**Django - szkolenie 1**

**Pierwsze uruchomienie aplikacji**

**Wstęp**

Muszę uroczyście przyznać, że wraz z tym plikiem, rozpoczynamy nową serię szkoleń związaną z rozbudowanym, jak i niezwykle użytecznym frameworkiem Django. Jeżeli przerabiasz już ten plik, to zakładam, że znasz podstawy związane z budowaniem mikroserwisów oraz tworzeniem funkcjonalnych aplikacji webowych (choćby dzięki poznaniu Flaska).

Poznanie Django na pewno przyniesie Ci dużo korzyści. Zobaczysz, jak efektywnie budować większe funkcjonalne aplikacje. Co więcej, szkolenie to będę opracowywał w formie kroków prowadzących do zbudowania pierwszego większego projektu. Tak więc wraz z zakończeniem całej serii, będziesz miał do dyspozycji w pełni użyteczny serwis (funkcjonujący jako blog)! Aby podsycić jeszcze w Tobie ciekawość - wiedziałeś, iż takie strony jak: Instagram, Mozilla czy Bitbucket są zbudowane właśnie w oparciu o Django? Nieźle, co?

Zanim przejdziemy do konkretów związanych z nawigacją po Djangowym projekcie i implementowaniu pierwszych funkcjonalności, chciałbym zrobić overview tego frameworka i porównać go z dotąd poznanym Flaskiem.

**Flask czy Django?**

Najprościej ujmując - Django sprawdzi się przy budowaniu bardziej wymagających projektów niż Flask. A to choćby dlatego, że Django dostarcza nam wiele wbudowanych już funkcjonalności. Gdybyś natomiast chcieli korzystać z nich podczas tworzenia flaskowego rozwiązania, musielibyśmy doinstalowywać wiele zewnętrznych blbliotek.

Przykładem może być tutaj integracja z bazą danych. Django zapewnia system migracji i ORM, które pozwalają łatwe zarządzanie bazą i dodawanie do niej rekordów. W przypadku Flaska natomiast, aby móc skorzystać z obiektowej implementacji bazy danych, konieczne jest choćby doinstalowanie biblioteki SQLAlchemy.

Idąc dalej, sytuacja ma się podobnie przy systemach autoryzacji. Django ułatwia zarządzanie użytkownikami i ich sesjami, gdzie Flask wymaga od nas doinstalowanie, np. JWT.

Aby nie być jednak tak bardzo stronniczym, powinieneś wiedzieć, iż Django raczej nie sprawdzi się przy projektowaniu dynamicznie zmieniających się pod względem wymagań aplikacji. Tutaj górować będzie Flask i jego łatwość w rozwijaniu i tworzeniu prostych projektów. Flask nie bez przyczyny jest nazywany przecież mikroframeworkiem. Posługując się tym frameworkiem, sprawny programista tak naprawdę “siada i od razu tworzy funkcjonalność”. Django wymaga nieco więcej konfiguracji i ustrukturyzowania plików w projekcie (co przekłada się również na większą ilość koniecznego do napisania kodu)...

