1. Instalação e Configuração Inicial

Primeiro, instale o pacote e adicione as configurações necessárias no seu `settings.py`.

pip install django-allauth

No **settings.py**, adicione os apps necessários:

```
INSTALLED_APPS = [
    ...
    'django.contrib.sites',
    'allauth',
    'allauth.account',
    'allauth.socialaccount',
    'allauth.socialaccount.providers.google', # Provedor Google
]

AUTHENTICATION_BACKENDS = [
    'django.contrib.auth.backends.ModelBackend',
    'allauth.account.auth_backends.AuthenticationBackend',
]

SITE_ID = 1
LOGIN_REDIRECT_URL = '/' # Para onde o user vai após o login
```

2. Configuração no Google Cloud Console

Para o Google "falar" com o seu site, você precisa de credenciais.

1. Vá ao Google Cloud Console.
2. Crie um novo projeto.
3. Em **Tela de consentimento OAuth**, configure como "Externo" e preencha os dados básicos.
4. Em **Credenciais**, clique em **Criar Credenciais > ID do cliente OAuth**.
    - **Tipo de aplicativo:** Aplicativo Web.
    - **Origens JavaScript autorizadas:** `http://127.0.0.1:8000` (para teste local).
    - **URIs de redirecionamento autorizados:** `http://127.0.0.1:8000/accounts/google/login/callback/`.
5. Copie o **Client ID** e o **Secret Key**.

3. Registrar o Provedor no Admin do Django

Com o servidor rodando, vá ao `/admin`:

1. Em **Sites**, certifique-se de que o domínio é `127.0.0.1:8000`.
2. Em **Social Applications**, adicione uma nova:
    - **Provider:** Google.
    - **Name:** Google Login.
    - **Client id:** (Cole o do Google Console).
    - **Secret key:** (Cole o do Google Console).
    - **Sites:** Mova o seu site para a coluna da direita.

4. Adicionar o botão de login no Template

O `allauth` já fornece as URLs. No seu HTML, basta carregar a tag e criar o link:

```
{% load socialaccount %}

<h1>Bem-vindo!</h1>
{% if user.is_authenticated %}
  <p>Olá, {{ user.username }}!</p>
  <a href="{% url 'account_logout' %}">Sair</a>
{% else %}
  <a href="{% provider_login_url 'google' %}">Entrar com o Google</a>
{% endif %}
```

5. Mapear dados do Google para o Usuário

O `allauth` faz o básico sozinho (email e nome). Se você precisar de dados extras ou lógica customizada ao criar o usuário, use um **Adapter**.

Crie um arquivo `adapter.py` no seu app:

```
from allauth.socialaccount.adapter import DefaultSocialAccountAdapter

class MySocialAccountAdapter(DefaultSocialAccountAdapter):
    def pre_social_login(self, request, sociallogin):
        # Lógica executada antes do login ser concluído
        user = sociallogin.user
        if not user.id:
            # Aqui você pode customizar o mapeamento
            # sociallogin.account.extra_data contém todo o JSON do Google
```

```
        pass
```

E registre no `settings.py`: `SOCIALACCOUNT_ADAPTER = 'seu_app.adapter.MySocialAccountAdapter'`

Dica de ouro

Sempre use `SOCIALACCOUNT_QUERY_EMAIL = True` no seu `settings.py` para garantir que o Django peça e valide o e-mail do usuário, evitando contas duplicadas ou sem contato.

==========

Pular essa página (Login Direto) Se você quer que, ao clicar em "Entrar com Google", o usuário vá direto para a tela do Google sem passar por esse "Continue", adicione isso ao seu `settings.py`:

```
# Pula a página de confirmação intermediária do Allauth
SOCIALACCOUNT_LOGIN_ON_GET = True
```

Atenção: Usar `LOGIN_ON_GET` pula a confirmação, o que é melhor para a experiência do usuário (UX), mas certifique-se de que seu botão de login seja um link simples ou um formulário POST básico.

==========

SE APARECER O SEGUINTE AVISO VIA TERMINAL QUANDO INICIAR O SERVIDOR⚠️:

```
WARNINGS:
?: settings.ACCOUNT_AUTHENTICATION_METHOD is deprecated, use:
settings.ACCOUNT_LOGIN_METHODS = {'email'}
?: settings.ACCOUNT_EMAIL_REQUIRED is deprecated, use:
settings.ACCOUNT_SIGNUP_FIELDS = ['email*', 'password1*', 'password2*']
?: settings.ACCOUNT_USERNAME_REQUIRED is deprecated, use:
settings.ACCOUNT_SIGNUP_FIELDS = ['email*', 'password1*', 'password2*']
```

USE OS CÓDIGOS ABAIXO PARA RESOLVER:

 # Mudar as configurações do settings.py para o django-allauth:

Para limpar seu terminal e deixar o código atualizado, vá no seu arquivo settings.py e faça as substituições sugeridas pelos avisos:

1. Troque:

   ```
   ACCOUNT_AUTHENTICATION_METHOD = 'email'  # (ou 'username_email')
   ```

   Por:

   ```
   ACCOUNT_LOGIN_METHODS = {'email'} # Se você usa login por email
   # ou {'email', 'username'} se permite os dois
   ```

   Troque:

   ```
   ACCOUNT_EMAIL_REQUIRED = True
   ACCOUNT_USERNAME_REQUIRED = False # (ou True)
   ```

   Por:

   ```
   # O asterisco * indica que o campo é obrigatório
   ACCOUNT_SIGNUP_FIELDS = ['email*', 'password1*', 'password2*']
   ```