Contoh Auth Django Apps dengan Keycloak / RH-SSO

1. Django Project

Di contoh ini, aplikasi Django yang kita buat berjalan di http://localhost:8000

1. Django / Python Library

Menambahkan library django-allauth

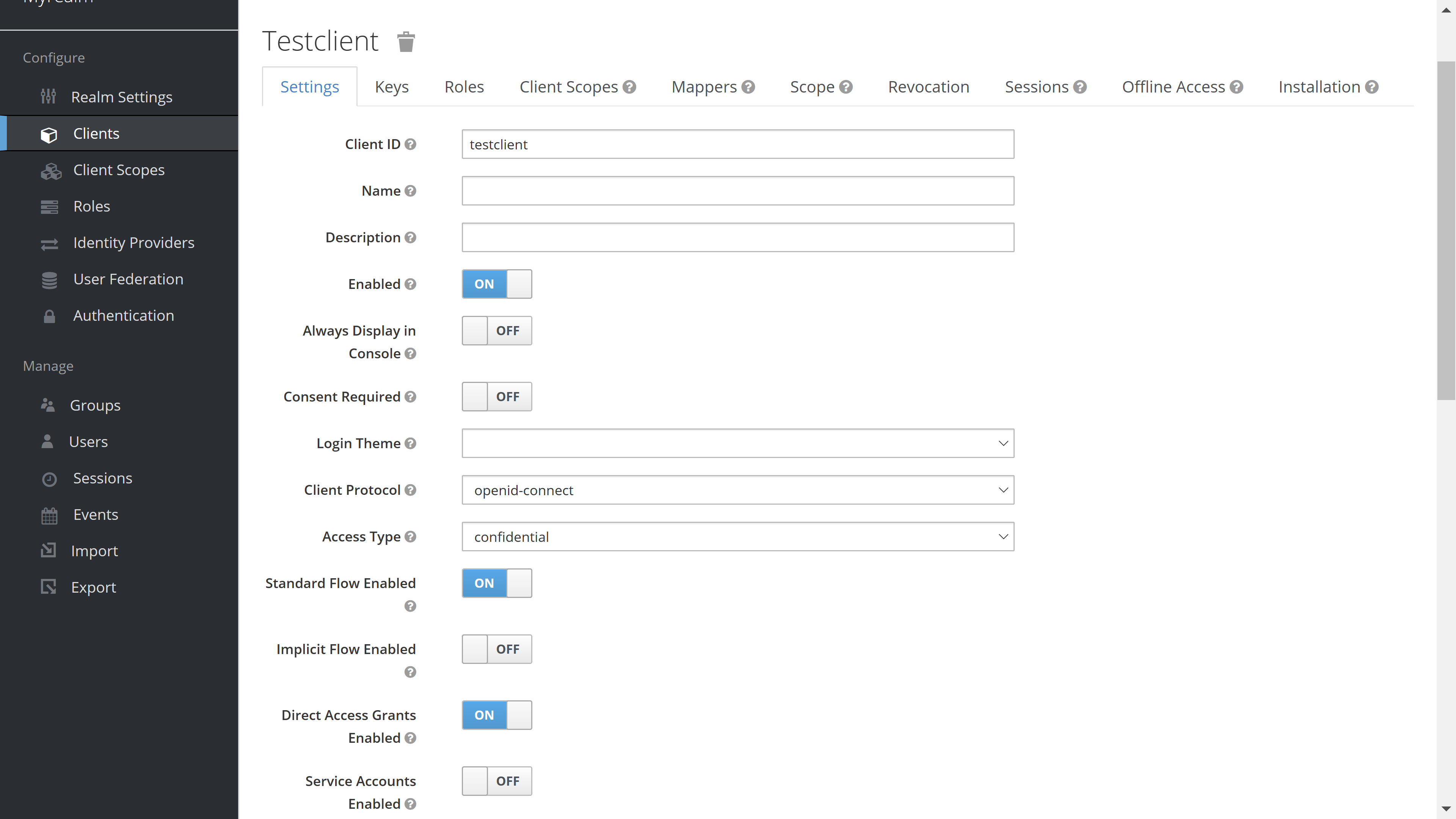
pip install django-allauth

1. Keycloak / RH-SSO Configuration

Buat realm dan client, misal:

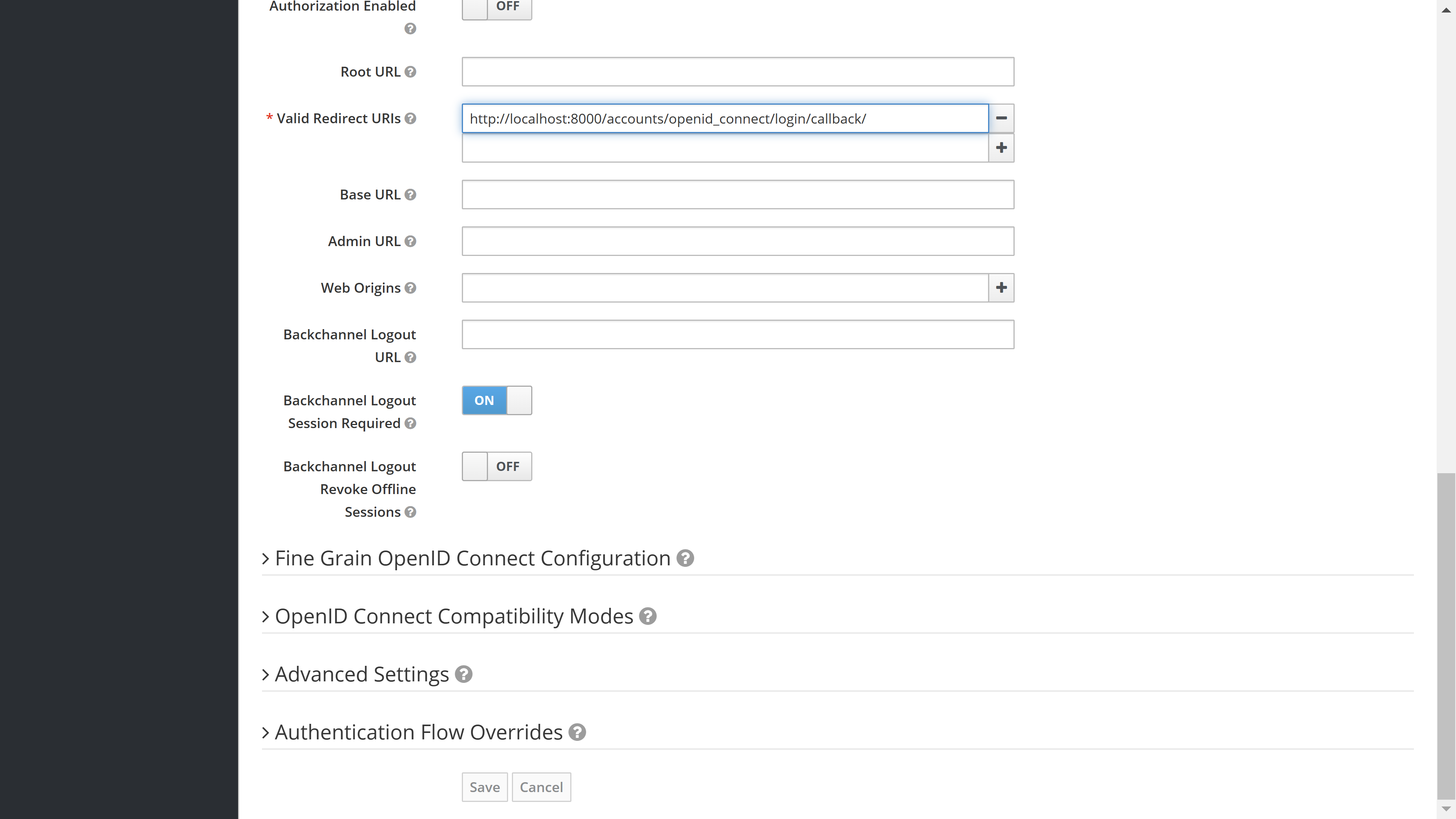
Realm: myrealm

Client: testclient

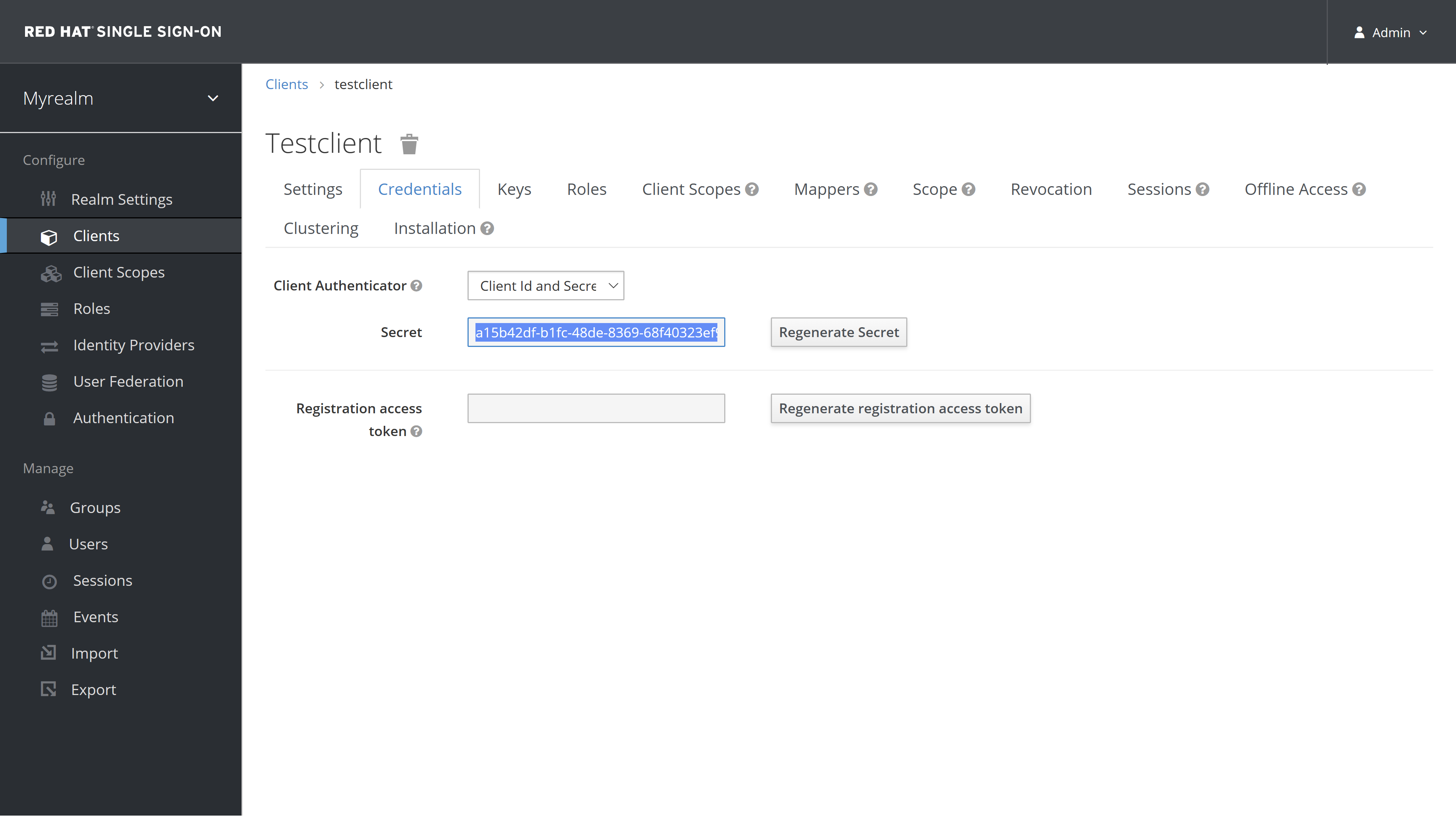


Set Access Type jadi “confidential”

Kemudian tambahkan Valid Redirect Urls: “http://localhost:8000/accounts/openid\_connect/login/callback/”



Kemudian ke tab “Credentials”, silahkan copy Client Secret



1. settings.py Configuration

Ikuti petunjuk di dokumentasi django-allauth

<https://django-allauth.readthedocs.io/en/latest/installation.html>

Tambahkan di bagian INSTALLED\_APPS

INSTALLED\_APPS = [

……

    'allauth',

    'allauth.account',

    'allauth.socialaccount',

    'allauth.socialaccount.providers.keycloak',

]

Tambahkan di bagian AUTHENTICATION\_BACKENDS

AUTHENTICATION\_BACKENDS = [

    # Needed to login by username in Django admin, regardless of `allauth`

    'django.contrib.auth.backends.ModelBackend',

    # `allauth` specific authentication methods, such as login by e-mail

    'allauth.account.auth\_backends.AuthenticationBackend',

]

Tambahkan di bagian SOCIALACCOUNT\_PROVIDERS

SOCIALACCOUNT\_PROVIDERS = { # changed

        'keycloak': {

        'KEYCLOAK\_URL': 'http://localhost:8080/auth',

        'KEYCLOAK\_REALM': 'myrealm',

    },

}

Tambahkan juga SITE\_ID, ini disesuaikan dengan konfigurasi di bagian Django-Admin di bawah nanti

SITE\_ID = 2

Tambahkan juga LOGIN\_REDIRECT\_URL, ini endpoint yang akan dipanggil setelah authentikasi sukses

LOGIN\_REDIRECT\_URL = '/hello'

1. urls.py Configuration

Ikuti petunjuk di dokumentasi django-allauth

<https://django-allauth.readthedocs.io/en/latest/installation.html>

from django.contrib import admin

from django.urls import path, include

from . import hello

urlpatterns = [

    path('admin/', admin.site.urls),

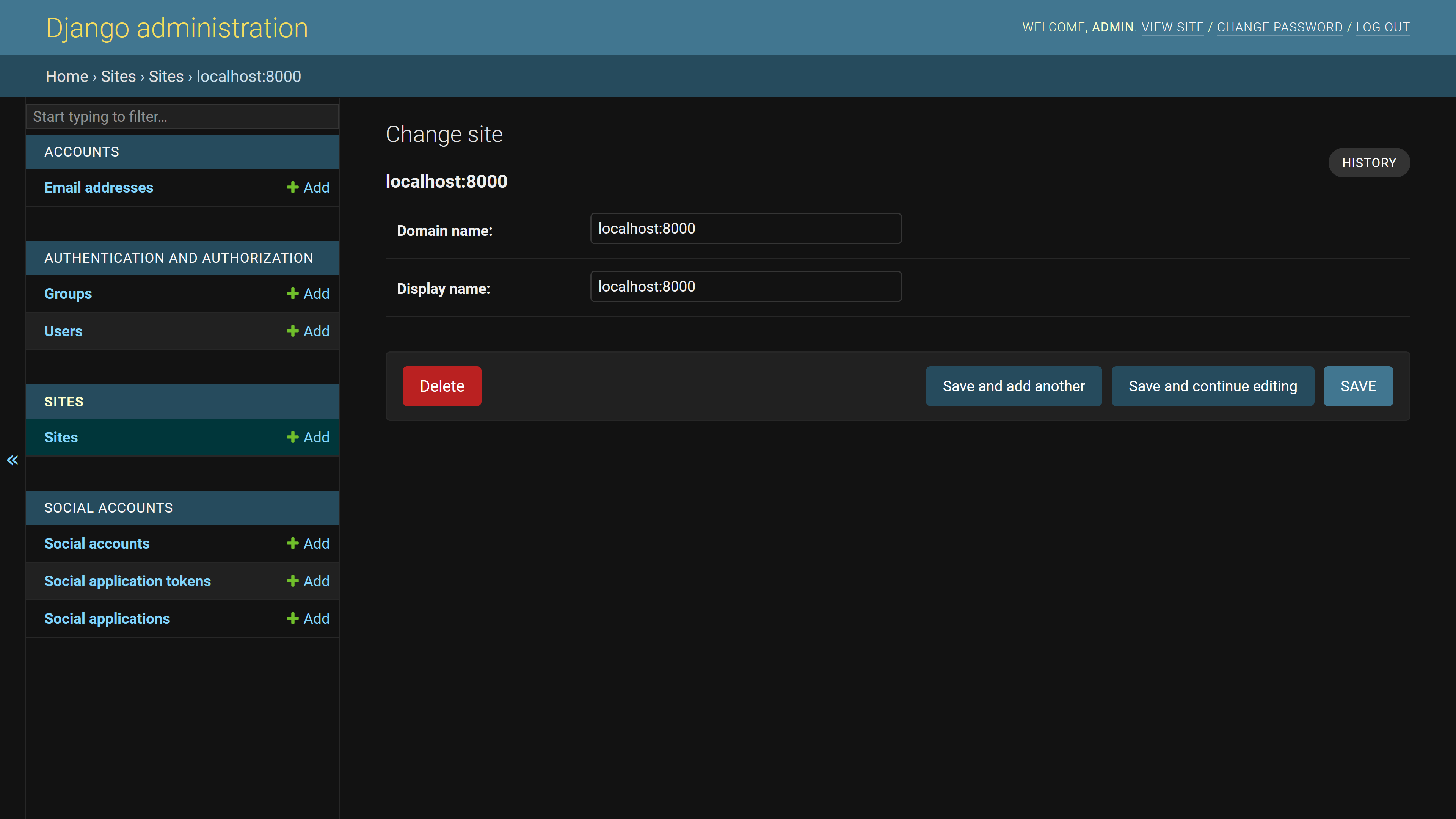
    path('accounts/', include('allauth.urls')),

