

Verificando se Login e senha estão corretos

```
26
27 def do_GET(self):
28     if self.path == '/login':
29         # Tenta abrir o arquivo login.html
30         try:
31             with open(os.path.join(os.getcwd(), 'login.html'), 'r') as login_file:
32                 content = login_file.read()
33                 self.send_response(200)
34                 self.send_header("Content-type", "text/html")
35                 self.end_headers()
36                 self.wfile.write(content.encode('utf-8'))
37         except FileNotFoundError:
38             self.send_error(404, "File not found")
39
40     elif self.path == '/login_failed':...
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57     else:
58         # Se não for a rota "/login", continua com o comportamento padrão
59         super().do_GET()
```

Verificando se Login e senha estão corretos


```
40 elif self.path == '/login_failed':
41     # Responde ao cliente com a mensagem de login/senha incorreta
42     self.send_response(200)
43     self.send_header("Content-type", "text/html; charset=utf-8")
44     self.end_headers()
45
46     # Lê o conteúdo da página login.html
47     with open(os.path.join(os.getcwd(), 'login.html'), 'r', encoding='utf-8') as login_
48         content = login_file.read()
49
50     # Adiciona a mensagem de erro no conteúdo da página
51     mensagem = "Login e/ou senha incorreta. Tente novamente."
52     content = content.replace('<!-- Mensagem de erro será inserida aqui -->',
53                             f'<div class="error-message">{mensagem}</div>')
54     # Envia o conteúdo modificado para o cliente
55     self.wfile.write(content.encode('utf-8'))
56
57 else:
```

## Construindo a rota /cadastro em do\_GET

```
| elif self.path.startswith('/cadastro'):  
  
    # Extraindo os parâmetros da URL  
    query_params = parse_qs(urlparse(self.path).query)  
    login = query_params.get('login', [''])[0]  
    senha = query_params.get('senha', [''])[0]  
  
    # Mensagem de boas-vindas  
    welcome_message = f"Olá {login}, seja bem-vindo! Percebemos que você é novo por aqui. Complete o seu cadastro."  
  
    # Responde ao cliente com a página de cadastro  
    self.send_response(200)  
    self.send_header("Content-type", "text/html; charset=utf-8")  
    self.end_headers()  
  
    # Lê o conteúdo da página cadastro.html  
    with open(os.path.join(os.getcwd(), 'cadastro.html'), 'r', encoding='utf-8') as cadastro_file:  
        content = cadastro_file.read()  
  
    # Substitui os marcadores de posição pelos valores correspondentes  
    content = content.replace('{login}', login)  
    content = content.replace('{senha}', senha)  
    content = content.replace('{welcome_message}', welcome_message)  
  
    # Envia o conteúdo modificado para o cliente  
    self.wfile.write(content.encode('utf-8'))  
  
    return # Adicionando um return para evitar a execução do restante do código  
  
else:  
    # Se não for a rota "/login", continua com o comportamento padrão  
    super().do_GET()
```

Criando arquivo com 3 colunas – login;senha;nome – Página HTML de Cadastro do novo campo “Nome”

```
53 <body>
54
55 <div class="login-container">
56   <div class="error-message">
57     <!-- Mensagem de erro será inserida aqui -->
58
59   </div>
60   <div class="welcome-message">
61     {welcome_message}
62   </div>
63   <h2>Cadastro</h2>
64   <form action="/confirmar_cadastro" method="post">
65     <label for="nome">Nome:</label>
66     <input type="text" id="nome" name="nome" required>
67
68     <label for="email">Email:</label>
69     <input type="email" id="login" name="login" value="{login}" required>
70
71     <label for="senha">Confirme a Senha:</label>
72     <input type="password" id="senha" name="senha" required>
73
74     <button type="submit">Confirmar Cadastro</button>
75   </form>
76 </div>
77
78 </body>
79 </html>
```



Criando arquivo com 3 colunas – login;senha;nome em do\_POST

```
157     if self.usuario_existente(login, senha):
158         # Responde ao cliente indicando que o usuário logou com sucesso
159         self.send_response(200)
160         self.send_header("Content-type", "text/html; charset=utf-8")
161         self.end_headers()
162         mensagem = f"Usuário {login} logado com sucesso!!!"
163         self.wfile.write(mensagem.encode('utf-8'))
164     else:
165         # Verifica se o login já existe no arquivo
166         if any(line.startswith(f"{login};") for line in open('dados_login.txt', 'r', encoding='utf-8')):
167             # Redireciona o cliente para a rota "/login_failed"
168             self.send_response(302)
169             self.send_header('Location', '/login_failed')
170             self.end_headers()
171             return # Adicionando um return para evitar a execução do restante do código
172         else:
173             # Adiciona o novo usuário ao arquivo
174             with open('dados_login.txt', 'a', encoding='utf-8') as file:
175                 file.write(f"{login};{senha};" + "none" + "\n")
176
177             # Redireciona o cliente para a rota "/cadastro" com os dados de login e senha
178             self.send_response(302)
179             self.send_header('Location', f'/cadastro?login={login}&senha={senha}')
180             self.end_headers()
181
182             return # Adicionando um return para evitar a execução do restante do código
```

Este if self.usuario\_existente está dentro do if self.path == '/enviar\_login'

Criando arquivo com 3 colunas – login;senha;nome – Nova rota `/confirmar_cadastro` em `do_POST`

```
elif self.path.startswith('/confirmar_cadastro'):  
  
    # Obtém o comprimento do corpo da requisição  
    content_length = int(self.headers['Content-Length'])  
    # Lê o corpo da requisição  
    body = self.rfile.read(content_length).decode('utf-8')  
    # Parseia os dados do formulário  
    form_data = parse_qs(body, keep_blank_values=True)  
  
    #query_params = parse_qs(urlparse(self.path).query)  
    login = form_data.get('login', [''])[0]  
    senha = form_data.get('senha', [''])[0]  
    nome = form_data.get('nome', [''])[0]  
  
    print("nome: " + nome)  
  
    # Verifica se o usuário já existe  
  
    if self.usuario_existente(login, senha):  
  
        # Atualiza o arquivo com o nome, se a senha estiver correta
```

```
def do_POST(self):  
    # Verifica se a rota é "/enviar_login"  
    if self.path == '/enviar_login':...  
  
    elif self.path.startswith('/confirmar_cadastro'):  
  
        # Obtém o comprimento do corpo da requisição  
        content_length = int(self.headers['Content-Length'])
```

Continua no próximo slide



Criando arquivo com 3 colunas – login;senha;nome – Nova rota `/confirmar_cadastro` em `do_POST`

```
if self.usuario_existente(login, senha):
```

← Continuação do slide anterior

```
# Atualiza o arquivo com o nome, se a senha estiver correta
```

```
with open('dados_login.txt', 'r', encoding='utf-8') as file:  
    lines = file.readlines()
```

```
with open('dados_login.txt', 'w', encoding='utf-8') as file:  
    for line in lines:  
        stored_login, stored_senha, stored_nome = line.strip().split(';')  
        if login == stored_login and senha == stored_senha:  
            line = f"{login};{senha};{nome}\n"  
            file.write(line)
```

```
# Redireciona o cliente para onde desejar após a confirmação
```

```
self.send_response(302)  
self.send_header("Content-type", "text/html; charset=utf-8")  
self.end_headers()  
self.wfile.write("Registro Recebido com Sucesso!!!".encode('utf-8'))
```

```
else:
```

```
# Se o usuário não existe ou a senha está incorreta, redireciona para outra página  
self.remove_ultima_linha('dados_login.txt')  
self.send_response(302)  
self.send_header("Content-type", "text/html; charset=utf-8")  
self.end_headers()  
self.wfile.write("A senha não confere. Retome o procedimento!".encode('utf-8'))
```

Nova função **usuario\_existente** com 3 colunas – login;senha;nome

```
2 usages
def usuario_existente(self, login, senha):
    # Verifica se o login já existe no arquivo

    with open('dados_login.txt', 'r', encoding='utf-8') as file:
        for line in file:
            if line.strip():
                stored_login, stored_senha, stored_nome = line.strip().split(';')
                if login == stored_login:
                    print("chequei aqui significando que localizei o login informado")
                    print("senha: " + senha)
                    print(" senha_armazenada: " + senha)
                    return senha == stored_senha

    return False
```



Nova função `remover_ultima_linha` com 3 colunas – login;senha;nome

```
def remover_ultima_linha(self, arquivo):  
    print("Vou excluir ultima linha")  
    with open(arquivo, 'r', encoding='utf-8') as file:  
        lines = file.readlines()  
    with open(arquivo, 'w', encoding='utf-8') as file:  
        file.writelines(lines[:-1])
```