UniSENAI

Trilha 04

Estilização com Vuetify

3 - Estilização com Vuetify

O **Vuetify** é uma biblioteca de componentes baseada em Material Design, amplamente utilizada no Vue.js para criar interfaces modernas e responsivas de forma rápida e eficiente. Ele oferece uma ampla gama de componentes pré-estilizados, suporte a temas e um sistema de grade poderoso, facilitando o desenvolvimento de aplicações complexas.

3.1 Dominando o Vuetify

Configuração Inicial e Introdução ao Vuetify

O que é o Vuetify?

Vuetify é uma biblioteca de componentes para Vue.js que segue os padrões do Material Design. Ele oferece uma estrutura robusta com elementos de UI pré-definidos, como botões, cards, tabelas, toolbars e muito mais.

Por que usar Vuetify?

1. **Padrão de Design:** Segue as diretrizes do Material Design, garantindo uma interface consistente e moderna.
2. **Alta Produtividade:** Reduz o tempo necessário para criar interfaces complexas.
3. **Responsividade:** Possui um sistema de grade poderoso que facilita o desenvolvimento de layouts responsivos.

Configuração do Vuetify

Para adicionar o Vuetify ao seu projeto Vue.js, siga os passos:

1. **Instale o Vuetify via Vue CLI:**

```
vue add vuetify
```

2. **Ou instale manualmente:**

```
npm install vuetify
npm install sass sass-loader fibers deepmerge -D
```

3. **Configure no `main.js`:**

```
import { createApp } from 'vue';
import App from './App.vue';
import vuetify from './plugins/vuetify';

const app = createApp(App);
app.use(vuetify);
app.mount('#app');
```

4. **Adicione o plugin `vuetify` na pasta `plugins/vuetify.js`:**

```
import { createVuetify } from 'vuetify';
import 'vuetify/styles';

export default createVuetify();
```

3.2 Grid System no Vuetify

O Grid System do Vuetify é baseado no flexbox e é dividido em 12 colunas. Ele permite criar layouts responsivos facilmente, ajustando os elementos automaticamente para diferentes tamanhos de tela.

Exemplo Básico de Layout Responsivo

```
<template>
  <v-container>
    <v-row>
      <v-col cols="12" md="6" lg="4">
        <v-card>Coluna 1</v-card>
      </v-col>
      <v-col cols="12" md="6" lg="4">
        <v-card>Coluna 2</v-card>
      </v-col>
      <v-col cols="12" md="12" lg="4">
        <v-card>Coluna 3</v-card>
      </v-col>
    </v-row>
  </v-container>
</template>
```

Explicação:

- `v-container`: Define um contêiner para o layout.
- `v-row`: Cria uma linha.
- `v-col`: Cria colunas dentro da linha. Os valores `cols`, `md` e `lg` especificam o número de colunas ocupadas para diferentes tamanhos de tela.

3.3 Componentes Principais do Vuetify

O Vuetify oferece diversos componentes úteis. Vamos explorar os principais:

Botões

```
<template>
  <v-btn color="primary" @click="alertar">Clique Aqui</v-btn>
</template>

<script>
export default {
  methods: {
    alertar() {
      alert('Botão clicado!');
    }
  }
};
</script>
```

Propriedades úteis:

- `color`: Define a cor do botão (ex.: `primary`, `secondary`, `success`).
- `outlined`: Adiciona borda ao botão.
- `block`: Faz o botão ocupar toda a largura do contêiner.

Cards

```
<template>
  <v-card>
    <v-card-title>Card Título</v-card-title>
    <v-card-text>Este é um card simples.</v-card-text>
    <v-card-actions>
      <v-btn color="primary">Ação</v-btn>
    </v-card-actions>
  </v-card>
</template>
```

Modais

```
<template>
  <v-dialog v-model="modalAberto" max-width="500">
    <template #activator="{ on }">
      <v-btn color="primary" v-on="on">Abrir Modal</v-btn>
    </template>
    <v-card>
      <v-card-title>Título do Modal</v-card-title>
      <v-card-text>Este é o conteúdo do modal.</v-card-text>
      <v-card-actions>
        <v-btn color="secondary" @click="modalAberto = false">Fechar</v-btn>
      </v-card-actions>
    </v-card>
  </v-dialog>
</template>

<script>
export default {
  data() {
    return {
      modalAberto: false
    };
  }
};
</script>
```