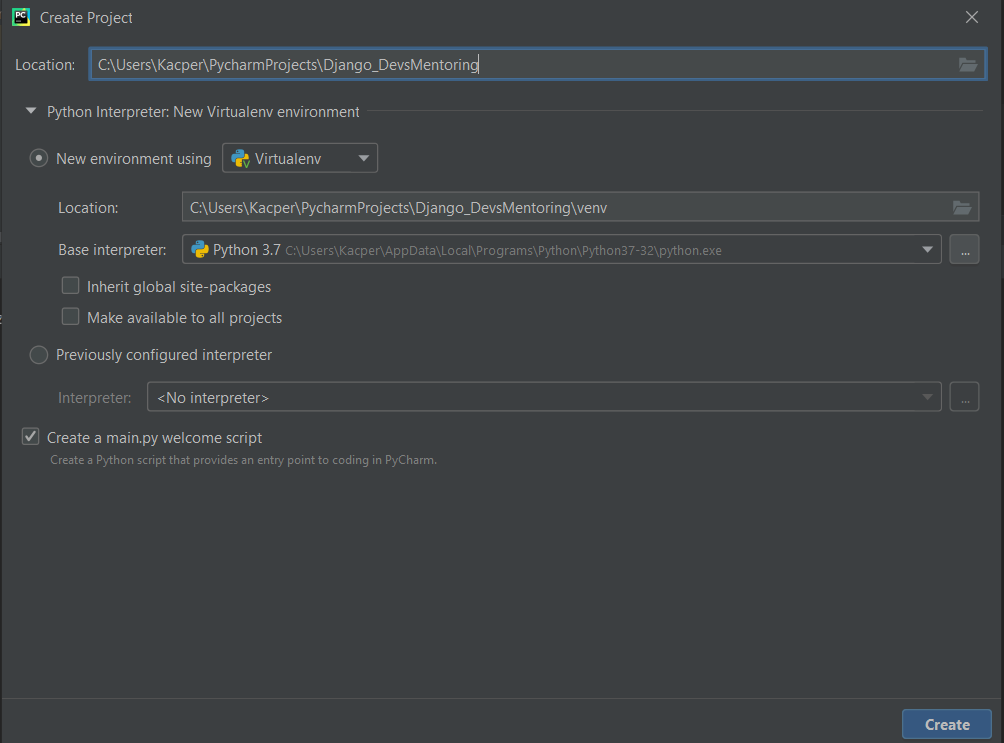
| **Django** | **Flask** |
| --- | --- |
| Łatwy dostęp do rozszerzeń | Konieczność instalacji rozszerzeń z zewnątrz |
| Wbudowany system ORM zapewnia łatwe zarządzanie bazą | Możliwość doinstalowania zewnętrznych modułów bazodanowych, zarówno w systemie ORM jak i non-ORM |
| Django jest lepszym rozwiązaniem dla większych projektów | Flask jest łatwy w obsłudze, zapewnia dużą szybkość w projektowaniu niewielkich projektów |
| Łatwy dostęp do panelu admina, skąd możemy kontrolować połączenie z bazą danych | Konieczność konfigurowania wszystkiego z “niskiego poziomu” |

.

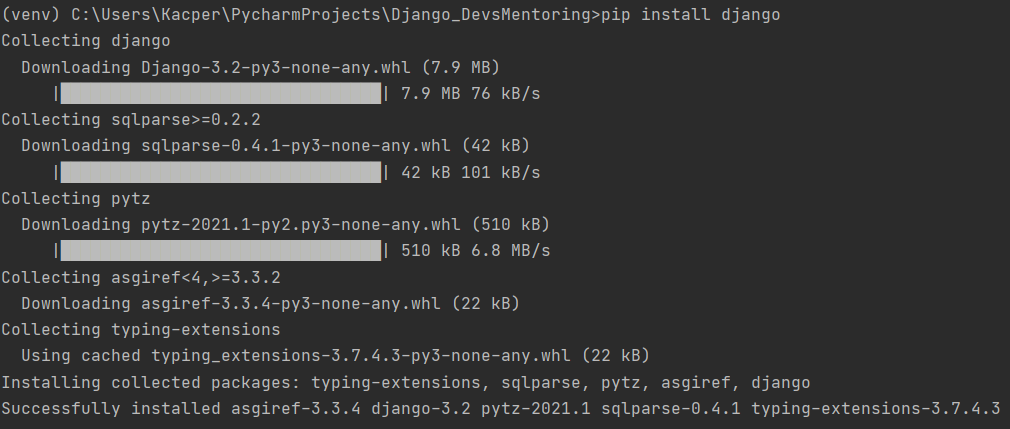
**Instalacja środowiska**

Przejdźmy do praktyki! Naukę rozpoczniemy od instalacji frameworka Django. Standardowo do budowania swojego projektu będę wykorzystywał PyCharma wraz z zintegrowanym środowiskiem wirtualnym, jednak wszystkie procesy instalacyjne będę przeprowadzał przy użyciu terminala i jego poleceń. Dzięki temu mój workflow będziesz mógł odwzorować w dowolnym IDE.

1. Utworzenie projektu



Po utworzeniu projektu i powiązaniu go z określonym venvem, w celu zainstalowania Django, wykonamy polecenie: pip install django



Zwróć uwagę na to, jaka wersja Django została u mnie zainstalowana. Konieczne jest, abyś również w swoim projekcie korzystał z co najmniej Django 3.2. Dzięki temu będziesz pewny, iż wszystkie operacje możesz przeprowadzić tak samo, jak ja w moim szkoleniu. Jeżeli chodzi o interpreter Pythona, to również powinieneś korzystać z co najmniej wersji 3.7 (możesz to sprawdzić, wpisując python --version).

