Introduction to Django Rest Framework

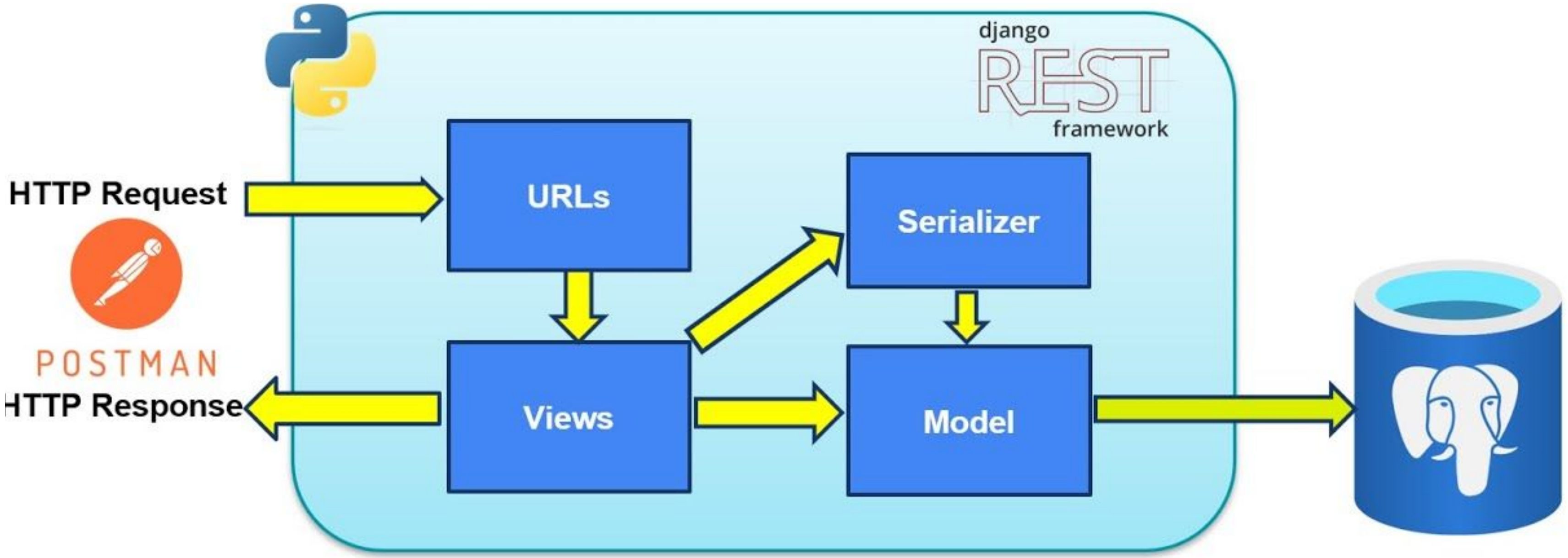
Urls, Views, Serializers, Models

First things first , install drf in the project virtual environment .