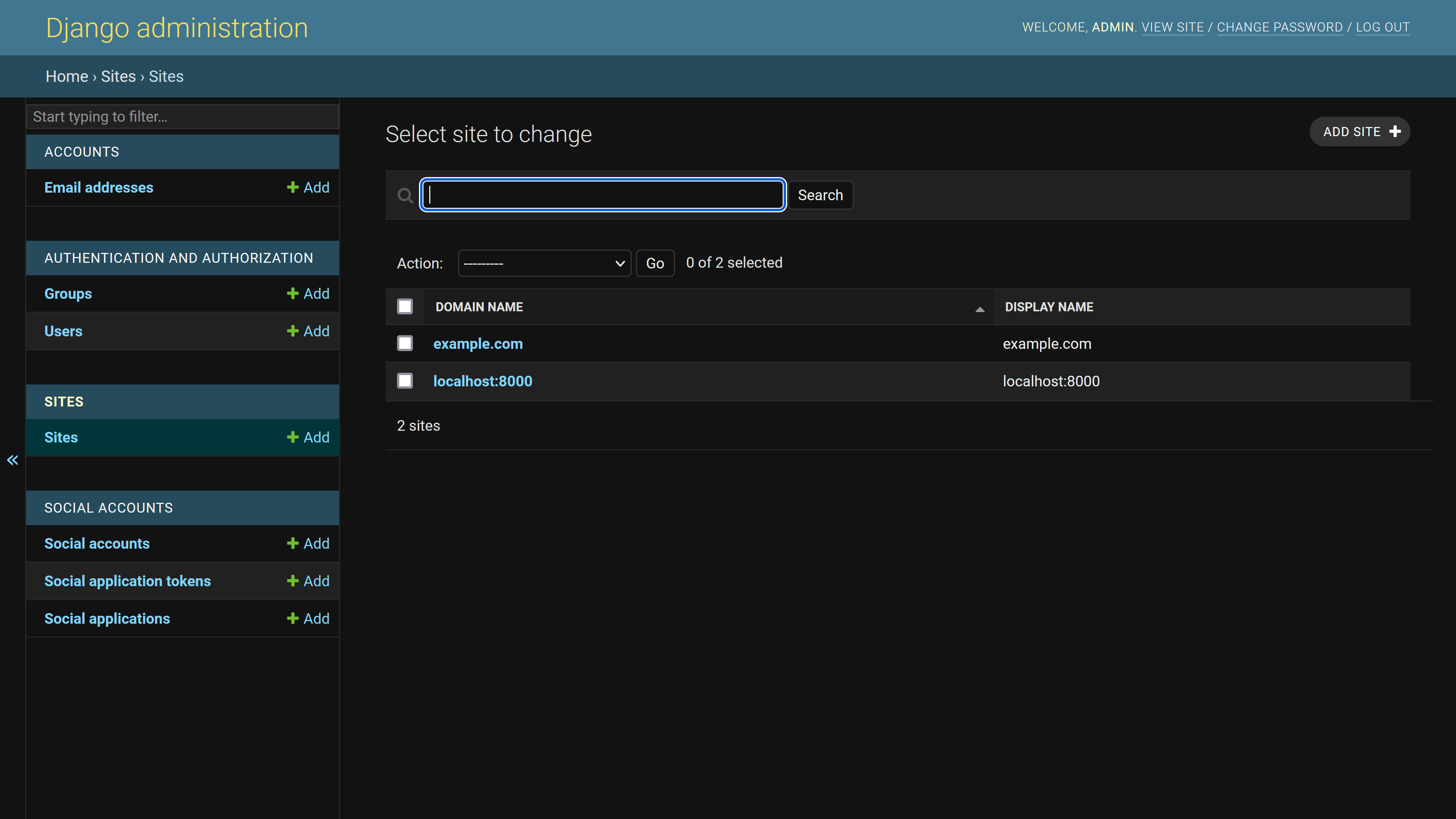
    path('hello/', hello.hello\_world, name='hello\_world'),

]

1. Django-Admin Configuration

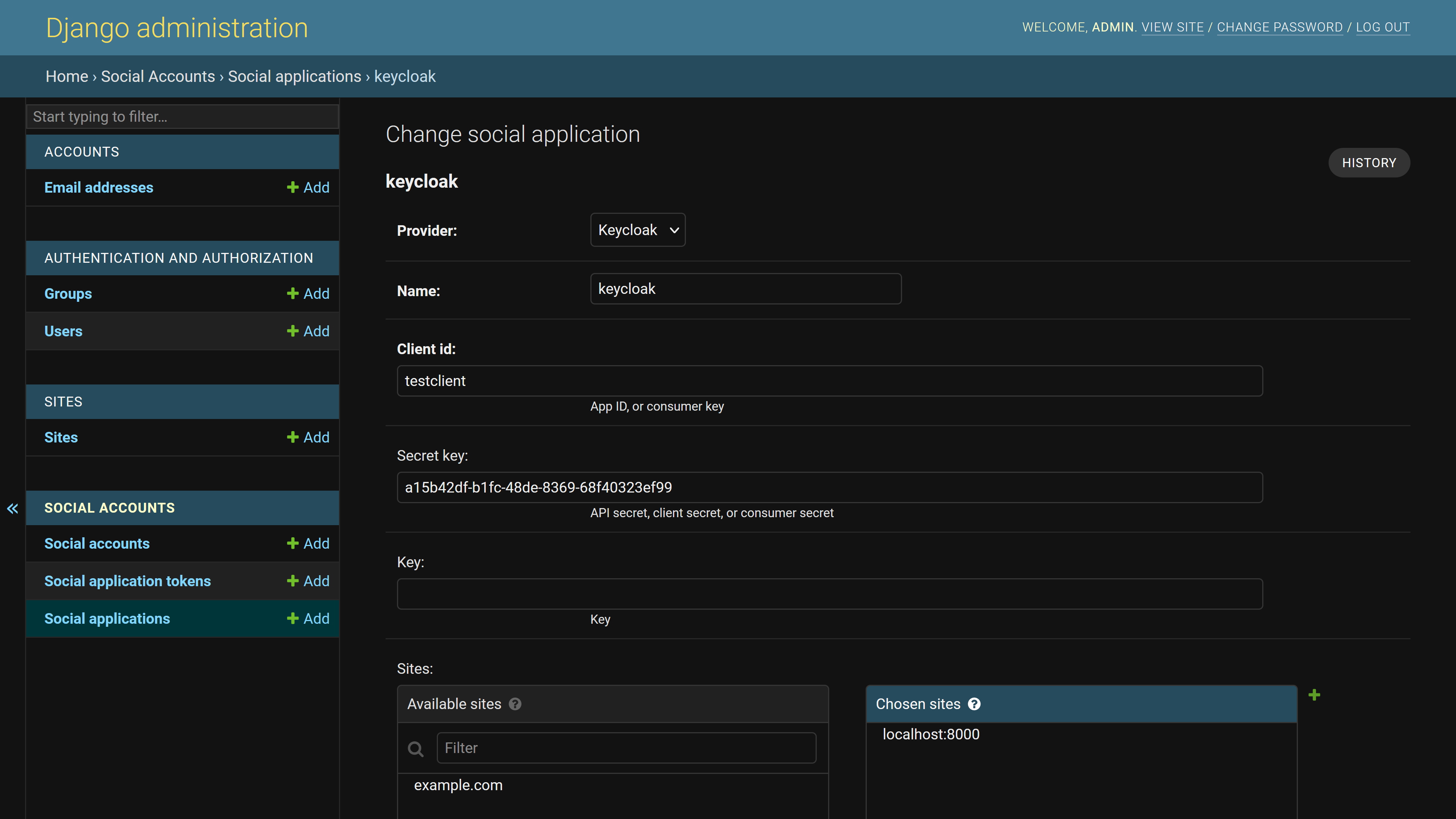
Tambahkan Site: localhost:8000, ini disesuaikan dengan aplikasi django kita





