

## Exercício Avaliativo – RF-001 [Resolução]

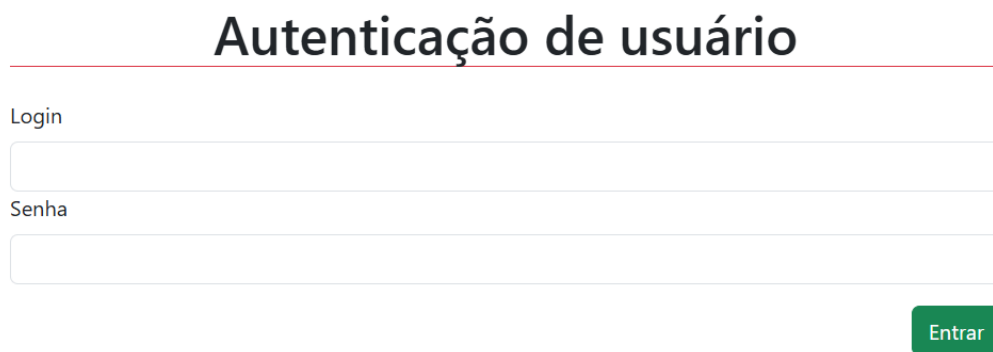
### [RF-001] Autenticação de Usuário

- O acesso à aplicação será restrito a usuários já cadastrados na plataforma OdinLine. Não é permitido realizar novos cadastros pela nova aplicação, pois todos os usuários são registrados exclusivamente na OdinLine.

### Proposta de resolução

É importante deixar claro que a aplicação a ser desenvolvida não deve permitir o cadastro de novos usuários. Ela deve apenas autenticar usuários já registrados na plataforma OdinLine, utilizando a API disponibilizada por essa plataforma.

#### 1º Passo: Criar o formulário de login



O formulário de autenticação de usuário apresenta o título "Autenticação de usuário" centralizado. Abaixo dele, há dois campos de entrada: "Login" e "Senha". O campo "Senha" possui uma máscara de pontos para ocultar o texto. À direita dos campos, há um botão verde com o texto "Entrar".

Figura 01: Formulário de autenticação

```
<body class="w-50 mx-auto">
  <div class="container mt-4">
    <h1 class="text-center border-bottom border-danger mb-4">Autenticação de usuário</h1>
    <form id="formulario" action="#" method="post">
      <fieldset>
        <div class="row">
          <div class="col-12">
            <label class="form-label" for="login">Login</label>
            <input class="form-control" id="login" name="login" type="text" required>
          </div>
        </div>
        <div class="row">
          <div class="col-12">
            <label class="form-label" for="senha">Senha</label>
            <input class="form-control" id="senha" name="senha" type="password" required>
          </div>
        </div>
        <div class="row">
          <div class="col-12 d-flex justify-content-end mt-3">
            <button class="btn btn-success" type="button" onclick="autenticar()">Entrar</button>
          </div>
        </div>
      </fieldset>
    </form>
  </div>
```

Arquivo: Trecho do arquivo index.html

### Observação:

- Foram utilizadas as classes do Bootstrap para criar o layout e a formatação do formulário.

## 2º Passo: Criar as regras de validação do formulário de login

```
$("#formulario").validate({
  {
    rules:{
      login:{
        required:true
      },
      senha:{
        required:true
      }
    },
    messages:{
      login:{
        required:"Campo obrigatório",
      },
      senha:{
        required:"Campo obrigatório"
      }
    }
  }
});
```

Arquivo: Trecho do arquivo index.js

## 3º Passo: Implementar uma função para autenticar os dados do usuário

```
22 async function autenticar() {
23   if ($("#formulario").valid()){
24     let login = $("#login").val();
25     let senha = $("#senha").val(); // Corrigido: estava pegando o valor de login novamente
26
27     try {
28       let resposta = await fetch(`https://api-odinline.odiloncorrea.com/usuario/${login}/${senha}/autenticar`);
29       let usuario = await resposta.json();
30
31       if (usuario.id > 0) {
32         localStorage.setItem('usuarioAutenticado', JSON.stringify(usuario));
33         window.location.href = "menu.html";
34       } else {
35         alert("Usuário ou senha inválidos.");
36       }
37     } catch (error) {
38       alert("Erro ao tentar autenticar.");
39     }
40   }
41 }
```

Arquivo: Trecho do arquivo index.js

### Observações:

- Linha 23
  - Verifica se o formulário com o ID **formulario** é válido. O método **.valid()** vem do plugin jQuery Validation e executa todas as regras de validação definidas para o formulário
- Linha 28
  - Template Literals (ou Template Strings) → são as crases ```, permitindo embutir variáveis no meio da string usando `${}`. Uma outra forma de escrever a mesma chamada e realizar a concatenação tradicional com `+`:

```
const resposta = await fetch("https://api-odinline.odiloncorrea.com/usuario/" + login + "/" + senha + "/autenticar");
```

- Linha 32
  - O nome e chave do usuário serão utilizados em outras páginas da aplicação. Sendo assim, é importante armazenar os dados do usuário no localStorage.