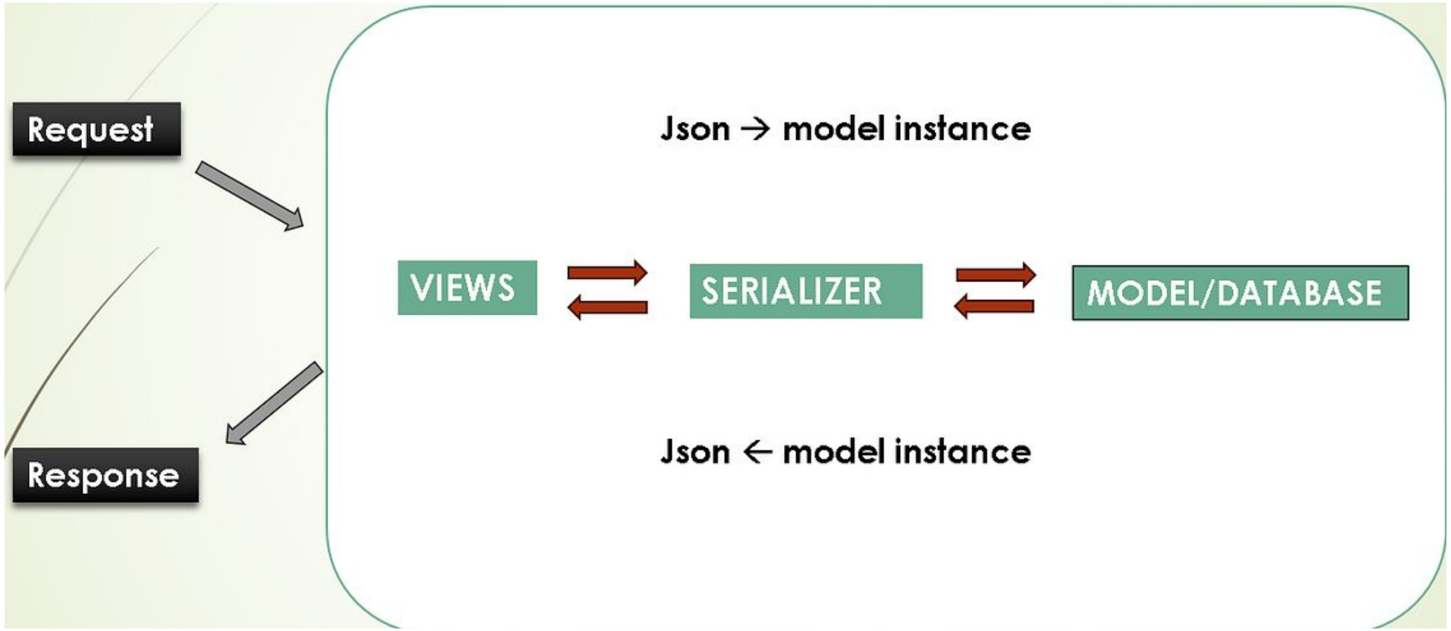
```
pip install djangorestframework
```

```
# settings.py
INSTALLED_APPS = [
    # other apps
    'rest_framework',
]
```