Penting juga untuk diperhatikan, karena sebelumnya sudah ada site example.com (dengan SITE\_ID = 1), maka site yang kita tambahkan akan mendapatkan SITE\_ID = 2. Ini yang kita sesuaikan dengan konfigurasi di settings.py yang dijelaskan di atas

Kemudian tambahkan Social Application



Dengan value sbb:

Provider: Keycloak

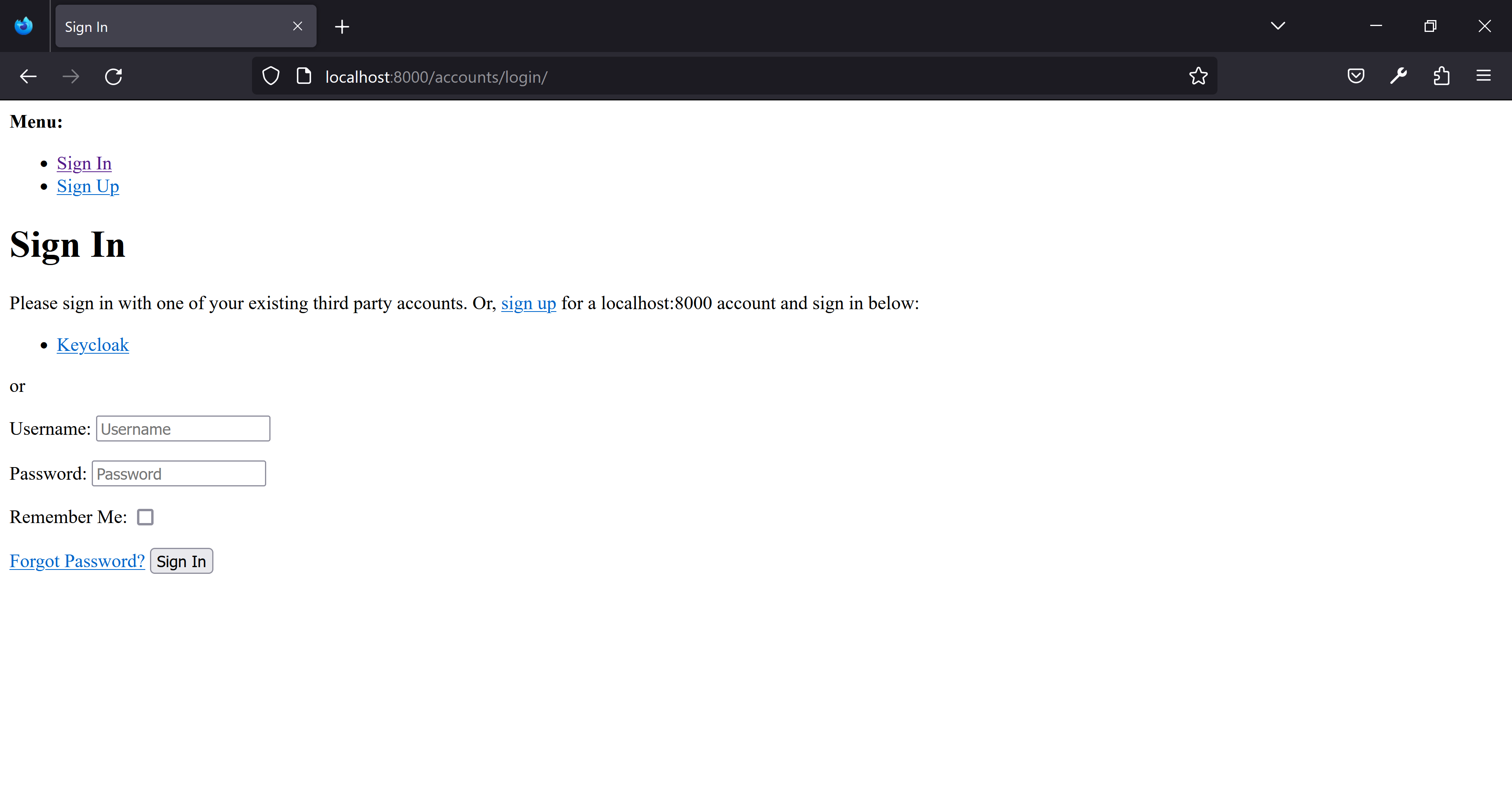
Client Id: testclient 🡪 sesuai dengan konfigurasi client di keycloak admin console

Secret Key: [client secret] 🡪 sesuai dengan konfigurasi client di keycloak admin console

Choosen Sites: 🡪 tambahkan localhost:8000 (aplikasi kita)

