**Rotas em uma aplicação VueJS**

1. Nesse capitulo iremos estudar sobre o Vue Ruter que é um plugin que irá controlar a navegação entre vários componentes.

**Aula 251 – Instalando o vue-router.**

1. vamos instalar a versão 3.0.2 do vue-router, para isso digite:
2. npm i --save vue-router@3.0.2 -E
3. serão instaladas todas as outras dependências.
4. Incluímos no arquivo index.html referências para o animate.css e fonte.
5. Criamos a pasta components e o componente inicio.vue
6. Criado a pasta usuário dentro de components e criado o componente Usuario.vue
7. Dentro da pasta src criado o arquivo router.js para configurarmos as rotas da aplicação.
8. No arquivo router.js
9. Importamos o Vue , importamos o vue-router e os dois componentes criados.
10. E registramos o plugin dentro do Vue.
11. E vamos exportar uma função construtora Router com objetos de configuração do vue-router.
12. Cada objeto que estiver dentro do array será responsável por mapear uma rota.
13. Configuramos o path: informando qual a rota que será aplicada e qual o componente será carregado.
14. import Vue from 'vue'
15. import Router from 'vue-router'
16. import Inicio from './components/Inicio'
17. import Usuario from './components/usuario/Usuario'
18. Vue.use(Router)
19. export default **new** Router({
20. routes: [{
21. path: '/',
22. component: Inicio
23. },{
24. path: '/usuario',
25. component: Usuario
26. }]
27. })
28. para funcionar devemos registrar o router no arquivo main.js importando e registrando dentro da instância do Vue.
29. import './style.css'
30. import Vue from 'vue'
31. import App from './App.vue'
32. import router from './router'
33. Vue.config.productionTip = false
34. **new** Vue({
35. router,
36. render: *h* => h(App),
37. }).$mount('#app')
38. Após realizar a configuração do vue-router temos que indicar onde os componentes serão exibido o conteúdo que será carregado pelo vue-router. Por isso vamos inserir no arquivo app.vue
39. A tag <router-view> e será neste ponto onde o vue-router irá carregar os componentes que foram importados no arquivo de rotas router.js.
40. <template>
41. <div *id*="app">
42. <h1>Rotas com VueRouter</h1>
43. <router-view />
44. </div>
45. </template>
46. C

**Aula 253 – Entendendo os modos de Rotas (hash vs history).**

1. Modo hash toda a requisição feita ao servidor passa obrigatoriamente pelo arquivo index.html e é onde é carregado o arquivo app.js.
2. Para mantermos o hash aparecendo na url do navegador, devemos fazer uma configuração no arquivo router.js informando o mode: ‘hash’.
3. export default **new** Router({
4. mode:'hash',
5. routes: [{
6. path: '/',
7. component: Inicio
8. },{
9. path: '/usuario',
10. component: Usuario
11. }]
12. })

Para retirar o hash da url basta utilizar o ‘history’ no mode, export default **new** Router({

    mode:'history',

    routes: [{

        path: '/',

        component: Inicio

    },{

        path: '/usuario',

        component: Usuario

    }]

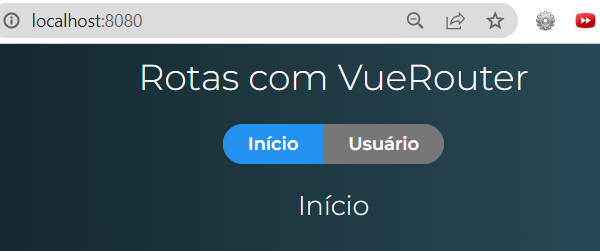
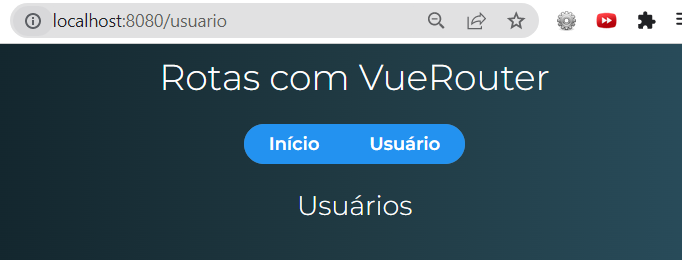
})

1. porém é necessário realizar configurações no servidor.
2. Para o modo history funcionar corretamente.

