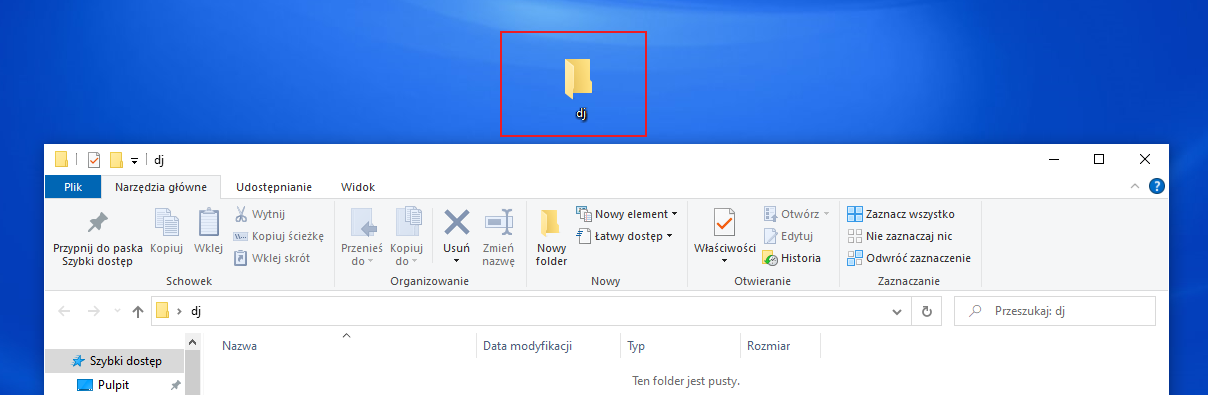
https://mega.nz/folder/mwlADAJR#tWRZp8vuJIBBvjxYcG5BRA/folder/q8ckQboY

Zakładam, że django już jest zainstalowane.

Odcinek 1:

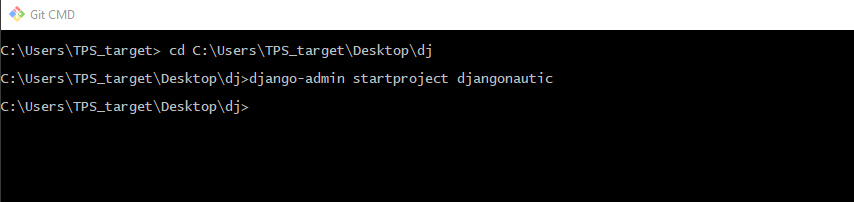
Tworzę folder o dowolnej nazwie (u mnie dj):



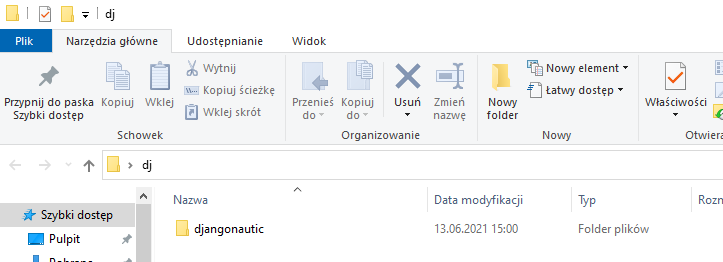
Otwieram CMD, przechodzę do stworzonego folderu (cd *C:\Users\TPS\_target\Desktop\dj*) i piszę:

*django-admin startproject djangonautic*

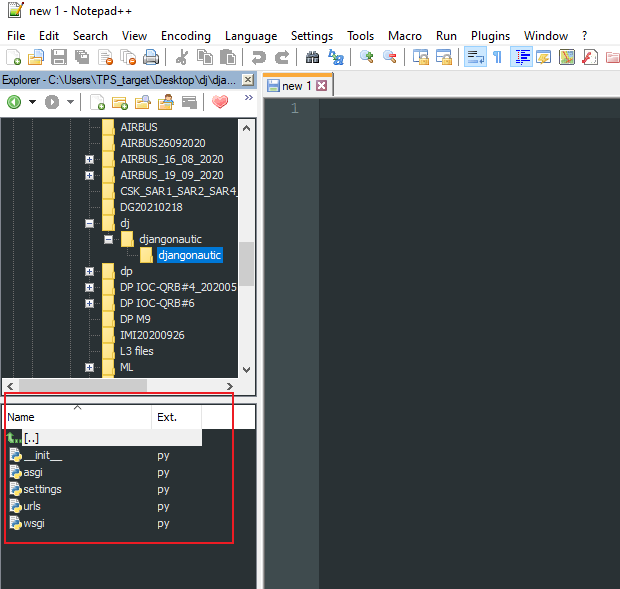
gdzie djangonautic to nazwa projektu:



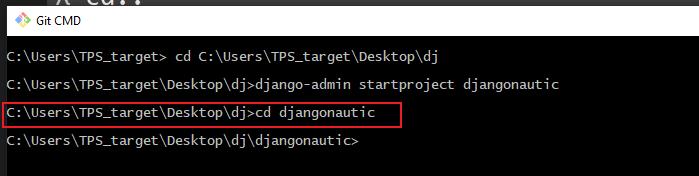
W folderze dj mamy projekt djangonautic:



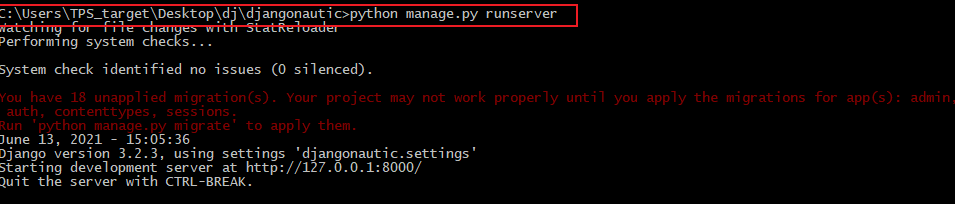
W folderze djangonautic mamy kolejny folder djngonautic. Otwieram Notepad++ w którym widać jakie pliki zostały stworzone:



Przechodzę do folderu djangonautic:

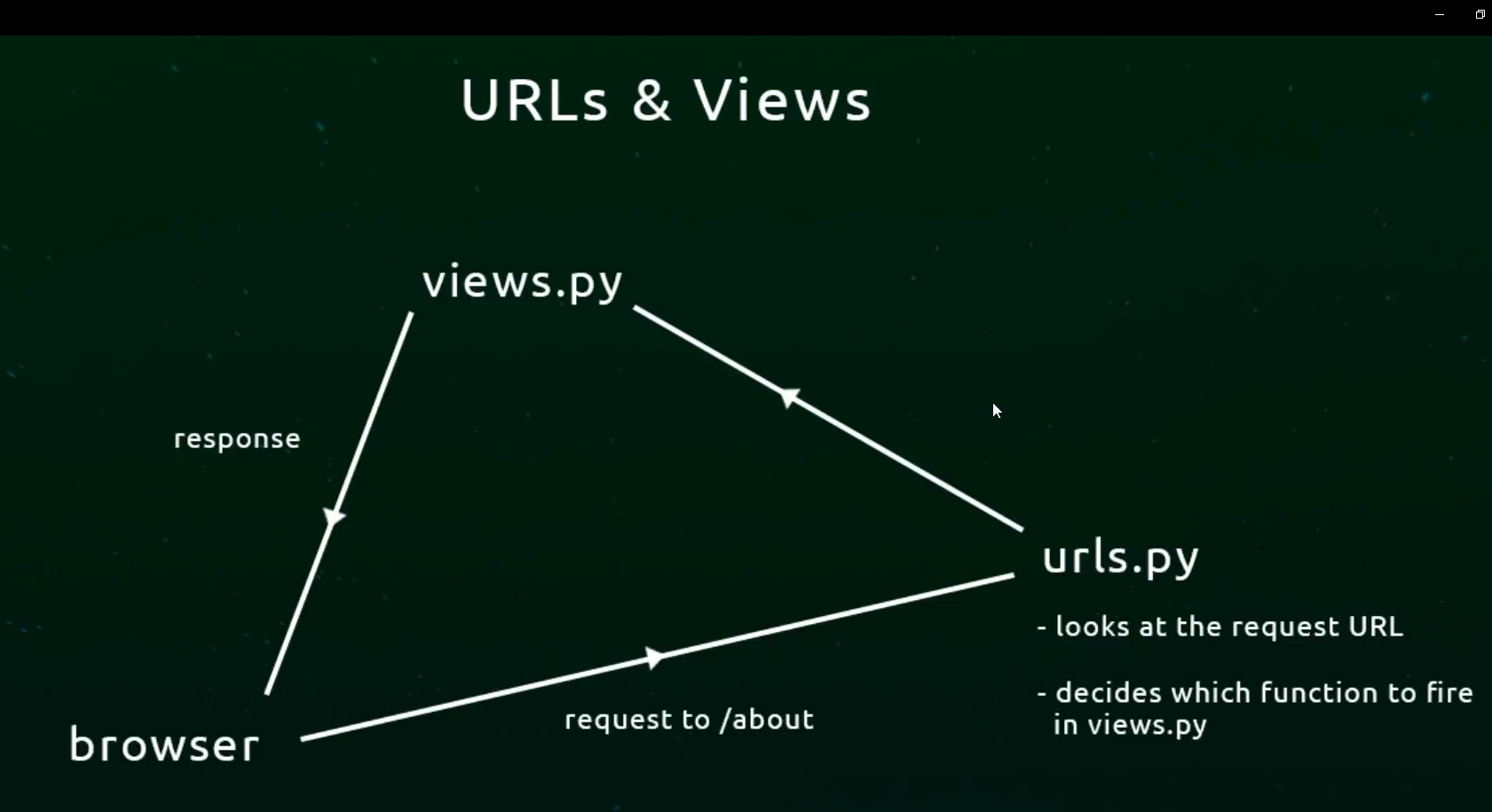


Przechodzę do CMD i odpalam server komendą *python manage.py runserver:*



Odcinek 2:

W tym odcinku konfigurujemy konkretne strony.



Otwieram plik urls.py:



Dodaję taki kod:

*from django.conf.urls import url*

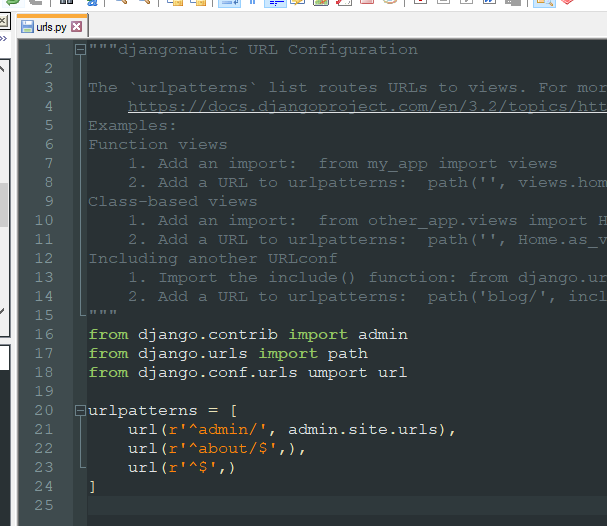
*urlpatterns = [*

*url(r'^admin/', admin.site.urls),*

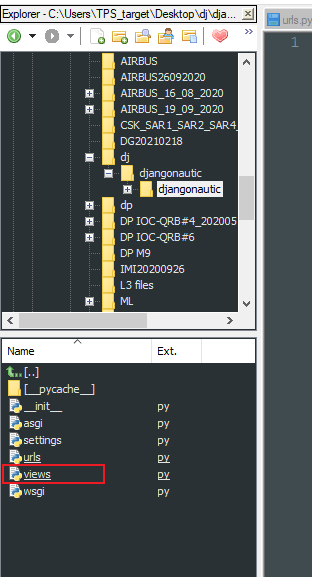
*url(r'^about/$',),*

*url(r'^$',)*

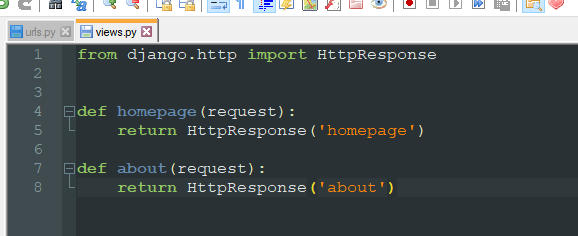
*]*



Tworzę plik views.py:



Zawartość pliku views.py:



*from django.http import HttpResponse*

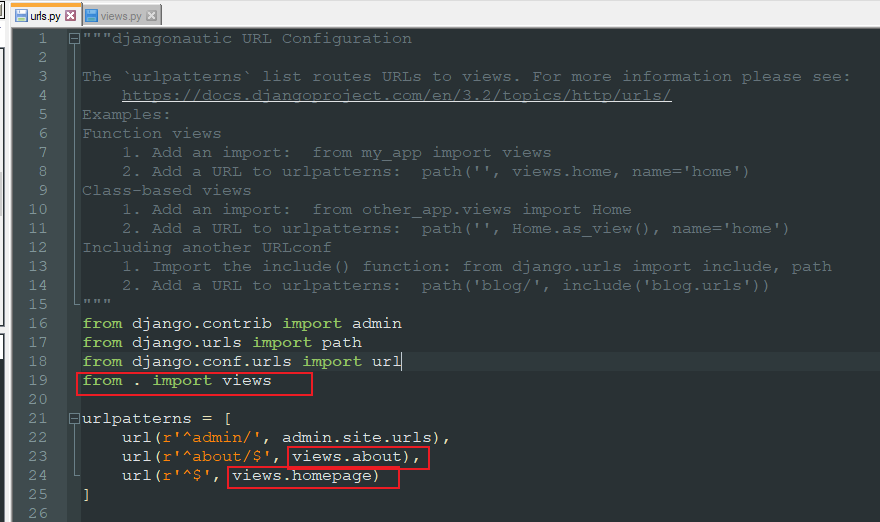
*def homepage(request):*

*return HttpResponse('homepage')*

*def about(request):*

*return HttpResponse('about')*

Modyfikujemy plik urls.py:



*from . import views*

*urlpatterns = [*

*url(r'^admin/', admin.site.urls),*

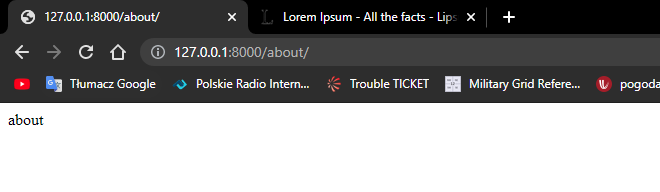
*url(r'^about/$', views.about),*

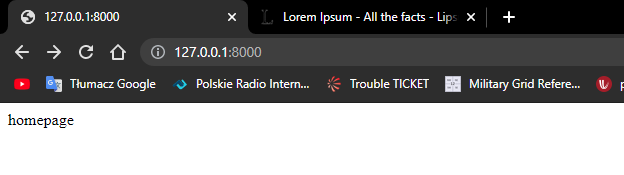
*url(r'^$', views.homepage)*

*]*

Dołączamy plik views a następnie określamy jakie funkcje mają być wywoływane po wejściu w about i po wejściu na stronę.

Tak to wygląda w przeglądarce:

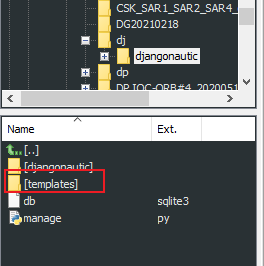




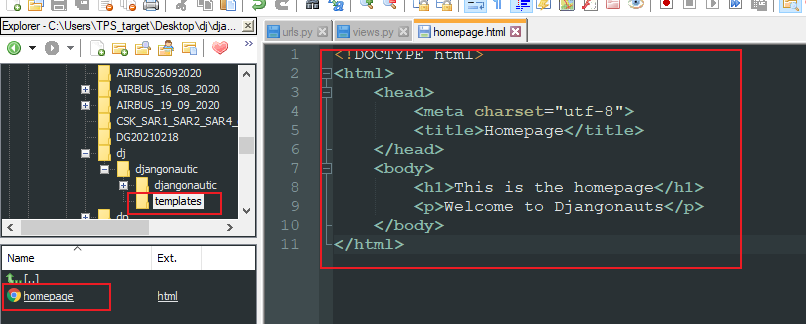
Odcinek 3:

W tym odcinku konfigurujemy strone tak żeby korzystała z plików html.

