UniSENAI

Trilha 01

Introdução ao Desenvolvimento com Vue.js

1. Introdução ao Desenvolvimento com Vue.js e Tecnologias Relacionadas

O desenvolvimento de aplicações modernas exige um conjunto de ferramentas que possam atender tanto à experiência do usuário no front-end quanto à robustez necessária no back-end. **Vue.js**, **Node.js**, **PHP** e **RESTful APIs** são algumas das tecnologias mais relevantes para criar aplicações completas e escaláveis, desde projetos simples até soluções empresariais.

Por que aprender Vue.js com Node.js, PHP e RESTful APIs?

O desenvolvimento full-stack permite ao desenvolvedor criar aplicações completas, onde o front-end e o back-end estão integrados de maneira eficiente.

- **Vue.js** é ideal para criar interfaces de usuário interativas e reativas.
- **Node.js** oferece alta performance para a criação de APIs e servidores.
- **PHP** continua sendo uma escolha sólida para sistemas tradicionais e gerenciadores de conteúdo.
- **RESTful APIs** conectam os dois mundos, garantindo uma comunicação fluida entre front-end e back-end.

1.1 Tecnologias Empregadas

1.1.1 Vue.js

Vue.js é um **framework progressivo de JavaScript**, projetado para ser simples de usar, mas poderoso o suficiente para atender às necessidades de aplicações complexas.

Por que escolher Vue.js?

- **Curva de aprendizado fácil**: Ideal tanto para iniciantes quanto para desenvolvedores experientes.
- **Versatilidade**: Pode ser usado para adicionar interatividade em pequenos projetos ou para criar aplicações de página única (SPAs).
- **Reatividade**: Possui um sistema de binding (ligação) entre a interface e os dados do aplicativo, atualizando automaticamente a tela sempre que os dados mudam.
- **Comunidade forte**: Possui uma vasta quantidade de plugins e bibliotecas.

Exemplo prático: Um sistema de controle de notas:

```
<div id="app">
  <h2>Controle de Notas</h2>
  <ul>
    <li v-for="nota in notas" :key="nota">{{ nota }}</li>
  </ul>
  <input v-model="novaNota" placeholder="Adicionar nota">
  <button @click="adicionarNota">Adicionar</button>
</div>

<script>
  const app = new Vue({
    el: '#app',
    data: {
      notas: ['Nota 1', 'Nota 2'],
      novaNota: ''
    },
    methods: {
      adicionarNota() {
        if (this.novaNota) {
          this.notas.push(this.novaNota);
          this.novaNota = '';
        }
      }
    }
  });
</script>
```

1.1.2 Node.js

Node.js é um **runtime de JavaScript para servidores**, que permite executar código JavaScript fora do navegador.
Ele foi projetado para construir aplicações escaláveis e de alta performance. Node.js se destaca pela sua arquitetura baseada em eventos e modelo assíncrono, tornando-o uma escolha ideal para APIs rápidas.

Usos comuns do Node.js:

- Construção de APIs RESTful.
- Desenvolvimento de servidores para aplicações em tempo real (ex.: chats).
- Processamento de grandes volumes de dados.

Destaques do Node.js:

- **NPM (Node Package Manager)**: Gerenciador de pacotes que oferece milhares de bibliotecas prontas para uso.
- **Ecossistema robusto**: Frameworks como Express.js tornam o desenvolvimento de APIs simples e eficiente.

1.1.3 PHP

PHP é uma **linguagem de programação back-end amplamente utilizada**. Embora seja mais antigo que Node.js, o PHP ainda é essencial para muitos projetos, especialmente devido à sua integração com bancos de dados e suporte em servidores como Apache.

Quando usar PHP?

- Em sistemas corporativos legados.
- Na construção de sistemas que utilizam CMS (ex.: WordPress, Joomla).
- Em projetos de pequeno e médio porte com menor exigência de performance.

Vantagens do PHP:

- Suporte nativo a bancos de dados como MySQL.
- Compatibilidade com servidores web amplamente usados (ex.: Apache e Nginx).
- Ferramentas modernas como Laravel, Slim e Symfony aumentam a produtividade.

1.1.4 APIs RESTful

Uma **API RESTful** é uma interface que permite a comunicação entre diferentes sistemas. Ela segue os princípios arquiteturais do REST (Representational State Transfer) e utiliza HTTP como protocolo de transporte.

