Django REST project

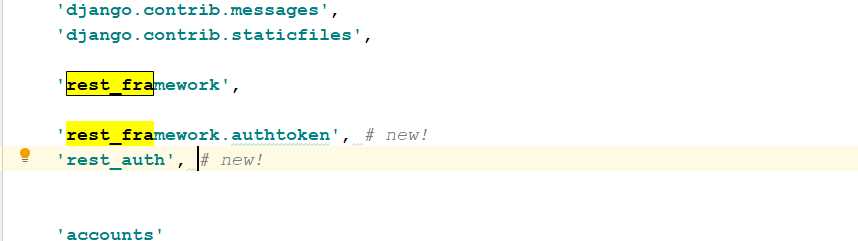
Pre-requirements:

1. Venv
2. Install Django
3. Create new Django project called {….}
4. Start app called accounts
5. Add it to settings
6. Install Django rest
7. Add it to installed apps
8. Create urls in the app
9. Include it in project urls

Now we have to define our goals – we want a user in our app to be able to login/ logout and register also reset password by email and change password in DRF (Django rest framework)

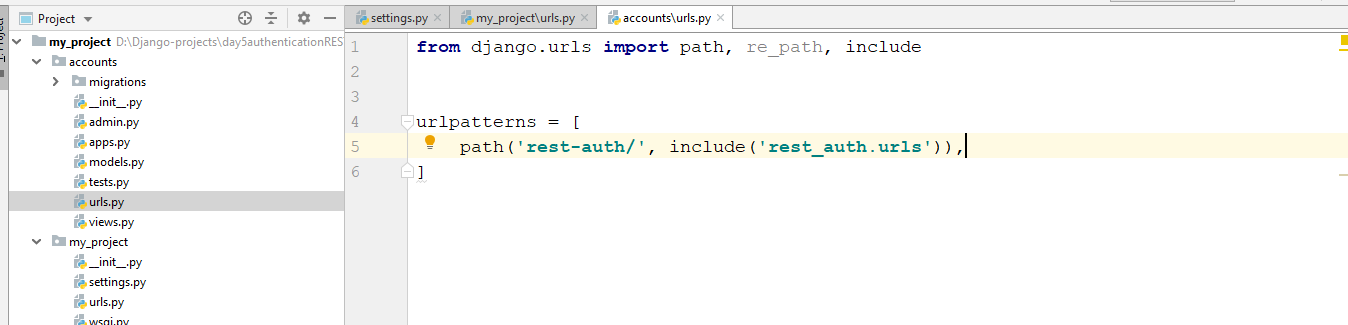
The third-party django-rest-auth library provides a set of pre-configured API endpoints for login, logout, registration, and can support social authentication too. We’ll use it here.

pip install django-rest-auth

We add a couple of things in installed apps 

Note that we’ve added two lines here: both rest\_framework.authtoken which is Django Rest Framework’s token auth app and also rest\_auth which uses it.

In our accounts/urls.py file –



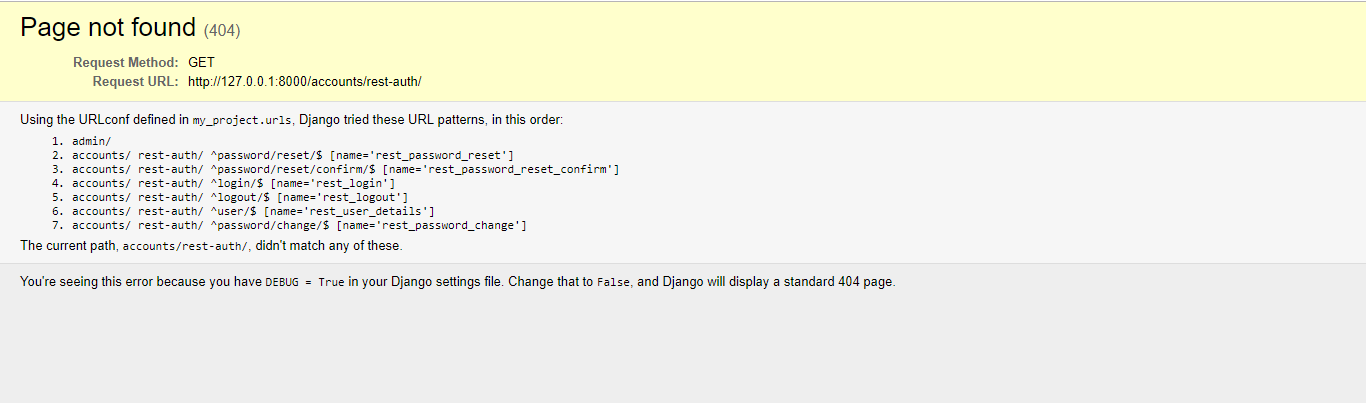
And that was almost all.

