VueRouter

DEVinHouse

Parcerias para desenvolver a sua carreira

SENAI

<LAB365>

AGENDA DA SEMANA

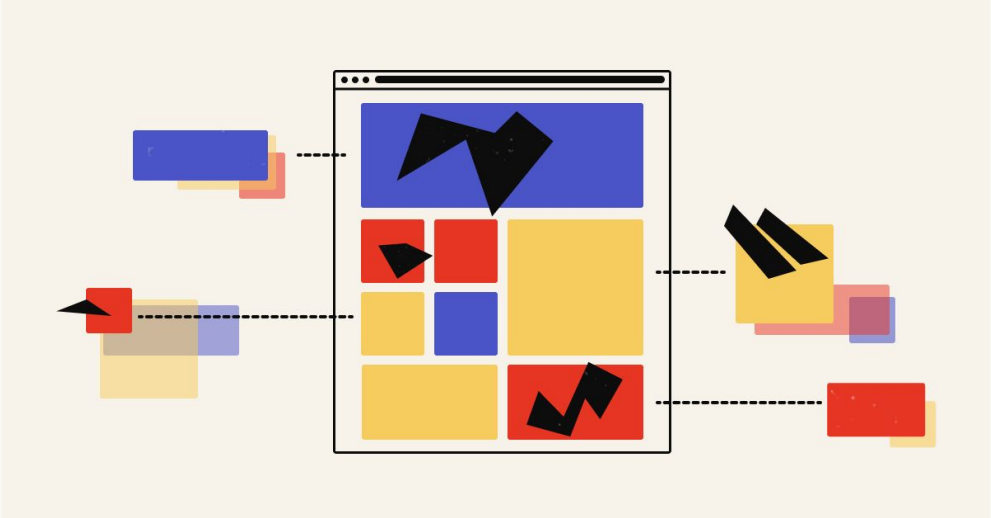
- Transition
- Transition Group
- **VueRouter**
- VueX e gerenciamento de estado

Vue Router

O Vue Router é a biblioteca de mapeamentos de rotas do Vue. O Vue Router irá mapear seus componentes e irá renderizar conforte a sua rota correspondente, criando uma aplicação de página única, conhecido também como SPA (Single Page Application).

O que é SPA?

"Single Page Applications (SPA) são aplicações cuja funcionalidade está concentrada em uma única página. Ao invés de recarregar toda a página ou redirecionar o usuário para uma página nova, apenas o conteúdo principal é atualizado de forma assíncrona, mantendo toda a estrutura da página estática." *(Fonte.: https://www.devmedia.com.br/ja-ouviu-falar-em-single-page-applications/39009)*



Instalando o Vue Router

A versão atual do Vue Router para uso no VUE 3 é a versão 4, sendo assim, as versões anteriores não irão funcionar. Para efetuar a instalação do VueRouter 4, basta instalar o módulo no projeto através do comando:

npm install vue-router@4

Configurações Iniciais

Após a instalação do VueRouter, iremos mapear as rotas numa lista e em seguida inicializar o roteador e importar no Vue, no main.js:

Importando o VueRouter:

```
import { createRouter, createWebHashHistory } from 'vue-router'
```

Criando lista de rotas:

```
const routes = [
    { path: '/', component: Home },
    { path: '/cadastro', component: Cadastro },
];
```

Configurações Iniciais

Criando as rotas:

```
const router = createRouter({
    routes,
    history: createWebHashHistory()
})
```

Importando no projeto:

```
app.use(router);
```

Configurações Iniciais

```
import { createApp } from 'vue'
import App from './App.vue'
import { createRouter, createWebHashHistory } from 'vue-router'

import Home from './components/home/Home.vue'
import Cadastro from './components/cadastro/Cadastro.vue'

const routes = [
    { path: '/', component: Home },
    { path: '/cadastro', component: Cadastro },
];

const router = createRouter({
    routes,
    history: createWebHashHistory()
})

const app = createApp(App);

app.use(router);

app.mount('#app')
```

