

Sprawozdanie nr 04

Temat:

Teoria:

Django jest frameworkiem przeznaczonym do tworzenia aplikacji internetowych, który bazuje na języku Python. Wydany on został na 3-klauzulowej licencji BSD. Dostarcza on strukturę aplikacji oraz ogólny mechanizm jej działania. Pomaga on w wypadku, gdy chcemy napisać stronę, ale nie chcemy lub nie mamy czasu na budowę strony od podstaw. Zapewnia on również dodatkowe funkcjonalności przyspieszające prace nad projektem, systemy zabezpieczeń oraz łatwe skalowanie.

W praktyce korzysta z niego wiele stron internetowych, z których, do tych bardziej rozpoznawalnych należą m. in. Instagram, Pinterest czy Mozilla.

API (Application Programming Interface) jest zbiorem reguł, dzięki którym programy lub podprogramy mogą komunikować się z jakąś usługą w określony sposób. API decyduje o tym, w jaki sposób użytkownik może się komunikować z systemem, określa reguły jak użytkownik może uzyskać dostęp do zasobów oraz w jakiej postaci je otrzymuje. W web development jednym z częściej spotykanych rodzajów API jest REST API (REST – Representational State Transfer) – sytuacja, w której back-end i front-end są osobno napisane, a komunikują się ze sobą za pomocą API napisanym w back-endzie. Całe REST API opiera się na protokole HTTP i metodach tam zawartych takich jak np. GET czy POST. REST więc jest stylem architektury, który definiuje kształt API.

Przebieg zadania:

Na samym początku należy zainstalować Django REST framework w naszym wirtualnym środowisku, wykonujemy to poleceniem `pip install`:

```
D:\Projekt\Sprawozdanie 4\Aplikacja>py -m venv venv

D:\Projekt\Sprawozdanie 4\Aplikacja>venv\Scripts\activate.bat

(venv) D:\Projekt\Sprawozdanie 4\Aplikacja>pip install djangorestframework
Collecting djangorestframework
  Using cached https://files.pythonhosted.org/packages/d7/fe/18d96db43d44065
```

Projekt nosić będzie nazwę CSTEams, jako że będzie on zbierał profesjonalne drużyny biorące udział w turniejach gry Counter-Strike: Global Offensive. Każda taka drużyna stanowić będzie jeden obiekt, gdzie na drużynę składają się:

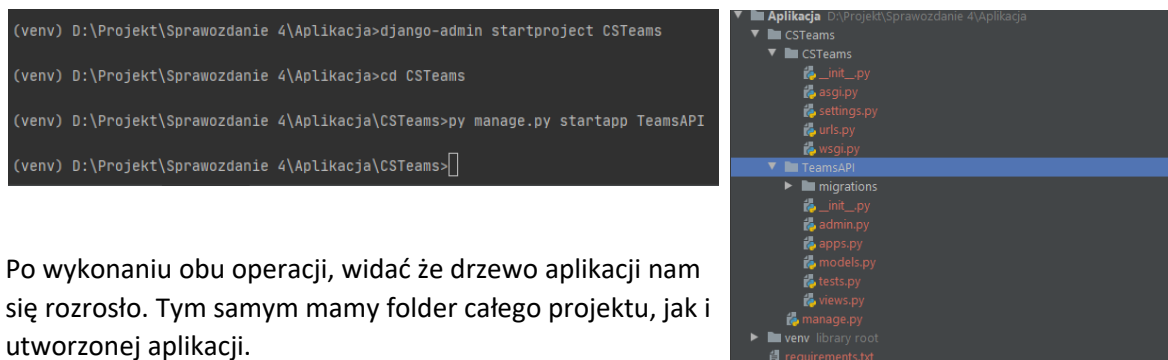
- Nazwa drużyny
- Kraj drużyny

- Trener
- Zawodnik 1
- Zawodnik 2
- Zawodnik 3
- Zawodnik 4
- Zawodnik 5

Żeby stworzyć projekt używamy polecenia: `django-admin startproject [name]` gdzie [name] oznacza nazwę projektu.

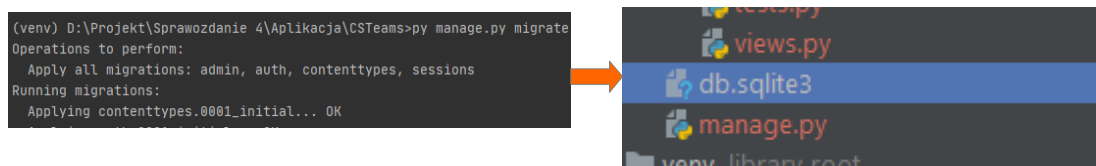
Następnym krokiem jest stworzenie samej aplikacji, w moim wypadku będzie to aplikacja pozwalająca uzyskać dostęp do powyżej zapisanych danych. Jej nazwą będzie: TeamsAPI

Stworzenie aplikacji osiągniemy komendą `python manage startapp TeamsAPI`



Po wykonaniu obu operacji, widać że drzewo aplikacji nam się rozrosło. Tym samym mamy folder całego projektu, jak i utworzonej aplikacji.

Kolejnym krokiem jest stworzenie bazy danych. Bazę danych stworzymy używając polecenia `python manage.py migrate` (tym samym poleceniem będziemy później mogli aplikować zmiany w bazie).



Ostatnim krokiem jest utworzenie administratora dla panelu administracyjnego. Służy do tego polecenie `python manage.py createsuperuser`

W celach edukacyjnych stworzone zostało konto admin z hasłem admin:

```
(venv) D:\Projekt\Sprawozdanie 4\Aplikacja\CSTEams>py manage.py createsuperuser
Username (leave blank to use 'oem'): admin
Email address: Filipiny2002@yahoo.pl
Password:
Password (again):
The password is too similar to the username.
This password is too short. It must contain at least 8 characters.
This password is too common.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```

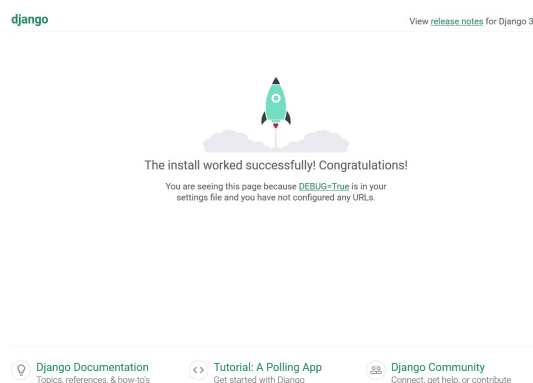
Po wykonaniu tych kroków jesteśmy gotów odpalić serwer za pomocą polecenia `python manage.py runserver`:

```
(venv) D:\Projekt\Sprawozdanie 4\Aplikacja\CSTeams>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

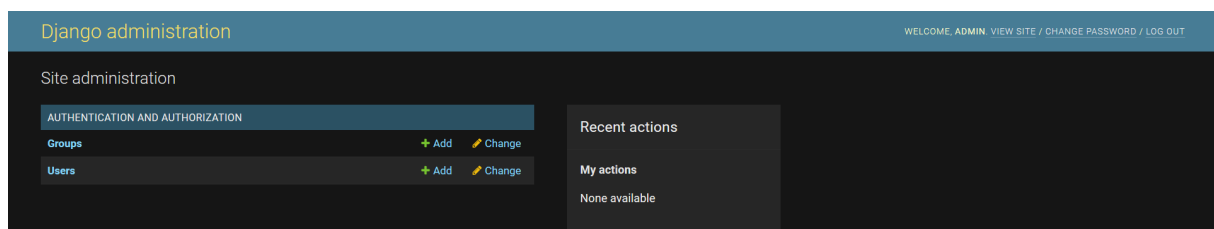
System check identified no issues (0 silenced).
May 20, 2021 - 17:23:05
Django version 3.2.3, using settings 'CSTeams.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[20/May/2021 17:24:57] "GET / HTTP/1.1" 200 10697
```

Widzimy że odpalił nam się serwer na adresie localhost na porcie 8000.

Żeby sprawdzić czy wszystko działa, wprowadzamy adres widoczny na zdjęciu do przeglądarki i jeśli widzimy następujący ekran to znaczy, że skonfigurowaliśmy projekt prawidłowo.



Wpisując adres: „<http://127.0.0.1:8000/admin/>”, możemy zalogować się do panelu administracyjnego, po logowaniu zobaczymy takie okno, w którym później będziemy mogli zarządzać danymi w naszych bazach.



W pliku `setting.py` musimy dodać do zainstalowanych aplikacji wpisy `'rest_framework'`, żeby zainstalować w naszym projekcie Django REST Framework oraz naszą aplikację `'TeamsAPI'`.

```
urls.py  models.py  settings.py
25 # SECURITY WARNING: don't run with debug turned on in production
26 DEBUG = True
27
28 ALLOWED_HOSTS = []
29
30 # Application definition
31
32 INSTALLED_APPS = [
33     'django.contrib.admin',
34     'django.contrib.auth',
35     'django.contrib.contenttypes',
36     'django.contrib.sessions',
37     'django.contrib.messages',
38     'django.contrib.staticfiles',
39     'rest_framework',
40     'TeamsAPI'
41 ]
```

Nasz model przechowujący drużyny umieszczony jest w pliku models.py, a jego deklaracja wygląda następująco:

```
class Team(models.Model):
    name = models.CharField(max_length=70)
    country = models.CharField(max_length=50)
    coach = models.CharField(max_length=70)
    p1 = models.CharField(max_length=70)
    p2 = models.CharField(max_length=70)
    p3 = models.CharField(max_length=70)
    p4 = models.CharField(max_length=70)
    p5 = models.CharField(max_length=70)

    def __str__(self):
        return self.name
```

Teraz w celu wprowadzenia naszych zmian należy wykonać migrację, w następujący sposób:

```
(venv) D:\Projekt\Sprawozdanie 4\Aplikacja\CSTeams>py manage.py makemigrations
Migrations for 'TeamsAPI':
  TeamsAPI\migrations\0001_initial.py
    - Create model Team

(venv) D:\Projekt\Sprawozdanie 4\Aplikacja\CSTeams>py manage.py migrate
Operations to perform:
  Apply all migrations: TeamsAPI, admin, auth, contenttypes, sessions
Running migrations:
  Applying TeamsAPI.0001_initial... OK

(venv) D:\Projekt\Sprawozdanie 4\Aplikacja\CSTeams>
```

Widzimy, że system sam zobaczył, że stworzyliśmy nowy model o nazwie „Team”

W tym momencie, aby móc uzyskać dostęp do modelu Team w panelu administracyjnym należy wykonać odpowiednie deklaracje w pliku admin.py:

```
1 from django.contrib import admin
2 from .models import Team
3
4 # Register your models here.
5
6
7 admin.site.register(Team)
8
```

Teraz możemy zarządzać tą tabelą z poziomu panelu administracyjnego:

The screenshot shows the Django admin interface. At the top, there's a 'Site administration' header. Below it, there's a section for 'AUTHENTICATION AND AUTHORIZATION' with links for 'Groups' and 'Users'. Then, there's a section for 'TEAMSAPI' with a link for 'Teams'. Below the 'Teams' link, there's a confirmation message: 'The team "Natus Vincere" was added successfully.' Below this, there's a form titled 'Select team to change' with an 'Action:' dropdown menu, a 'Go' button, and a list of teams. The list shows 'TEAM' and 'Natus Vincere' (selected). At the bottom, it says '1 team'.

Żeby móc przekonwertować takie dane na natywne wartości Python'a, które następnie mogą zostać łatwo przedstawione w JSON, należy stworzyć tzw. serializer.

```
CSTeams\urls.py x TeamsAPI\urls.py x models.py x admin.py x serializers.py x
1 from rest_framework import serializers
2 from .models import Team
3
4
5 class TeamSerializer(serializers.ModelSerializer):
6     class Meta:
7         model = Team
8         fields = '__all__'
9
10
```

zapis: fields = '__all__' oznacza wszystkie dostępne pola

W tym momencie, należy obsłużyć operacje na danych i w jaki sposób możemy się do nich dostać.

Obsługę danych tworzymy w pliku views.py. Możemy w nim zdefiniować funkcję dla odpowiednich wypadków, np. Funkcja team_list – wyświetlać będzie listę wszystkich drużyn w bazie lub obsłuży dodanie danych metodą POST:

```
@csrf_exempt
def team_list(request):
    if request.method == 'GET':
        teams = Team.objects.all()
        serializer = TeamSerializer(teams, many=True)
        return JsonResponse(serializer.data, safe=False)

    elif request.method == 'POST':
        data = JSONParser().parse(request)
        serializer = TeamSerializer(data=data)

        if serializer.is_valid():
            serializer.save()
            return JsonResponse(serializer.data, status=201)
        return JsonResponse(serializer.errors, status=400)
```

Funkcja team_detail z kolei obsługuje wyświetlenie pojedynczego rekordu drużyny, na podstawie klucza podstawowego. Obsługuje również dodanie/ edycję rekordu metodą PUT, jak i usunięcie rekordu metodą DELETE:

```
@csrf_exempt
def team_detail(request, pk):
    try:
        team = Team.objects.get(pk=pk)

    except Team.DoesNotExist:
        return HttpResponse(status=404)

    if request.method == 'GET':
        serializer = TeamSerializer(team)
        return JsonResponse(serializer.data)

    elif request.method == 'PUT':
        data = JSONParser().parse(request)
        serializer = TeamSerializer(team, data=data)

        if serializer.is_valid():
            serializer.save()
            return JsonResponse(serializer.data)
        return JsonResponse(serializer.errors, status=400)

    elif request.method == 'DELETE':
        team.delete()
        return HttpResponse(status=204)
```

Na samym końcu należy określić, kiedy dane funkcje mają się wykonać. Musimy więc dodać do pliku `urls.py` naszego projektu deklaracje. W tym projekcie zostało wykonane podłączenie pliku `urls.py` z `TeamsAPI`, w którym znajdują się konkretne ścieżki, więc najpierw w pliku `urls.py` w projekcie `CSTeams` dodajemy zapis o podłączeniu adresów URL z `TeamsAPI`:

```
16 from django.contrib import admin
17 from django.urls import path, include
18
19 urlpatterns = [
20     path('admin/', admin.site.urls),
21     path('', include('TeamsAPI.urls'))
22 ]
23
```

Następnie dopiero w pliku `urls.py` należącym do `TeamsAPI` określamy dokładne ścieżki i odpowiadające funkcje:

```
CSTeams\urls.py TeamsAPI\urls.py models.py
1 from django.urls import path
2 from .views import team_list, team_detail
3
4
5 urlpatterns = [
6     path('teams/', team_list),
7     path('team/<int:pk>/', team_detail),
8 ]
9
```

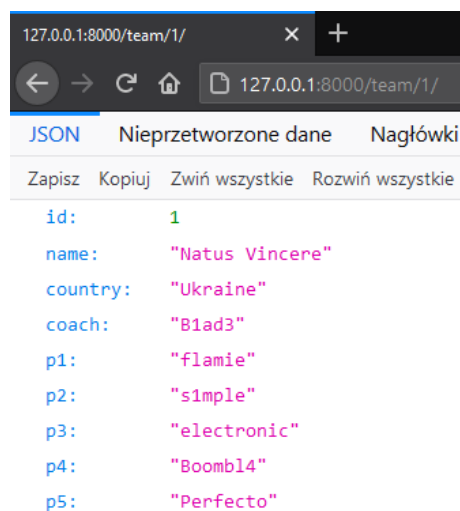
Wynik jest następujący

(ukazanie wszystkich drużyn):



adres: 127.0.0.1:8000/teams/

ukazanie konkretnej drużyny po id):



adres: 127.0.0.1:8000/team/1/

Oczywiście całą zawartością bazy możemy zarządzać z poziomu panelu administracyjnego dostępnego po adresie 127.0.0.1:8000/admin/

Select team to change

ADD TEAM +

Action: Go 0 of 3 selected

TEAM

Astralis

FaZe Clan

Natus Vincere

3 teams

Change team

HISTORY

FaZe Clan

Name:

FaZe Clan

Country:

United States

Coach:

RobbaN

P1:

rain

P2:

coldzera

P3:

broky

P4:

Twistzz

P5:

karrigan

Delete

Save and add another

Save and continue editing

SAVE