Do not forget to include accounts urls in {project\_name}/urls.py.

Migrate everything and start the server.

Go to <http://127.0.0.1:8000/accounts/rest-auth/>

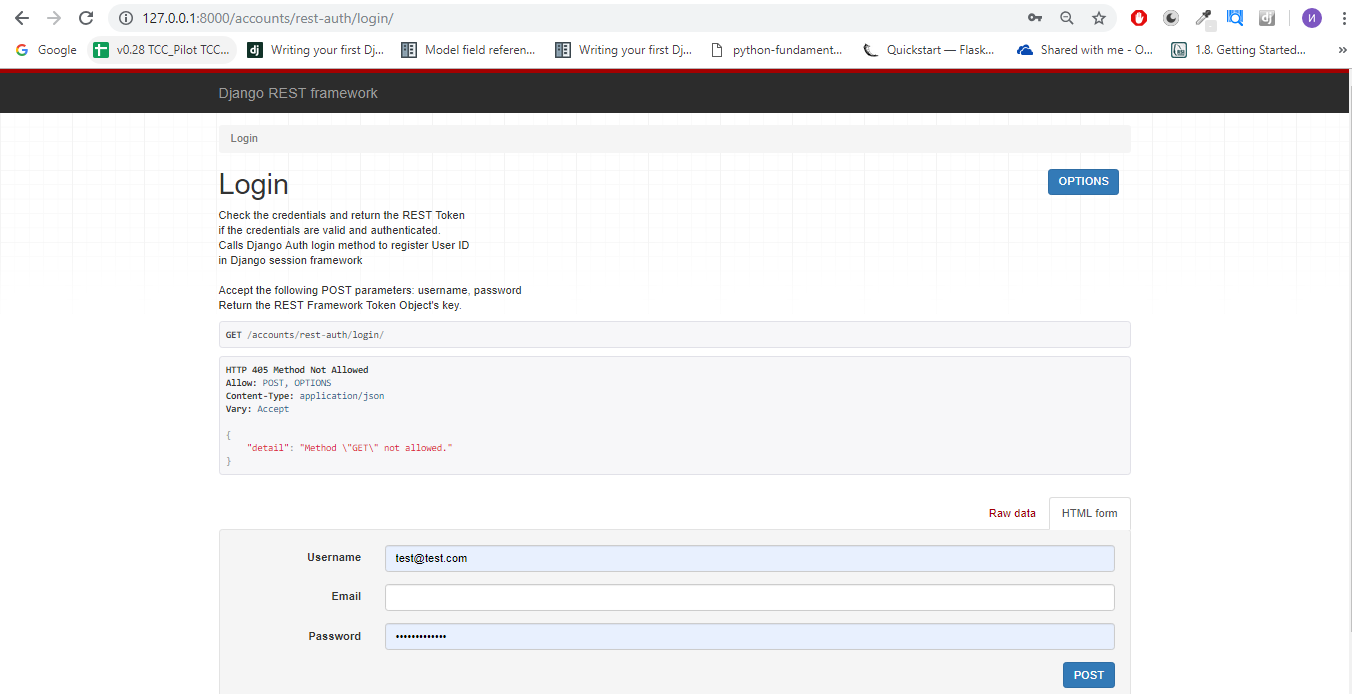
And you will see something like this:



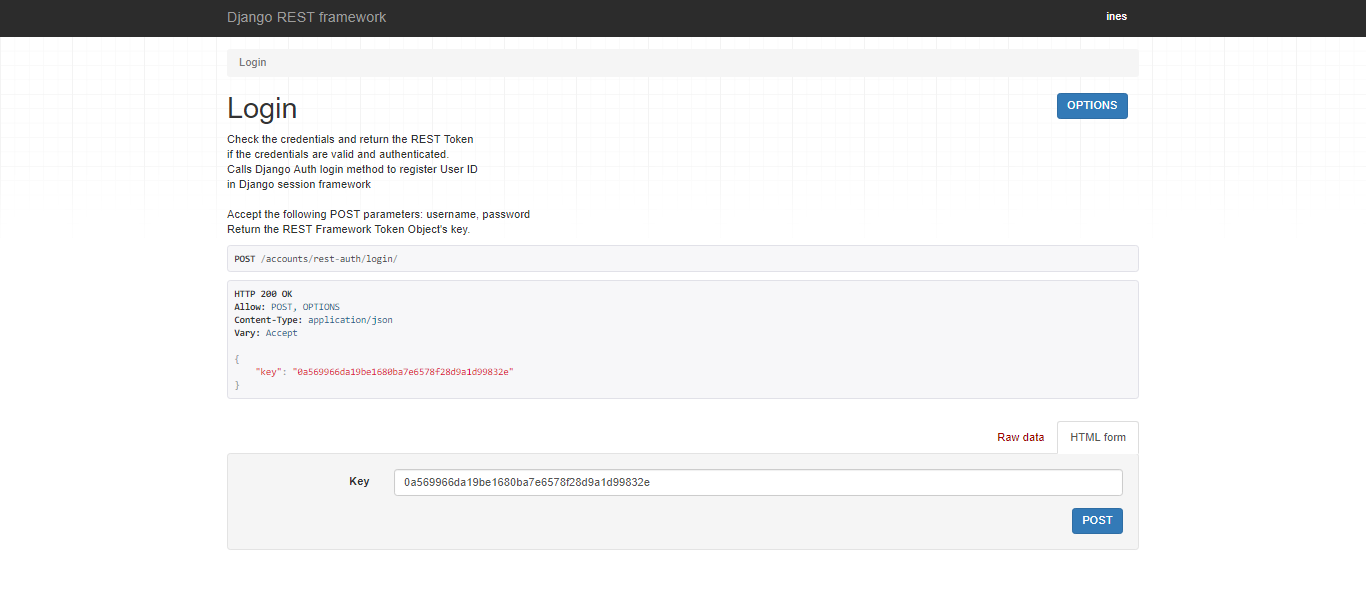
NB! A best practice is the url to be <http://127.0.0.1:8000/api/v1/rest-auth/login/> . Implement it at home

1. **Login functionality**

Out of the box it is done for us. Go to <http://127.0.0.1:8000/accounts/rest-auth/login/> and you will see:



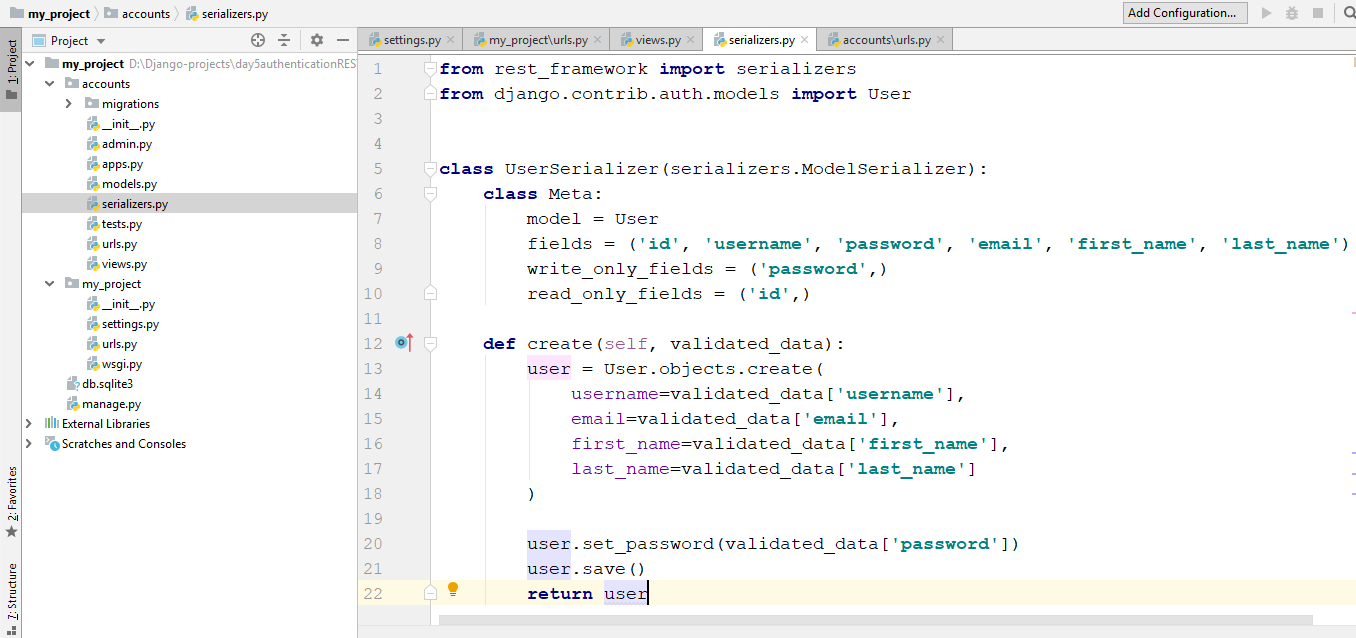
Create a superuser through the consol and try to login. You will see a strange thing after submit



The result is a token for the user which can be used to authenticate future HTTP requests!

1. **Register functionality**

Register functionality is a bit more specific. First create a serializers.py file



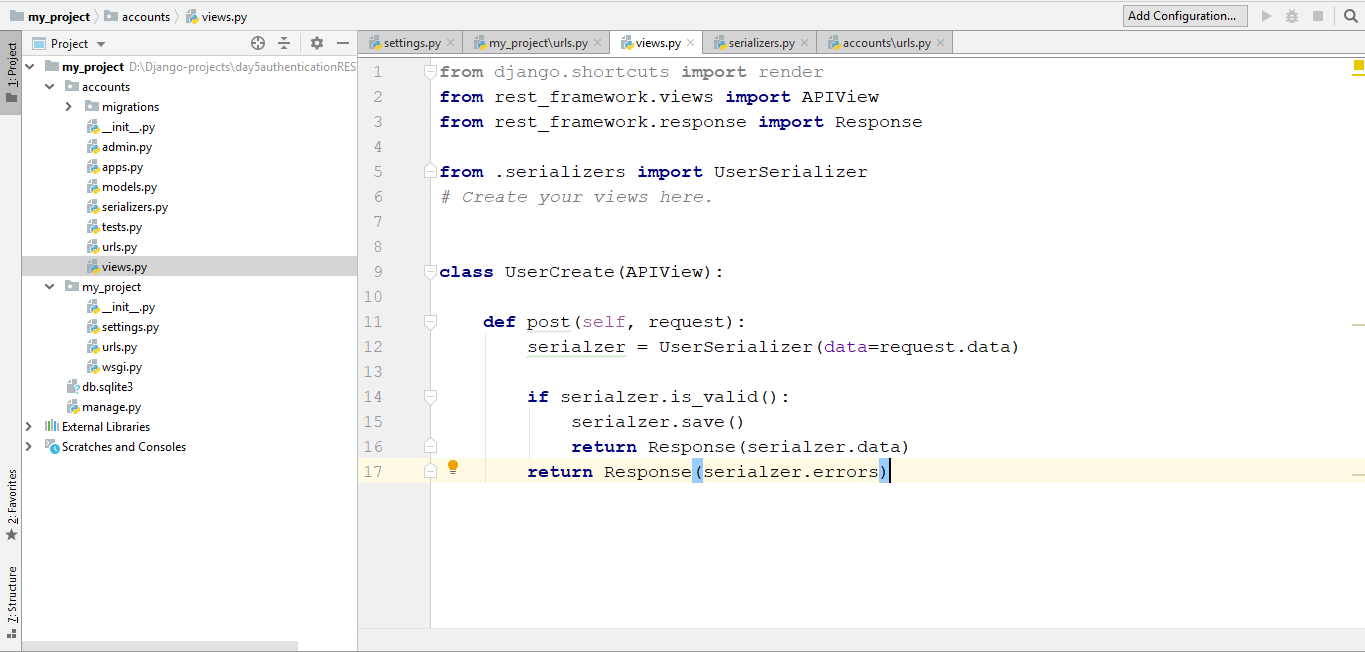
A couple of things to note here

* Write\_only\_fields is a password field, and read\_only is the id (best practices)
* We rewrite create function – NOTE how we call User.objects.create() –

