04

Interceptando as requisições HTTP

Transcrição

Analisando o nosso projeto, estamos importando uma nova instância do `axios` tanto no componente `Login.vue` quanto na tela de `NovoUsuario.vue`.

Isso faz com que cada componente renderizado tenha uma nova instância do `axios` criada. Porém, quando tivermos que implementar alguma regra por exemplo, teremos que procurar todos os lugares que usam o `axios` para adicionar a regra, e podemos adicionar a mesma em componentes diferentes, o que pode ser um problema.

Para evitarmos essa situação, criaremos um único arquivo responsável por criar essa instância do `axios` e deixá-la disponível para o elemento que for utilizá-la, usando sempre a mesma instância.

Todas as vezes em que for usada, sairá do mesmo local, e portanto toda regra ficará encapsulada dentro do mesmo arquivo. Então, no VSCode, criaremos uma nova pasta em "src" chamada "http", a qual receberá um novo arquivo chamado `index.js`.

Este será responsável por exportar essa instância do `axios`. Da mesma forma que fizemos com os depois, importaremos o `axios` de `'axios'`, depois criaremos uma `const http` que receberá `axios.create()`, e por fim a exportaremos como `default`.

```
import axios from 'axios'
```

```
const http = axios.create()

export default http
```

COPIAR CÓDIGO

Este método de criar uma nova instância nos permite definir algumas propriedades comuns para todas as requisições, como por exemplo, a nossa `baseURL`, e todas as nossas requisições começam do mesmo jeito: `'http://localhost:8000'`.

Caso isso mude, teremos que ficar procurando qual elemento está usando essa url para podermos utilizar a nova. Desta forma, todos os componentes que usarem a instância do `axios` já possuem a `baseURL:` definida, e só precisaremos pensar no que vem depois do endereço padrão.

Também poderemos definir algumas propriedades do `headers:`, então diremos, por exemplo, que aceitaremos tanto o `'Accept:'` quanto o `'Content:'` como `'application/json'`.

Uma vez que já temos o nosso objeto `http` que fará as requisições para o *back-end*, poderemos parar de importar o `axios` para importarmos este `http`.

```
import axios from 'axios'

const http = axios.create({
    baseURL: 'http://localhost:8000/',
    headers: {
        'Accept': 'application/json',
        'Content': 'application/json'
    }
})

export default http
```

COPIAR CÓDIGO

Então refatoraremos o `Login.vue`. Ao invés de importarmos o `axios`, importaremos o `http` no `<script>`.

Em seguida, queremos que, a partir da raiz, acessar o `htpp` e o `index.js`. Como usamos o projeto construído com `axios` que já é padrão do *view*, teremos um *alias* ou apelido para acessarmos a pasta "src", sem precisarmos fazer o caminho relativo com `'../..'`.

A partir da "src", queremos acessar a pasta `'@/http` pegando o arquivo `/index.js'`. Porém, como usamos o **webpack**, somos permitidos a não explicitar a extensão do arquivo, então podemos escrever apenas `'@/http/index'`.

Além disso, se o arquivo que está dentro desta pasta que estamos procurando se chamar `index`, também não precisaremos dizer seu nome. Portanto, podemos simplesmente importar `htpp` de `'@/http'`.

E ao invés de fazermos `axios.post()`, faremos `http.post()` e não precisaremos mais passar toda URL, apenas o *endpoint* que iremos acessar. Salvaremos e verificaremos se tudo está funcionando como o esperado no navegador.

```
//código anterior omitido

<script>
import http from '@/http'

export default {
  data() {
    return {
      usuario: {}
    };
  },
  methods: {
    efetuarLogin() {
```

```
          http.post("auth/login", this.usuario)
          .then(response => {
            console.log(response);
            localStorage.setItem("token",
response.data.access_token);
            this.$router.push({ name: "gerentes" });
          })
          .catch(erro => console.log(erro));
      }
    }
};
</script>
```

COPIAR CÓDIGO

Preenchendo os campos da aplicação com nossos dados, clicaremos em "Logar" e, na aba "Network" do inspetor de código do navegador, receberemos o "login" na coluna "Name".

O outro lugar que usa isso é o `NovoUsuario.vue`, portanto teremos que alterar o `import`. É normal que vários componentes necessitem desse `http`, e pensando nesses objetos comuns a todos os componentes, o Vue.js nos permite criá-los e deixá-los disponíveis em todas as instâncias de todos os componentes.

Para isso, iremos ao `main.js` de `App.vue` e importaremos o `http` a partir de `'@/http'`, e depois diremos ao `Vue` que coloque um objeto chamado `$http` dentro do protótipo usando `.prototype`, o qual será o nosso `http`.

O cifrão ou dólar `$` é uma convenção de nomes do Vue.js. Por exemplo, usamos o `$router` para identificarmos objetos globais. Portanto, disponibilizaremos o `$http`.

```
import Vue from 'vue'
import App from './App.vue'
import router from './router'
```

```
import 'bootstrap'
import 'bootstrap/dist/css/bootstrap.min.css'

import http from '@/http'

Vue.config.productionTip = false

Vue.prototype.$http = http

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

COPIAR CÓDIGO

Se voltarmos ao `Login.vue`, não precisaremos mais do `import http from '@/http'`, e poderemos simplesmente fazer `this.$http.post()` para ver se continua funcionando.

Salvando e indo de volta ao navegador, atualizaremos a página, faremos o Logout, preencheremos os campos e logaremos novamente.

