Zadanie 1

1. Przygotowanie środowiska
2. Stworzenie podstawowej aplikacji
3. Wypchnięcie do repozytorium gita
4. Wstawienie read.me z używanymi technologiami w projekcie

Zadanie 2

1. Odpalamy aplikację do modelowania bazy danych (np. MySQL Workbench),
2. Tworzymy nowy model bazy danych zawierający wszystkie tabele i relacje, które będą potrzebne w naszym projekcie.
3. Zapisujemy projekt
4. Plik ze schematem wrzucamy do głównego katalogu repozytorium.
5. Dodajemy podpunkt w README.md repozytorium:
   - Model Danych
6. Wyświetlamy w nim plik db_schema

Zadanie 3

W przypadków innych technologii niż Python/Django proszę o stworzenie aplikacji, dodanie endpointów, stworzenie przykładowego widoku na jednym z endpointów, przygotowanie modelu danych w aplikacji, podpięcie modeli danych pod zewnętrzną bazę danych.

1. Zainstaluj pakiety django i djangorestframework,
2. Dodaj 'rest_framework' do zainstalowanych aplikacji w swoim projekcie Django,
3. Stwórz projekt Django,
4. Stwórz aplikację Django w projekcie z nazwą oddającą cel projektu,
5. W aplikacji stwórz plik urls.py
6. W głównym pliku urls.py dodaj odwołanie do pliku urls w aplikacji (jako link ustaw nazwę aplikacji),
7. Przygotuj plik models.py, tak aby odzwierciedlał stworzony model w MySQL Workbench na poprzednich zajęciach
8. Podepnij się pod bazę MySQL(najłatwiej lokalnie wrzucić XAMPP)/Postgres,
9. Stwórz migrację,
10. Zmigruj bazę na serwer MySQL/Postgres

Zadanie 4

Celem ćwiczeń będzie stworzenie klas odpowiedzialnych za serializację danych w naszym projekcie API. Następujące polecenia powinny być wykonane dla każdego modelu w projekcie, do którego będziemy mieli dostęp zewnętrzny (przez wykonanie endpointu, np. tworzenie zamówienia).

1. W **aplikacji**, w której znajduje się nasz model otwieramy/tworzymy plik serializers.py,
2. W nagłówku pliku dodajemy import:
   `from rest_framework import serializers`

3. Tworzymy klasę dziedziczącą po serializers.HyperlinkedModelSerializer lub serializers.ModelSerializer, a następnie tworzymy odpowiednie pola
4. Jeżeli dany model wymaga dodatkowej walidacji (np. data stworzenia zamówienia nie może być w przyszłości), tworzymy odpowiednie funkcje

Zadanie 5

Celem ćwiczeń będzie stworzenie widoków (endpointów), dzięki którym użytkownik będzie mógł łączyć się z naszym API by pobierać, edytować czy usuwać dane. W tym celu zostanie użyte djangorestframework.

1. Dodaj widoki drf do urlpatterns swojego projektu:
   url(r'^api-auth/', include('rest_framework.urls'))

2. Stwórz widoki dla swoich modeli. Następnie dodaj je do urlpatterns by wyświetliły się w liście dostępnych endpointów API.
   - Zwróć uwagę, które modele powinny mieć możliwość dodawania, usuwania czy edytowania informacji. Może niektóre powinny tylko wyświetlać dane?
   - Pamiętaj by zastosować serializery z poprzednich zajęć.
3. Dodaj zezwolenia do aplikacji, tak by tylko zarejestrowani uzytkownicy mogli korzystać z endpointów,
4. Dodaj endpointy, które są dostępne tylko dla administratora.

Zadanie 6

1. Dodaj widok oparty na GenericAPIView pozwalający na nawigację po API
2. Zamiast wartości kluczy obcych wstaw pole opisujące dany rekord (np. nazwisko, nazwa, tytuł) - użyj SlugRelatedField z klasy HyperlinkedModelSerializer
3. W związkach jeden do wielu po stronie jeden wstaw linki do rekordów po stronie wiele - użyj HyperlinkedRelatedField z klasy HyperlinkedModelSerializer
4. Ustaw globalną paginację z liczbą pozycji na stronę 5
5. Ustaw filtry i sortowanie na poszczególne widoki
6. Dodaj własne filtry na datę i liczby jako przedział od - do