**Aula 254 – Navegando com Router links**

1. Iremos criar um menu para navegarmos entre os componentes sem ter que digitar a url.
2. Para isso vamos criar dentro de comonents uma nova pasta chamada template e dentro dela um novo arquivo chamado menu.vue.
3. Que terá a estrutura de um componente e em template uma tag nav e classe menu.
4. Dentro da tag nav teremos os links, porém, devemos utilizar o <router-link> utilizando a propriedade to: para informar os valores de path que estão configurados no arquivo router.js.
5. <template>
6. <nav *class*="menu">
7. <router-link *to*="/">Início</router-link>
8. <router-link *to*="/usuario">Usuário</router-link>
9. </nav>
10. </template>
11. Agora vamos importar o componente Menu.vue para o arquivo app.vue e registra-lo.
12. Vamos inserir o componente Menu após o h1 e antes do router-view.
13. Porém para ficar mais dinâmico, criei uma variável do tipo array no arquivo App.vue, chamada ‘itensMenu’ que possui como atributos ‘title: texto que irá aparecer no menu’ e ‘tag: caminho que foi configurado no arquivo router.js’.
14. <div *id*="app">
15. <h1>Rotas com VueRouter</h1>
16. <Menu :*itensMenu*="itensMenu" */*>
17. <router-view />
18. </div>
19. No arquivo App.js passo esse array para o componente Menu.
20. <div *id*="app">
21. <h1>Rotas com VueRouter</h1>
22. <Menu :*itensMenu*="itensMenu" */*>
23. <router-view />
24. </div>
25. no menu tenho um props do tipo array que irá receber o conteúdo do array.
26. export default{
27. props:{
28. itensMenu: {type: *Array*, required: true}
29. }
30. }
31. No componente Menu tenho também um v-for que irá percorrer os itensMenu e irá criar os roter-links para cada item.
32. <router-link v-for="item in itensMenu" :*key*="item.tag"
33. :*to*="item.tag"
34. >{{item.title}}</router-link>
35. C

**Aula 255 – Estilizando o link ativo do Menu**

1. Para aplicar os estilos que foram configurados nas classes css do arquivo style.css vamos envolver o <router-link> do componente Menu.vue, dentro de uma ul e li.
2. <ul>
3. <li v-for="item in itensMenu" :*key*="item.tag">
4. <router-link
5. :*to*="item.tag"
6. >{{item.title}}</router-link>
7. </li>
8. </ul>
9. Mas existe uma outra forma de alteramos a tag que envolve o <router-link>.
10. Para isso vamos utilizar o atributo tag dentro de <li> e passar para ela ‘li’, temos também que colocar o texto do <router-link> entre duas tag <a> </a>. O resultado final será o mesmo.
11. <nav *class*="menu">
12. <ul>
13. <router-link v-for="item in itensMenu" :*key*="item.tag"
14. :*to*="item.tag"
15. *tag*="li"
16. ><a>{{item.title}}</a>
17. </router-link>
18. </ul>
19. </nav>
20. 
21. Existe uma propriedade no router-link que define se o link está ativo ou não, essa propriedade é class-active. No arquivo style.css definimos uma classe chamada active que muda o estilo do link quando está ativo. Vamos então definir essa propriedade no router-link para aplicamos este estilo.
22. 
23. Porem existe um erro quando clicamos no outro link ‘usuario’, os dois ficam selecionados.
24. 
25. Para resolvermos esse erro basta utilizar a propriedade exact dentro do router-link, e desta forma ele irá aplicar a propriedade somente se o link for exatamento o que foi passado na propriedade to=.
26. <nav *class*="menu">
27. <ul>
28. <router-link v-for="item in itensMenu" :*key*="item.tag"
29. :*to*="item.tag"
30. *tag*="li"
31. *active-class*="active"
32. *exact*
33. ><a>{{item.title}}</a>
34. </router-link>
35. </ul>
36. 

**Aula 256 – Navegação via código (Navegação imperativa)**

1. Para fazer este tipo de navegação iremos utilizar código Javascript e não somente o <router-link>.
2. Para verificar isso vamos abrir o componente Usuario e incluir um botão para voltar na página anterior.
3. <template>
4. <div *class*="usuario">
5. <h2>Usuários</h2>
6. <hr>
7. <button *sucesso*>Voltar</button>
8. </div>
9. </template>
10. Porem estamos passando um atributo chamado ‘sucesso’para o botão esse atributo foi definido no arquivo style.css e está entre [] neste arquivo.
11. [*sucesso*] {
12. color: #FFF;
13. background-color: #47af55;
14. }
15. Desta forma aplicamos um estilo através de um atributo e não somente apartir de uma classe definida.
16. 
17. Agora vamos criar um método chamado irParaInicio() chamando a variável global $router. Vamos passar para a variável $router, através do método push, o path e ele irá navegar para o componente que possui o path associado.
18. export default{
19. methods:{
20. irParaInicio(){
21. *this*.$router.push('/')
22. }
23. }
24. }
25. Vamos inserir este método no evento @click do botão.
26. <button @*click*="irParaInicio" *sucesso*>Voltar</button>
27. Outra possibilidade também seria ao invés de passar ‘/’ para o método .push(), poderíamos passar um json com o path já definido.
28. methods:{
29. irParaInicio(){
30. // this.$router.push('/')
31. *this*.$router.push({ path: '/'})
32. }
33. }

**Aula 257 – Passando parâmetros para as Rotas.**

1. Para atribuirmos parâmetros em nossa url, devemos incluir no path do arquivo router.js os parâmetros utilizando ‘:’com o nome do paramentro.
2. export default **new** Router({
3. mode:'history',
4. routes: [{
5. path: '/',
6. component: Inicio
7. },{
8. path: '/usuario/:id',
9. component: Usuario
10. }]
11. })
12. Para nossa navegação continuar funcionando vamos subsituir na variável tag do array itensMenu o valor ‘/usuario’ por ‘/usuário/10’. Veremos que irá funcionar normalmente.
13. 

**Aula 258 – Lendo e usando parâmetros de rotas**

1. Estamos passando o parâmetro diretamente na variável tag do array. Para verificar o funcionamento de passagem de parâmetro vamos inserir no componente <Usuario />.
2. Uma tag <p> com um Strong com a palavra Código dentro de <p> iremos interpolar a variável id que em data(), recebe o parâmetro ‘id’ configurado no arquivo router.js.
3. <div *class*="usuario">
4. <h2>Usuários</h2>
5. <hr>
6. <p><strong>Código:</strong> {{id}}</p>
7. <button @*click*="irParaInicio" *sucesso*>Voltar</button>
8. </div>
9. Em data() teremos então o id recebendo o parâmetro.
10. data() {
11. return {
12. id: *this*.$route.params.id
13. }
14. },
15. Vamos alterar o array de itensMenu e incluir mais doi itens.
16. Quando clicamos no item do menu a variável id que está no template de <Usuario> para resolvermos isso iremos utilizar a propriedade watch:
17. Dentro de watch iremos monitorar a propriedade $route que possui dois parâmetros to e from, iremos setar a variável this.id com to.params.id isso fará com que o parâmetro id da url seja passado automaticamente para this.id.
18. watch:{
19. $route(*to*, *from*){
20. *this*.id = *to*.params.id
21. }
22. }