```
//código anterior omitido

<script>
export default {
  data() {
    return {
      usuario: {}
    };
  },
  methods: {
    efetuarLogin() {
        this.$http.post("auth/login", this.usuario)
        .then(response => {
          console.log(response);
```

```
            localStorage.setItem("token",
response.data.access_token);
            this.$router.push({ name: "gerentes" });
          })
          .catch(erro => console.log(erro));
      }
    }
};
</script>
```

COPIAR CÓDIGO

Se tudo funcionar corretamente, não precisaremos mais fazer o `import` do `axios`.

Então iremos em `NovoUsuario.vue`, tiraremos a importação e, ao invés de chamarmos `axios.post()`, escreveremos `this.$http.post()`. Além disso, não precisaremos mais de toda a URL completa, apenas o *endpoint* que queremos acessar.

```
//código anterior omitido

<script>
export default {
  data: function() {
    return {
      usuario: {
        nome: '',
        senha: '',
        email: ''
      }
    };
  },
  methods: {
    enviarFormulario() {
      this.$http.post("auth/register", this.usuario)
        .then(resposta => {
```

```
                console.log(resposta)
                this.$router.push({ name: 'login' })
            })
            .catch(erro => console.log(erro))
        }
    }
}
</script>
```

COPIAR CÓDIGO

Tudo continua funcionando.

Outra coisa que é bastante comum é que, inclusive usaremos agora, quando fazemos o login do nosso usuário, pegamos o *token* e o enviamos nas requisições seguintes para o servidor conseguir nos identificar e calcular se temos a permissão ou não.

Então faremos com que o `axios` automaticamente adicione esse *token* a todas as requisições para que o servidor saiba quem somos neste caso.

No `index.js` de `http`, implementaremos o comportamento de adicionar o *header* com autorização por padrão. Acessando o `axios` que é nosso objeto `http`, diremos que acessaremos os `interceptors`, ou seja, os intereceptadores que irão interceptar cada vez que fizerem uma requisição no nosso caso.

Então a cada `request`, usaremos a função `function ()` em `use()` para o caso de sucesso, a qual receberá a configuração `config` do `axios`. Depois faremos outra `function ()` recebendo o caso de `erro`.

Este erro é bem simples, então apenas retornaremos uma `Promise` com `.reject()` rejeitada pelo `erro`. Como algo de errado aconteceu, não é responsabilidade do interceptador de tratar o erro neste caso.

Mas no `request` de sucesso, queremos pegar o `token` recebendo `localStorage` com `.getItem()` cujo nome é `'token'`. Neste caso, se o `token`

existir no `if()`, queremos adicionar um `headers` chamado `Authorization` com o valor `'Bearer'` concatenado com o `${token}`.

Isso é o que a nossa API do servidor está esperando. Quando uma requisição for protegida, precisaremos enviar um `headers` cujo nome é `Authorization` e o valor é o nosso `token` em si, concatenado com `Bearer`. Por fim, faremos o `return config`

```
import axios from 'axios'

const http = axios.create({
    baseURL: 'http://localhost:8000/',
    headers: {
        'Accept': 'application/json',
        'Content': 'application/json'
    }
})

http.interceptors.request.use(function (config) {
    const token = localStorage.getItem('token')
    if (token) {
        config.headers.Authorization = `Bearer ${token}`
    }
    return config
}, function (erro) {
    return Promise.reject(erro)
})

export default http
```

COPIAR CÓDIGO

Para testarmos isso, iremos ao componente `gerentes:` de `Gerentes.vue`, os quais já estão definidos *hard coded*, e não queremos isso, e sim que venham da API do servidor.

Portanto, quando alguém acessar a URL e montar o componente com `mounted()`, queremos que o Vue.js acesse o `.$http` com `this` para pegar com `get()` o `'gerentes'`.

Em caso de sucesso com `.then()`, queremos pegar a `response` e dizer que `gerentes` receberá `response.data`. Já se der errado, faremos o `.catch()` pegando o `erro` e logando-o no `console.log()`.

```
//código anterior omitido

<script>
import Gerente from "@/components/Gerente.vue";

export default {
  components: {
    Gerente
  },
  data() {
    return {
      gerentes: []
    };
  },
  mounted() {
    this.$http.get('gerentes')
      .then(response => (this.gerentes = response.data))
      .catch(erro => console.log(erro));
  }
}
</script>

<style>
</style>
```

COPIAR CÓDIGO

Com este código, voltaremos à aplicação no navegador para verificarmos se funciona.

Ao recarregarmos, notaremos na aba "Network" do inspetor de código que os "gerentes" da coluna "Name" já vêm da API. Observando o "Preview", veremos os nomes e os `id:` cada qual com sua `agencia:` .

Não vêm mais já pronto, então toda vez que este componente for montado, o Vue.js irá ao servidor, pegará os gerentes e já colocará o *token*.

Poderemos ver isso atualizando a página novamente e vendo a requisição `Authorization: Bearer...` de "gerentes" no "Headers" de Network". É este campo no cabeçalho que nos identificará para o servidor.

Se fizermos uma requisição deste tipo sem o token, como por exemplo copiando a URL através do clique com o botão direito em "gerentes" na coluna "Name" e acessando "Copy > Copy link address" para colar no campo do navegador, receberemos a mensagem de `"status": 401` .

Ou seja, não temos permissão para fazer isso e nosso *token* é inválido. Só conseguiremos ter acesso às informações dos gerentes quando autenticado e quando possui o *token* válido.