1. Test

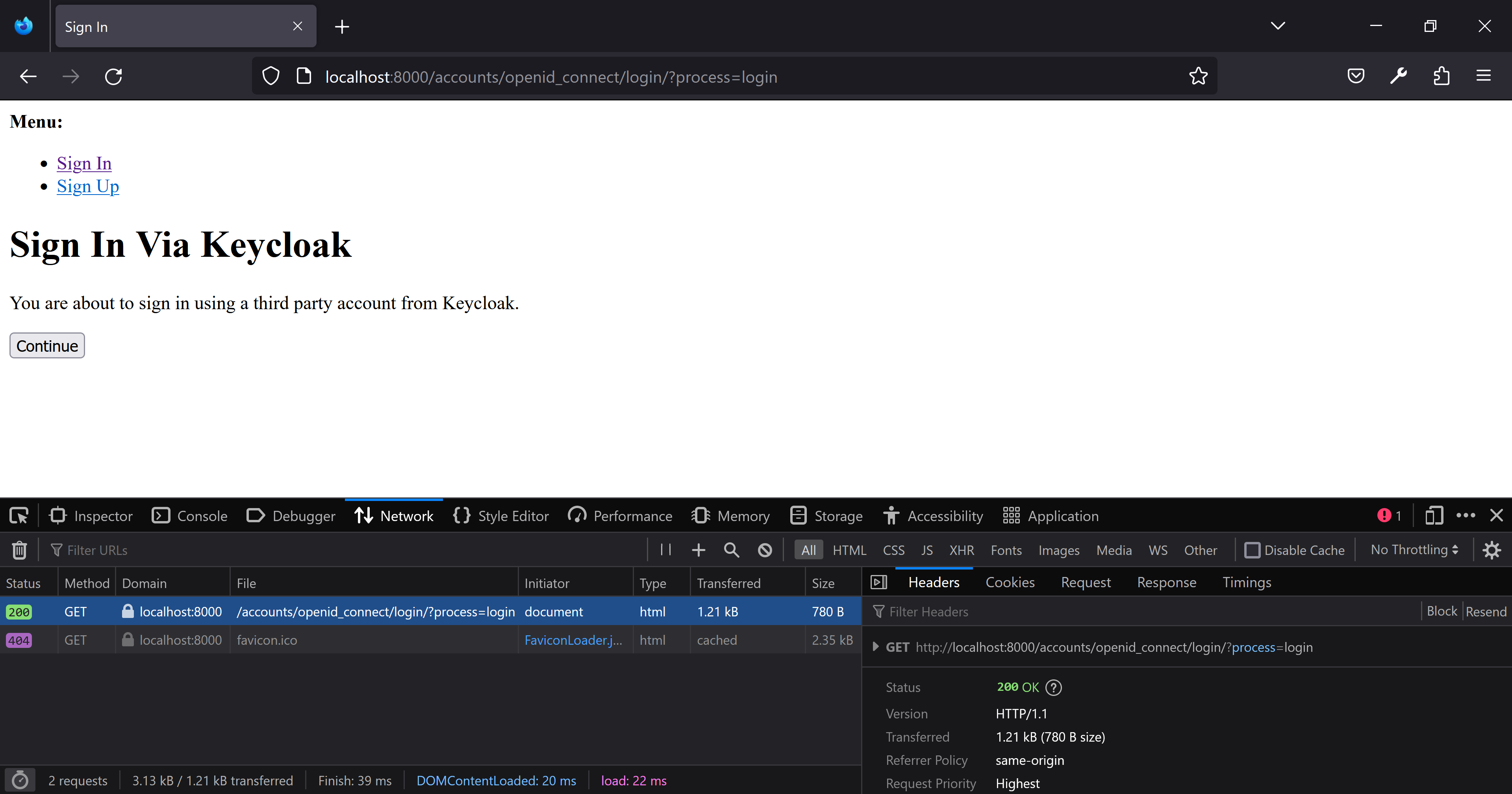
Membuka login page: <http://localhost:8000/accounts/login>



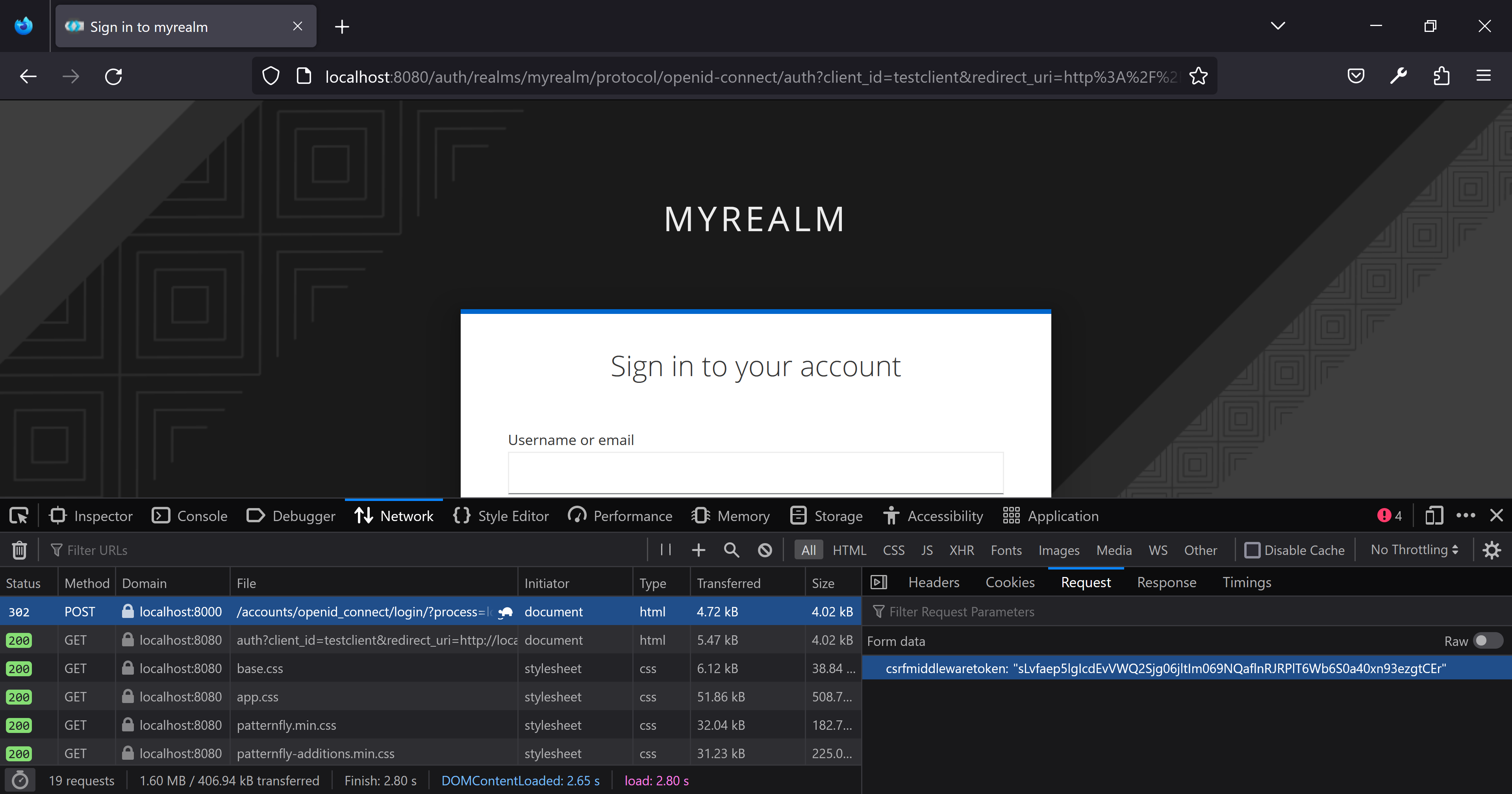
Klik Keycloak, akan membuka halaman selanjutnya

GET

<http://localhost:8000/accounts/openid_connect/login/?process=login>



Klik Continue



POST

<http://localhost:8000/accounts/openid_connect/login/?process=login>

{

"csrfmiddlewaretoken": "sLvfaep5lgIcdEvVWQ2Sjg06jltIm069NQaflnRJRPlT6Wb6S0a40xn93ezgtCEr"