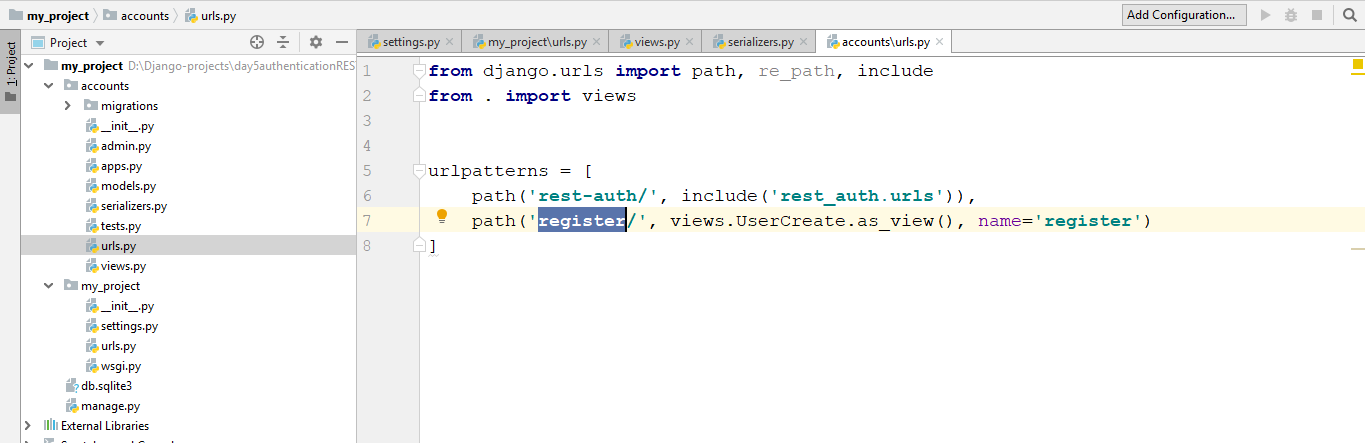
This is a safe function for creation

* We set user password with set\_password function – we do not want to store in plain text user’s password so this function makes it a hash string
* We save and return the user

Let’s go to views and create a class



And the last step – urls



You can go to <http://127.0.0.1:8000/accounts/register/>

And post this:

{

"username": "test\_user",

"first\_name": "test",

"last\_name": "test",

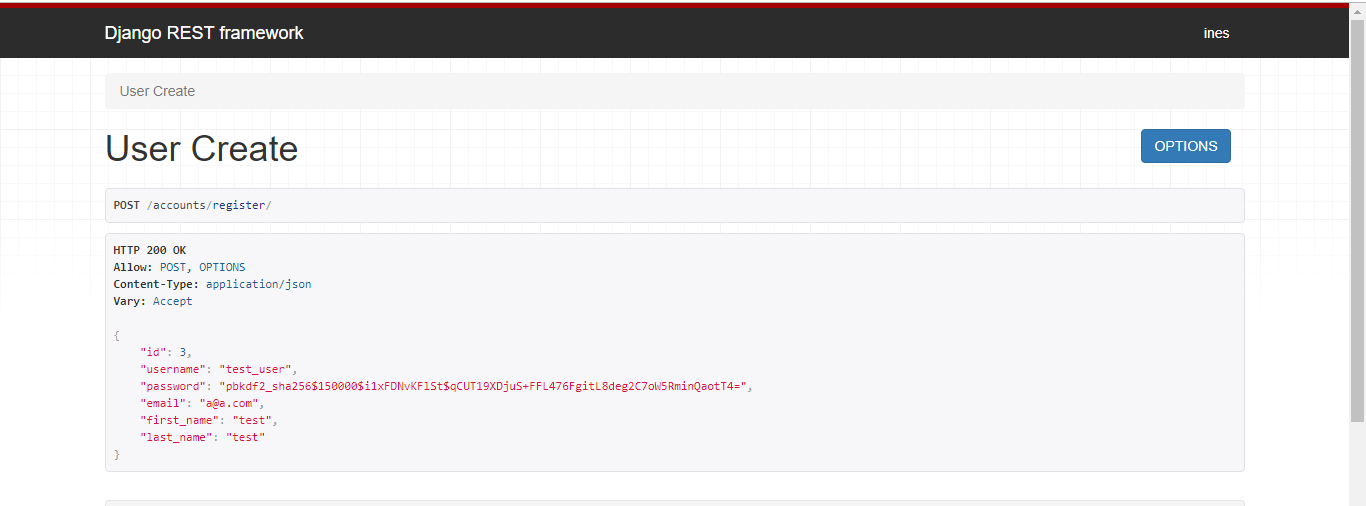
"email": "a@a.com",

"password": "12345678t"

}

Be careful for the double “

You will get something like this as a response:



And if we check the admin panel:



So it is working ☺

TASK FOR YOU: Discover on your own how not to return the password field when the user is registered

1. **Reset and password change functionality**

Literary there are just a few things:

The endpoint <http://127.0.0.1:8000/accounts/rest-auth/password/change/>

Give us the functionality to change password.

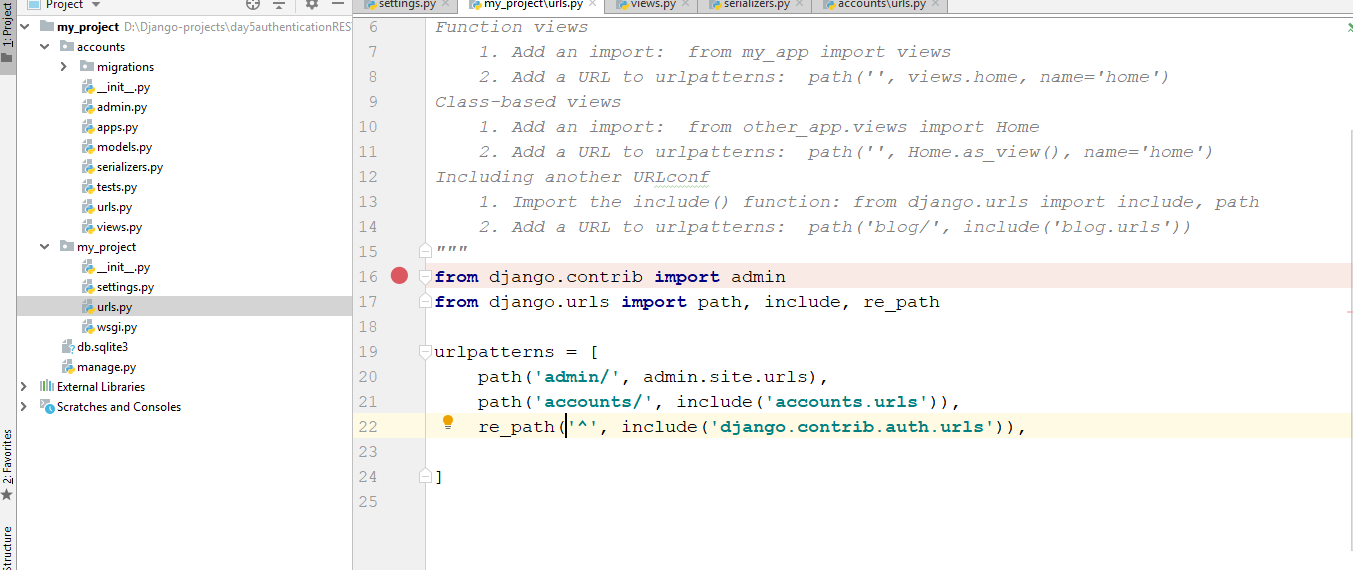
And - <http://127.0.0.1:8000/accounts/rest-auth/password/reset/>

It won’t work though -> for testing purposes we will log the email to the console.