Tworzymy template directory:



W środku tego katalogu robimy plik homepage.html z następującą zawartością:



*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="utf-8">*

*<title>Homepage</title>*

*</head>*

*<body>*

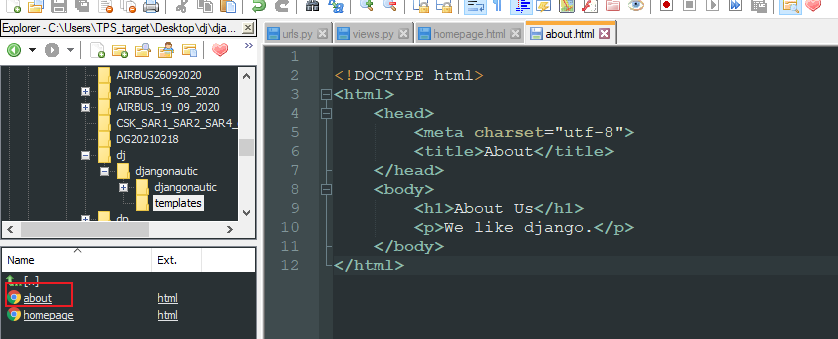
*<h1>This is the homepage</h1>*

*<p>Welcome to Djangonauts</p>*

*</body>*

*</html>*

Analogicznie tworzymy plik about.html:



*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="utf-8">*

*<title>About</title>*

*</head>*

*<body>*

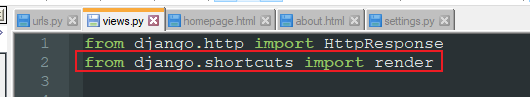
*<h1>About Us</h1>*

*<p>We like django.</p>*

*</body>*

*</html>*

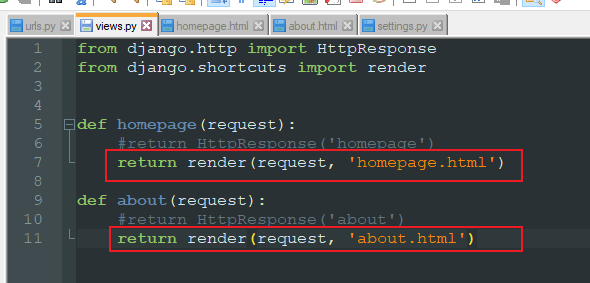
Przechodzimy do pliku views.py i uzupełniamy go o następujący kod:



*from django.shortcut import render*

Pozwala to na renderowanie plików html.

Następnie wskazujemy które pliki html mają być renderowane dla naszych funkcji:



*def homepage(request):*

*#return HttpResponse('homepage')*

*return render(request, 'homepage.html')*

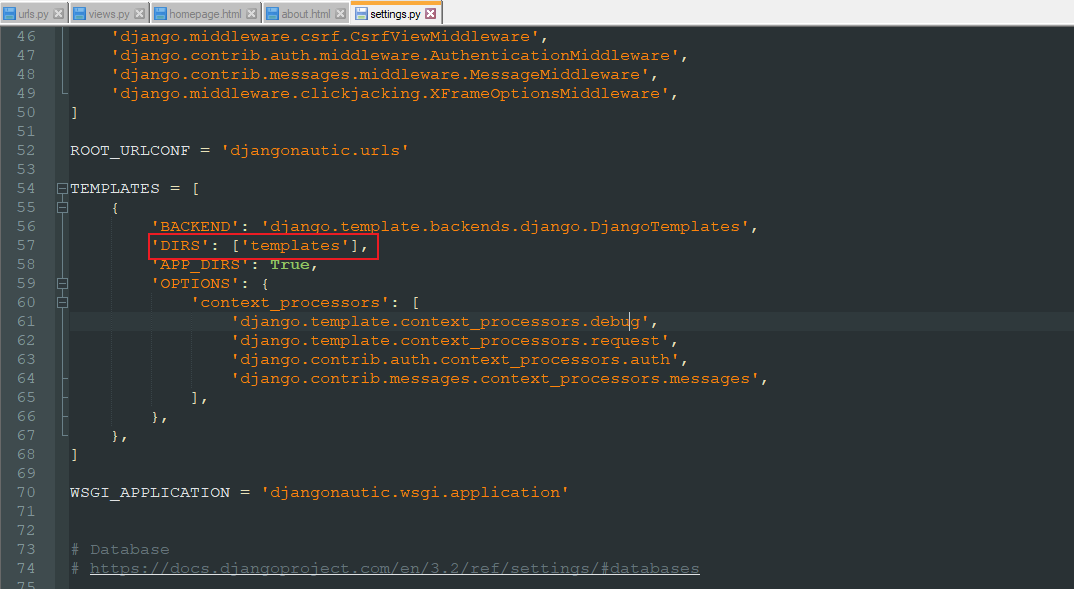
*def about(request):*

*#return HttpResponse('about')*

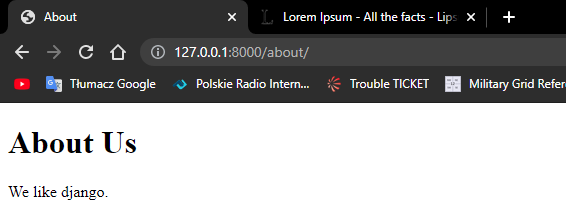
*return render(request, 'about.html')*

Funkcja render jako pierwszy argument zawsze przyjmuje render. Drugi argument to strona jaka ma być renderowana.

Przechodzimy do pliku settings.py i wskazujemy gdzie nasze templatki htmlowe są przechowywane:

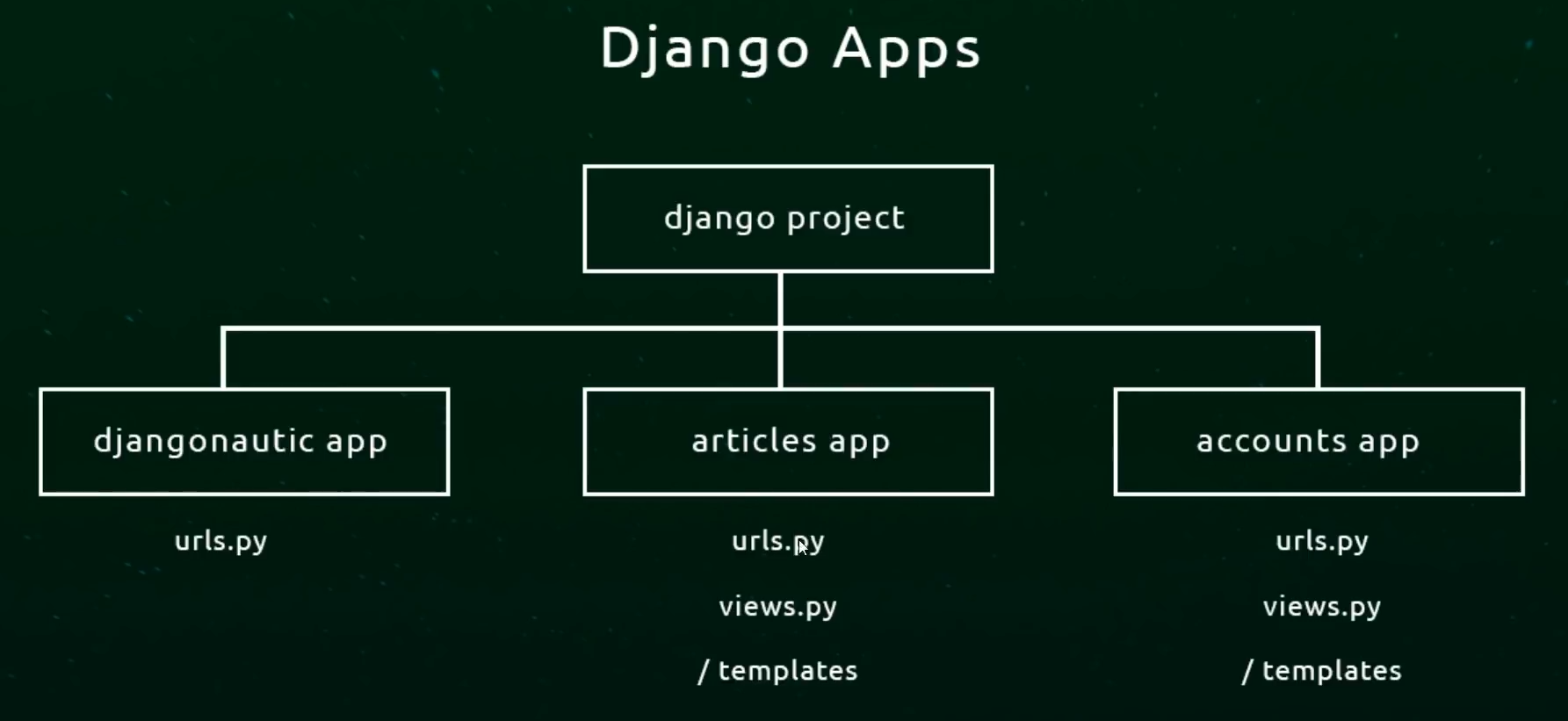


Patrzymy czy nasze htmlki są renderowane:

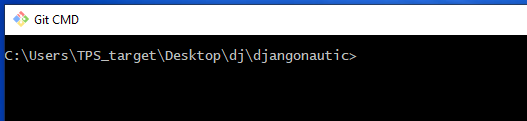


Odcinek 5:

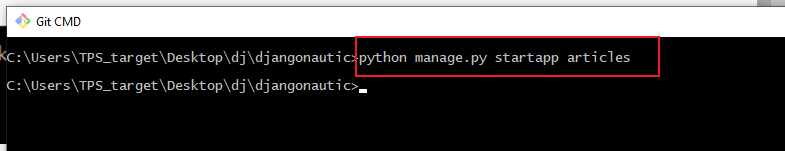
Tworzymy aplikacje w naszym projekcie. Docelowo będziemy mieli 3 aplikacje:



Tworzymy aplikację articles. W tym celu przechodzimy do „zewnętrznego” pliku djangonautic:

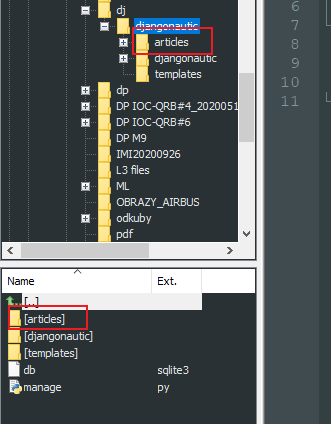


I wpisujemy komendę:

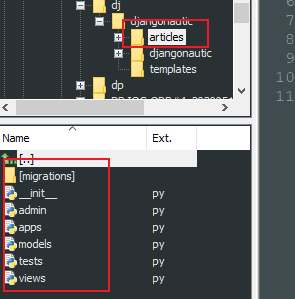


*python manage.py startapp articles*

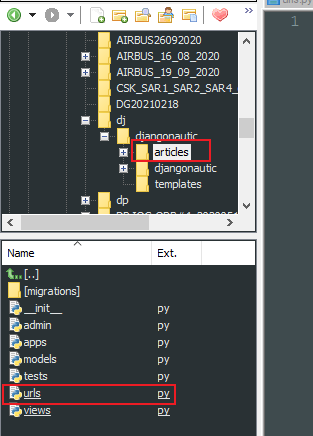
W djangonautics pojawił nam się folder articles:



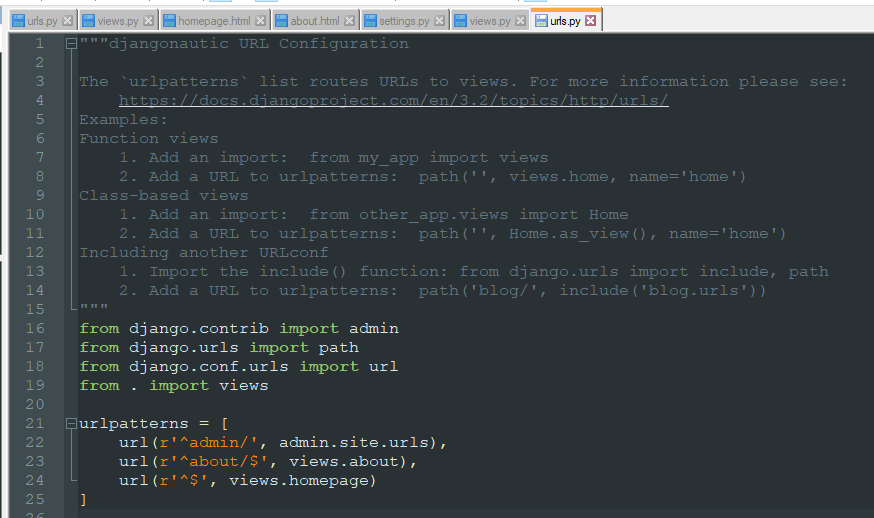
W środku mamy następujące pliki:



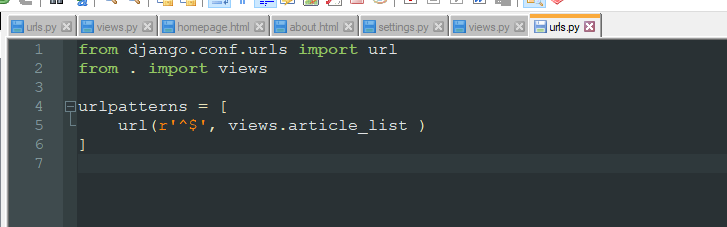
Tworzymy plik urls.py:



I kopiujemy do niego zawartość pliku urls.py z katalogu djangonautic:



Modyfikujemy tę zawartość do postaci docelowej:



*from django.conf.urls import url*

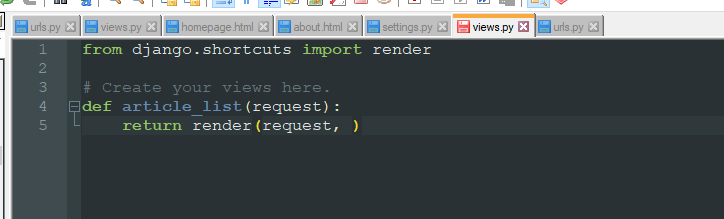
*from . import views*

*urlpatterns = [*

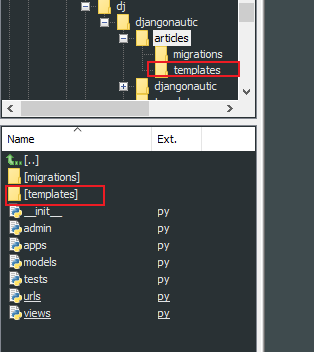
*url(r'^$', views.article\_list )*

*]*

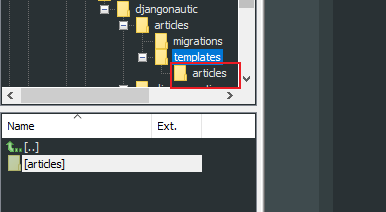
I przechodzimy do pliku views.py i wstępnie przygotowujemy kod, który ma renderować htmlkę:



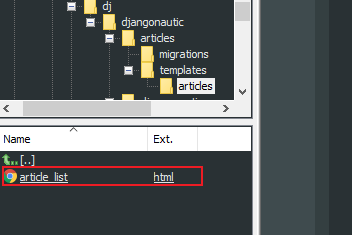
Teraz przygotowujemy folder w którym mają być templatki dla aplikacji artices. W folderze articles tworzymy folder templates:



W tym stworzonym folderze templates tworzymy z kolei folder articles:



W środku tworzymy plik article\_list.html:



Ten plik ma mieć następującą zawartość:

*<!DOCTYPE html>*

*<html>*

*<head>*

*<meta charset="utf-8">*

*<title></title>*

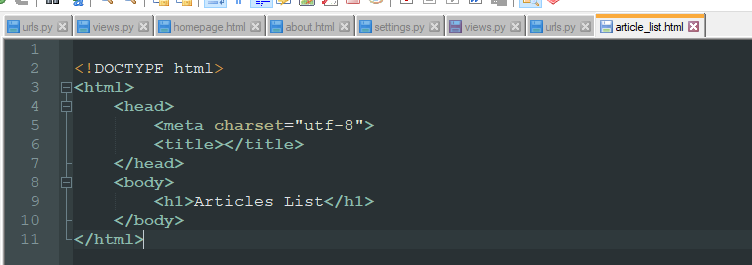
*</head>*

*<body>*

*<h1>Articles List</h1>*

*</body>*

*</html>*



Wracamy do views.py i uzupełniamy kod tak żeby renderował stworzoną przed chwilą htmlkę:



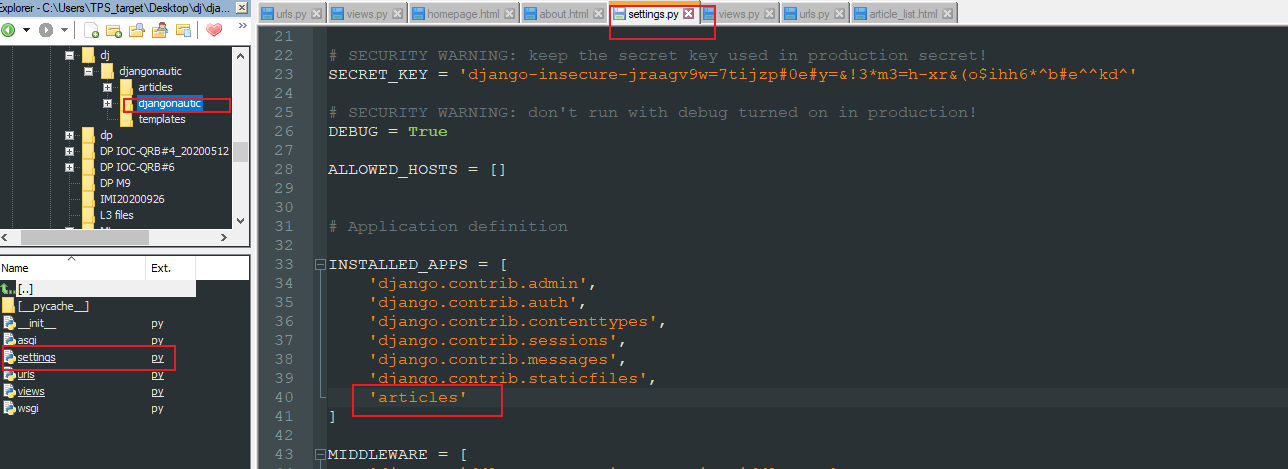
*from django.shortcuts import render*

*# Create your views here.*

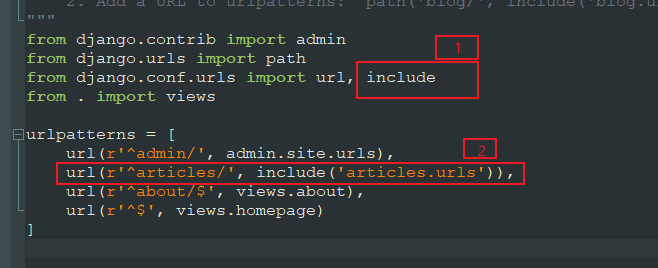
*def article\_list(request):*

*return render(request, 'articles/article\_list.html')*

Za każdym razem gdy tworzymy nową aplikację (w naszym przypadku articles) należy ją zarejestreować w projekcie. Robi się to w pliku settings.py znajdującym się w katalogu djangodautic:



Teraz chcemy spowodować, żeby po wejściu w link strony kończący się na /articles zostaniemy przekierowani do strony aplikacji articles. W tym celu musimy dokonać odpowiedniej konfiguracji w pliku urls.py znajdującym się w katalogu djangonautic urls.py. Importujemy moduł include (1), który pozwoli nam odwołać się do innej aplikacji (articles) i dodajemy linijkę w której wskazujemy link i odwołanie (2):



*from django.contrib import admin*

*from django.urls import path*

*from django.conf.urls import url, include*

*from . import views*

*urlpatterns = [*

*url(r'^admin/', admin.site.urls),*

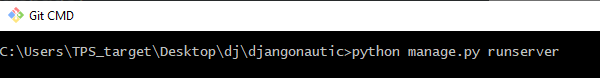
*url(r'^articles/', include('articles.urls')),*

*url(r'^about/$', views.about),*

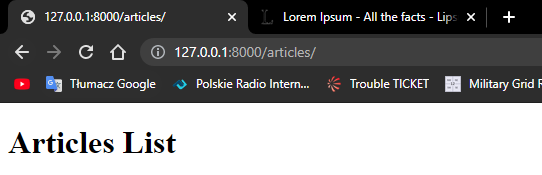
*url(r'^$', views.homepage)*

*]*

W razie potrzemy restartujemy serwer w cmd:

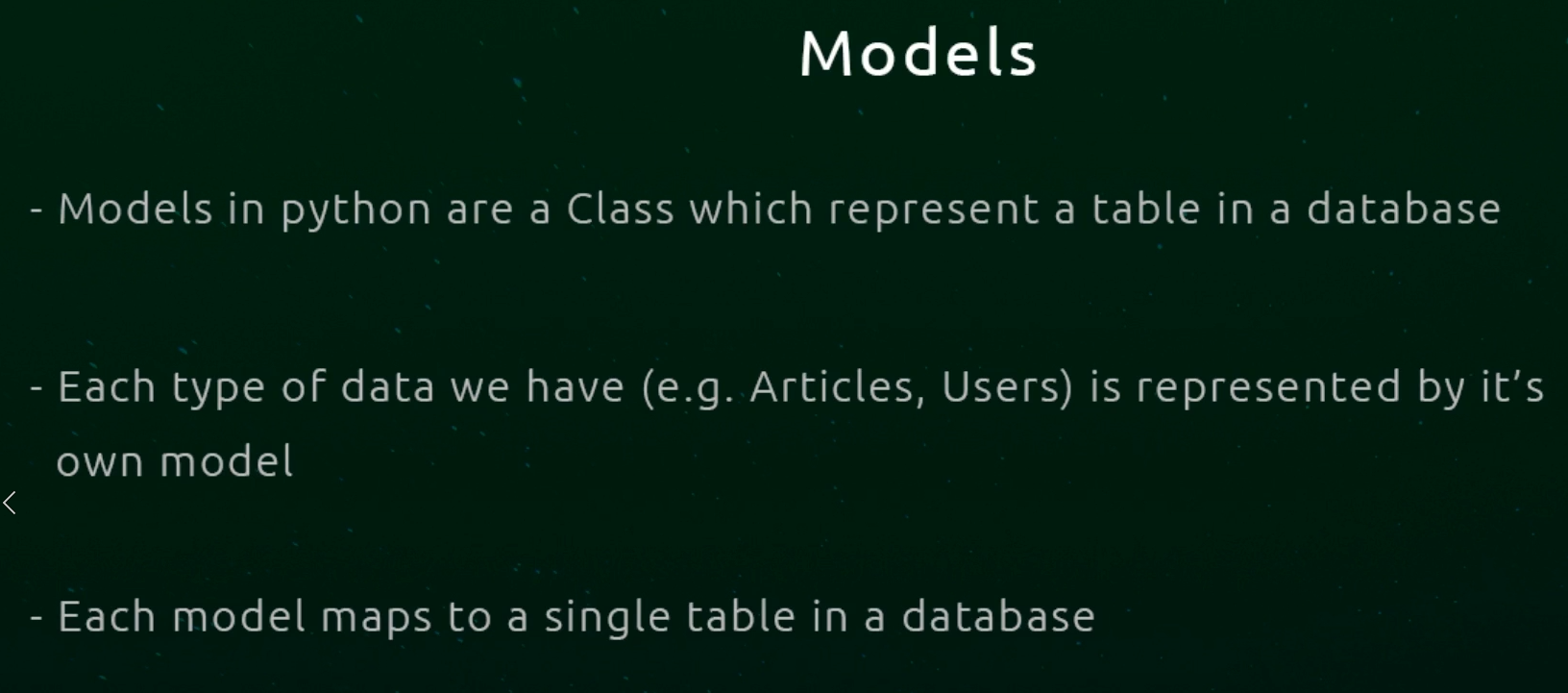


I już articles powinno działać:



Odcinek 6:

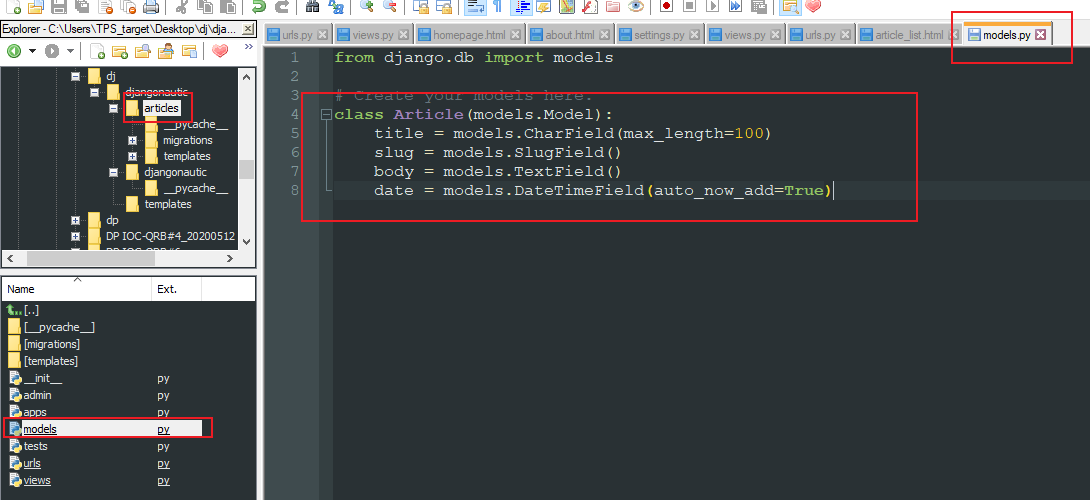
Stworzymy model czyli tabelę w bazie danych.





Później będziemy używać ORM żeby wprowadać dane do bazy.