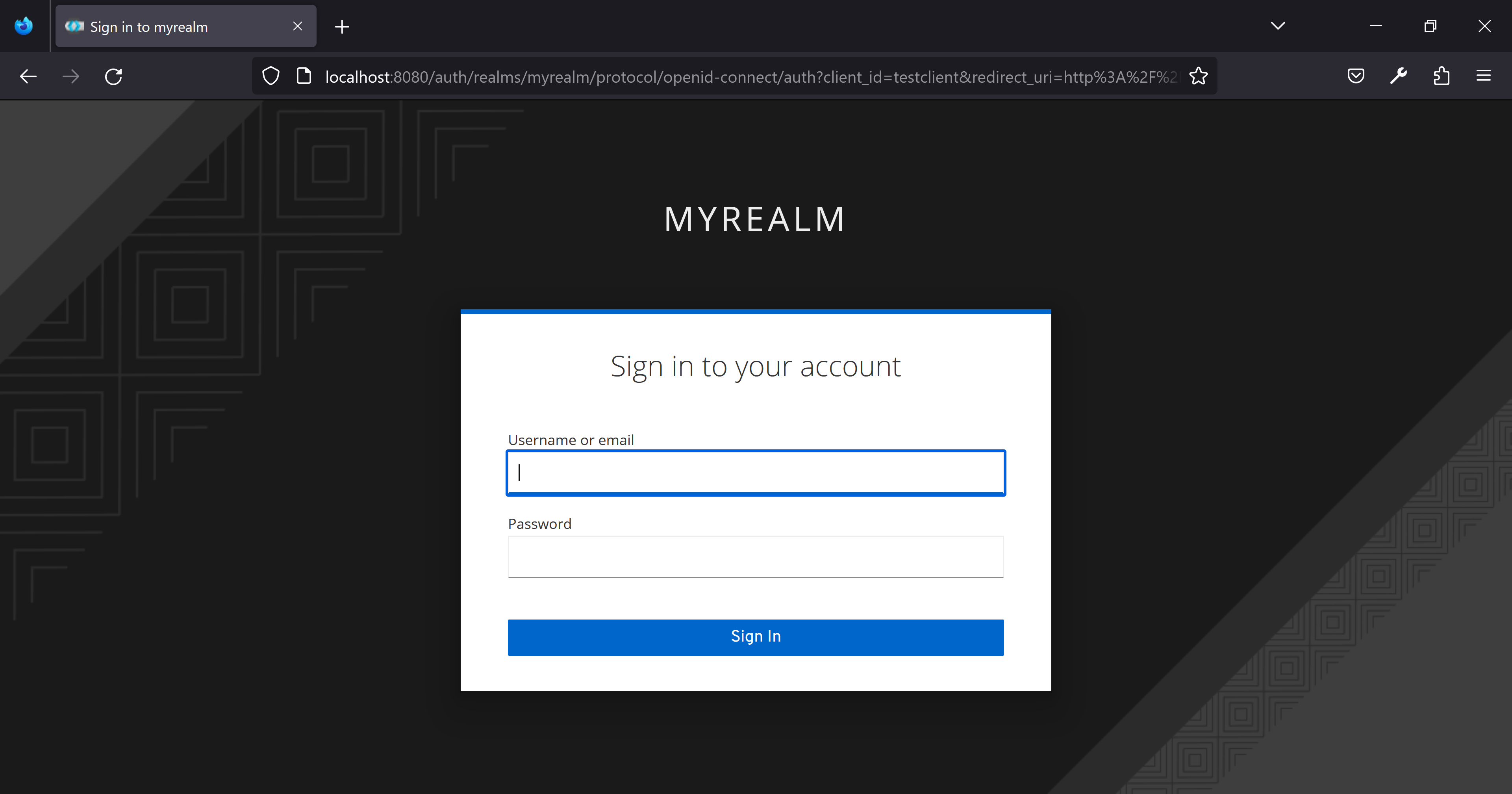
}

Response: 302

Kemudian akan diredirect ke…

GET

<http://localhost:8080/auth/realms/myrealm/protocol/openid-connect/auth?client_id=testclient&redirect_uri=http%3A%2F%2Flocalhost%3A8000%2Faccounts%2Fopenid_connect%2Flogin%2Fcallback%2F&scope=profile+openid+email&response_type=code&state=fgjSloWwTQJL>



Masukkan username dan password sesuai user di realm keycloak, klik Sign In, akan di-redirect ke endpoint yang kita set di LOGIN\_REDIRECT\_URL

\* Alternatif lain: menggunakan provider OpenID Connect (instead of Keycloak)

Di **settings.py**

Di bagian INSTALLED\_APPS, ganti 'allauth.socialaccount.providers.keycloak' dengan 'allauth.socialaccount.providers.openid\_connect'

INSTALLED\_APPS = [

    ……

    'django.contrib.sites',

    'allauth',

    'allauth.account',

    'allauth.socialaccount',

    'allauth.socialaccount.providers.openid\_connect',

]

Di bagian SOCIALACCOUNT\_PROVIDERS

SOCIALACCOUNT\_PROVIDERS = {

    "openid\_connect": {

        "SERVERS": [

            {

                "id": "openid\_connect",  # 30 characters or less

                "name": "OpenId Connect",

                "server\_url": "http://localhost:8080/auth/realms/myrealm",

                # Optional token endpoint authentication method.

                # May be one of "client\_secret\_basic", "client\_secret\_post"

                # If omitted, a method from the the server's

                # token auth methods list is used

                "token\_auth\_method": "client\_secret\_basic",

                "APP": {

                    "client\_id": "testclient",

                    "secret": "dccc6fea-f521-4b08-969d-7a6b0136bdbf",

                },

            },

        ]

    }

}

Referensi:

django-allauth Documentation:

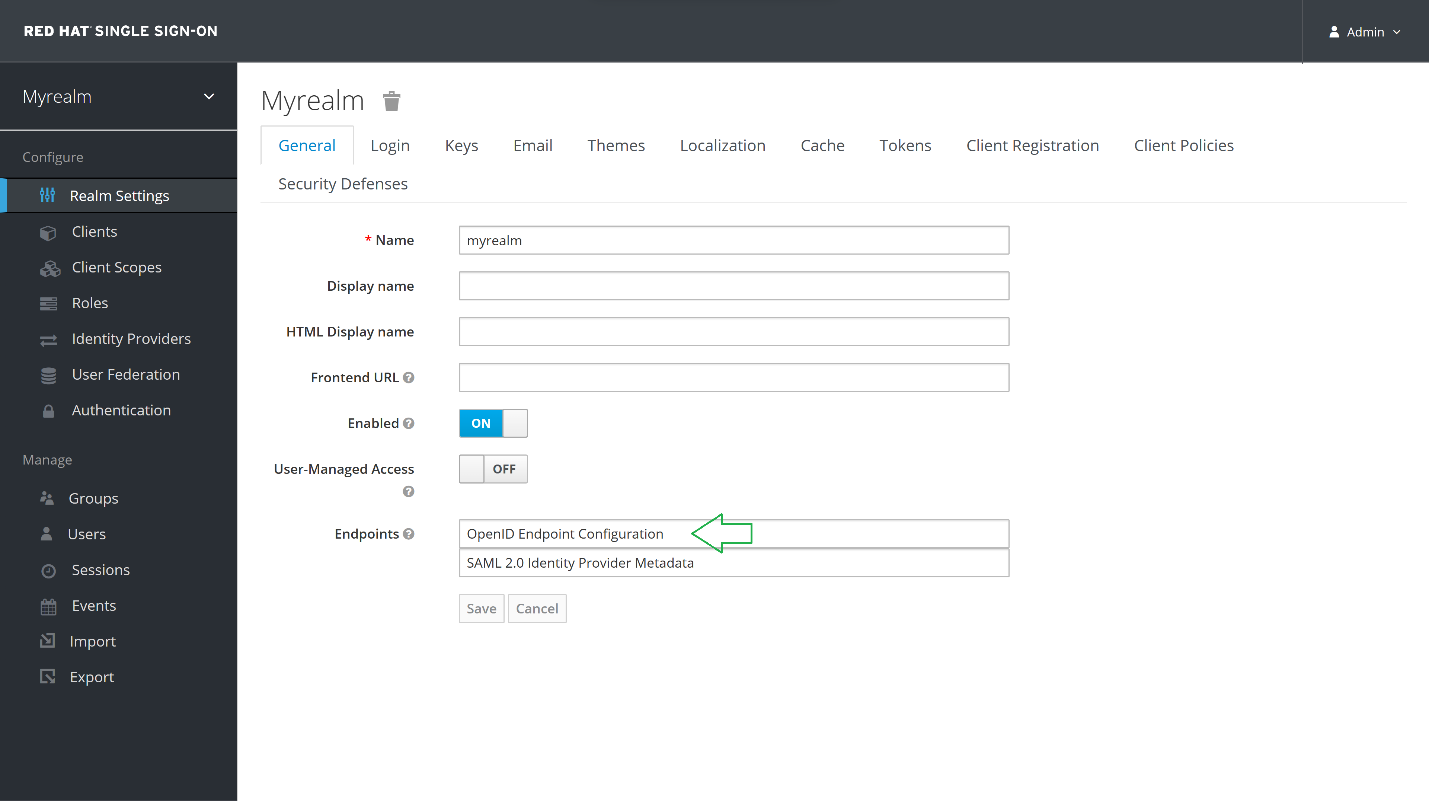
<https://django-allauth.readthedocs.io/en/latest/installation.html>

SSO - Authenticate GeoNode against a keycloack server

<https://gist.github.com/t-book/0fb30804e217bdeb064dd91b5041fbc9>

Mendapatkan Role dan User Profile dari JWT Keycloak

Keycloak mengikuti standard OpenID Connect, termasuk dalam hal JWT Token dan juga endpoint-endpoint yang dapat diakses melalui REST service. Untuk mengetahui beberapa endpoint yang mungkin berguna bagi kita, bisa melalui web console, ke menu Realm Settings, kemudian klik link Endpoints – OpenID Endpoint Configuration





Endpoint [/.well-known/openid-configuration](http://localhost:8080/auth/realms/myrealm/.well-known/openid-configuration) adalah untuk mendapatkan list endpoint-endpoint yang tersedia.

