Django

Protokół http (powtórzenie)

HTTP (ang. *Hypertext Transfer Protocol*) – protokół przesyłania dokumentów hipertekstowych WWW. Najważniejsze cechy:

- Znormalizowany sposób komunikacji, niezależny od platformy
- Ma formę **transakcji**, składających się z żądania (request) oraz odpowiedzi (response)
- Jest **bezstanowy** – nie zachowuje żadnych informacji o zrealizowanej transakcji; Stanowość zapewnia najczęściej mechanizm sesji kontrolowanej przez serwer, do czego wykorzystuje się mechanizm ciasteczek (cookies) zapisywanych przez klienta

Żądanie HTTP:

```
Metoda Zasób Wersja-http
Nagłówek
Nagłówek
 [pusta linia]
Zawartość (jeżeli potrzebna)
```

Przykład:

```
GET / HTTP/1.1
Host: www.domena.com
Connection: keep-alive
```

Odpowiedź HTTP

```
Wersja-http Kod-odpowiedzi Uzasadnienie
Nagłówek
Nagłówek
 [pusta linia]
Zawartość (jeżeli potrzebna)
```

Przykład:

```
HTTP/1.1 200 OK.
Server: Apache
Content-type: text/html

[Dokument HTML]
```

Metody żądań HTTP

Zdefiniowano 8 metod (rodzajów żądań) HTTP:

- GET – pobranie wskazanego przez URI zasobu; najczęściej wykorzystywany
- POST – przesłanie danych od klienta do serwera, np. danych formularza
- HEAD – pobranie informacji o wskazanym zasobie
- PUT – umieszczenie zasobu (pliku) na serwerze
- DELETE – usunięcie zasobu z serwera
- OPTIONS – informacje o kanale komunikacyjnym
- TRACE – do celów diagnostycznych
- CONNECT – dotyczące tunelowania

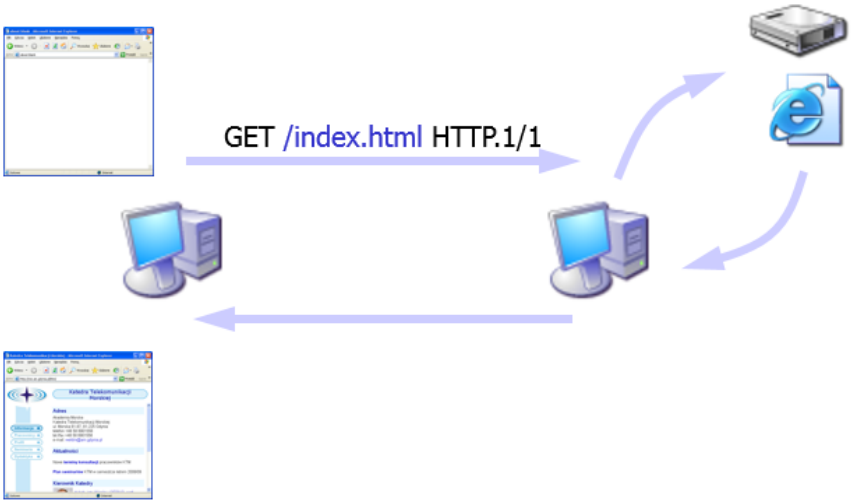
Ważniejsze kody odpowiedzi serwera HTTP:

- 200 OK – serwer przesyła żądany dokument w zawartości
- 204 No contenet – serwer zrealizował żądanie, jednak nie wymaga ono zawartości
- 301 Moved prmanently – zasób został przeniesiony
- 401 Unauthorized – dostęp do zasobu wymaga autoryzacji
- 403 Forbidden – serwer zrozumiał zapytanie, jednak dostęp do zasobu jest zabroniony
- 404 Not found – serwer nie odnalazł zasobu
- 500 Internal Server Error – serwer nie może zrealizować żądania z powodu błędów w oprogramowaniu (np. CGI)
- 501 Not implemented – metoda nie jest obsługiwana (np. PUT)
- 503 Service unavailable – usługa niedostępna (przeciążenie)

Statyczne i dynamiczne strony WWW

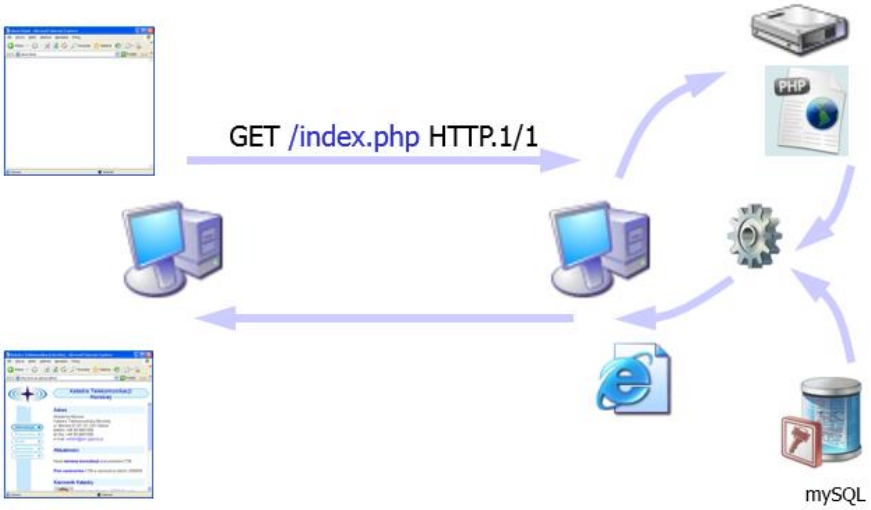
Statyczne strony WWW

- Dokumenty HTML umieszczone na serwerze; zmiana zawartości witryny WWW wymaga modyfikacji plików HTML; niewielkie obciążenie serwera



Dynamiczne strony WWW

- Serwer ma dodatkowe moduły (np. interpreter PHP);
  Zamiast plików HTML na serwerze umieszczone są skrypty generujące dokumenty HTML (np. zamiast x.html – x.php)
- Zawartość dokumentu HTML zwracanego do przeglądarki może być uzależniona od wielu czynników, np. parametrów wybranych przez użytkownika w oknie przeglądarki, zawartości baz danych, odpowiedzi innych serwerów na zapytania itp.
- Większość serwisów do składowania informacji prezentowanych na witrynie wykorzystuje bazy danych



Zastosowania:

- Portale informacyjne
  zawartość informacyjna modyfikowana odrębnymi narzędziami, zapisywana w bazie danych,
- Sklepy internetowe
  dostępne towary opisane w bazie danych, podobnie zamówienie złożone przez użytkownika,
- Bankowość internetowa
- Systemy nauczania na odległość
  materiały dydaktyczne, testy zaliczeniowe, dane instruktorów oraz słuchaczy (w tym wyniki testów) zapisane w bazie danych
- Fora internetowe, serwisy społecznościowe, serwisy Wiki, …
- Systemy CMS (*Content Management System)*

Wymagania:

- Obsługa protokołu HTTP
  odczyt i tworzenie nagłówków, obsługa formularzy HTML
- Obsługa baz danych
  wbudowane lub dołączane moduły, przynajmniej dla najpopularniejszych systemów baz danych
- Programowanie obiektowe
  najbardziej popularna metodyka programowania
- Bezpieczeństwo
  obsługa plików cookie (ciasteczek), obsługa sesji, zabezpieczenia przeciw atakom (np. SQL injection, XSS, …)

Platformy:

- Dedykowane serwery
  Programy obsługujące komunikację TCP/IP na porcie 80 TCP, dowolny język (np. Pascal/C/C++/…)
- Skrypty CGI (*Common Gateway Interface*)
  skrypty (Perl, **Python**) lub programy (dowolny język)
- PHP
  skrypty generujące HTML, z możliwością załączania HTML; wbudowane wsparcie HTTP i baz danych
- JSP (*Java Server Pages*), JSF (*Java Server Faces*)
  znaczniki (nie HTML), przetwarzane przez JSP/JSF na HTML, środowisko RAD, automatyczne powiązania z kodem (zdarzenia)
- ASP.NET (*Active Server Pages*)
  technologia MS, funkcjonalnie podobna do JSF; różne języki programowania (Visual Basic, C#, Java)

Implementacja dynamicznych i statycznych stron WWW

Statyczne strony WWW

Dokumenty HTML umieszczone na serwerze

- Konfiguracja serwera www określa folder, którego zawartość jest udostępniona przez http (żadne pliki spoza tego folderu nie mogą być przez serwer udostępnione), np.:

```
/var/www/
```

- Lokalizacja i nazwa pliku wynika wprost z żądania http (jeżeli w żądaniu jest nazwa folderu, serwer szuka w tym folderze pliku o nazwie domyślnej, zwykle index.html):

```
GET /                - /var/www/index.html
GET /style.css       - /var/www/style.css
GET /njiti/w.html    - /var/www/njiti/w.html
GET /img/logo.jpg    - /var/www/img/logo.jpg
```

- Oprócz samego żądania http, serwer uwzględnia również nagłówki http (np. nagłówek "Host: domena.com" – jeżeli na serwerze jest kilka domen)
- Serwer generuje również nagłówki odpowiedzi

Dynamiczne strony WWW - PHP

Skrypty PHP umieszczone na serwerze

- Przykład skryptu PHP:

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <?php

     echo ("<h1>Hello world!</h1>");

     ?>
  </body>
</html>
```

- Plik php zawiera html i instrukcje php – fragmenty html oraz dane "drukowane" przez php na standardowe wyjście – funkcja echo() – są wysyłane do przeglądarki
- Przeglądarka otrzymuje dokument html jak przy stronach www statycznych – nie potrzebuje nic dedykowanego dla php, żeby wyświetlać strony generowane w php
- Pliki php są jednocześnie plikami html, wszystkie zasady odnajdowania i udostępniania plików dotyczą też skryptów php: lokalizacja i nazwa pliku wynika wprost z żądania http (jeżeli w żądaniu jest nazwa folderu, serwer szuka w tym folderze pliku o nazwie domyślnej, zwykle index.php lub default.php):

```
GET /                - /var/www/index.php
GET /style.css       - /var/www/style.css
GET /njiti/w.php     - /var/www/njiti/w.php
GET /img/logo.jpg    - /var/www/img/logo.jpg
```

- Pliki skryptów php oraz wszelkie inne pliki potrzebne w serwisie (arkusze CSS, obrazki, czcionki, …) koegzystują w tym samym folderze – pliki "statyczne" są wysyłane do przeglądarki, skrypty – uruchamiane przez interpreter PHP
- Uruchamiany skrypt otrzymuje wszystkie informacje z żądania http – w tym żądany zasób oraz wszystkie nagłówki; Zasób z żądania może zawierać dane z formularza html, również nagłówki mogą zawierać dane z formularza, a ponadto ciasteczka, np.:

```
GET /find.php?str=xxx&options=strict
```

- Interpreter udostępnia wszelkie dane z żądania http (nagłówki, dane formularzy, ciasteczka, …) w postaci tablic, bardzo wygodnej dla programisty, np. tablica $_GET, zawiera dane z formularza html, tablica $_COOKIES – ciasteczka, itd.
- Do nagłówków odpowiedzi serwera są dodawane nagłówki wygenerowane przez skrypt php, np. nowe wartości ciasteczek

Aplikacje internetowe – PHP

Skrypty PHP umieszczone na serwerze

- Aplikacje internetowe php, np. systemy CMS, zmieniają domyślny sposób działania serwera przez przekierowania; W serwerze Apache służy do tego moduł mod_rewrite:

```
--- .htaccess ---
RewriteEngine On

RewriteCond %{REQUEST_FILENAME} -f
RewriteRule ^ - [L]

RewriteRule ^(*)$ /index.php?rq=$1 [QSA,NC,L]
```

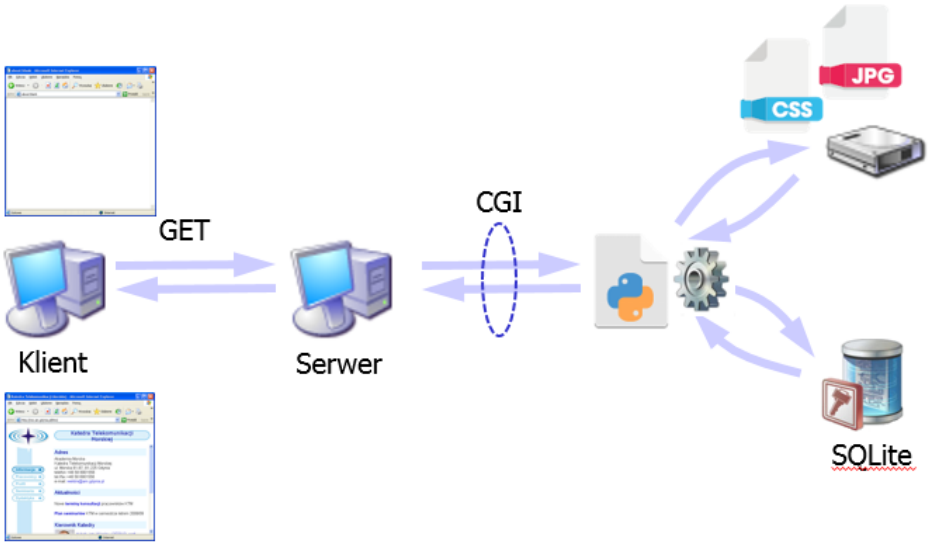
- Niezależnie co będzie w żądaniu, wykonany zostanie skrypt index.php (dane z żądania zostaną przekazane do tego skryptu)

```
GET /studenci -> /index.php?rq=studenci
```

Dynamiczne strony WWW – CGI

Dowolne programy CGI umieszczone na serwerze

- Serwer lub proxy http przekazuje – przez interfejs CGI – żądanie do programu, otrzymuje zwrotnie kompletną odpowiedź, którą odsyła do klienta
- Program generuje strony dynamiczne, najczęściej posiłkując się bazą danych;
- Program może być napisany w dowolnym języku. Często wykorzystuje się języki skryptowe (Perl, Python), stąd określenie "skrypty CGI"
- Interfejs CGI określa sposób komunikacji między serwerem a programem
- Do wymiany danych pomiędzy serwerem www a skryptami CGI wykorzystuje się różne mechanizmy IPC (inter-proces communication):
  – wirtualny strumień (plik) łączący procesy (named pipe),
  – połączenie TCP/IP
  – połączenie IPC socket (Unix domain socket)



Rodzaje interfejsów

- CGI (Common Gateway Interface) – najstarszy, obsługiwany przez większość serwerów www, niezależny od platformy, stabilny; Wada: każde żądanie wymaga uruchomienia nowego procesu, stąd duże obciążenie serwera
- FCGI (Fast CGI) – ulepszony wariant CGI: proces pracujący w sposób ciągły, obsługujący wiele kolejnych żądań
- SCGI (Simple CGI) – podobny do FCGI, uproszczona składnia
- Dedykowane dla języków programowania lub platform:
  – WSGI (Web Server Gateway Interface) – dedykowany dla Pythona,
  – RACK – dedykowany dla Ruby,
  – JSGI (Javascript Gateway Interface),
  – PSGI (Perl Web Server Gateway Interface)
  …

Python i protokoły CGI

- Python nie ma "wbudowanej" obsługi żadnego protokołu CGI – jako język ogólnego przeznaczenia może obsługiwać każdy z nich, jednak implementacja nie jest prosta
- Każdy framework www musi obsługiwać któryś wariant CGI: początkowo różne frameworki obsługiwały różne interfejsy
  (CGI, FCGI, mod_python, …), co prowadziło do chaosu
- WSGI (Web Server Gateway Interface) został opracowany przez Python Software Foundation, jako uzupełnienie specyfikacji Python 3;
- ASGI (Asynchronous Server Gateway Interface) – następca protokołu WSGI, może obsługiwać żądania asynchroniczne; Zachowuje kompatybilność wstecz z WSGI

  Django początkowo wykorzystywał FCGI, obecnie WSGI i ASGI

Protokół WSGI

WSGI składa się z dwóch "części":

- Serwerowej – moduł serwera www (jak Apache, Nginx, …) lub serwer aplikacyjny dedykowany dla Pythona (np. Gunicorn, uWSGI)
- Aplikacyjnej, wbudowanej we frameworku – w praktyce sprowadza się do funkcji, która otrzymuje dane żądania i zwraca gotowy dokument html;
  Obsługę WSGI mają Django, Flask, Pyramid, Tornado, …
- Obsługa WSGI sprowadza się do implementacji jednej funkcji:

```
def application(environ, start_response):
    html = ' …tu cały dokument html… '
    status = '200 OK'
    header = [('Content-type','text/html')]
    start_response(status, header)
    return [html]
```

Dynamiczne strony WWW – WSGI

- Serwer www Apache wymaga zainstalowania modułu mod_wsgi;
  W konfiguracji modułu określa się adres aplikacji WSGI, np.:

```
WSGIScriptAlias /app /var/python/app.py
```

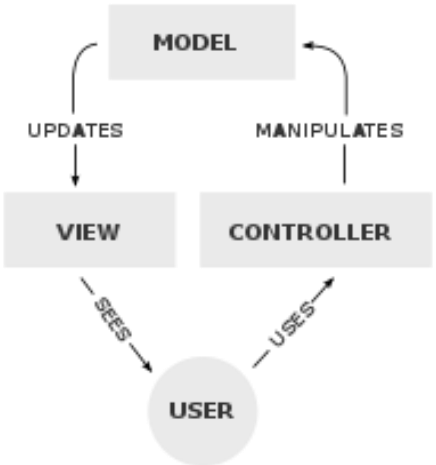
- Wszystkie żądania domena.com/app/<cokolwiek> będą obsługiwane wyłącznie przez WSGI – np. /app/style.css nie zostanie obsłużony przez Apache, tylko przez WSGI
- Kod aplikacji może (a ze względów bezpieczeństwa powinien) znajdować się poza folderem www (np. www: /var/www, aplikacje Python: /var/python) – dzięki temu serwer www nie udostępni przez www źródeł aplikacji Pythona
- Aplikacja musi sama "zdekodować" żądanie: jeżeli żądanie dotyczy podstrony serwisu (np. /studenci/plany), aplikacja generuje i odsyła dokument html, ale jeżeli żądanie dotyczy plików statycznych (np. /obrazki/logo.jpg) – aplikacja znajduje, odczytuje i odsyła plik

Django

- Framework do aplikacji internetowych;
  Autorzy: Adrian Holovaty, Simon Willison, pierwsze wydanie w 2003 r.
- Obsługuje interfejsy WSGI oraz ASGI;
  Dekoduje adres żądania ze zmiennymi (np. /archiwum/<rok>), automatycznie obsługuje pliki statyczne (np. /static/style.css)
- Ma zaawansowany system mapowania obiektowo-relacyjnego (ORM, Object-Relational Mapping) typu "code first" – baza danych jest tworzona na podstawie kodu, a w przypadku zmian w kodzie sam tworzy tzw. migracje – przenosi zmiany kodu na zmiany w bazie danych
- Ma zaawansowany system szablonów, z obsługą wyrażeń oraz instrukcji (if, for), z możliwością dodawania znaczników i filtrów
- Wbudowany ORM i szablony mogą być łatwo zastąpione przez inne rozwiązania, np. SQLAlchemy, Jinja, …
- Wbudowane, rozszerzalne aplikacje:
  – system uwierzytelniania
  – panel administracyjny (zarządzanie użytkownikami i bazą danych)
  – generowanie mapy witryny (Google sitemap)
- Wbudowane zabezpieczenia przed atakami na serwisy www (cross-site request forgery, cross-site scripting, SQL injection, password cracking) i logowania zdarzeń
- Obsługuje formaty danych XML i JSON
- Zawiera własny system cache
- Zawiera prosty serwer www (tylko do testowania!)
- Zawiera moduł do testów jednostkowych

Model MVC

- Django jest zbudowany w oparciu o wariant modelu MVC (Model-View-Controller):
  – Model (model) – logika biznesowa
  – View (widok) – sposób prezentacji danych w GUI
  – Controller (kontroler) – pośredniczy między modelem a widokiem (aktualizacja modelu oraz odświeżanie widoków)
- Model MVC separuje logikę aplikacji od interfejsu użytkownika – umożliwia zmianę części modelu (np. bazy danych) bez zmian w widoku i na odwrót



Warianty modelu MVC

- MVP (model-view-presenter)
- MVVC (model-view-view model)
- Django: MVT (model-view-template)
  Sugestia z dokumentacji (formalnie takiego modelu nie ma):
  – Model (model) – klasy z ORM i logiką biznesową,
  – View (widok) – wybiera dane z modelu (wg adresu URL)
  – Template (szablon) – sposób prezentacji danych

  Model MVT Django separuje również dane (view) od sposobu ich prezentacji (template); Kontrolerem z modelu MVC w Django jest sam Django

Dokumentacja Django:

- Oficjalna strona Django, pełna dokumentacja aktualnej wersji:
  https://docs.djangoproject.com/en/3.1/

Poradniki:

- Oficjalna strona Django:
  https://docs.djangoproject.com/en/3.1/intro/
- Liczne alternatywne, np.:
  https://tutorial.djangogirls.org/pl/

Tworzenie projektu Django

(1) Utworzenie i uruchomienie środowiska wirtualnego (Ważne! Konsola z uprawnieniami administratora)

```
C:\Users\Admin\Desktop\Django>py --version
Python 3.8.0

C:\Users\Admin\Desktop\Django>py -m venv venv

C:\Users\Admin\Desktop\Django>venv\Scripts\activate.bat

(venv) C:\Users\Admin\Desktop\Django>py -m pip install --upgrade pip
Collecting pip
  Using cached
https://files.pythonhosted.org/packages/cb/28/91f26bd088ce8e22169
032100d4260614fc3da435025ff389ef1d396a433/pip-20.2.4-py2.py3-none-
any.whl
Installing collected packages: pip
  Found existing installation: pip 19.2.3
    Uninstalling pip-19.2.3:
      Successfully uninstalled pip-19.2.3
Successfully installed pip-20.2.4
```

(2) Instalacja Django

```
(venv) C:\Users\Admin\Desktop\Django>pip install django
Collecting django
  Using cached Django-3.1.3-py3-none-any.whl (7.8 MB)
Collecting asgiref<4,>=3.2.10
  Using cached asgiref-3.3.1-py3-none-any.whl (19 kB)
Collecting sqlparse>=0.2.2
  Using cached sqlparse-0.4.1-py3-none-any.whl (42 kB)
Collecting pytz
  Using cached pytz-2020.4-py2.py3-none-any.whl (509 kB)
Installing collected packages: asgiref, sqlparse, pytz, django
Successfully installed asgiref-3.3.1 django-3.1.3 pytz-2020.4
```

(3) Utworzenie projektu i aplikacji, pierwsza migracja

```
(venv) C:\Users\Admin\Desktop\Django>django-admin startproject project

(venv) C:\Users\Admin\Desktop\Django>cd project

(venv) C:\Users\Admin\Desktop\Django\project>py manage.py startapp blog

(venv) C:\Users\Admin\Desktop\Django\project>py manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  …
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
```

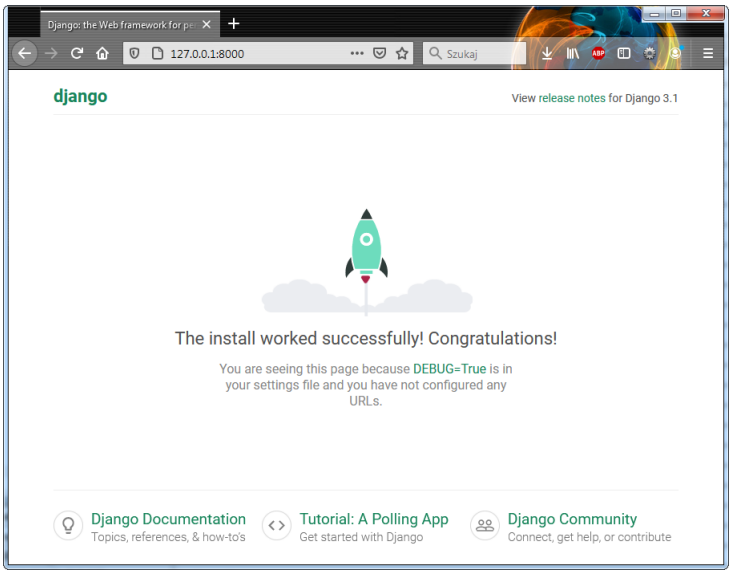
(4) Uruchomienie serwera testowego

```
(venv) C:\Users\Admin\Desktop\Django\project>py manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
November 27, 2020 - 00:00:52
Django version 3.1.3, using settings 'project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

[27/Nov/2020 00:01:52] "GET / HTTP/1.1" 200 16351
[27/Nov/2020 00:01:52] "GET /static/css/fonts.css HTTP/1.1" 200 423
```

(5) Sprawdzenie w przeglądarce



(6) Wyłączenie serwera i środowiska wirtualnego

```
Django version 3.1.3, using settings 'project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.


[…]


(venv) C:\Users\Admin\Desktop\Django\project>cd ..

(venv) C:\Users\Admin\Desktop\Django>venv\Scripts\deactivate.bat

C:\Users\Admin\Desktop\Django>exit
```

Zadania

Proszę utworzyć i uruchomić projekt Django, a w tym celu:

1. Założyć nowy folder, np. na pulpicie, uruchomić konsolę z uprawnieniami administratora i przejść do nowo utworzonego folderu
2. Utworzyć i uruchomić wirtualne środowisko Pythona
3. Zaktualizować pip, instalator pakietów Pythona
4. Korzystając z pip zainstalować Django
5. Korzystając z narzędzi linii poleceń Django utworzyć projekt
6. Korzystając z managera projektu (manage.py), utworzyć w projekcie aplikację, a następnie wykonać pierwszą migrację bazy danych
7. Korzystając z managera projektu uruchomić serwer www
8. Sprawdzić, czy aplikacja działa, wpisując w przeglądarce internetowej adres: 127.0.0.1:8000
9. Zamknąć przeglądarkę, wyłączyć serwer www i środowisko wirtualne