Características principais das APIs RESTful:

- São baseadas em recursos (ex.: `/usuarios`, `/produtos`).
- Utilizam métodos HTTP como:
  - **GET**: Para buscar dados.
  - **POST**: Para criar novos registros.
  - **PUT/PATCH**: Para atualizar registros existentes.
  - **DELETE**: Para remover registros.
- Retornam respostas no formato **JSON**, o que facilita o consumo pelo front-end.

Exemplo prático de uma API RESTful:

```
const express = require('express');
const app = express();
app.use(express.json());

let usuarios = [{ id: 1, nome: 'João' }, { id: 2, nome: 'Maria' }];

// Rota GET
app.get('/usuarios', (req, res) => {
  res.json(usuarios);
});

// Rota POST
app.post('/usuarios', (req, res) => {
  const novoUsuario = req.body;
  usuarios.push(novoUsuario);
  res.status(201).json(novoUsuario);
});

app.listen(3000, () => console.log('API rodando na porta 3000'));
```

Consumo dessa API com Vue.js usando Axios:

```
export default {
  data() {
    return {
      usuarios: [],
      novoUsuario: { id: '', nome: '' }
    };
  },
  methods: {
    async carregarUsuarios() {
      const resposta = await axios.get('http://localhost:3000/usuarios');
      this.usuarios = resposta.data;
    },
    async adicionarUsuario() {
      await axios.post('http://localhost:3000/usuarios', this.novoUsuario);
      this.carregarUsuarios(); // Atualiza a lista
    }
  },
  mounted() {
    this.carregarUsuarios();
  }
};
```

1.2 Instalação do Node.js e Vue CLI

Passos para instalação do Node.js

1. Acesse https://nodejs.org.
2. Baixe a versão LTS (indicado para estabilidade).
3. Após instalar, verifique:

```
node -v
npm -v
```

Passos para instalar o Vue CLI

1. Após o Node.js estar configurado, execute:

```
npm install -g @vue/cli
```

2. Crie um projeto:

```
vue create meu-projeto
```

3. Inicie o servidor de desenvolvimento:

```
cd meu-projeto
npm run serve
```

1.3 Uso do npm e Yarn

npm e **yarn** são gerenciadores de pacotes para o Node.js. Eles permitem instalar bibliotecas e dependências de forma fácil.

- **Instalar pacotes locais**:

```
npm install axios
```

- **ou com Yarn:**

```
yarn add axios
```

- **Instalar pacotes globais**:

```
npm install -g @vue/cli
```

1.4 Conceitos Básicos de APIs RESTful

O que são APIs RESTful?

APIs (Application Programming Interfaces) são interfaces que permitem a comunicação entre sistemas, seja entre um front-end e um back-end, ou entre dois serviços distintos. APIs RESTful seguem os princípios arquiteturais do REST (**Representational State Transfer**), um padrão amplamente adotado por sua simplicidade, eficiência e escalabilidade.

Uma API RESTful utiliza o protocolo HTTP para manipular recursos (dados), representando-os por meio de URLs. Cada recurso pode ser acessado e modificado usando os métodos HTTP apropriados, como **GET**, **POST**, **PUT** e **DELETE**.

Por que usar APIs RESTful?

1. Integração simplificada entre diferentes plataformas
APIs RESTful são independentes da linguagem de programação ou da plataforma. Isso significa que um back-end desenvolvido em PHP pode ser consumido por um front-end em Vue.js, um aplicativo mobile em Flutter ou qualquer outro cliente que entenda HTTP.

Exemplo: Um sistema de gerenciamento de produtos com APIs RESTful pode fornecer endpoints que podem ser consumidos tanto por um site de e-commerce quanto por um aplicativo mobile.

```
GET /api/produtos -> Retorna uma lista de produtos.
```

2. Modularidade
Ao seguir os princípios do REST, você pode criar APIs altamente modulares, com endpoints que representam recursos específicos. Essa abordagem facilita o desenvolvimento de sistemas escaláveis, onde cada módulo (ex.: autenticação, produtos, pedidos) pode ser tratado de forma independente.

Exemplo: Um sistema pode ter módulos distintos:

- Autenticação: `/api/auth/login`
- Produtos: `/api/produtos`
- Pedidos: `/api/pedidos`

Com isso, o mesmo back-end pode ser usado para diferentes front-ends:

- Um site web consumindo os dados.
- Um aplicativo mobile acessando os mesmos recursos.
- Um painel administrativo utilizando outros endpoints.