**Utworzenie odpowiedniej struktury projektu**

Następnym krokiem będzie zapewnienie odpowiedniej struktury katalogów i plików w projekcie Django. Musisz być bowiem świadomy tego, iż aby móc tworzyć własne rozwiązania, framework ten przewiduje z góry określoną hierarchię katalogów w projekcie. Na szczęście, nie musimy tworzyć wszystkich plików ręcznie. Wystarczy tak naprawdę skorzystać z submodułu django-admin i wywołać na nim odpowiednie poleceniem.

Po wpisaniu polecenia django-admin, ukaże się:



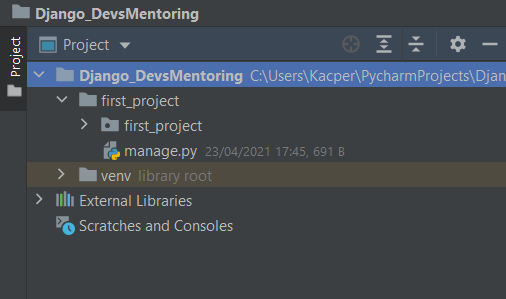
Jest to lista wszystkich poleceń, które możemy wykorzystywać w celu “zarządzania” projektem. Nie przejmuj się ich ilością. Sam nie znam wszystkich poleceń i korzystam z nich raczej wybiórczo.

Na ten moment wykorzystamy polecenie startproject. Posłuży Nam ono do wygenerowania gotowej struktury projektu.

Tak więc utwórzmy pierwszy katalog z projektem (o nazwie first\_project), wykonując polecenie:

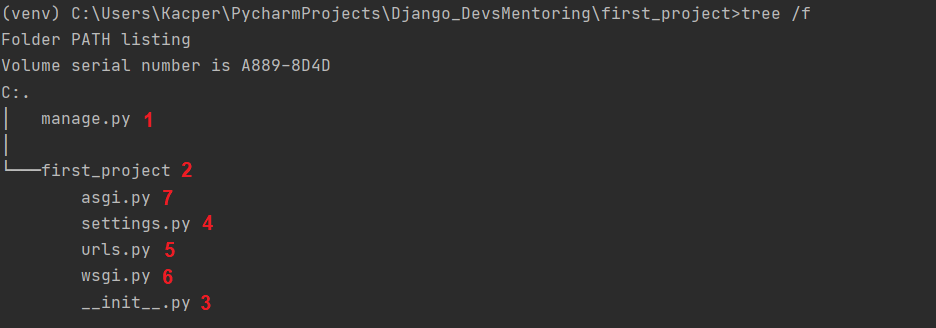
django-admin startproject first\_project

W tym momencie, w katalogu, w którym się znajdujemy (Django\_DevsMentoring), powinien znaleźć się nowy folder o nazwie first\_project z całą strukturę Django-wego projektu.



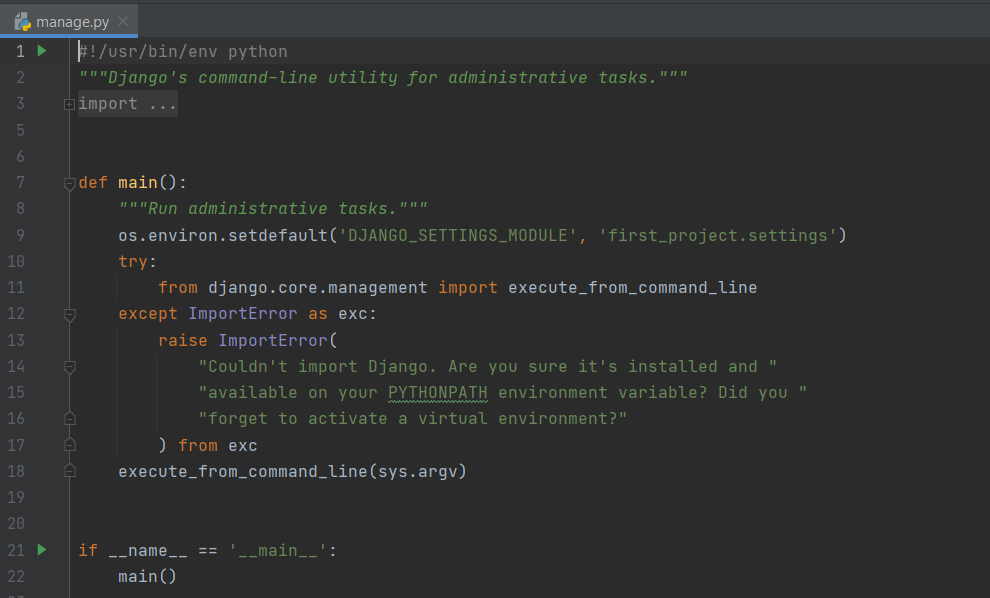
**Analiza struktury projektu**

Prześledźmy, z jakich części składa się utworzony projekt. Dla łatwiejszej analizy skorzystam z polecenia: tree /f, które wyświetli podkatalogi i pliki w formie drzewa, przez co lepiej zobrazuje sktrukturę bieżącego katalogu.



Omówienie katalogów i plików:

1. Plik manage.py



Jest to plik, którego zawartości raczej nie będziemy modyfikowali na przestrzeni rozwijania projektu. Plik ten uruchamiany jest jako pierwszy wraz ze startem aplikacji. Umożliwia prawidłową “inicjalizację” wszystkich zasobów.

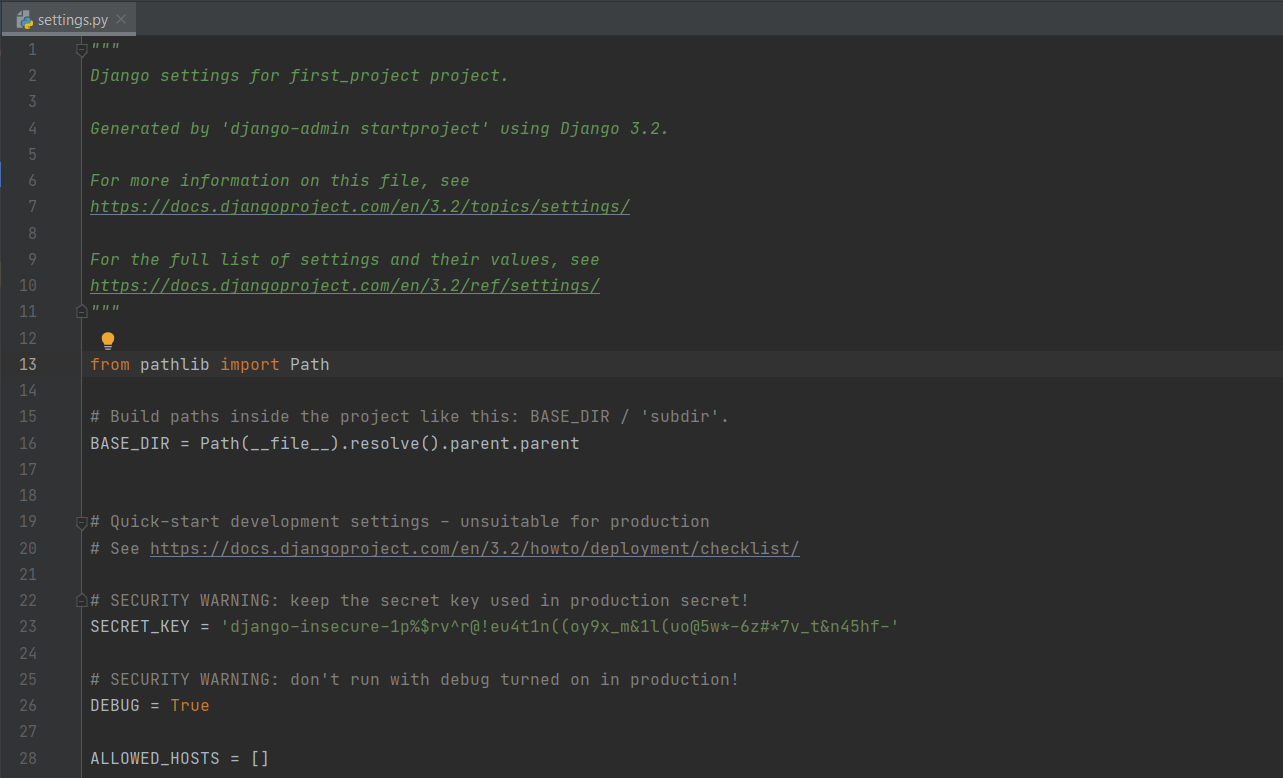
1. Katalog first\_project

To właściwa paczka będąca reprezentacją projektu. Zwróć uwagę, iż umieszczony jest on w katalogu o tej samej nazwie (w tym momencie w projekcie mamy do dyspozycji dwa katalogi first\_project - nadrzędny i podrzędny).

