



Relatório de Tecnologia para Desenvolvimento de Sistemas

Título: AP04 – Implementação de Aplicação Web com Node.js e Vue.js.

Aluno: Gustavo Pauli da Luz

Data: 26/07/2023

1. INTRODUÇÃO

Atualmente, o desenvolvimento de aplicações tem se concentrado em arquiteturas escaláveis, entrega de soluções em tempo real e ênfase na componentização e segurança. Essa evolução é impulsionada pela revolução iniciada pelos smartphones, que trouxeram um crescente uso de mídias sociais e um aumento significativo de soluções de IoT (Internet das Coisas). Nesse cenário, os paradigmas tradicionais de desenvolvimento de aplicações têm passado por diversas mudanças, com uma abordagem mais holística, considerando o consumo de dados e a disponibilidade de infraestrutura (BESSA, 2023).

Uma tecnologia que tem contribuído para o desenvolvimento de aplicações no lado do servidor é o Node.js. Trata-se de um ambiente de execução do código JavaScript, permitindo criar aplicações autossuficientes em uma máquina servidora, dispensando a necessidade do navegador (BESSA, 2023).

Outro destaque no desenvolvimento de aplicações é o Vue.js, um framework JavaScript de código aberto conhecido por sua reatividade. Amplamente utilizado na construção de Single Page Applications e interfaces de usuário, o Vue.js se destaca pela facilidade de desenvolvimento ágil e pelo uso de componentes reutilizáveis. Sua arquitetura baseada em componente independentes, compostos por HTML, CSS e JavaScript, oferece liberdade em relação ao estilo e lógica de implementação. Além disso, o Vue.js adota uma renderização eficiente por meio de DOM, o que contribui para o desempenho do framework (PINHEIRO, 2021).

2. OBJETIVOS

Este presente trabalho terá então como objetivo a implementação de uma aplicação web com Node.js e Vue.js. Para isto será utilizado as linguagens de programação HTML, CSS e JavaScript.

No presente trabalho será desenvolvido uma implementação de lista de afazeres, do inglês. Todo list, nela será possível cadastrar uma nova tarefa, visualizar as já cadastradas, além de editá-las e excluí-las.

3. MATERIAL UTILIZADO

- Linguagens: HTML, CSS e JavaScript
- Frameworks: Bootstrap, Node.js e Vue.js
- Ferramentas: NPM, VueCLI e Webpack

4. PREPARAÇÃO DO AMBIENTE

Primeiramente para fazermos a instalação do ambiente de desenvolvimento com vue.js, precisamos fazer a instalação do Node.js e do NPM com o comando “sudo apt install nodejs npm”. Depois disso podemos fazer a instalação do Vue.js utilizando o comando “sudo npm install -g @vue/cli”.

- Versão instalada Vue: @vue/cli 5.0.8
- Versão instalada Node: v20.4.0
- Versão instalada npm: 8.5.1

Para o desenvolvimento do projeto, foi decidido a escolha da IDE Vscode, utilizando os plugins/extensões de indentação fornecidos pela biblioteca da própria IDE do Vscode (Vue Language Features (Volar) v1.8.5). Essa extensão pode ser encontrada utilizando o atalho CTRL + Shift + X e na barra de pesquisa, buscar por Vue Language Features (Volar) e em seguida instalar a extensão.

Em seguida foi necessário criar um repositório git pelo próprio site do GitHub, e em seguida foi realizada a autorização do vscode utilizar a conta do GitHub, para que dessa forma fique mais fácil a utilização do Git com o Vscode.

5. CRIANDO O PROJETO

Após a preparação do ambiente de desenvolvimento, foi preciso criar o nosso projeto com o comando “vue create .”, esse comando faz a criação de todos os arquivos necessários para iniciarmos o projeto.

Para as configurações de criação foi selecionado para criar o projeto na pasta atual. Em seguida foi escolhido as features de forma manual e foi adicionado a feature Router. A versão do Vue.js escolhida foi 3.x. Foi escolhido utilizar o History mode para o Router que é uma opção de configuração que permite criar URLs amigáveis e remover o símbolo de cerquilha (#) da URL. Quando o History Mode é ativado, a navegação entre páginas da aplicação não exige o uso de fragmentos (hashes) na URL, proporcionando uma experiência de usuário mais limpa e semelhante à navegação tradicional em sites. Também foi definido que cada arquivo terá um arquivo de configuração individual.