---

3. Facilidade no consumo de dados no formato JSON
RESTful APIs geralmente utilizam o formato JSON (JavaScript Object Notation) para enviar e receber dados. O JSON é leve, fácil de ler e escrever, e compatível com a maioria das linguagens de programação modernas.

Exemplo de resposta JSON de uma API RESTful:

```json
{
  "id": 1,
  "nome": "João",
  "email": "joao@example.com",
  "telefone": "123456789"
}
```

Esse formato pode ser facilmente interpretado por um aplicativo front-end, transformando os dados recebidos em uma interface interativa.

Princípios Básicos de APIs RESTful

1. Recursos e Representações
Em REST, tudo é tratado como um recurso. Cada recurso é identificado por um **URI** (Uniform Resource Identifier).
Exemplo:

- `/api/usuarios` (conjunto de usuários)
- `/api/usuarios/1` (detalhes do usuário com ID 1)

2. Métodos HTTP
Os métodos HTTP definem as operações que podem ser realizadas nos recursos:

- **GET**: Obtém dados de um recurso.
- **POST**: Cria um novo recurso.
- **PUT**: Atualiza um recurso existente.
- **DELETE**: Remove um recurso.

Exemplo prático de um recurso "Usuários":

Métodos HTTP	Endpoint	Descrição
GET	/api/usuarios	Retorna todos os usuários.
GET	/api/usuarios/1	Retorna o usuário com ID 1.
POST	/api/usuarios	Cria um novo usuário.
PUT	/api/usuarios/1	Atualiza o usuário com ID 1.
DELETE	/api/usuarios/	Remove o usuário com ID 1.

Exemplo Prático: Criando uma API RESTful com Node.js e Express

Passo 1: Configuração do ambiente

1. Instale o Node.js.
2. Crie um novo projeto e instale o Express:

```
npm init -y
npm install express
```

Passo 2: Criando uma API simples

```
const express = require('express');
const app = express();
app.use(express.json());

let usuarios = [
  { id: 1, nome: 'João', email: 'joao@example.com' },
  { id: 2, nome: 'Maria', email: 'maria@example.com' }
];

// Rota GET para retornar todos os usuários
app.get('/api/usuarios', (req, res) => {
  res.json(usuarios);
});

// Rota GET para retornar um único usuário
app.get('/api/usuarios/:id', (req, res) => {
  const usuario = usuarios.find(u => u.id === parseInt(req.params.id));
  if (!usuario) return res.status(404).send('Usuário não encontrado');
  res.json(usuario);
});

// Rota POST para adicionar um novo usuário
app.post('/api/usuarios', (req, res) => {
  const novoUsuario = {
    id: usuarios.length + 1,
    nome: req.body.nome,
    email: req.body.email
  };
  usuarios.push(novoUsuario);
  res.status(201).json(novoUsuario);
});

// Rota DELETE para remover um usuário
app.delete('/api/usuarios/:id', (req, res) => {
  const usuarioIndex = usuarios.findIndex(u => u.id === parseInt(req.params.id));
  if (usuarioIndex === -1) return res.status(404).send('Usuário não encontrado');
  const usuarioRemovido = usuarios.splice(usuarioIndex, 1);
  res.json(usuarioRemovido);
});

// Inicia o servidor
app.listen(3000, () => console.log('API rodando na porta 3000'));
```

Consumindo APIs RESTful com Vue.js usando Axios

No front-end, utilizamos o **Axios** para consumir dados de APIs RESTful.
Exemplo de consumo:

```
export default {
  data() {
    return {
      usuarios: [],
      novoUsuario: { nome: '', email: '' }
    };
  },
  methods: {
    async carregarUsuarios() {
      const resposta = await axios.get('http://localhost:3000/api/usuarios');
      this.usuarios = resposta.data;
    },
    async adicionarUsuario() {
      await axios.post('http://localhost:3000/api/usuarios', this.novoUsuario);
      this.carregarUsuarios(); // Atualiza a lista
    }
  },
  mounted() {
    this.carregarUsuarios();
  }
};
```

Interface Vue.js:

```
<div id="app">
  <h2>Lista de Usuários</h2>
  <ul>
    <li v-for="usuario in usuarios" :key="usuario.id">
      {{ usuario.nome }} ({{ usuario.email }})
    </li>
  </ul>

  <h3>Adicionar Usuário</h3>
  <input v-model="novoUsuario.nome" placeholder="Nome">
  <input v-model="novoUsuario.email" placeholder="Email">
  <button @click="adicionarUsuario">Adicionar</button>
</div>
```

Benefícios ao Desenvolver com APIs RESTful

1. Manutenção Facilitada: A separação entre front-end e back-end facilita a manutenção e a atualização do sistema.
2. Reutilização: O mesmo back-end pode ser usado para múltiplos clientes (web, mobile, etc.).
3. Escalabilidade: APIs RESTful podem ser facilmente escaladas horizontalmente para suportar mais usuários.

Lista de Exercícios

Parte 1: Questões Teóricas (15 questões)

1. Defina o conceito de Vue.js e cite pelo menos três de suas principais características.
2. Explique a importância de usar Node.js em aplicações web. Como ele difere de outras tecnologias de back-end como PHP?
3. O que são APIs RESTful? Cite suas principais vantagens no desenvolvimento de sistemas modernos.
4. Qual a função do `npm` no ecossistema Node.js? Explique como ele é usado no gerenciamento de dependências.
5. Explique a diferença entre os métodos HTTP `GET`, `POST`, `PUT` e `DELETE` em APIs RESTful.
6. Descreva o que é o Vue CLI e como ele facilita o desenvolvimento de projetos Vue.js.
7. Qual é o propósito do formato JSON em APIs RESTful? Por que ele é amplamente utilizado?
8. Compare o uso de Vue.js em uma aplicação front-end com o uso de bibliotecas como jQuery. Quais as vantagens do Vue.js?
9. Cite dois cenários em que o PHP seria mais indicado que o Node.js para o desenvolvimento de um projeto.
10. Explique a relação entre Vue.js e o DOM reativo. Como isso melhora a performance de uma aplicação?
11. Qual é a importância de separar o front-end (Vue.js) do back-end (Node.js ou PHP) no desenvolvimento de aplicações?
12. O que é o comando `npm install` e como ele se diferencia de `npm install -g`?
13. Por que o método `POST` em APIs RESTful é usado para criação de recursos? Dê um exemplo prático.
14. Explique o conceito de modularidade em APIs RESTful e como isso pode beneficiar um projeto grande.
15. Qual o papel do `yarn` no ecossistema de desenvolvimento? Compare-o com o `npm`.

Parte 2: Questões Práticas (10 questões)

1. Instale o Node.js em sua máquina e, utilizando o terminal, verifique a versão instalada. Envie um print do comando `node -v`.
2. Crie um novo projeto Vue.js utilizando o Vue CLI. Certifique-se de que o servidor de desenvolvimento seja iniciado com sucesso.
3. Utilizando o npm, instale a biblioteca Axios em um projeto Vue.js e configure-a para realizar uma requisição `GET` para `https://jsonplaceholder.typicode.com/posts`. Exiba os dados retornados no console.
4. Implemente uma API RESTful com Node.js e Express que possua os seguintes endpoints:
   - `GET /usuarios`: Retorna uma lista de usuários.
   - `POST /usuarios`: Adiciona um novo usuário.
5. No Vue.js, crie uma interface que consuma os dados da API criada no exercício anterior e exiba a lista de usuários em uma tabela.
6. Adapte a API do exercício 4 para incluir um endpoint `DELETE /usuarios/:id`, que permita remover um usuário pelo ID.
7. Implemente uma página no Vue.js que permita adicionar e remover usuários, conectando-se à API do exercício anterior.
8. Utilizando o PHP, implemente uma API básica para listar produtos de uma tabela fictícia chamada `produtos` no banco de dados MySQL. Crie o endpoint `GET /produtos`.
9. Configure um projeto com Vue CLI e adicione um formulário reativo que permita o cadastro de usuários. O formulário deve conter os campos Nome, Email e Telefone.
10. Consuma a API pública de CEP (ex.: ViaCEP) no Vue.js. Crie uma interface onde o usuário insira o CEP e os dados retornados sejam exibidos na tela (logradouro, bairro, cidade e estado).

- *Responda todas as questões teóricas e práticas.*
- *Organize as respostas em um arquivo PDF, contendo nome e data.*
- *Envie o PDF aos professores responsáveis, seguindo o padrão de nomenclatura.*
- *Valide o material com um professor antes de prosseguir.*
- *Certifique-se de cumprir o prazo de entrega.*

Padrão de nomeclatura
NomeCompleto_TrilhaX_DataDeEntrega.pdf
Exemplo: JoãoSilva_Trilha1_2025-01-30.pdf