Toolbars

```
<template>
  <v-app-bar color="primary" dark>
    <v-app-bar-title>Título</v-app-bar-title>
  </v-app-bar>
</template>
```

Snackbar

```
<template>
  <v-snackbar v-model="mostrarSnackbar" color="success">
    Ação realizada com sucesso!
    <template #actions>
      <v-btn text @click="mostrarSnackbar = false">Fechar</v-btn>
    </template>
  </v-snackbar>
  <v-btn @click="mostrarSnackbar = true">Mostrar Snackbar</v-btn>
</template>

<script>
export default {
  data() {
    return {
      mostrarSnackbar: false
    };
  }
};
</script>
```

3.4 Trabalhando com Tabelas (Data Tables)

As data tables permitem exibir dados em formato tabular, com suporte a paginação, filtros e ordenação.

Exemplo Básico

```
<template>
  <v-data-table :items="usuarios" :headers="cabecalhos" class="elevation-1"></v-data-table>
</template>

<script>
import axios from 'axios';

export default {
  data() {
    return {
      cabecalhos: [{ text: 'Nome', value: 'nome' }],
      usuarios: []
    };
  },
  mounted() {
    axios.get('https://jsonplaceholder.typicode.com/users').then(response => {
      this.usuarios = response.data.map(user => ({ nome: user.name }));
    });
  }
};
</script>
```

3.5 Validações de Formulários com Vuetify

O Vuetify facilita a validação de formulários por meio de regras configuráveis.

Exemplo: Validação de Campos

```
<template>
  <v-form ref="form" v-model="formValido">
    <v-text-field
      v-model="nome"
      label="Nome"
      :rules="[requerido]"
    ></v-text-field>
    <v-btn :disabled="!formValido" @click="submeter">Enviar</v-btn>
  </v-form>
</template>

<script>
export default {
  data() {
    return {
      nome: '',
      formValido: false,
      requerido: value => !!value || 'Este campo é obrigatório'
    };
  },
  methods: {
    submeter() {
      alert(`Nome enviado: ${this.nome}`);
    }
  }
};
</script>
```

3.6 Personalização Avançada no Vuetify

Criando Temas Customizados

O Vuetify suporta temas personalizados para ajustar cores e estilos.

```
import { createVuetify } from 'vuetify';

export default createVuetify({
  theme: {
    themes: {
      light: {
        primary: '#4CAF50',
        secondary: '#FFC107'
      }
    }
  }
});
```

Utilizando SCSS para Modificar Estilos

```
<style lang="scss" scoped>
.v-btn {
  background-color: #ff5722 !important;
}
</style>
```

3.7 Integração com APIs

Integre tabelas ou formulários ao back-end para preencher dados dinamicamente.

Exemplo: Preenchendo Tabela com API

```vue
<template>
  <v-data-table :items="usuarios" :headers="cabecalhos" class="elevation-1"></v-data-table>
</template>

<script>
import axios from 'axios';

export default {
  data() {
    return {
      cabecalhos: [{ text: 'Nome', value: 'nome' }],
      usuarios: []
    };
  },
  mounted() {
    axios.get('https://jsonplaceholder.typicode.com/users').then(response => {
      this.usuarios = response.data.map(user => ({ nome: user.name }));
    });
  }
};
</script>
```

Resumo

- O Vuetify é uma ferramenta poderosa para criar interfaces modernas e responsivas.
- Ele oferece suporte para temas customizados, validação de formulários e componentes avançados como tabelas e modais.
- A integração com APIs permite dinamizar o conteúdo exibido, aumentando a interatividade das aplicações.

Material Complementar: Documentação Oficial do Vuetify

Para aprofundar seus conhecimentos sobre o **Vuetify**, é altamente recomendável explorar a **documentação oficial** disponível em:

🔗 **https://vuetifyjs.com/en/**

A documentação oficial do Vuetify contém informações detalhadas sobre todos os aspectos da biblioteca, incluindo:

✅ **Guia de Instalação:** Passo a passo para configurar o Vuetify em diferentes ambientes de desenvolvimento.
✅ **Componentes e APIs:** Explicação completa sobre cada componente disponível, incluindo suas propriedades, eventos e exemplos de uso.
✅ **Sistema de Grade e Responsividade:** Como utilizar o flexbox e os layouts dinâmicos para criar interfaces adaptáveis.
✅ **Customização e Temas:** Métodos avançados para personalizar cores, estilos e criar temas específicos para sua aplicação.
✅ **Boas Práticas e Exemplos:** Casos de uso e exemplos práticos para aplicar no desenvolvimento de projetos reais.

Por que consultar a documentação do Vuetify?

📌 **Mantida pela Comunidade e Desenvolvedores Oficiais** – Sempre atualizada com novos recursos e melhorias.
📌 **Exemplos Interativos** – Possui trechos de código prontos para copiar e testar diretamente.
📌 **Referência Completa** – Útil tanto para iniciantes quanto para desenvolvedores avançados.

Conclusão

A documentação oficial do Vuetify é um recurso indispensável para quem deseja dominar essa poderosa biblioteca. Consultá-la frequentemente ajudará você a expandir seus conhecimentos, aprender novas funcionalidades e otimizar o desenvolvimento de interfaces modernas e responsivas com Vue.js. 🚀

Lista de Atividades - Estilização com Vuetify

Questões Teóricas (10 questões)

1. O que é o Vuetify e quais são suas principais vantagens ao desenvolver interfaces no Vue.js?
2. Explique o processo de configuração inicial do Vuetify em um projeto Vue.js. Quais são os passos necessários para adicioná-lo ao projeto?
3. Descreva o funcionamento do sistema de Grid do Vuetify. Como ele se compara ao sistema tradicional de CSS Flexbox e Bootstrap?
4. Quais são os principais componentes do Vuetify? Escolha pelo menos três e explique suas funcionalidades e usos.
5. Explique a diferença entre os componentes `v-btn`, `v-card` e `v-dialog`. Em quais situações cada um deles é mais adequado?
6. Como funciona a validação de formulários no Vuetify? Qual a importância do atributo `rules` ao definir um campo de entrada de dados?
7. Explique como funcionam as `Data Tables` no Vuetify e como elas podem ser utilizadas para exibir grandes volumes de dados.
8. O Vuetify permite a personalização avançada da interface. Explique como criar e configurar temas customizados.
9. Qual é a importância do uso de SCSS no Vuetify? Como ele pode ser utilizado para modificar estilos globalmente?
10. Como integrar APIs com Vuetify para preencher dinamicamente tabelas e formulários? Cite um exemplo de requisição e consumo de dados.

Questões Práticas (10 questões)

1. Crie um projeto Vue.js com Vuetify e implemente um layout básico utilizando o sistema de Grid. O layout deve conter três colunas que se ajustam para telas pequenas.
2. Crie um botão (`v-btn`) estilizado com uma cor personalizada e um evento de clique que exibe uma mensagem no console.
3. Implemente um `v-card` contendo uma imagem, um título e um botão que exibe um alerta ao ser pressionado.
4. Crie um modal (`v-dialog`) que seja ativado ao clicar em um botão. O modal deve conter um título, uma mensagem e um botão para fechá-lo.
5. Desenvolva um `v-toolbar` fixo na parte superior da tela que contenha um menu lateral (`v-navigation-drawer`) e um título centralizado.
6. Crie uma `Data Table` (`v-data-table`) que carregue uma lista de produtos a partir de um array estático no `data()`, contendo colunas para nome, preço e estoque.
7. Implemente um formulário de login utilizando Vuetify, com validação de campos para email e senha. Os campos devem exibir mensagens de erro quando não preenchidos corretamente.
8. Crie um tema customizado no Vuetify, alterando a cor primária para verde e a cor secundária para amarelo. Aplique essas cores nos componentes da interface.
9. Utilizando SCSS, modifique a aparência dos botões no Vuetify para que fiquem arredondados e com uma sombra suave.
10. Faça a integração de uma API externa com Vuetify. Os dados da API devem ser exibidos dinamicamente em uma `Data Table`, atualizando os registros ao carregar a página.

- *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