Wchodzimy do models.py w folderze articles. Tworzymy tam model, który natępnie zostanie zmapowany na tabelę Article:



*class Article(models.Model):*

*title = models.CharField(max\_length=100)*

*slug = models.SlugField()*

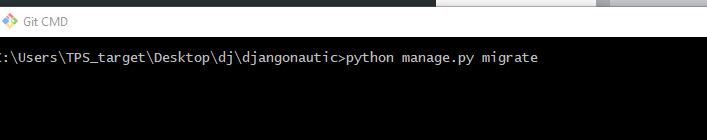
*body = models.TextField()*

*date = models.DateTimeField(auto\_now\_add=True)*

Odcinek 7:

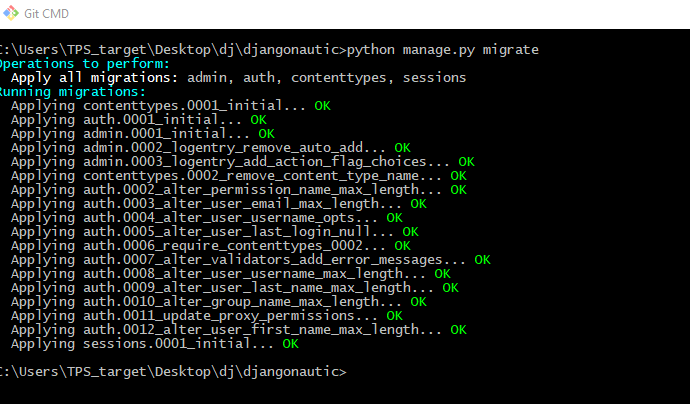
Do tej pory stworzyliśmy tylko model a tabela jeszcze nie istnieje. Trzeba teraz zrobić migrację, która spowoduje stworzenie tabeli.

W cmd wykonujemy następującą komendę:

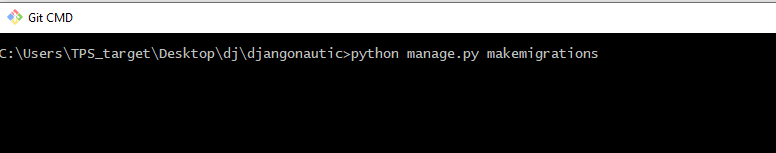


*python manage.py migrate*

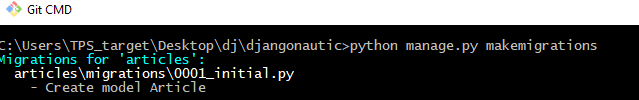
Poniżej wynik:



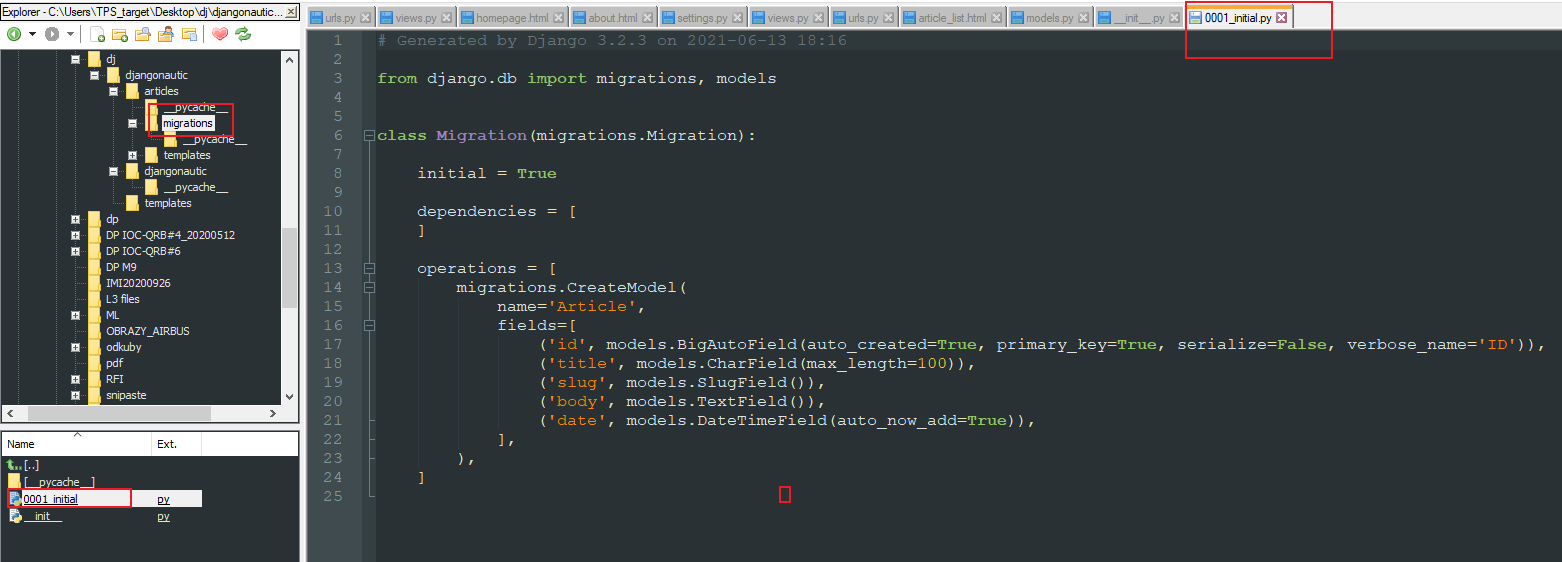
Tworzymy plik migracji:



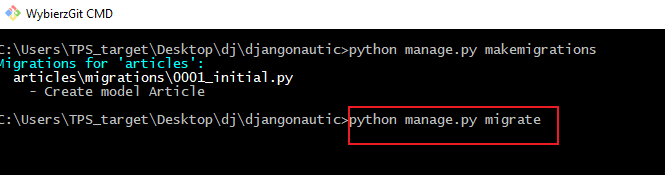
Wynik:



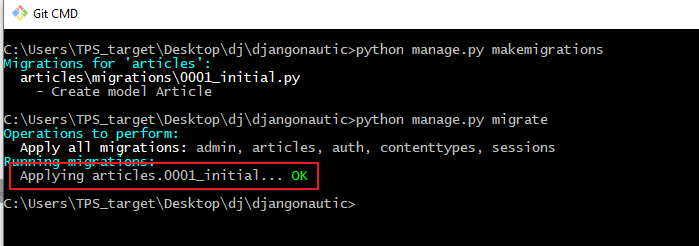
W katalogu migrations powstał taki plik:



Następnie ponownie trzeba zmigrować ten nowo powstały plik:



Wynik:

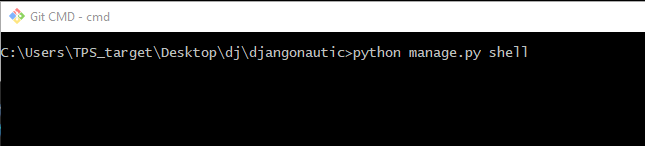


Model został zmigrowany. Za każdym razem przy zmianie trzeba będzie robić plik za pomocą komendy makemigrations a następnie migrate.

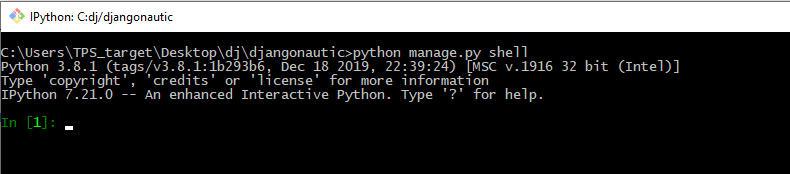
Odcinek 8:

ORM – interakcja z bazą danych.

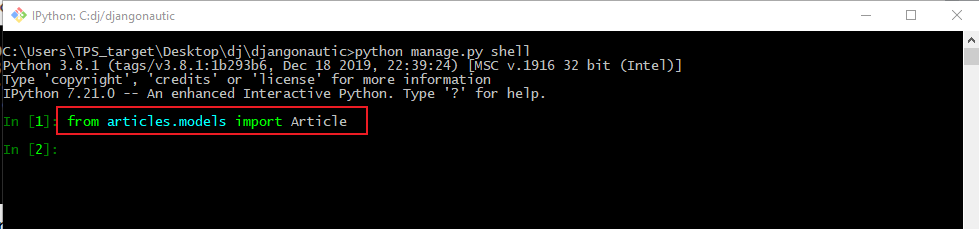
Otwieramy w cmd „interactive shell”:



Wynik:

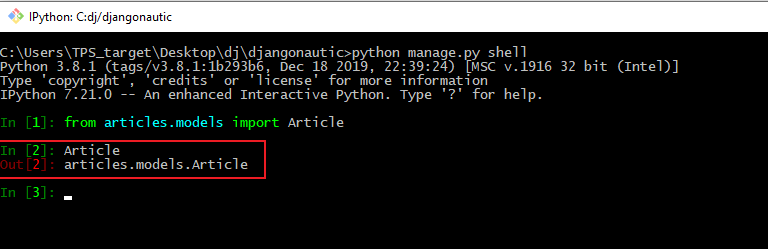


Imporujemy tabelę Article:

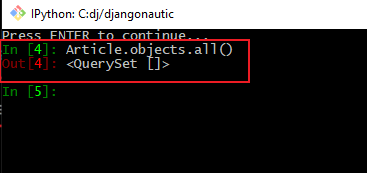


*From articles.models import Article*

Teraz możemy przywołać Article:

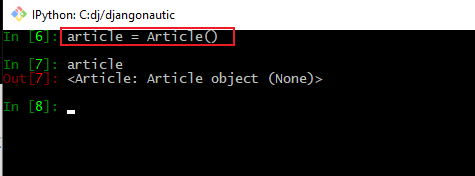


Chcemy zobaczyć wszystkie rekordy w tej tabeli:



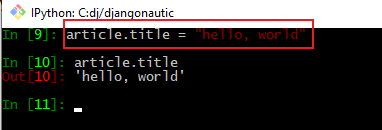
*Article.objects.all()*

Tabela jest póki co pusta więc komenda nic nam nie zwróciła. W związku z tym dodamy rekord. Tworzymy obiekt klasy Article:



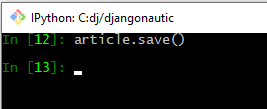
*article = Article()*

W polu title dodajemy „hello world”:



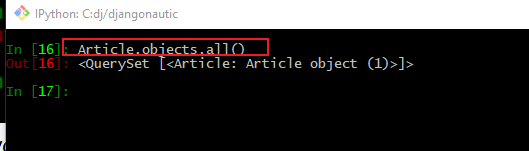
*article.title = „hello world”*

Teraz zapisujemy w bazie obiekt article:



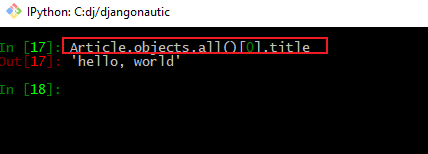
*article.save()*

Teraz sprawdzamy jakie obiekty klasy Article mamy:



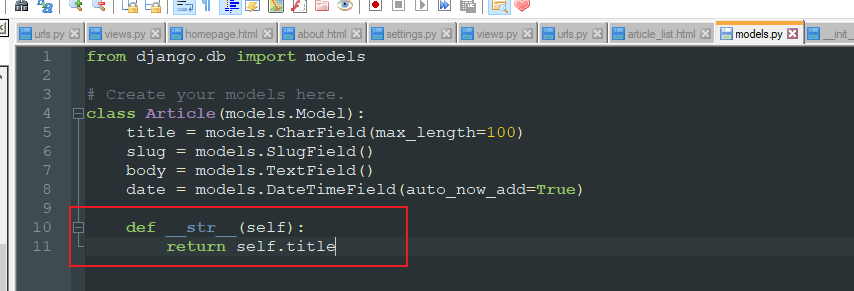
*Article.objects.all()*

Możemy odwołać się do pierwszego obiektu i jego konkretnego parametru:



*Articles.objects.all()[0].title*

Możemy zrobić tak, że po odwołaniu się do obiektu będzie się nam wyświetlał konkretny jego parametr, np. title. W tym celu wchodzimy do models.py i dodajemy następujący kod:

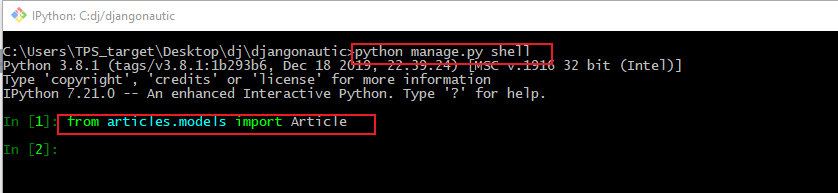


*def \_\_str\_\_(self):*

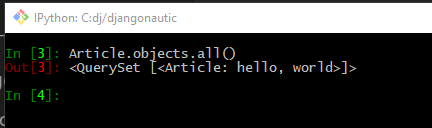
*return self.title*

Uważaj zeby zrobić wcięcia tak, żeby metoda \_\_str\_\_ należała do klasy Article!

Musimy teraz ponownie uruchomić shell i zaimportować model Articles:

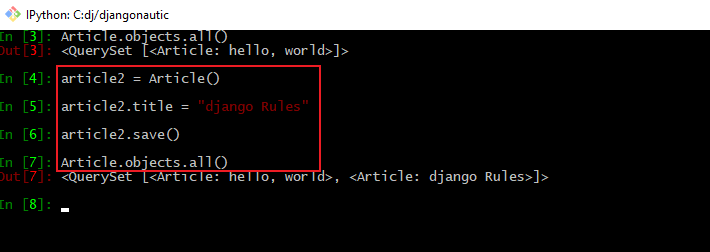


A następnie odwołujemy się do obiektów klasy Article:



I widzimy, że wyświetla nam się tytuł.

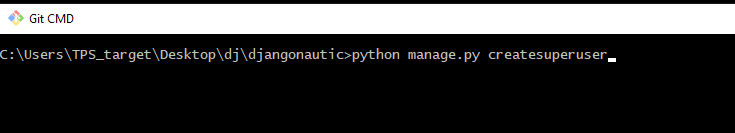
Dodajemy kolejny rekord:



Odcinek 9:

Django Admin.

Konfigurujemy konto administarcyjne:



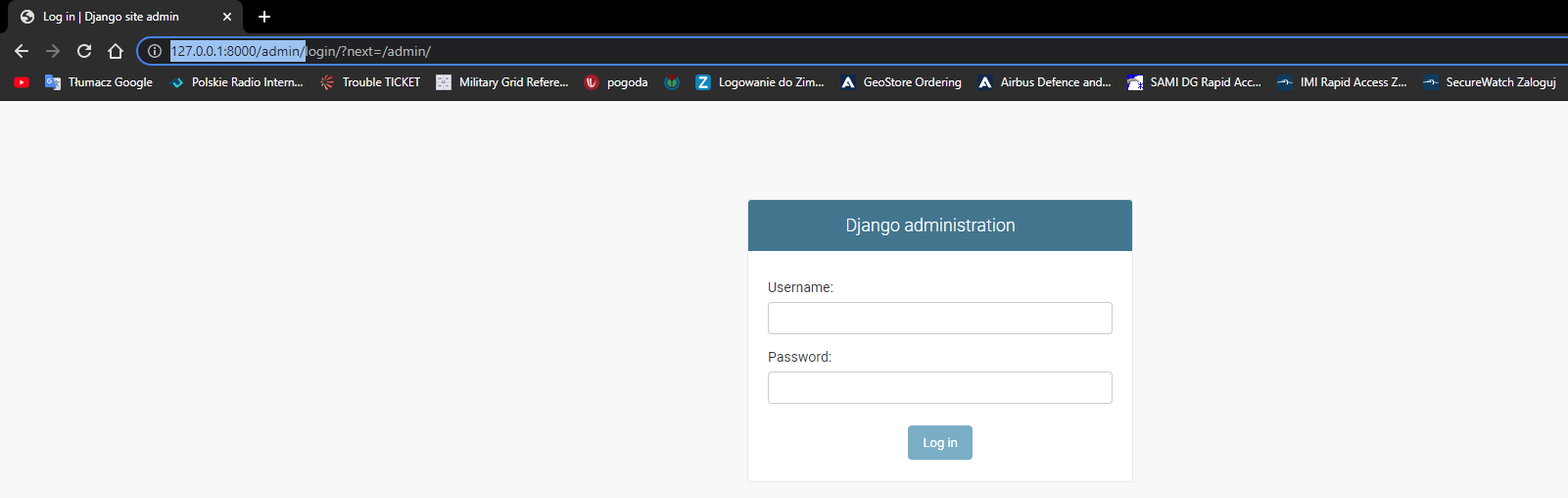
*python manage.py createsuperuser*

Username: admin

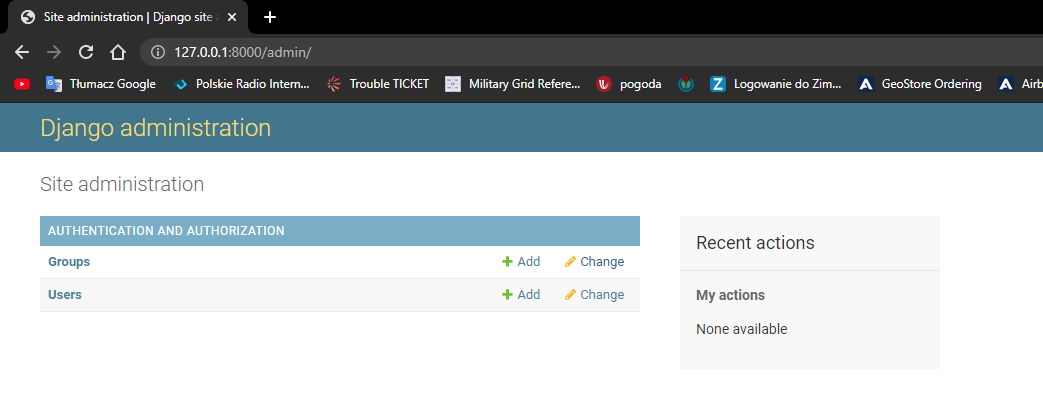
Mail: [admin@admin.com](mailto:admin@admin.com)

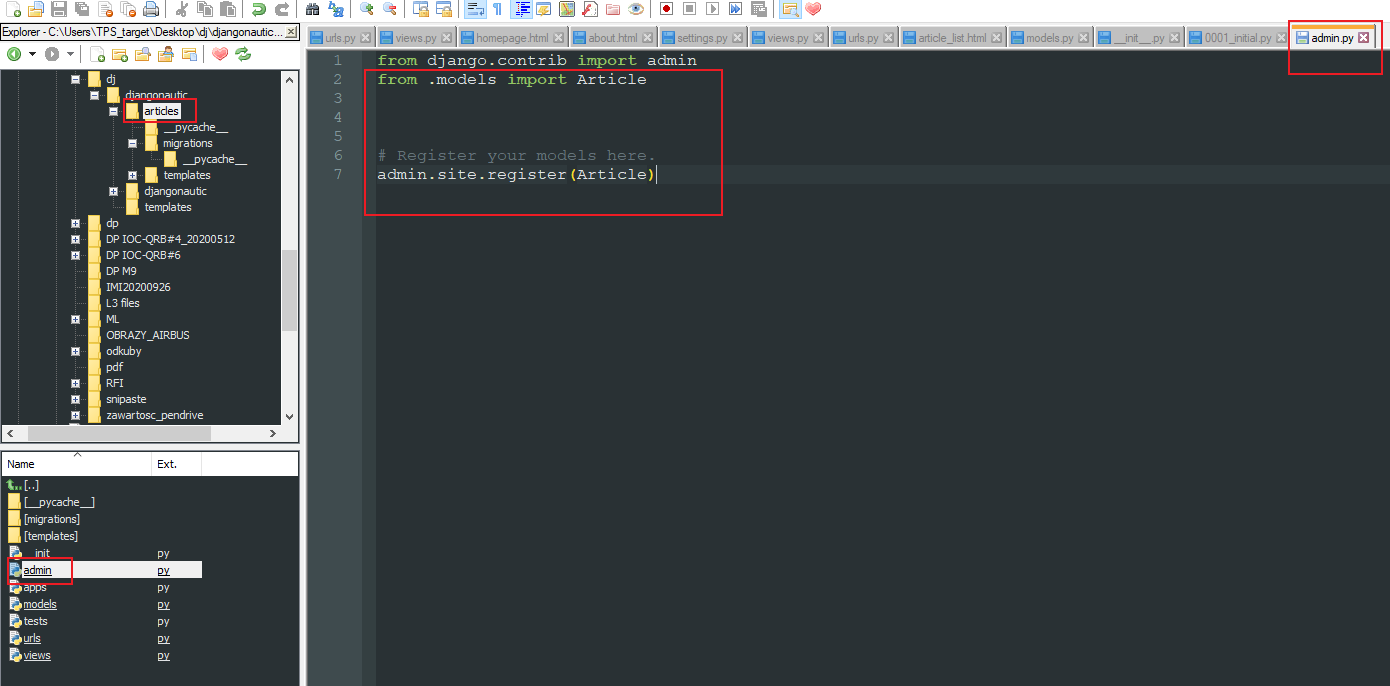
Password: admin

Przechodzimy do <http://127.0.0.1:8000/admin/>:



I logujemy się wcześniej wprowadzonymi credentialami. Tak wyglada nasze admin area:



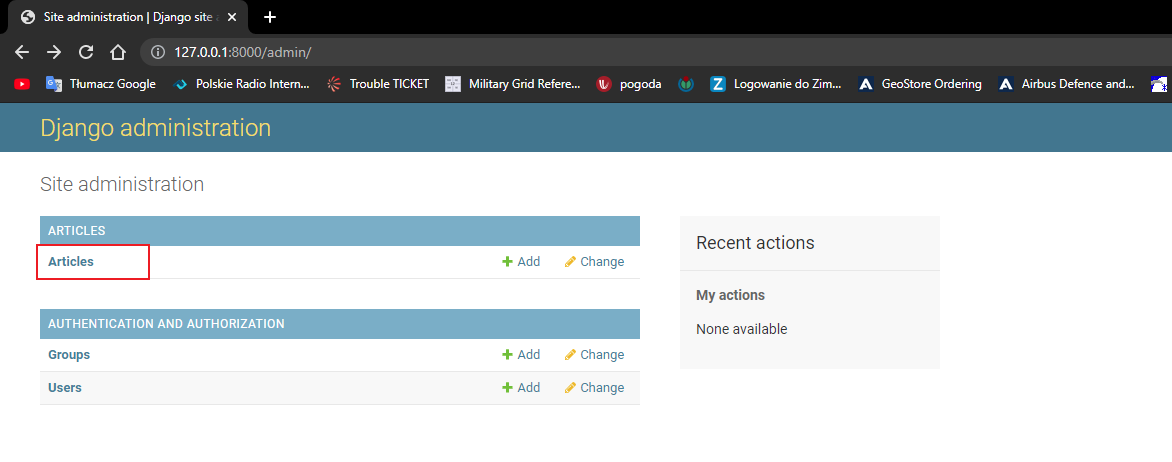
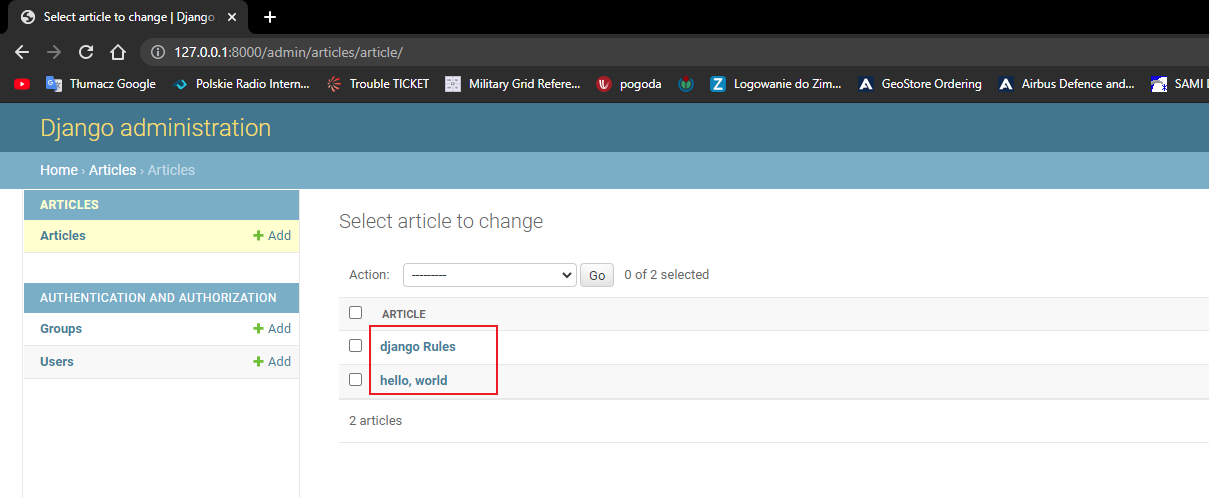
Chcemy żeby pojawił się nasz moduł Articles. W tym celu przechodzimy do pliku admin.py w katalogu articles i rejestrujemy w tym pliku moduł Articles: 

*from .models import Article*

*# Register your models here.*

*admin.site.register(Article)*

Po kliknięciu w Aricles na stronie administracyjnej widzimy nasze artykuły:

Widzimy ich tytuły bo wcześniej w pliku models.py skonfigurowaliśmy, że to właśnie tytuły mają się wyświetlać:

