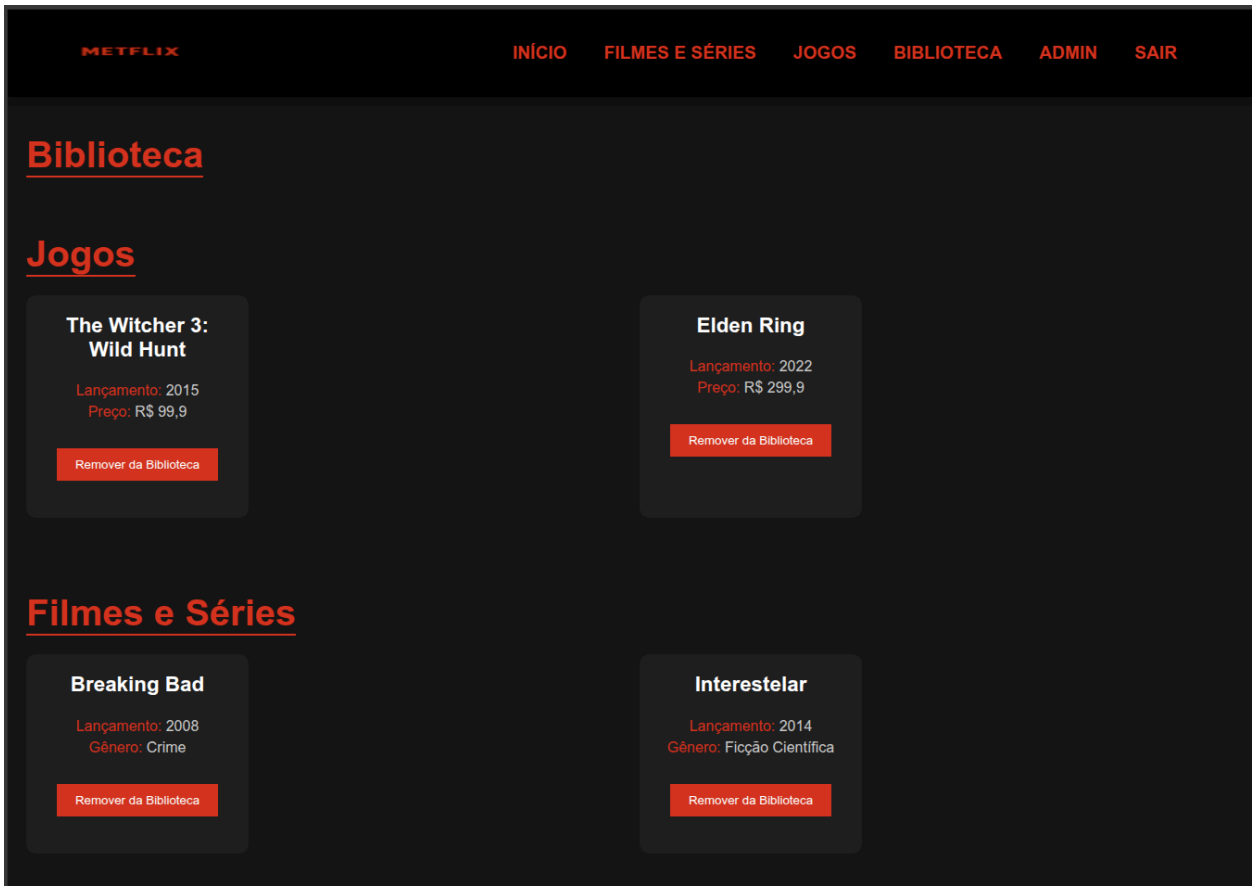
```vue
<template>
  <div class="center-container">
    <h2 class="section-title">Bem-vindo ao Metflix</h2>

    <form @submit.prevent="buscar">
      <input
        type="text"
        v-model="termoBusca"
        placeholder="Buscar por título..."
      />
      <button type="submit">Buscar</button>
    </form>

    <div class="card-container">
      <div v-if="resultados.length > 0">
        <div class="card" v-for="(item, index) in resultados" :key="index">
          <strong>{{ item.nome }}</strong>
          <span>
            <template v-if="item.preco !== null">
              Jogo - R$ {{ item.preco }}
            </template>
            <template v-else>
              {{ item.categoria }}
            </template>
          </span>
        </div>
      </div>
      <p v-else>Nenhum resultado encontrado.</p>
    </div>
  </div>
</template>

<script>
export default {
  name: "DashboardView",
  data() {
    return {
      termoBusca: "",
      resultados: [],
    };
  },
  methods: {
    buscar() {
      this.resultados = [
        { nome: "The Witcher 3", preco: 99.9 },
        { nome: "Breaking Bad", categoria: "Série" },
      ].filter((item) =>
        item.nome.toLowerCase().includes(this.termoBusca.toLowerCase())
      );
    },
  },
};
</script>
```

No VueJs a principal diferença é que temos a construção de um componente reutilizável

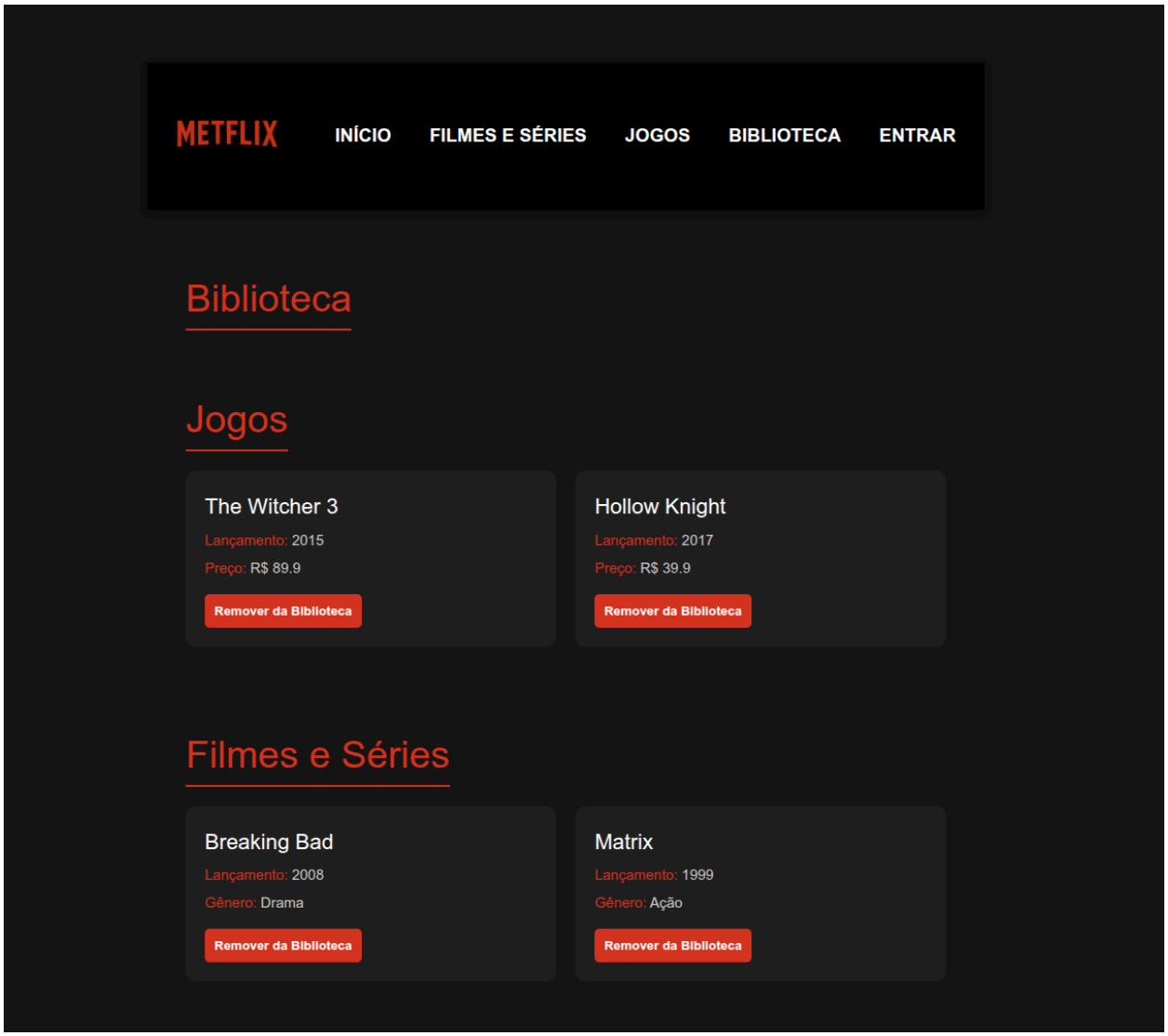
---

Diferenças na Interface:

- **Templates do Django**: /biblioteca



- **VueJs**: /app/biblioteca



Considerações Finais

A integração do Vue.js no projeto Metflix trouxe **ganhos práticos de organização e apresentação**, sem alterar o funcionamento tradicional das views do Django. Essa escolha permitiu evoluir a interface **sem reestruturar todo o backend**.

Mesmo sem uso de API, foi possível obter uma interface moderna, com um fluxo de desenvolvimento mais atual e modular. Com o que já foi construído, o projeto está bem posicionado para evoluir em direção a uma arquitetura SPA com backend em Django REST, caso isso se torne necessário no futuro.