06

Implementando o login do usuário

Crie o arquivo `js/login/LoginUsuario.js` e adicione em `index.html` as tags `<script>` que precisaremos para fazer o login funcionar.

```
<script src="js/login/LoginUsuario_render.js"></script>
<script src="js/login/LoginUsuario.js"></script>
```

Crie em `js/login/LoginUsuario.js` uma variável `login` com o valor `false` . Mais pra frente ela terá seu valor alterado quando o usuário fizer login por meio do formulário. Ainda em `js/login/LoginUsuario.js` , execute a função LoginUsuario_render passando o valor da variável logado para a propriedade logado do objeto que passamos no parâmetro:

```
let logado = false
LoginUsuario_render({
    logado: logado
})
```

Altere `js/mural/Mural.js` para que os cartões só possam ser adicionados caso o usuário esteja logado. Crie um `if` em volta do código da função `adiciona` verificando se o usuário está logado ou não, caso contrário mostre para ele um popup dizendo que ele não está logado.

```
if(logado){
            cartoes.push(cartao)
            salvaCartoes()
            cartao.on("mudanca.**", render)
            cartao.on("remocao", ()=>{
                cartoes = cartoes.slice(0)
                cartoes.splice(cartoes.indexOf(cartao),1)
                render()
            })
            render()
            return true
} else {
alert("Você não está logado")
}
```

Teste! Tente adicionar cartões no mural e veja se você está sendo bloqueado. Caso não esteja, revisite os exemplos anteriores e veja se está tudo igual.

Agora em `js/login/LoginUsuario.js` precisamos alterar o valor da variável `logado` para `true` ou `false` quando o usuário fizer *login* ou *logout* respectivamente:

```
let logado = false
LoginUsuario_render({
    logado: logado
    ,onLogin: () => {
        logado = true
```

```
    }
    ,onLogout: () => {
        logado = false
    }
})
```

Teste! Tente adicionar cartões no mural e veja que você está sendo bloqueado. Faça login e veja que você consegue adicionar cartões.

Por enquanto, o usuário precisa fazer login toda vez que recarrega a página. Podemos melhorar usando o `localStorage` para armazenar os dados do usuário no navegador. Quando ele fizer *login*, podemos salvar uma entrada `logado` com o valor `true` e quando fizer logout, podemos salvar a mesma entrada `logado` com o valor `false`. Se utilizarmos essa entrada do `localStorage` sempre, podemos salvar se o usuário estava logado ou não. Não esqueça de converter o que vem do localStorage como String com o `JSON.parse`.

```
let logado = JSON.parse(localStorage.getItem("logado"))

LoginUsuario_render({
    logado: logado
    ,onLogin: () => {
        logado = true
        localStorage.setItem("logado", true)
    }
    ,onLogout: () => {
        logado = false
        localStorage.setItem("logado", false)
    }
})
```

Outra coisa que precisamos fazer é armazenar o nome do usuário que está logado e podemos obter esse nome como parâmetro da função de callback `onLogin`. Quando o usuário logar, criaremos uma entrada `nomeUsuario` no `localStorage` e quando deslogar, removeremos essa entrada.

```
let logado = JSON.parse(localStorage.getItem("logado"))

LoginUsuario_render({
    logado: logado
    ,usuario: localStorage.getItem("nomeUsuario")
    ,onLogin: (nomeUsuario) => {
        logado = true
        localStorage.setItem("logado", true)
        localStorage.setItem("nomeUsuario", nomeUsuario)
    }
    ,onLogout: () => {
        logado = false
        localStorage.setItem("logado", false)
        localStorage.removeItem("nomeUsuario")
    }
})
```

Dica: Como estamos alterando arquivos constantemente, deve-se atentar ao `Cache` do navegador. Por sorte o DevTools do Chrome pode nos facilitar o trabalho, limpando o cache toda vez que atualizamos a página. Para habilitar essa opção, vá na

aba `Network` e habilite a opção `Disable Cache` .