**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

**Redirect e Alias**

1. Redireciona rotas e pode acessar um mesmo caminho por mais de uma rota
2. O redirect deve vir sempre após o ***path***

***Exemplo:***

const routes = [{

path: ‘/admin’,

redirect: ‘/home’ -> ao acessar /admin, será redirecionar para a home

}, {

path: ‘/admin’,

redirect: {name: ‘homepage’} -> Podemos usar como redirect uma rota com nome também

}]

1. const routes = [{

path: ‘/admin’,

alias: ‘/administration’, -> Ao acessar /administration, verá o componente de /admin sem erros

component: adminComponent

}]

**Navigation Guards - Global**

1. Monitora as rotas de nossa aplicação, dando a nós a opção de realizarmos algum tipo de verificação antes ou depois de renderizá-las no navegador.
2. O VueRouter possui ***3 métodos globais*** principais para controlarmos o acesso de cada rota da aplicação.
   1. ***beforeEach()*** -> Método chamado antes de a rota ser acessada.
   2. ***beforeResolve()*** -> Método chamado assim que a navegação é confirmada.
   3. ***afterEach()*** -> Método chamado assim que a navegação é finalizada.
3. Cada método recebe 3 parâmetros
   1. ***to*** -> Rota para onde a navegação irá
   2. ***from*** -> Rota de onde a navegação está vindo
   3. ***next*** -> Função que pode ser usada para cancelar a navegação, realizar um redirecionamento ou exibir um erro (Obs: O método afterEach() não tem acesso a esse recurso.)

***Exemplo:***

***const router = new VueRouter({ })***

***router.beforeEach( ( to, next, from ) => {***

***if(!user.isLogged()){***

***next(‘/login’)***

***} else {***

***next()***

***}***

***} )***

***router.beforeEach( ( to, from, next ) => {***

***console.log(‘Navegação confirmada’)***

***} )***

***router.afterEach( ( to, from ) => {***

***console.log(‘Navegação concluída ok’)***

***} )***

**Navigation Guards - Local**

1. O VueRouter possui 1 ***método*** para criar guardas por rotas, de maneira individual.
   1. ***beforeEnter()*** -> Método chamado antes de a rota ser acessada. Tem a mesma função do beforeEach(), mas é usado para aplicar a monitoração em apenas uma rota. Ele recebe os mesmos parâmetros dos guardas globais.
2. O método recebe 3 parâmetros
   1. ***to*** -> Rota para onde a navegação irá
   2. ***from*** -> Rota de onde a navegação está vindo
   3. ***next*** -> Função que pode ser usada para cancelar a navegação, realizar um redirecionamento ou exibir um erro.

***Exemplo:***

***const router = new VueRouter({***

***routes: [{***

***path: ‘/profile’,***

***beforeEnter: ( to, from, next ) => {***

***if(!user.isLogged()) {***

***next(‘/login’)***

***} else {***

***next()***

***}***

***}***

***}]***

***})***

**Navigation Guards – Components**

1. É possível criar guardas de navegação até mesmo para os componentes. Para isso, temos à nossa disposição três métodos que podemos usar quando criamos um componente.
   1. ***beforeRouterEnter()*** -> Método chamado antes de a rota renderizar o componente na aplicação.
   2. ***beforeRouterUpdate()*** -> Método chamado antes de a rota mudar. Usado principalmente quando o componente continua sendo utilizado mesmo que a rota seja alterada.
   3. ***beforeRouterLeave()*** -> Método chamado antes de acessar uma nova rota, que não renderizará o componente.
2. O método recebe 3 parâmetros
   1. ***to*** -> Rota para onde a navegação irá
   2. ***from*** -> Rota de onde a navegação está vindo
   3. ***next*** -> Função que pode ser usada para cancelar a navegação, realizar um redirecionamento ou exibir um erro.

***Exemplo:***

    beforeRouteEnter: (to, from, next) => {

        let params = to.params

        if (!parseInt(params.idnotice) > 0) {

            next('/')

        } else {

            next()

        }

    }