Após a criação do projeto ser finalizada, foi utilizado o comando “npm run serve” para rodar o host em uma porta para poder rodar o projeto inicial no navegador.

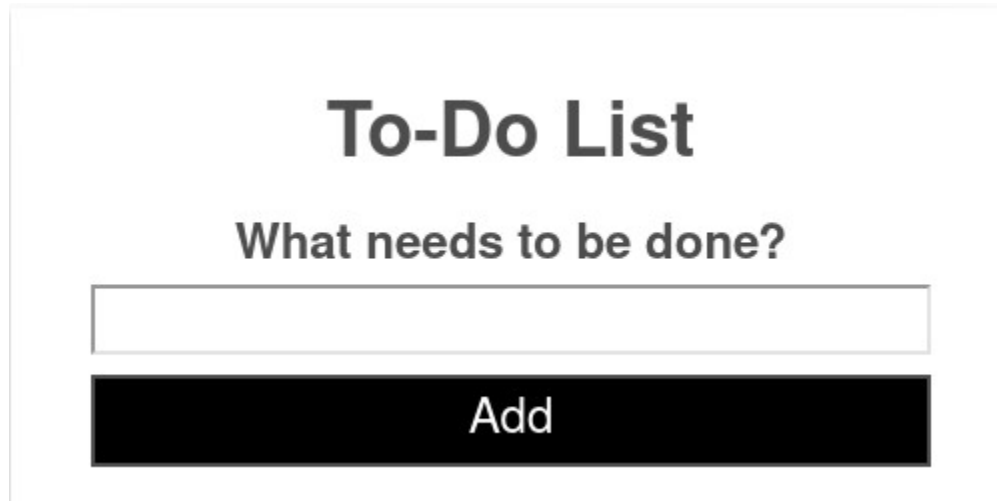
6. DESENVOLVIMENTO

Para essa to-do list foi necessário criar apenas quatro arquivos: ToDoForm.vue, ToDoItem.vue, ToDoItemEditForm.vue e App.vue. Dentro de cada arquivo foram feitas as edições de style necessárias para cada um.

6.1 ToDoForm

```
<template>
  <form @submit.prevent="onSubmit">
    <h2 class="label-wrapper">
      <label for="new-todo-input" class="label_lg">
        What needs to be done?
      </label>
    </h2>
    <input
      type="text"
      id="new-todo-input"
      name="new-todo"
      autocomplete="off"
      v-model.lazy.trim="label"
      class="input_lg" />
    <button type="submit" class="btn btn_primary btn_lg">Add</button>
  </form>
</template>
```

Primeiro criamos o template do form com o texto inicial, a caixa para entrada de texto e o botão para criar uma nova tarefa.



Aqui também temos um script que verifica se o campo label está preenchido, quando o formulário é enviado, caso esteja ele emite um evento chamado “todo-added” com o valor do campo label como parâmetro e em seguida limpa o campo para a entrada do próximo valor.

```
<script>
export default {
  methods: {
    onSubmit() {
      if (this.label === "") {
        return;
      }
      this.$emit("todo-added", this.label);
      this.label = "";
    },
  },
  data() {
    return {
      label: "",
    };
  },
};
</script>
```

6.2 ToDoItem

Neste método, é feita a formatação das entradas da lista de afazeres, é criado um template que cria uma check-box e as opções de editar e excluir cada entrada da lista. Aqui também é feita todas as decisões de edição de cada item, ou seja, caso o botão de edição seja clicado, ele muda a entrada para o formato de edição, verifica se foi editado ou não, caso tenha sido editado muda o dado da entrada e volta para seu estado normal. Aqui ele também chama o

evento "item-deleted" caso o botão delete seja ativado.

aaaaaaaaa

EditDelete

6.3 ToDoItemEditForm

Este código de template representa um formulário de edição de nome, onde o usuário pode editar o valor de um campo de entrada de texto e, em seguida, salvar ou cancelar as alterações. O formulário chama os métodos "onSubmit" e "onCancel" quando os botões de envio e cancelamento são clicados, respectivamente, para executar as ações apropriadas.

```
<template>
  <form class="stack-small" @submit.prevent="onSubmit">
    <div>
      <label class="edit-label">Edit Name for &quot;{{ label }}&quot;</label>
      <input
        :id="id"
        ref="labelInput"
        type="text"
        autocomplete="off"
        v-model.lazy.trim="newLabel" />
    </div>
    <div class="btn-group">
      <button type="button" class="btn" @click="onCancel">
        Cancel
        <span class="visually-hidden">editing {{ label }}</span>
      </button>
      <button type="submit" class="btn btn__primary">
        Save
        <span class="visually-hidden">edit for {{ label }}</span>
      </button>
    </div>
  </form>
</template>
```