History

A history nos permite escolher diferentes modos de histórico da aplicação, podendo ser:

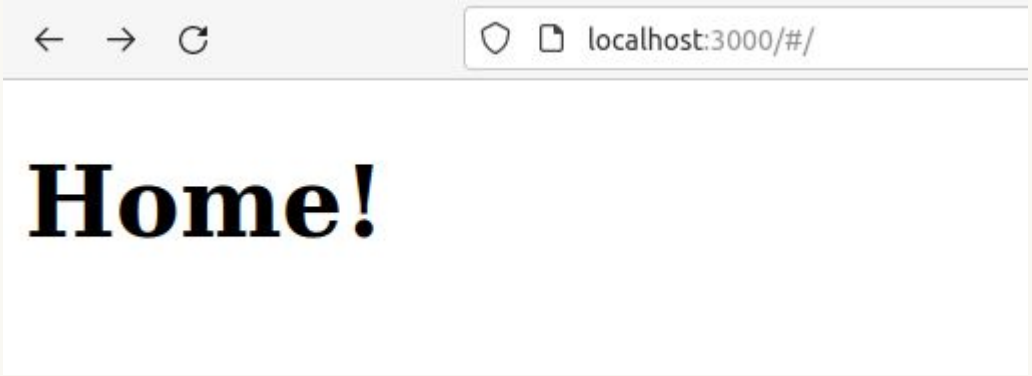
Web Hash History: Utiliza a hash (#) antes da URL real que é passada internamente. Como essa seção da URL nunca é enviada ao servidor, ela não requer nenhum tratamento especial em nível de servidor.

Web History: Neste caso, a URL aparecerá “normal”, porém se não houver as configurações devidas de redirecionamento no lado do servidor, ao acessar o URL, o usuário receberá o erro 404 (Not Found).

Router view

Para utilizar o componente mapeado na rota, no componente na qual está aberto pela aplicação, basta adicionar a tag <router-view>. Ele exibirá o componente correspondente à url e você pode colocá-lo em qualquer lugar para adaptá-lo ao seu layout.

```
<div id="app">
  <router-view></router-view>
</div>
```



Router link

Para acessar uma rota, não utilizaremos a tag <a>, mas sim a tag <router-link>, que ficará responsável por renderizar a rota solicitada na aplicação.

```
<template>
    <div>
        <h1>Home!</h1>
        <router-link to="/cadastro">Cadastro</router-link>
    </div>
</template>
```

Router link

Também podemos acessar o router link pelo método através do “this.$router”:

```
<template>
    <div>
        <h1>Home!</h1>
        <button @click="abrirCadastro">Cadastro</button>
    </div>
</template>
<script>
    export default {
        methods: {
            abrirCadastro() {
                this.$router.push('/cadastro')
            }
        }
    }
</script>
```

Rotas filhas

Na lista de rotas podemos definir os componentes filhos através da variável “children”, onde será renderizado os componentes de acordo com a URL, assim como na rota principal

```
const routes = [
    { path: '/', component: Home },
    { path: '/cadastro', component: Cadastro, children: [
        { path: 'pagina1', component: Pagina1 },
        { path: 'pagina2', component: Pagina2 },
    ] },
];
```

Navigation Guard

Os protetores de navegação fornecidos pelo Vue Route são usados principalmente para proteger as navegações redirecionando-as ou cancelando-as. Há várias maneiras de se conectar ao processo de navegação de rota: globalmente, por rota ou no componente.

```
const routes = [
    { path: '/', component: Home },
    { path: '/cadastro', component: Cadastro, children: [
        // Guard por rota
        { path: 'pagina1', component: Pagina1, beforeEnter: (to, from, next) => {
            return true
        } },
        { path: 'pagina2', component: Pagina2 },
    ] },
];
```

```
// Global
router.beforeEnter((to, from) => {
    return true
})
```

DEVinHouse

Parcerias para desenvolver a sua carreira

OBRIGADO!

SENAI

<LAB365>