Sedangkan **Token Endpoint** [/protocol/openid-connect/token](http://localhost:8080/auth/realms/myrealm/protocol/openid-connect/token) adalah untuk mendapatkan JWT Token. Sesuai standard OpenID Connect, JWT Token terdiri dari: ID Token, Access Token, dan Refresh Token. ID Token berisi data authentication atau identity user profile, Access Token berisi data authorization, termasuk role dan hak akses, ini sama dengan yang digunakan di standard OAuth2, sedangkan Refresh Token adalah token yang digunakan untuk mendapatkan token yang baru apabila token lama telah expired.

Berikutnya kita akan menggunakan Token Endpoint ini, untuk mendapatkan informasi yang kita butuhkan dari JWT, misal user profile dan access role

Dengan Realm dan Client yang sudah kita buat sebelumnya lewat SSO Web Admin Console, kita memiliki konfigurasi sbb:

* realm: myrealm
* client-id: testclient
* client-secret: a15b42df-b1fc-48de-8369-68f40323ef99
* username: myuser
* password: P455w0rd!

Kemudian kita akan menggunakan Postman untuk melakukan POST Request ke Token Endpoint SSO

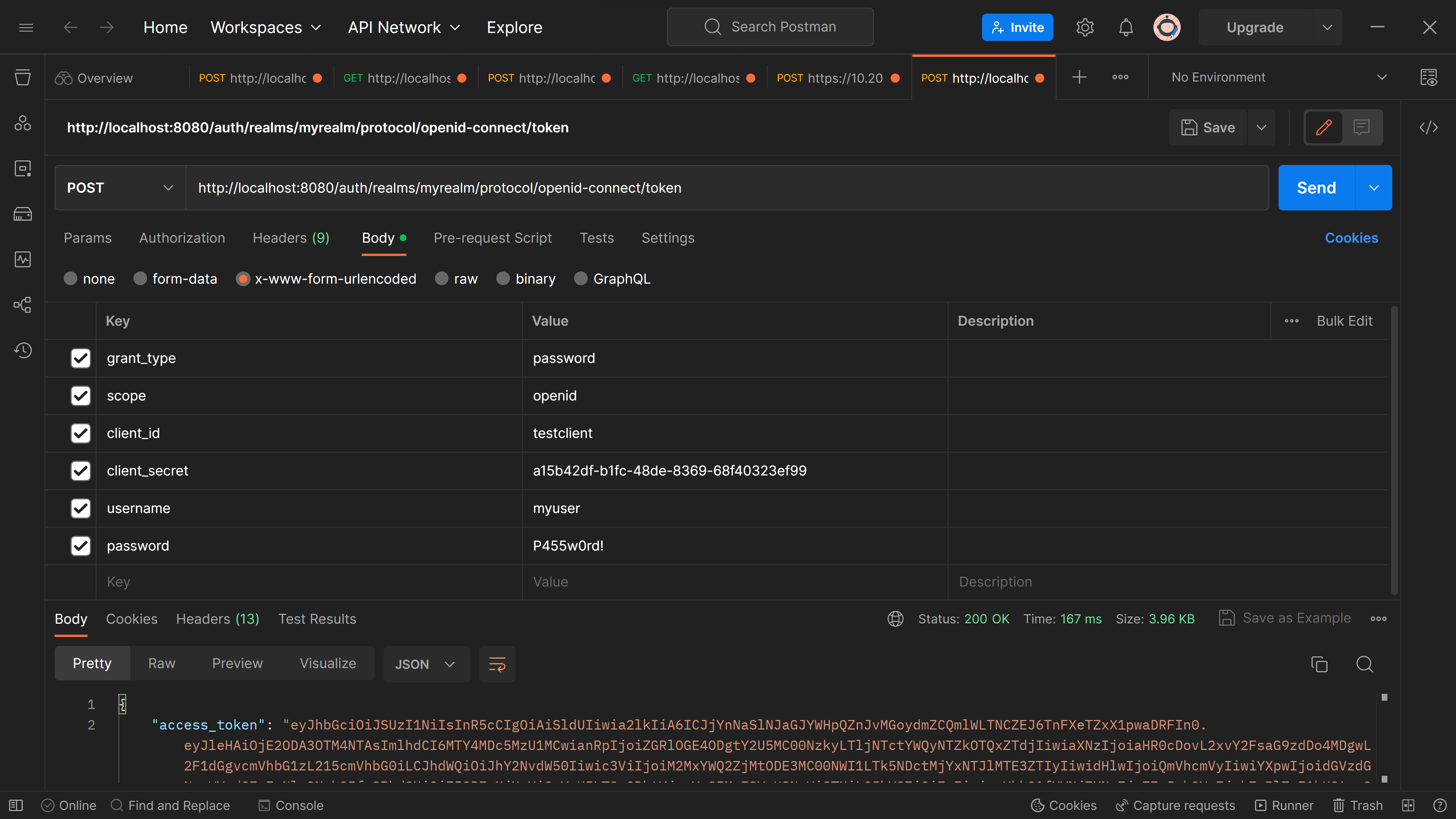
POST: <http://localhost:8080/auth/realms/myrealm/protocol/openid-connect/token>

Header:

* Content-Type: application/x-www-form-urlencoded

Body:

* grant\_type: password
* scope: openid
* client\_id: testclient
* client\_secret: a15b42df-b1fc-48de-8369-68f40323ef99
* username: myuser
* password: P455w0rd!



Response:

{

    "access\_token": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJjYnNaSlNJaGJYWHpQZnJvMGoydmZCQmlWLTNCZEJ6TnFXeTZxX1pwaDRFIn0..XxvwJ3biNMLVSgZoxw20Cv9fPp7xkcGWS2Sq\_Ykhth7LqPH1sWhbzOUYYbkMpffqHmoylWpCCWOTwOkq3owioTUr1huk0s5Z7H9wqNL0Ljj7tg6yjus70uXfTzWxgTVjmvHyJ77euBsDH\_ylDosbF3sD5\_pjBNjSARPYjCOQcTAbQQTvoYgoI6T9b1fBO0iGT9iSMDrIBwSuyg8aInecvl5SlXw1wOUw9qIs86U25dEKpW1LWbhip4-EEU\_So6sTmHGbcY6jWT0b50BGKgMaF57pSgCoa8gm5BvjstOx2zg0YbO8zFnMVOzDhtySvl-YOmEo1xEtAVgffr1yiR-Sfw",

    "expires\_in": 300,

    "refresh\_expires\_in": 1800,

    "refresh\_token": "eyJhbGciOiJIUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICI5NDlmMDFjNC00OTBiLTRlODYtOGEyZi00MTQ4MDM3NjQzMmEifQ..7gQIo7TjUB2lPBx0kqt12O7HWepo-MaXzVDRo5CexMw",

    "token\_type": "Bearer",

    "id\_token": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJjYnNaSlNJaGJYWHpQZnJvMGoydmZCQmlWLTNCZEJ6TnFXeTZxX1pwaDRFIn0..aMipGpVOvR4A62DF6gWjFWqto8rHpfmBoHymh33CE6pzBwzsnZZWsxOI-EPiWBQYAgvO5sbyAonRu\_GAbCKZhlYdSDW63E9eWqNMtxKXm2py6n377OZJPg5W8PsXLpFQR42pj\_7uv\_swVO4QkN9Jw8qLY3RCeqy9Ucb4iwTc3dYItjInnbsN42HxluIDCEPiIf9Oi72bIP\_tZiosd9o5VZdgNzdARTIOQUr6OkRz5nevVtmCexifL-hdzyZAffgLZ4qWC\_rPeZrWAjUE5OuG9r8P2\_jct9gHiVIn49RQ23f8fZwVwPOgsXzni0oG6OMzwXKBDLgEWiW9ISDiCRhdIQ",