Temos também outro componente Vue que é usado para editar um rótulo de um item específico. Ele recebe o rótulo original e um ID como propriedades e permite que o usuário edite o rótulo em um campo de entrada de texto. Quando o formulário é enviado (submit), o componente emite um evento "item-edited" contendo o novo valor do rótulo, desde que tenha sido editado e seja diferente do valor original. Além disso, há um botão de cancelamento que emite um evento "edit-cancelled" se o usuário decidir cancelar a edição.

6.4 App

E por fim, temos o App.vue, este código representa a lista de tarefas (to-do list) com a capacidade de adicionar novas tarefas e interagir com tarefas existentes (marcar como concluídas, editar e excluir). Ele utiliza componentes personalizados "to-do-form" e "to-do-item"

para lidar com as operações relacionadas à lista de tarefas. Ou seja, esse arquivo é onde todos os outros são relacionados.

```
<template>
  <div id="app">
    <h1>To-Do List</h1>
    <to-do-form @todo-added="addToDo"></to-do-form>
    <h2 id="list-summary" ref="listSummary" tabindex="-1">{{ listSummary }}</h2>
    <ul aria-labelledby="list-summary" class="stack-large">
      <li v-for="item in ToDoItems" :key="item.id">
        <to-do-item
          :label="item.label"
          :done="item.done"
          :id="item.id"
          @checkbox-changed="updateDoneStatus(item.id)"
          @item-deleted="deleteToDo(item.id)"
          @item-edited="editToDo(item.id, $event)">
        </to-do-item>
      </li>
    </ul>
  </div>
</template>
```

7. EXECUÇÃO DO PROGRAMA

Para executar a aplicação web, no sistema Linux, primeiramente, foi baixado a versão final do projeto, localizado no link: (<https://github.com/GusPauli/to-do-list>). Em seguida, foi aberto um terminal dentro da pasta do projeto e inserido o comando “npm install” para fazer a instalação das dependências e logo depois o comando “npm run serve”. Com isso, como todas as dependências citadas no tópico “Preparação do ambiente” foram devidamente instaladas, a aplicação está devidamente no ar e rodando.

8. CONCLUSÃO

Durante todo este relatório, pudemos acompanhar o processo de desenvolvimento de uma aplicação web utilizando o framework Vue.js. Desde a configuração inicial até a criação de interfaces interativas e responsivas, ficou claro que o Vue.js é uma ferramenta poderosa e flexível para construir aplicações modernas e ricas em recursos.

O Vue Router possibilitou a implementação da navegação de página em uma aplicação single-page (SPA), o que resultou em transições suaves entre as rotas, sem a necessidade de recarregamento completo da página. Isso contribuiu significativamente para aprimorar a experiência do usuário, tornando a aplicação mais ágil e responsiva.

Ao longo do desenvolvimento, o Vue.js demonstrou ser uma escolha acertada, proporcionando uma curva de aprendizado suave, facilidade de uso e uma gama de recursos poderosos para a construção da aplicação web. Além disso, a comunidade ativa e a vasta documentação do Vue.js foram valiosas aliadas, tornando o processo de desenvolvimento mais tranquilo e produtivo.

No geral, a jornada de desenvolvimento com o Vue.js foi gratificante, permitindo-nos criar uma aplicação web robusta e moderna. Com sua abordagem intuitiva e suas capacidades

impressionantes, o Vue.js se destaca como uma excelente escolha para projetos web de diferentes complexidades, garantindo um produto final de qualidade e uma experiência do usuário satisfatória.

BIBLIOGRAFIA

NODE.JS: o que é, como funciona esse ambiente de execução JavaScript e um Guia para iniciar. [S. l.], 20 jun 2023.

Disponível em: <https://www.alura.com.br/artigos/node-js>. Acesso em: 25 jul. 2023.

VUE.JS: tudo sobre o framework para trabalhar com mídias interativas. [S. l.], 22 jul. 2021. Disponível em: <https://rockcontent.com/br/talent-blog/vue-js/>. Acesso em: 25 jul. 2023.