

# BackEnd

# BackEnd

## Django

Bom já temos:

Um servidor funcional;

Temos uma index;

Temos uma base de dados;

Temos dados iniciais;

Agora vamos fazer a página de login funcional.

# BackEnd

## URLS do App\_Escola

```
1 from django.urls import path, include
2 from . import views
3 urlpatterns = [
4     path('', views.abre_index, name='abre_index'),
5     path('enviar_login', views.enviar_login, name='enviar_login'),
6     path('confirmar_cadastro', views.confirmar_cadastro, name='confirmar_cadastro')
7 ]
8
```

# BackEnd

```
51     # Fim da População da tabela Atividade
52
53     print("Populei")
54
55     1 usage
56     def abre_index(request):
57         #return render(request, 'Index.html')
58         # mensagem = "OLÁ TURMA, MUITO BOM DIA!"
59         # return HttpResponse(mensagem)
60
61         # query set Tipos de Look Up
62         # nome__exact='SS' - tem que ser exatamente igual
63         # nome__contains='H' - contém o H maiusculo
64         # nome__icontains='H' - ignora se maiúsculo ou minúsculo
65         # nome__startswith='M' - traz o que começa com a letra M ou sequencia de letras
66         # nome__istartswith='M' - traz o que começa com a letra M ignorando se maiusculo ou minusculo u sequencia de letras
67         # nome__endswith='a' - traz o que termina com a letra a minusculo ou sequencia de letras
68         # nome__iendswith='a' - traz o que termina com a letra a ignorando maiúsculo ou minusculo
69         # nome__in=['Michael', 'Obama']) traz somente os nome que estão na lista
70         # Pode ser feito uma composição 'and' utilizando , (virgula entre os campos) ou 'or' utilizando | (pipe entre os campos)
71
72         dado_pesquisa = 'Obama'
```

## Views.py App\_Escola

Quando desejamos fazer uma comparação, temos algumas possibilidades que nos ajudam a fazer esse comparativo, considerando ou desprezando algumas características, por exemplo:

Se usarmos o `Extract` o texto comparado deve ser exatamente igual.

Se usarmos o `contains = 'H'`, contém o H (maiúsculo).

Se usarmos o `endwith = 'a'`, o texto comparado deve terminar com a

# BackEnd Views.py App\_Escola

```
# nome__endswith='a' - traz o que termina com a letra a minuscúlo ou sequência de letras
# nome__iendswith='a' - traz o que termina com a letra a ignorando maiúsculo ou minuscúlo
# nome__in=['Michael', 'Obama']) traz somente os nome que estão na lista
# Pode ser feito uma composição 'and' utilizando , (virgula entre os campos) ou 'or' utilizando | (pipe entre os campos)

dado_pesquisa = 'Obama'

verifica_populado = Professor.objects.filter(nome__icontains=dado_pesquisa)
#verifica_populado = Professor.objects.filter(nome='Prof. Barak Obama')

if len(verifica_populado) == 0:
    print ("Não está Populado")
    initial_population()
else:
    print ("Achei Obama", verifica_populado)
```

Colocando um dos valores cadastrados em uma variável, posso fazer a verificação se os dados estão devidamente populados.

# BackEnd

```
def enviar_login(request):  
  
    if (request.method == 'POST'):  
        email = request.POST.get('email')  
        senha = request.POST.get('senha')  
        senha_criptografada = sha256(senha.encode()).hexdigest()  
        dados_professor = Professor.objects.filter(email=email).values('nome', 'senha', 'id')  
        print("Dados do Professor ", dados_professor)  
        #if len(dados_professor) > 0:  
        if dados_professor:  
            senha = dados_professor[0]  
            senha = senha['senha']  
            usuario_logado = dados_professor[0]  
            usuario_logado = usuario_logado['nome']  
            if (senha == senha_criptografada):  
                #Se logou corretamente, traz as turmas do professor  
                #Para isso instanciamos o model turmas do professor  
                id_logado = dados_professor[0]  
                id_logado = id_logado['id']  
                turmas_do_professor = Turma.objects.filter(id_professor=id_logado)  
                print("Turma do Professor ", turmas_do_professor)  
                return render(request, 'Cons_Turma_Lista.html', {'usuario_logado': usuario_logado,  
                                                                    'turmas_do_professor': turmas_do_professor,  
                                                                    'id_logado': id_logado})  
            else:  
                messages.info(request, 'Usuario ou senha incorretos. Tente novamente.')  
                return render(request, 'login.html')  
  
        messages.info(request, "Olá " + email + ", seja bem-vindo! Percebemos que você é novo por aqui. Complete o seu cadastro.")  
        return render(request, 'cadastro.html', {'login': email})
```

# BackEnd

## Views.py App\_Escola

Crio uma função chamada enviar\_login que aceita um request.

Caso a solicitação seja feita pelo método POST, capturo os dados que estão nos campos e-mail e senha.

A senha, deve ser convertido para um valor criptografado usando o sha256. Fazemos uma consulta no banco de dados na tabela Professor considerando como filtro, o email, e resgato o nome, senha e o ID.

Caso haja um retorno dessa consulta

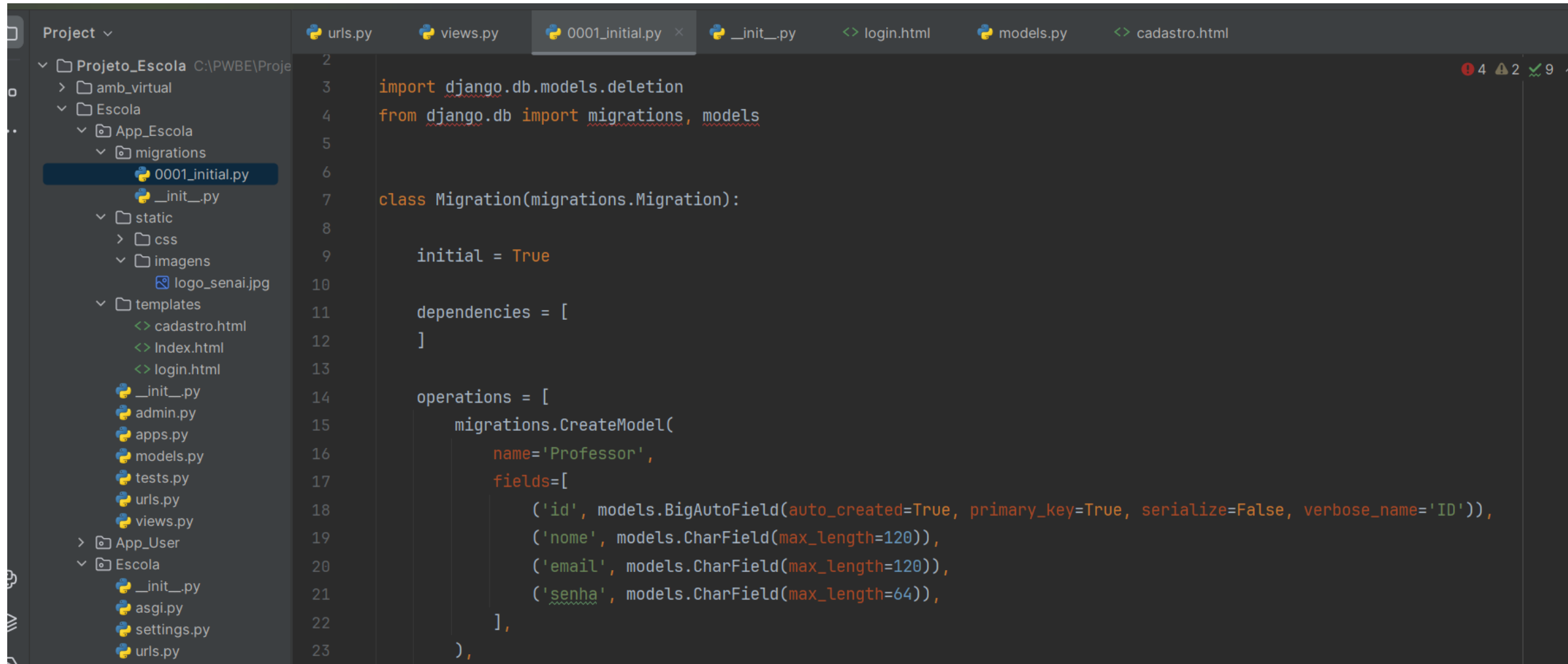
Em senha recupero o valor correspondente a senha.

E em usuário logado o nome do professor correspondente.



# BackEnd

## Migrations



The screenshot shows a code editor with a project explorer on the left and a code editor on the right. The project explorer shows a project named 'Projeto\_Escola' with a folder 'Escola' containing a 'migrations' folder. The 'migrations' folder is expanded, showing a file '0001\_initial.py' which is selected. The code editor shows the content of '0001\_initial.py'.

```
2
3 import django.db.models.deletion
4 from django.db import migrations, models
5
6
7 class Migration(migrations.Migration):
8
9     initial = True
10
11     dependencies = [
12     ]
13
14     operations = [
15         migrations.CreateModel(
16             name='Professor',
17             fields=[
18                 ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
19                 ('nome', models.CharField(max_length=120)),
20                 ('email', models.CharField(max_length=120)),
21                 ('senha', models.CharField(max_length=64)),
22             ],
23         ),
```

# BackEnd

## Views.py App\_Escola

Caso a senha recuperada seja igual a senha criptografada informada no formulário, então exibo a mensagem: Bem vindo e em mensagem coloco: "Ola Professor" o email dele seja bem vindo Envio essa mensagem pelo response como retorno.

Caso o e-mail exista no banco de dados, mas a senha esteja diferente, caio no else, que retorna que os dados de acesso não estão corretos, tendo como resposta a renderização da página de login.

E caso esse email não exista na base de dados ele é encaminhado para tela de cadastro de novos professores.

```
53 <body>
54
55 <div class="login-container">
56   <div class="error-message">
57     <!-- Mensagem de erro será inserida aqui -->
58   </div>
59   <h2>Login</h2>
60   {% if messages %}
61     <ul class="mensagem_do_servidor" style="...">
62       {% for mensagem in messages %}
63         <li style="..." {% if mensagem.tags %}class="alert alert-{{message.tags}}"{% endif %}>{{mensagem}}</li>
64       {% endfor %}
65     </ul>
66   {% endif %}
67   <form action="{% url 'enviar_login' %}" method="post">{% csrf_token %}
68
69   <label for="email">Email:</label>
70   <input type="email" id="email" name="email" required>
71
72   <label for="senha">Senha:</label>
73   <input type="password" id="senha" name="senha" required>
74
75   <button type="submit">Enviar</button>
76 </form>
77 </div>
```

```
{% if messages %}
    <ul class="mensagem_do_servidor" style="...">
        {% for mensagem in messages %}
            <li style="..." {% if mensagem.tags %}class="alert alert-{{mensagem.tags}}"{% endif %}>{{mensagem}}</li>
        {% endfor %}
    </ul>
{% endif %}
```

Caso existam mensagens de retorno de validações no formulário, conseguimos fazer uma lista com os retornos desta forma, como por exemplo credenciais de acesso erradas.

```
{% endif %}
<form action="{% url 'enviar_login' %}" method="post">{% csrf_token %}
```

Para enviar a requisição, ao trecho de código adequado, usamos a sintaxe {% %}

```
usage
def confirmar_cadastro(request):
    if (request.method == 'POST'):
        nome = request.POST.get('nome')
        email = request.POST.get('login')
        senha = request.POST.get('senha')
        senha_criptografada = sha256(senha.encode()).hexdigest()

        grava_professor = Professor(
            nome=nome,
            email=email,
            senha=senha_criptografada
        )
        grava_professor.save()

        mensagem = "OLÁ PROFESSOR " + nome + ", SEJA BEM VINDO!"
        return HttpResponse(mensagem)
```

Quando o email informado não é encontrado na base de dados, o professor é direcionado para outra página, a de Cadastro.

Nesta página de Cadastro, consigo fazer a captura dos dados:

Nome;

Email;

Senha, a qual eu faço sua criptografia;

E em seguida faço seu armazenamento, instanciando a classe Professor. Concluído isso é exibida a mensagem de Olá professor seja bem vindo.

```
</style>
</head>
<body>

<div class="login-container">
  <div class="error-message">
    {{ error_message }}
  </div>
  <div class="welcome-message">
    {{welcome_message}}
  </div>
  <h2>Cadastro</h2>
  <form action="/confirmar_cadastro" method="post">
    {% csrf_token %}
    <label for="nome">Nome:</label>
    <input type="text" id="nome" name="nome" required>

    <label for="email">Email:</label>
    <input type="email" id="login" name="login" value="{{login}}" required>

    <label for="senha">Confirme a Senha:</label>
    <input type="password" id="senha" name="senha" required>

    <button type="submit">Confirmar Cadastro</button>
  </form>
</div>

</body>
</html>
```