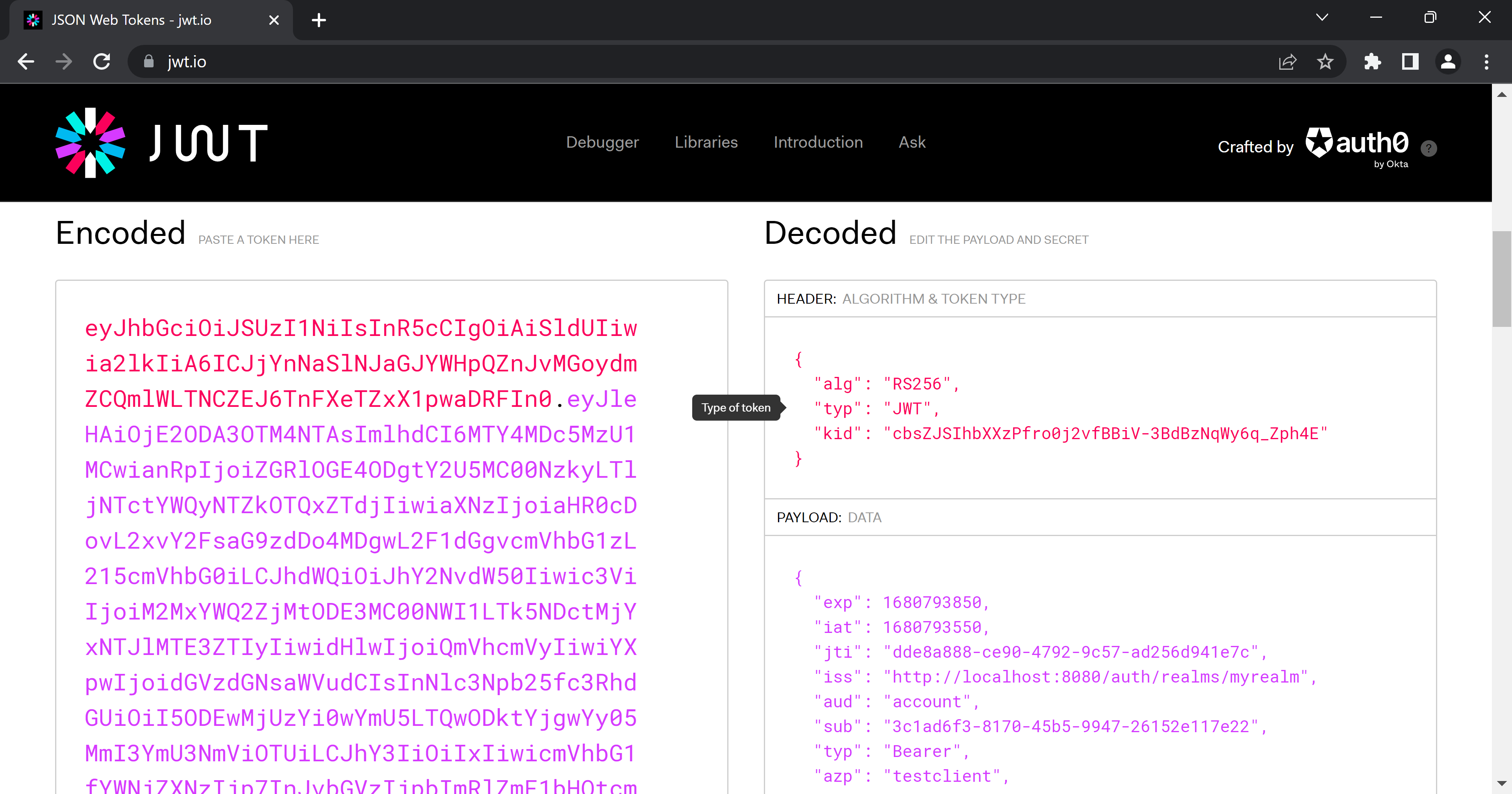
    "not-before-policy": 0,

    "session\_state": "9810253b-0be9-4089-b80c-92b7be76eb95",

    "scope": "openid profile email"

}

Kemudian kita akan mencoba melihat isi dari Access Token melalui web <https://jwt.io/>



Kita dapatkan:

**{**

**"exp": 1680793850,**

**"iat": 1680793550,**

**"jti": "dde8a888-ce90-4792-9c57-ad256d941e7c",**

**"iss": "http://localhost:8080/auth/realms/myrealm",**

**"aud": "account",**

**"sub": "3c1ad6f3-8170-45b5-9947-26152e117e22",**

**"typ": "Bearer",**

**"azp": "testclient",**

**"session\_state": "9810253b-0be9-4089-b80c-92b7be76eb95",**

**"acr": "1",**

**"realm\_access": {**

**"roles": [**

**"default-roles-myrealm",**

**"offline\_access",**

**"uma\_authorization"**

**]**

**},**

**"resource\_access": {**

**"account": {**

**"roles": [**

**"manage-account",**

**"manage-account-links",**

**"view-profile"**

**]**

**}**

**},**

**"scope": "openid profile email",**

**"sid": "9810253b-0be9-4089-b80c-92b7be76eb95",**

**"email\_verified": false,**

**"name": "My User",**

**"preferred\_username": "myuser",**

**"given\_name": "My",**

**"family\_name": "User",**

**"email": "myuser@gmail.com"**

**}**

Kemudian kita akan mencoba mendapatkan value dari **"resource\_access"** 🡪 **"account"** 🡪 **"roles"** dengan menggunakan Django

Seperti flow aplikasi Django kita sebelumnya, setelah sukses authentication oleh Keycloak, kita akan di-redirect ke route **/hello**, yang berada di file **hello.py** dan di method **hello\_world**. Di sini kita akan melakukan:

* membuat POST request ke Token Endpoint
* mengambil Access Token
* decode Access Token untuk mendapatkan access role

Untuk menangani (encode/decode) JWT Token, kita akan menggunakan library PyJWT (<https://pyjwt.readthedocs.io>)

Dan code di hello.py sbb:

from django.http import HttpResponse

import requests

import json

import logging

import jwt

from django.conf import settings

def hello\_world(request):

    logger = logging.getLogger(\_\_name\_\_)

    # Define the endpoint URL

    host = 'http://localhost:8080/auth'

    realm = 'myrealm'

    url = host + '/realms/' + realm + '/protocol/openid-connect/token'

    logger.info('url: ' + url)

    # Define the data to be sent in the request

    client\_id = 'testclient'

    client\_secret = 'a15b42df-b1fc-48de-8369-68f40323ef99'

    username = 'myuser'

    password = 'P455w0rd!'

    data = 'grant\_type=password&scope=openid&client\_id=' + client\_id + '&client\_secret=' + client\_secret + '&username=' + username + '&password=' + password

    logger.info('data: ' + data)

    # Set the content type of the request to JSON

    headers = {'Content-Type': 'application/x-www-form-urlencoded'}

    # Send the POST request

    logger.info('Begin sending POST')

    response = requests.post(url, data=data, headers=headers)

    logger.info('Finish sending POST')

    # Check if the request was successful

    if response.status\_code == 200:

        # Request was successful, do something with the response

        logger.info('response.status\_code: ' + str(response.status\_code))

        response\_str = response.content.decode('utf-8')

        response\_obj = json.loads(response\_str)

        access\_token = response\_obj['access\_token']

        # decoded\_token = jwt.decode(access\_token, settings.SECRET\_KEY, algorithms=['HS256'])

        decoded\_token = jwt.decode(access\_token, options={"verify\_signature": False})

        roles = decoded\_token['resource\_access']['account']['roles']

        return HttpResponse(roles)

    else:

        # Request failed, handle the error

        logger.error('response.status\_code: ', str(response.status\_code))

        return HttpResponse(response, response.status\_code)

#   return HttpResponse("Hello, " + request.user.username + '!')

Dan hasilnya sbb:

