**Propriedades do <router-link>**

1. <router-link to=”/profile” replace > -> Navegação não irá criar um histórico
2. <router-link to=”/profile” append > -> O link atual será mesclado com o novo link(“/link\_atual/profile”)
3. <router-link to=”/profile” tag=”p” > -> Podemos mudar a tag de link que será renderizada no HTML. Por padrão é a tag <a>
4. <router-link to=”/profile” active-class=”minha-classe” > -> Podemos adicionar uma classe personalizada para quando o link estiver ativo na página
5. <router-link to=”/profile” event=”dbclick” > -> Podemos definir qual evento pode ser usado para acionar a mudança de link. Essa propriedade também aceita um array de eventos.

**Propriedades do <router-view>**

1. <router-view name=”section” > -> Propriedade útil quando precisamos renderizar mais de um componente na mesma rota. Ao dar um nome para o router-view, podemos associá-lo com um componente de mesmo nome na instância VueRouter.

**Métodos para Navegação**

1. this.$router.push() -> Método que muda a URL atual, adicionando uma nova informação para o histórico do navegador.
2. this.$router.replace() -> Método que muda a URL atual, mas não adiciona uma nova informação para o histórico.
3. this.$router.go() -> Método que volta ou avança uma página já acessada. Se informarmos -1 para ele, voltaremos para a última página. Se informarmos 1, avançaremos para a página seguinte.

**Propriedades do $route**

1. $route.path -> O caminho da rota. Exemplo: “/users”
2. $route.fullPath -> O caminho da rota junto com a URL completa (domínio, hash, etc).

Exemplo: <https://meufocoetii.com.br/users>

1. $route.query -> Um objeto que contém os valores informados na query string da URL.

Exemplo: “/users?name=Joao&age=30”

1. $route.redirectFrom -> Se a rota atual estiver vindo de um redirect, essa propriedade mostrará de qual URL ela veio.
2. $route.name -> O nome da rota, se tivermos definido um
3. $route.hash -> O hash da rota

**Rotas Filhas(children)**

1. Usada para que o sistema não fique muito verboso com rotas repetitivas

***Exemplo***: const routes = [

{ path: ‘/products/:id’

component: productComponent,

Children: [

{ Path: ‘photos’,

Componente: ‘photosComponent’

}] -> Isso corresponde a /products/:id/photos

}]

**Nome para Rotas (router-link)**

1. Podemos dar nome para uma rota usando a propriedade ***(name)*** ao criar nosso array de rotas.
2. No HTML, podemos informar seu nome para o <router-link>, usando a propriedade (to).

Contudo, iremos informar um objeto para ela, não apenas uma string

***Exemplo***: const routes = [{

path: ‘/products/:id’

component: productComponent,

name: ‘product’

}]

<router-link to=’{ name: ‘product’ }’ > Link </router-link>

1. Também podemos informar parâmetros para as rotas que definirmos um nome

No HTML, devemos informar uma propriedade ***(params)*** além do ***(name)***

***Exemplo***: const routes = [{

path: ‘/products/:id’

component: productComponent,

name: ‘productEdit’

}]

<router-link to=’{ name: ‘productEdit’, params: { id: 23 } }’ >

Link

</router-link>

**Nome para as Views (router-view)**

1. Podemos dar um nome para uma view usando a propriedade ***(name)*** no componente <router-view>
2. Devemos definir no arquivo de rotas quais componentes específicos a rota irá renderizar usando a propriedade components ao criar nosso array de rotas, informando o mesmo identificador criado no HTML

Exemplo:

<router-view name=’home’ > Link </router-view>

<router-view name=’profile’ > Link </router-view>

<router-view> Link </router-view> Corresponde ao component default

const routes = [{

path: ‘/’,

components: {

home: homeComponent,

profile: profileComponent,

default: errorComponent,

}

}]

**Ordem na criação das Rotas**

1. Sempre colocar as rotas que possuem parâmetros em sua estrutura, por último no array de rotas