1. Plik \_\_init\_\_.py

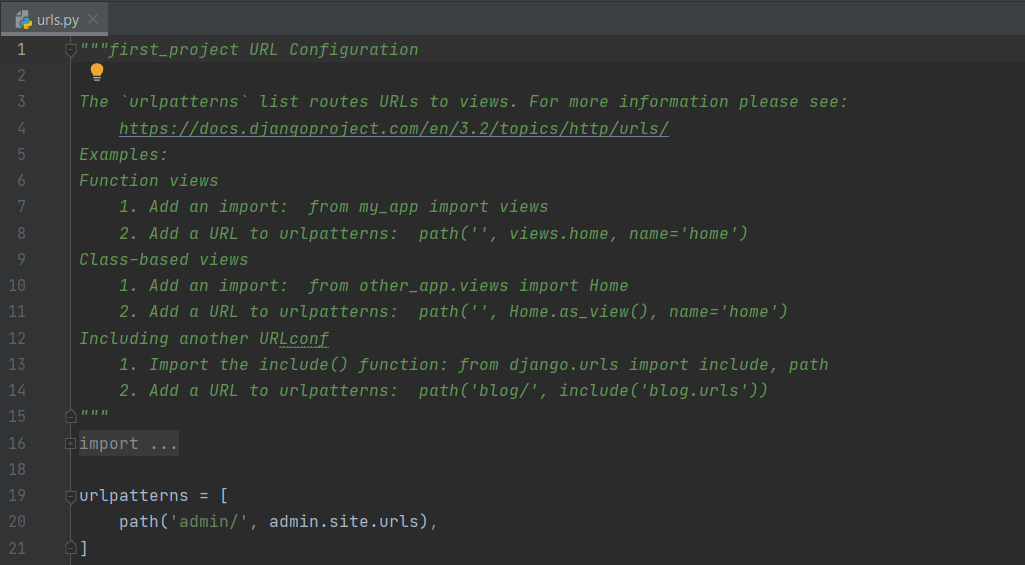
Pusty plik, który służy jedynie do zaznaczania, iż katalog w którym się znajduje (first\_project) jest Pythonową paczką.

1. Plik settings.py



Plik konfiguracyjny całego projektu. To tutaj znajdują się wszystkie właściwości projektu, które możemy zmieniać, tj. domyślnie zainstalowane moduły, rodzaj implementacji bazy danych, klucz do szyfrowania danych w sesji, ustawienia cookie, możliwe do obsługi endpointy itd. Wraz z rozwojem aplikacji, do tego pliku będziemy dodawali kolejne parametry konfiguracyjne.

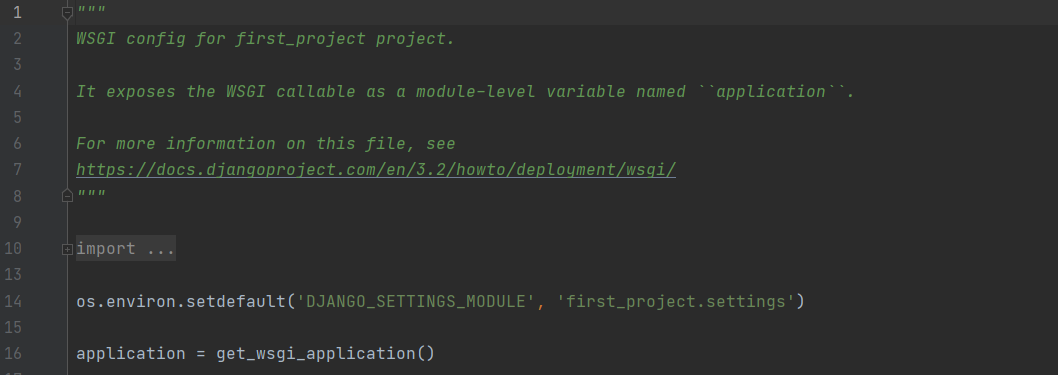
1. Plik urls.py



Miejsce dostępnych adresów URL Naszego serwisu. W pliku tym określać będziemy referencje do głównych endpointów całej aplikacji. Mam na myśli wszelki funkcjonalności, jak np. część zarządzająca użytkownikami (rejestracja, logowanie, profil), funkcjonalność blogu itd.

W pliku urs.py domyślnie mamy umieszczony adres admin/, który przekierowuje Nas nas do panelu administratora i umożliwia z jego poziomu zarządzanie, np. bazą danych.

1. Plik wsgi.py

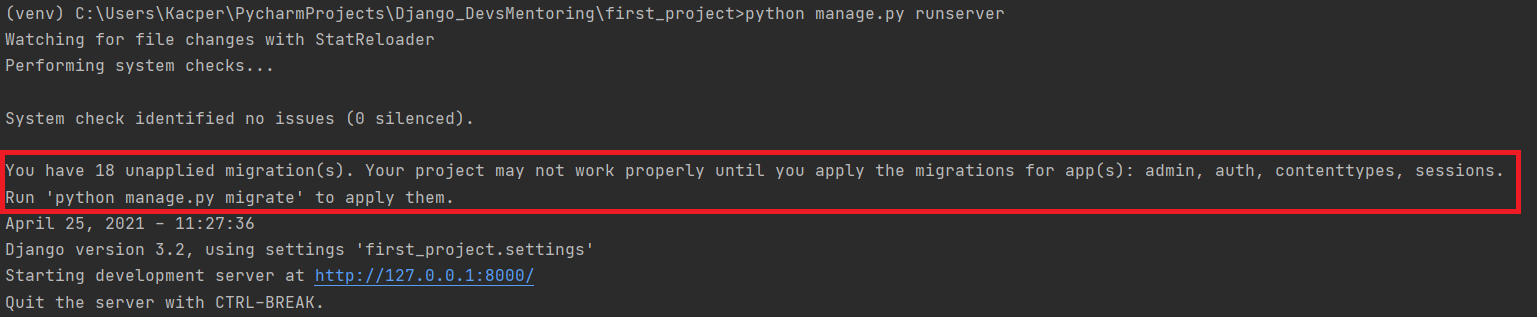


Domyślna konfiguracja Web Server Gateway Interface w naszym projekcie. Jednym słowem - to specyfikacja, w jaki sposób serwer ma komunikować się ze swoimi aplikacjami oraz jak aplikacje mają się zachowywać przy obsłudze danego requestu. Wprowadzanie zmian do tego pliku również będzie należało do rzadkości.

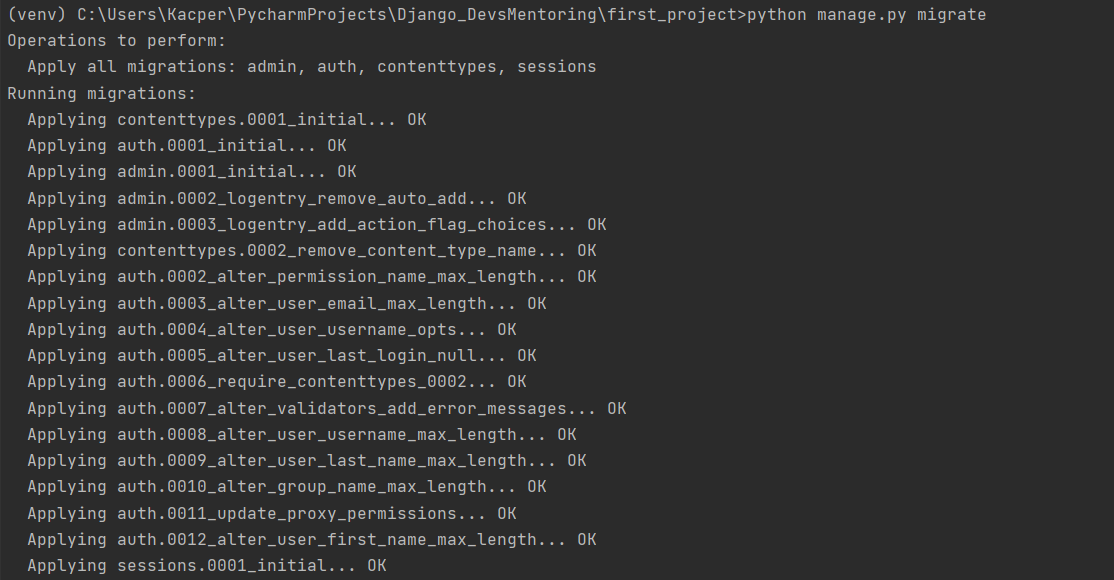
**Uruchamianie aplikacji**

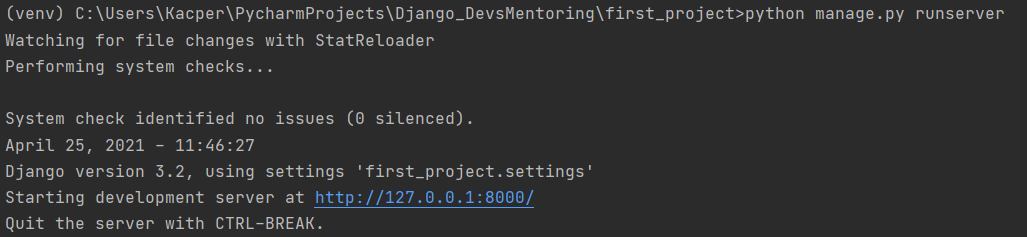
Teraz uważaj - za chwilę pokażę Ci chyba jedną z najczęściej wykorzystywanych komend przy pracy z Django. Jest to polecenie umożliwiające uruchomienie aplikacji. Jak już wiesz, to właśnie w pliku manage.py ma się odbywać inicjalizacja wszystkich zasobów, dlatego też to właśnie ten plik będzie przez Nas uruchamiany. Uruchomienie pozostałych część aplikacji Django zadzieje się automatycznie.

Znajdując się w katalogu first\_project, wykonajmy polecenie: python manage.py runserver

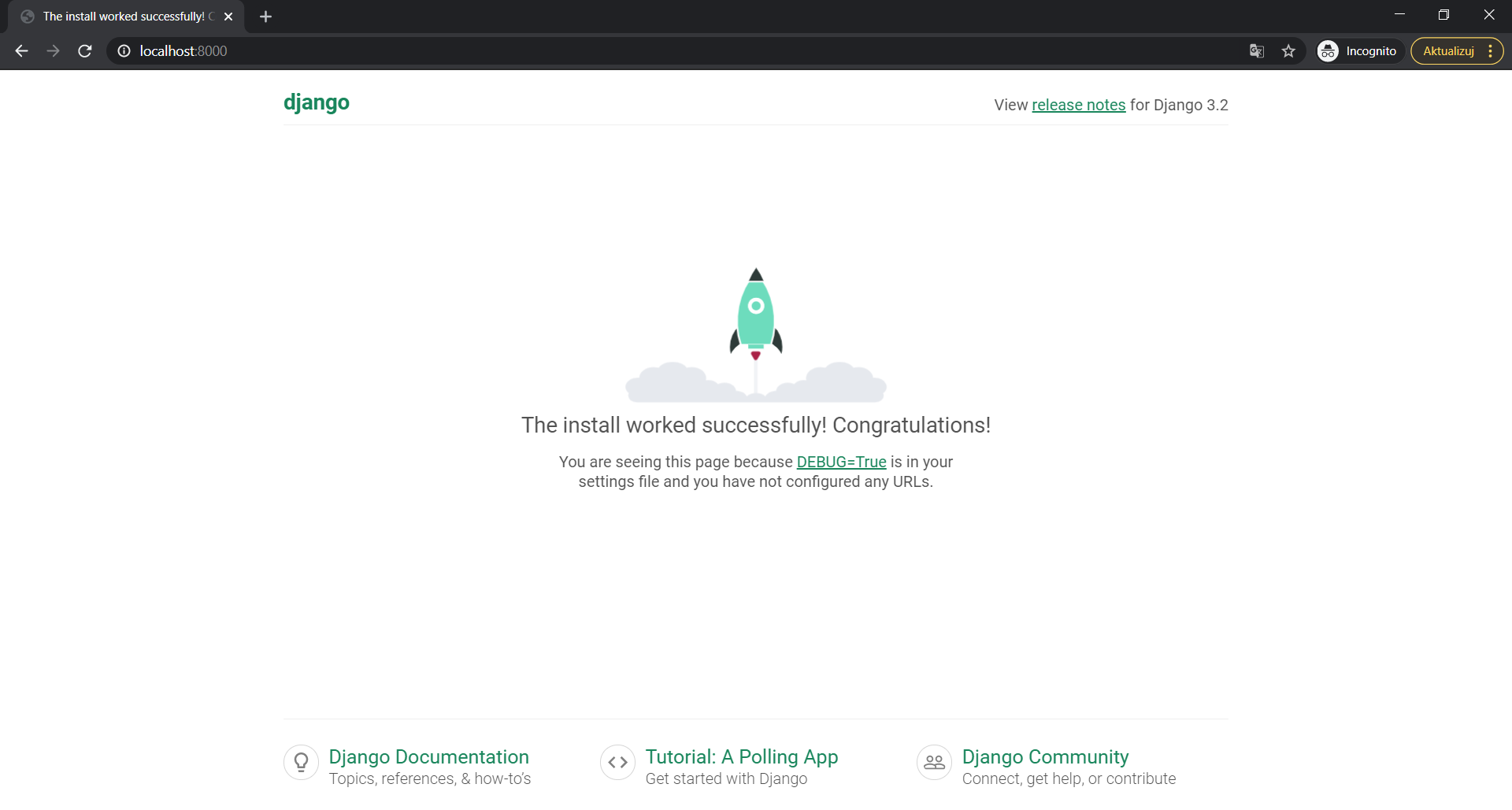


Po uruchomieniu jednak zwróć uwagę na zaznaczony komunikat. Czasami po zmianach w strukturze projektu, potrzebne będzie wywołanie dodatkowego polecenia, jakim jest python manage.py migrate. A to wszystko po to, aby “zmigrować” zmiany i zapewnić prawidłowe działanie aplikacji. Wykonajmy zatem to polecenie i spróbujmy ponownie uruchomić aplikację (aby zamknąć aktualnie wykonywaną aplikację, wciśnij CTRL + C)





Przejdźmy na wskazany adres: localhost:8000, a Naszym oczom ukaże się:

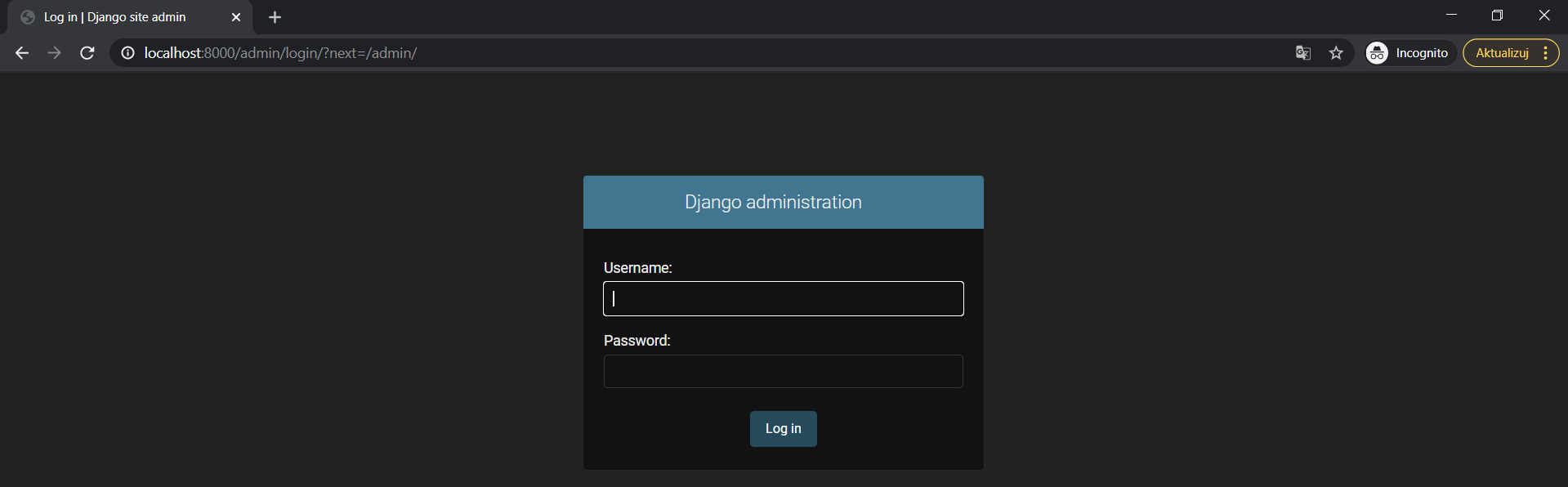


Gratulacje! Uruchomiłeś swoją pierwszą aplikację Django. Jesteśmy gotowi do tworzenia czegoś o wiele większego!

**Zakończenie**

W następnej sekcji szkolenia, chciałbym przede wszystkim omówić, w jaki sposób dodawać nowe endpointy do aplikacji. Zanim jednak zakończymy ten plik szkoleniowy, chciałbym wrócić jeszcze na chwilę do pliku urls.py w naszym projekcie. Pamiętasz, że znajduje się w nim domyślnie umieszczona ścieżka **admin/**?

Sprawdźmy, dokąd Nas przeniesie, gdy wpiszemy ten adres w przeglądarce (**localhost:8000/admin**)



Jest to panel logowania do konta administratorskiego. Na ten moment nie możemy się do niego zalogować, ponieważ proces ten wymaga nieco więcej konfiguracji (którą zajmiemy się niebawem). Na ten moment, chciałbym Cię tylko uświadomić, że dostęp do takiego panelu zapewnia łatwe zarządzanie Naszą projektową bazą danych. Zarządzanie rekordami - ich usuwanie, dodawanie może w pełni odbywać się z poziomu GUI, a nie kodu Pythona.