03

Refatorando a localstorage

Transcrição

Temos tudo quase funcionando como desejado, "quase" pois há um pequeno bug no sistema, vamos a ele. Ao fazermos o login em nossa aplicação e acessarmos a página de gerentes, o menu não se altera, sendo exibido ainda para o usuário as opções "login" e "registre-se". A atualização para a lista correta só acorre quando recarregamos a página. O mesmo problema ocorre quando realizamos o logout e o status não é automaticamente modificado.

Como parâmetro para a base de navegação, solicitamos que se retire do local storage o item `token` . Caso haja um token, o usuário está logado. Tal execução é chamada uma vez,pois por padrão o vue não monitora alterações dentro da local storage.

O local storage é uma solução de armazenamento durante o uso da aplicação, mas também queremos controlar o estado do usuário. Para esta nossa segunda necessidade, o local storage não funciona bem, e este é um desafio muito comum no front-end e existem diversas soluções possíveis.

Nós utilizaremos o Vuex, uma biblioteca pronta que serve para controlarmos o estado do usuário na aplicação. Na linha de comando,escreveremos `npm instal vuex -- save` , dessa maneira já teremos esse recurso disponível e então executaremos novamente o servidor.

Começaremos a configurar a biblioteca. Primeiramente, trabalharemos no provedor, aquele que será responsável pela monitoração do estado do usuário e não permitir que nenhuma alteração fora de escopo aconteça.

Criaremos um novo arquivo chamado `preovedor.js`. Começaremos a explicitar o estado inicial da aplicação: `token` que inicia nulo, e do `usuario` com um objeto literal. Feito isso, faremos a importação do vuex, então o exportaremos por padrão, além disso, ordenaremos que o vue utilize o vuex. O vuex possui um método chamado `Store()`, que receberá como estatos a instância que acabamos de criar.

```
import Vuex from 'vuex'
import Vue from 'vue'

Vue.use(Vuex)

const estado = {
    token: null,
    usuario: {}
}

export default new Vuex.Store({
    state: estado
})
```

COPIAR CÓDIGO

e deixá-lo disponível dentro do ambiente o importando em `main.js`, por isso importaremos. Em `Vue`, acrescentaremos que a `store` é `provedor`.

```
new Vue({
    router,
    store: provedor,
    render: h => h(App)
}).$mount('#app')
```

COPIAR CÓDIGO

Ao recarregarmos o navegador, veremos que aplicação funciona normalmente. Como utilizaremos o Vuex ao longo do curso, é interessante que instalemos e

nossa máquina no **Vue.js dvtools**, disponível na webstore do Google Chrome. Com esta ferramenta, conseguimos visualizar as alterações de estado ao inspecionarmos a página.

Com o Vuex configurado, podemos parar de usar a local storage. Em `Login.vue` comentaremos a `localStorage` e salvaremos o `token` e `usuario`.

```
methods: {
    this.$http.post('auth/login', this.usuario)
        .then(response => {
            console.log(response)
            // localStorage.setItem('token',
response.data.access_token)
            this.$store.state.token =
response.data.access_token
            this.$store.state.usuario = response.data.user
            this.$router.push({ name: 'gerentes '})
        })
        .catch(erro=> console.log(erro))
}
```

COPIAR CÓDIGO

Realizadas as modificações, carregaremos a página e faremos o login. Ao inspecionarmos a página veremos que o Viux não alterou nem o token ou o usuário. Isso se deu porque o token não está correto, mas por que isso aconteceu?

Comentamos anteriormente que o Viux é para controle de estado, e então ele não permite que alterações sejam feitas de maneira tão aberta. Precisamos utilizar as próprias ferramentas do Veux para realizar alterações dentro do estado.

Realizaremos uma mutação, mutation em inglês. Em `provedor.js` adicionaremos o `const mutations`. Precisaremos explicitar com clareza o procedimento que realizamos e qual deverá ser o resultado dado o contexto.

Em nossa mutação, iremos definir o usuário logado, para o que Vuex disponibilize o estado e dados (`token` e `usuario`).

```
const mutations = {
    DEFINIR_USUARIO_LOGADO (state, {token, usuario }) {
        state.token = token
        state.usuario = usuario
    }
}
```

COPIAR CÓDIGO

O Vuex sempre nos deixa dois parâmetros e os dados enviados para a mutação, realizamos a object destruction, isto é, separamos em duas variáveis: `token` e `usuario` .

Assim como definimos que nosso `state` é `estado` , precisaremos fazer o mesmo com `mutations` .

```
export  default new Vuex.Store({
    state: estado
    mutations
})
```

COPIAR CÓDIGO

Feito isso, alteraremos o método em `Login.vue` :

```
methods: {
    this.$http.post('auth/login', this.usuario)
        .then(response => {
            console.log(response)
            // localStorage.setItem('token',
response.data.access_token)
            // this.$store.state.token =
response.data.access_token
```

```
                // this.$store.state.usuario =
    response.data.user
                // this.$router.push({ name: 'gerentes '})
                this.$store.commit('DEFINIR_USUARIO_LOGADO', {
                    token: response.data.access_token
                    usuario: response.data.user
                })
```

COPIAR CÓDIGO

Feito isso, já conseguiremos efetuar o login com as devidas informações sendo armazenadas pelo Viuex. Mas ainda precisamos propagar essa lógica para o restante da aplicação.