Add this line to seeting.py

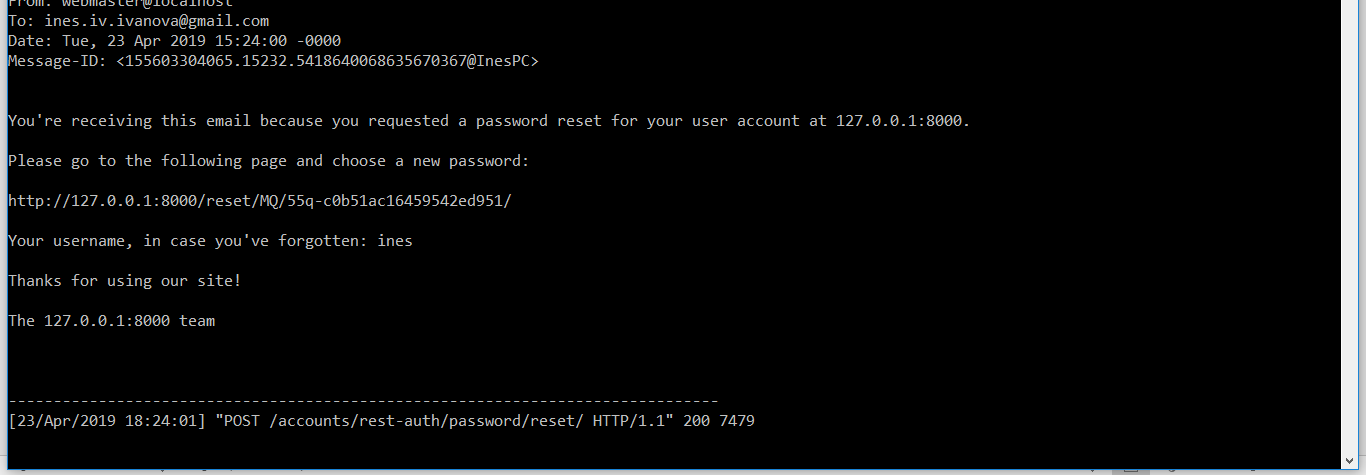
EMAIL\_BACKEND = **'django.core.mail.backends.console.EmailBackend'**

And one more thing in {project\_name}/urls.py



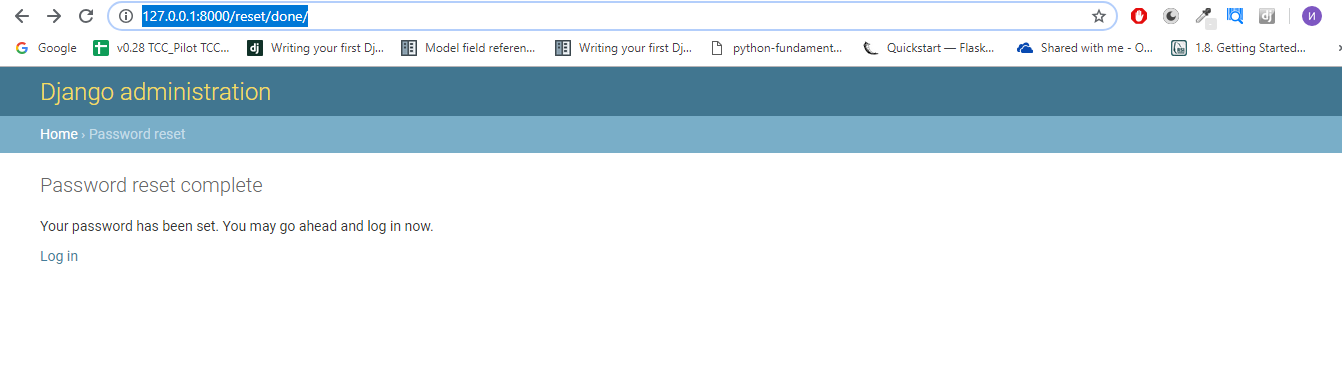
This line is required because Django rest reset works with Django authentication urls.

Submit an email and check the console



Copy the link and paste it to the browser. Follow the steps

TASK FOR YOU: Discover on you own how to tell Django when you are at this step



To go to this link <http://127.0.0.1:8000/accounts/rest-auth/login/> and not to this one <http://127.0.0.1:8000/accounts/login/> when tou click Log in