PYSTACK WEEK 3.0 | AULA 01

Recomendamos acessar o material direto pelo Notion, segue o link:

```
https://grizzly-amaranthus-f6a.notion.site/PYSTACK-WEEK-3-0-AULA-01-5a288e9287584eceb44482dc844f2244
```

Primeiro vamos criar e ativar o ambiente virtual:

```
# Criar
  # Linux
    python3 -m venv venv
  # Windows
    python -m venv venv

#Ativar
  # Linux
    source venv/bin/activate
  # Windows
    venv/Scripts/Activate

# Caso algum comando retorne um erro de permissão execute o código e tente novamente:

  Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned
```

Instale as bibliotecas:

```
pip install django
pip install pillow
```

Inicio um projeto Django

```
django-admin startproject freelaway .
```

Configure os arquivos estáticos:

```
STATIC_URL = '/static/'
STATICFILES_DIRS = (os.path.join(BASE_DIR, 'templates/static'),)
STATIC_ROOT = os.path.join('static')

MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
MEDIA_URL = '/media/'
```

Crie um app autenticacão:

```
python3 manage.py startapp autenticacao
```

Crie uma URL para nosso novo APP:

```
path('auth/', include('autenticacao.urls')),
```

Crie uma URL para login e cadastro:

```
path('cadastro/', views.cadastro, name='cadastro'),
path('logar/', views.logar, name='logar'),
```

Crie as views:

```
def cadastro(request):
    return render(request, 'cadastro.html')

def logar(request):
    pass
```

Crie o arquivo cadastro.html

Crie um arquivo base.html

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    {% block 'head' %}

    {% endblock %}
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css">
    <title>{% block 'title' %}{% endblock %}</title>
  </head>
  <body class="fundo">
    {% block 'body' %}{% endblock %}

    <script src="https://cdn.jsdelivr.net/npm/jquery@3.5.1/dist/jquery.slim.min.js" ></script>
```

```
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/js/bootstrap.bundle.min.js"></script>

  </body>
</html>
```

Desenvolva o cadastro.html

```
{% extends 'base.html' %}
{% load static %}

{% block 'head' %}
<link rel="stylesheet" href="{% static 'autenticacao/css/cadastro.css' %}">
{% endblock %}

{% block 'body' %}
    <div class="box">
        <form>

            <h1 class="titulo font-degrade borda-bottom-degrade">INSCREVA-SE</h2>
            <br>
            <span class="span-descricao">Nome de usuário:</span>
            <input type="text" class="form-control input-cadastro" name="username">
            <br>
            <span class="span-descricao">Senha:</span>
            <input type="password" class="form-control input-cadastro" name="password">
            <br>
            <span class="span-descricao">Confirmar senha:</span>
            <input type="password" class="form-control input-cadastro" name="confirm-password">
            <br>
            <input  type="submit" value="CADASTRAR" class="btn btn-success btn-lg">
            <a href="#" class="btn btn-primary btn-lg">LOGIN</a>
        </form>

    </div>
{% endblock %}
```

Agora crie o arquivo cadastro.css

```
.fundo{

    background-color: #222;
    color: white;

}

.box{

    background-color: #444;
    width: 500px;
    padding: 40px;
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%,-50%);
    box-shadow: 0 0px 20px 0 rgba( 255, 207, 0, 0.3 );


}
```

```
.span-descricao{
    font-family:'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande', 'Lucida Sans Unicode', Geneva, Verdana, sans-serif;
    font-weight: bold;
}

.input-cadastro{
    margin-top: 15px;

}

.font-degrade{
    background-image: linear-gradient(90deg, rgba(255,207,0,1) 0%, rgba(231,71,220,1) 100%);
    background-clip: text;
    -webkit-background-clip: text;
    color: transparent;
    padding-bottom: 10px;
}

.titulo{
    text-align: center;
    font-family: Arial, Helvetica, sans-serif;
    font-weight: bold;
    font-size: 52px;
}

.borda-bottom-degrade{

    border-bottom: 1px solid transparent;
    border-image: linear-gradient(90deg,rgba(231,71,220,1) 0%,rgba(255,207,0,1) 100%);
    border-image-slice: 1;


}
```

Altere a view para diferenciar POST e GET:

```
def cadastro(request):
    if request.method == "GET":
        return render(request, 'cadastro.html')
    elif request.method == "POST":
        username = request.POST.get('username')
        senha = request.POST.get('password')
        confirmar_senha = request.POST.get('confirm-password')

        return HttpResponse(confirmar_senha)
```

Faça as validações:

```
if not senha == confirmar_senha:
    return redirect('/auth/cadastro')

if len(username.strip()) == 0 or len(senha.strip()) == 0:
    return redirect('/auth/cadastro')

user = User.objects.filter(username=username)

if user.exists():
    return redirect('/auth/cadastro')
```

Salve os dados do novo usuário no banco:

```
try:
    user = User.objects.create_user(username=username,
                                    password=senha)
    user.save()
    return redirect('/auth/logar')
except:
    return redirect('/auth/cadastro')
```

Redireciona para a página de login:

```
<a href="{% url 'logar' %}" class="btn btn-primary btn-lg">LOGIN</a>
```

Vamos agora preparar as mensagens de erro:

Primeiro fazemos as configurações no settings.py

```
from django.contrib.messages import constants

MESSAGE_TAGS = {
    constants.DEBUG: 'alert-primary',
    constants.ERROR: 'alert-danger',
    constants.SUCCESS: 'alert-success',
    constants.INFO: 'alert-info',
    constants.WARNING: 'alert-warning',
}
```

Adicione as mensagens onde achar necessário:

```
from django.contrib import messages
from django.contrib.messages import constants

def cadastro(request):
    if request.method == "GET":
        return render(request, 'cadastro.html')
    elif request.method == "POST":
        username = request.POST.get('username')
        senha = request.POST.get('password')
        confirmar_senha = request.POST.get('confirm-password')

        if not senha == confirmar_senha:
            messages.add_message(request, constants.ERROR, 'As senhas não coincidem')
            return redirect('/auth/cadastro')

        if len(username.strip()) == 0 or len(senha.strip()) == 0:
            messages.add_message(request, constants.ERROR, 'Preencha todos os campos')
            return redirect('/auth/cadastro')

        user = User.objects.filter(username=username)

        if user.exists():
            messages.add_message(request, constants.ERROR, 'Já existe um usário com esse username')
```

```
            return redirect('/auth/cadastro')

        try:
            user = User.objects.create_user(username=username,
                                            password=senha)
            user.save()
            messages.add_message(request, constants.SUCCESS, 'Usuário criado com sucesso')
            return redirect('/auth/logar')
        except:
            messages.add_message(request, constants.ERROR, 'Erro interno do sistema')
            return redirect('/auth/cadastro')
```

Exiba as mensagens no template HTML:

```
{% if messages %}
        {% for message in messages %}
            <div class="alert {{message.tags}}">
                {{message}}
            </div>
        {% endfor %}
{% endif %}
```

Agora vamos para o login

Crie o arquivo login.html:

```
{% extends 'base.html' %}
{% load static %}

{% block 'head' %}
<link rel="stylesheet" href="{% static 'autenticacao/css/cadastro.css' %}">
{% endblock %}

{% block 'body' %}
    <div class="box">
        <form action="{% url 'logar' %}" method="post">{% csrf_token %}

            <h1 class="titulo font-degrade borda-bottom-degrade">LOGAR</h2>
            <br>
            <span class="span-descricao">Nome de usuário:</span>
            <input type="text" class="form-control input-cadastro" name="username">
            <br>
            <span class="span-descricao">Senha:</span>
            <input type="password" class="form-control input-cadastro" name="password">
            <br>
            <input  type="submit" value="LOGAR" class="btn btn-success btn-lg">
            <a href="{% url 'logar' %}" class="btn btn-primary btn-lg">CADASTRO</a>
        </form>

    </div>
{% endblock %}
```

Crie a view logar para realizar a autenticação do usuário:

```
def logar(request):
    if request.method == "GET":
        return render(request, 'logar.html')
    elif request.method == "POST":
        username = request.POST.get('username')
        senha = request.POST.get('password')

        usuario = auth.authenticate(username=username, password=senha)

        if not usuario:
            messages.add_message(request, constants.ERROR, 'Username ou senha inválidos')
            return redirect('/auth/logar')
        else:
            auth.login(request, usuario)
            return redirect('/')
```

Por fim é só verificar se o usuários já está logado antes de exibir logar e cadastro:

```
if request.user.is_authenticated:
        return redirect('/')
```

Para realizar o logout do usuário cria uma URL para sair:

```
path('sair/', views.sair, name="sair")
```

Crie a view para sair:

```
def sair(request):
    auth.logout(request)
    return redirect('/auth/logar')
```

E com isso finalizamos nossa primeira aula da PSW 3.0