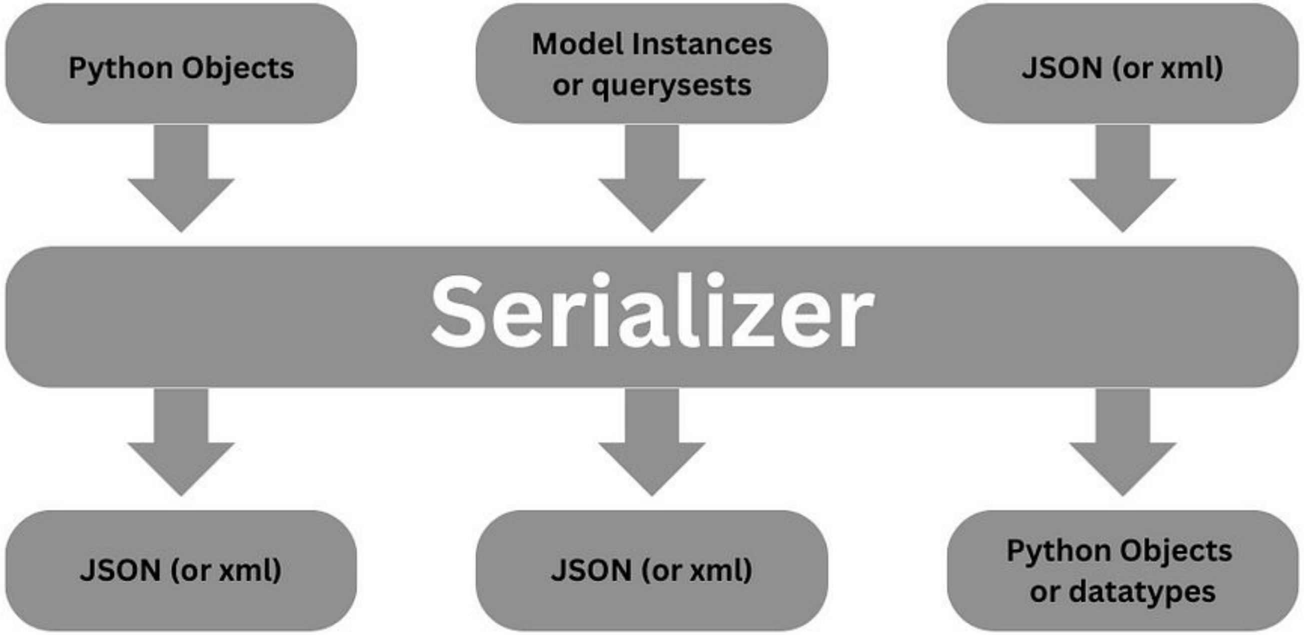
DRF Flow Diagram



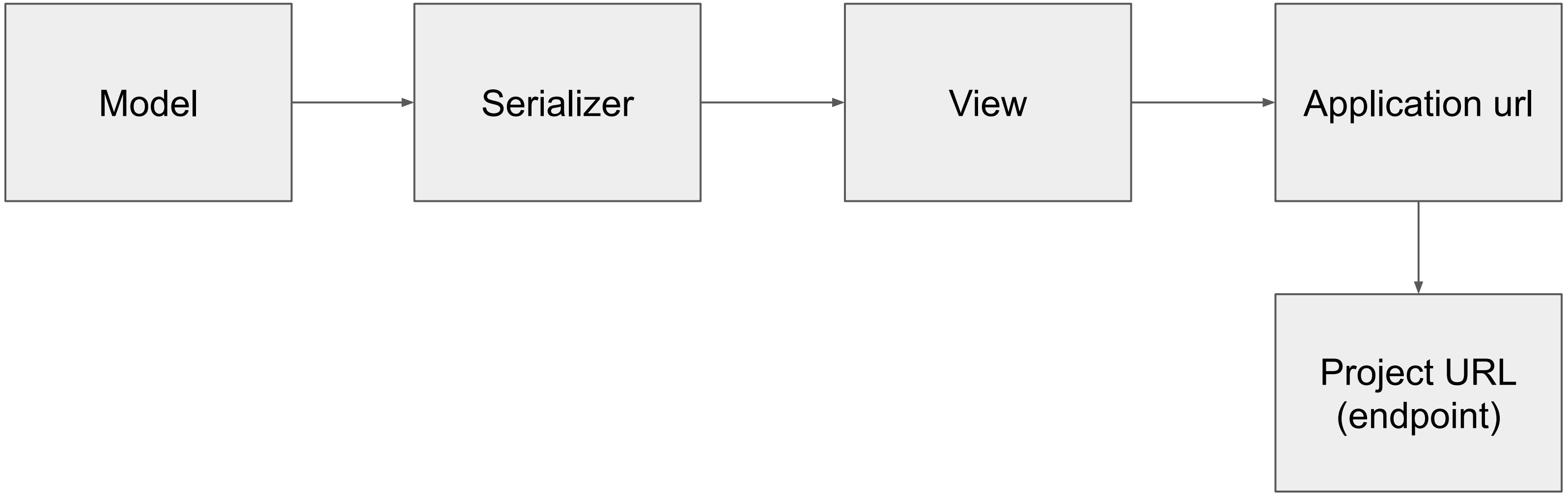
DRF Serialization



DRF Serializer



Our Initial approach



What we will build ?

A Todo System:

- Create
- Read
- Update
- Delete

Project Intialization

```
django-admin startproject src
```

```
django-admin startapp todo
```

```
#settings.py
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    # 3rd party
    'rest_framework',

    'todo',
]
```

Project Level Route

```
# src/urls.py
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/', include('todo.urls'))
]
```

We will create this ‘urls’ file under todo app!

Design our model

```
# todo/models.py
from django.db import models


class Todo(models.Model):
    title = models.CharField(max_length=255)
    completed = models.BooleanField(default=False)
```

```
python3 manage.py makemigrations
```

```
python3 manage.py migrate
```

Design our serializer

Create a file named ‘`serializers.py`’ under ‘todo’ folder

```
# todo/serializers.py
from rest_framework import serializers
from .models import Todo

class TodoSerializer(serializers.ModelSerializer):
    id = serializers.ReadOnlyField()

    class Meta:
        model = Todo
        fields = ["id", "title" , "completed"]
```

Writing the Views

Import something

```
# views.py
from rest_framework.decorators import api_view
from rest_framework.response import Response
from rest_framework import status

from .models import Todo
from .serializers import TodoSerializer
```

For list of object we know the related method is 'GET' and for create it's 'POST' . Let's continue editing in views.py

```
@api_view(['GET', 'POST'])
def todos(request):
    if request.method == 'GET':
        todos = Todo.objects.all()
        serializer = TodoSerializer(todos, many=True)
        return Response(data=serializer.data, status=status.HTTP_200_OK)

    elif request.method == 'POST':
        serializer = TodoSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(
                data={"success": "Todo created successfully"},
                status=status.HTTP_201_CREATED
            )
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

What’s rest ? Yes it is single object (‘GET’) , update (‘PUT’) & delete (‘DELETE’) of a single todo. Let's do it, same in the views.py

```
@api_view(['GET' , 'PUT', 'DELETE'])
def todo_details(request, pk):
    try:
        todo = Todo.objects.get(pk=pk)
    except Todo.DoesNotExist:
        return Response(status=status.HTTP_404_NOT_FOUND)

    if request.method == 'GET':
        serializer = TodoSerializer(todo)
        return Response(serializer.data)

    elif request.method == 'PUT':
        serializer = TodoSerializer(todo, data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(
                data={"success": "Todo updated successfully"}
            )
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

    elif request.method == 'DELETE':
        todo.delete()
        return Response(status=status.HTTP_204_NO_CONTENT)
```

Application url

Create a file named ‘urls.py’ under ‘todo’

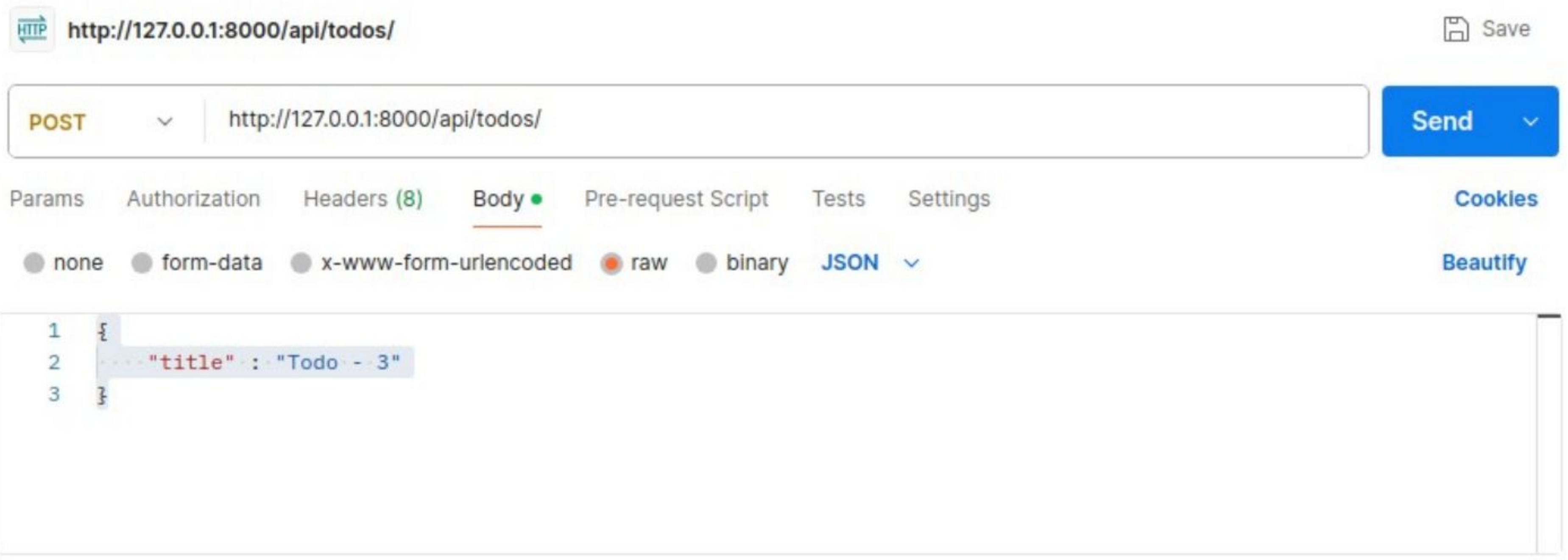
```
# urls.py
from django.urls import path
from . import views

urlpatterns = [
    path("todos/", views.todos ,name="todos"),
    path("todos/<int:pk>/", views.todo_details ,name="todos-details"),
]
```

Test the urls. What are the options ?

Insomnia

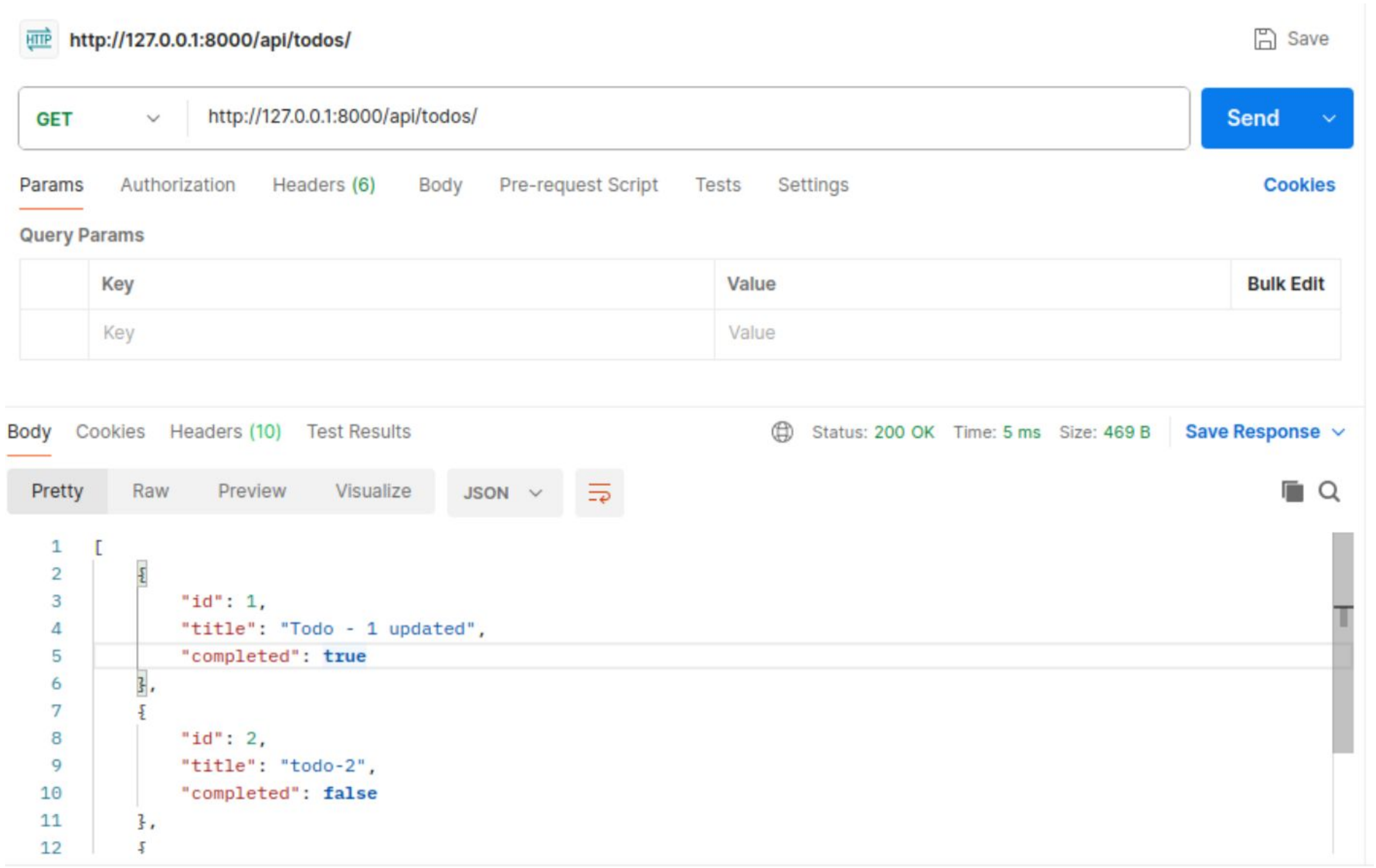
POSTMAN

RestFox



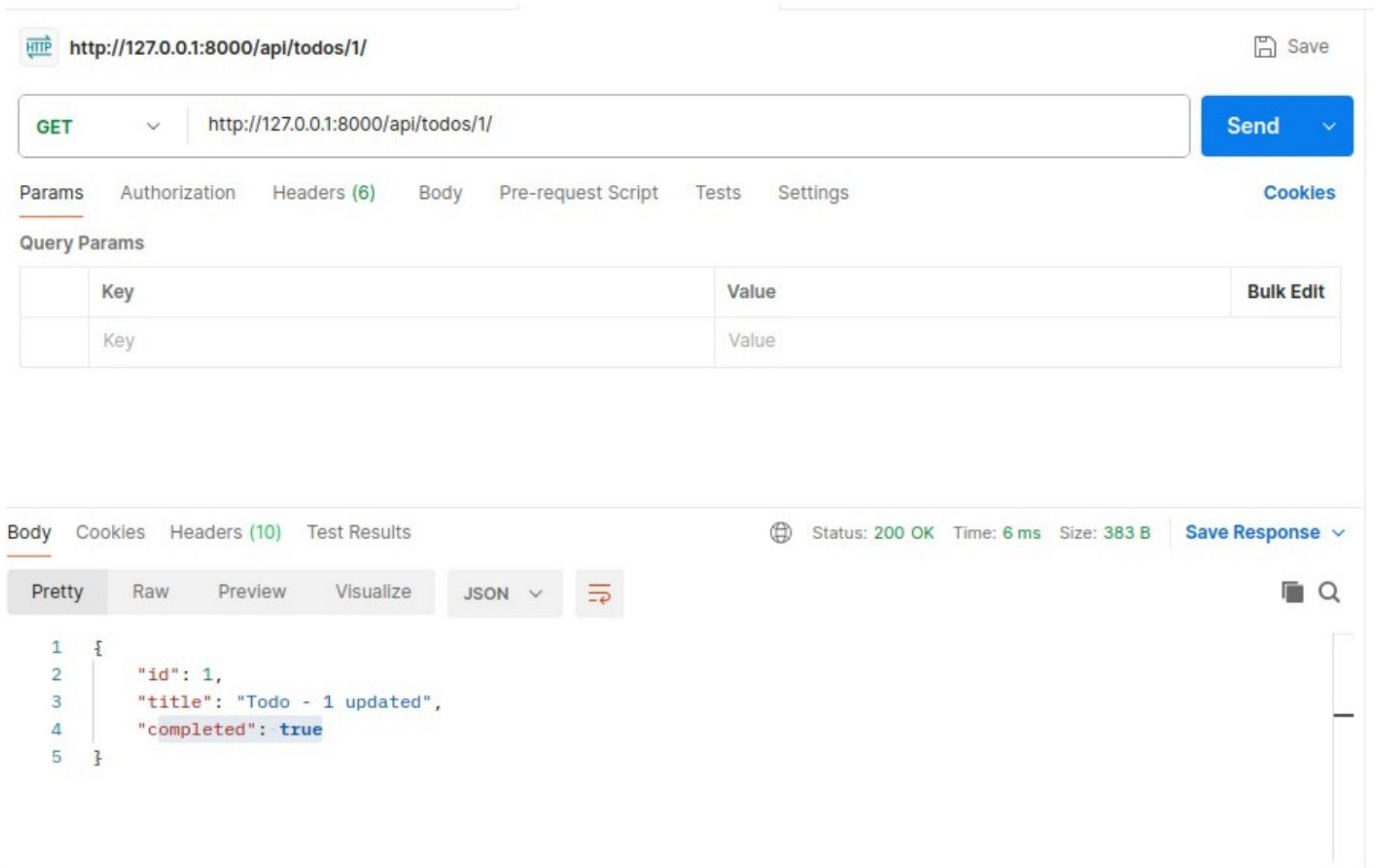
POST http://127.0.0.1:8000/api/todos/

```
{
    "title" : "Todo - 3"
}
```

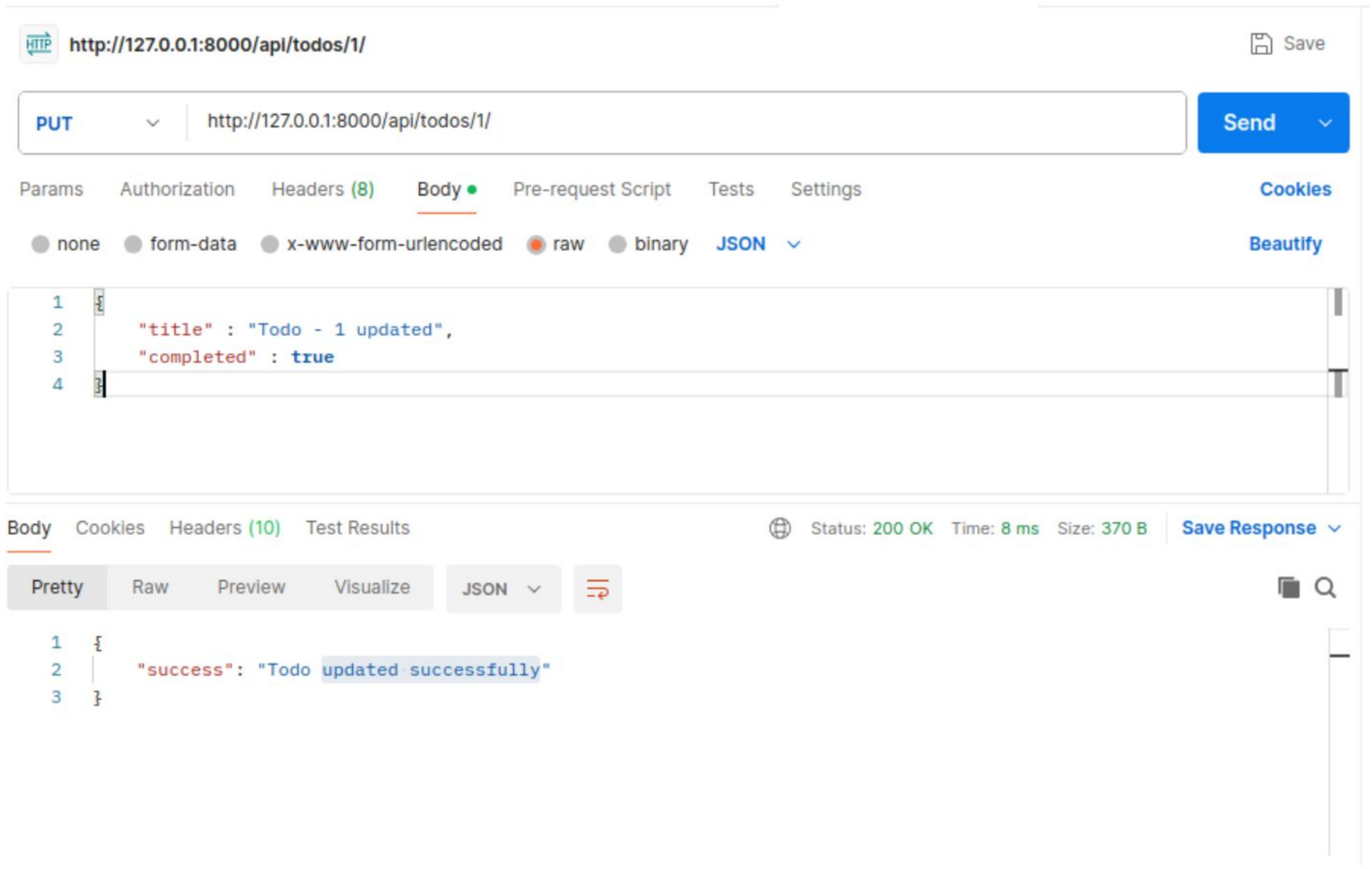


Key	Value	Bulk Edit
Key	Value	

```
[
    {
        "id": 1,
        "title": "Todo - 1 updated",
        "completed": true
    },
    {
        "id": 2,
        "title": "todo-2",
        "completed": false
    },
    {
```



```
{
    "id": 1,
    "title": "Todo - 1 updated",
    "completed": true
}
```


http://127.0.0.1:8000/api/todos/1/

PUT http://127.0.0.1:8000/api/todos/1/

Send
Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies
none form-data x-www-form-urlencoded raw binary JSON Beautify

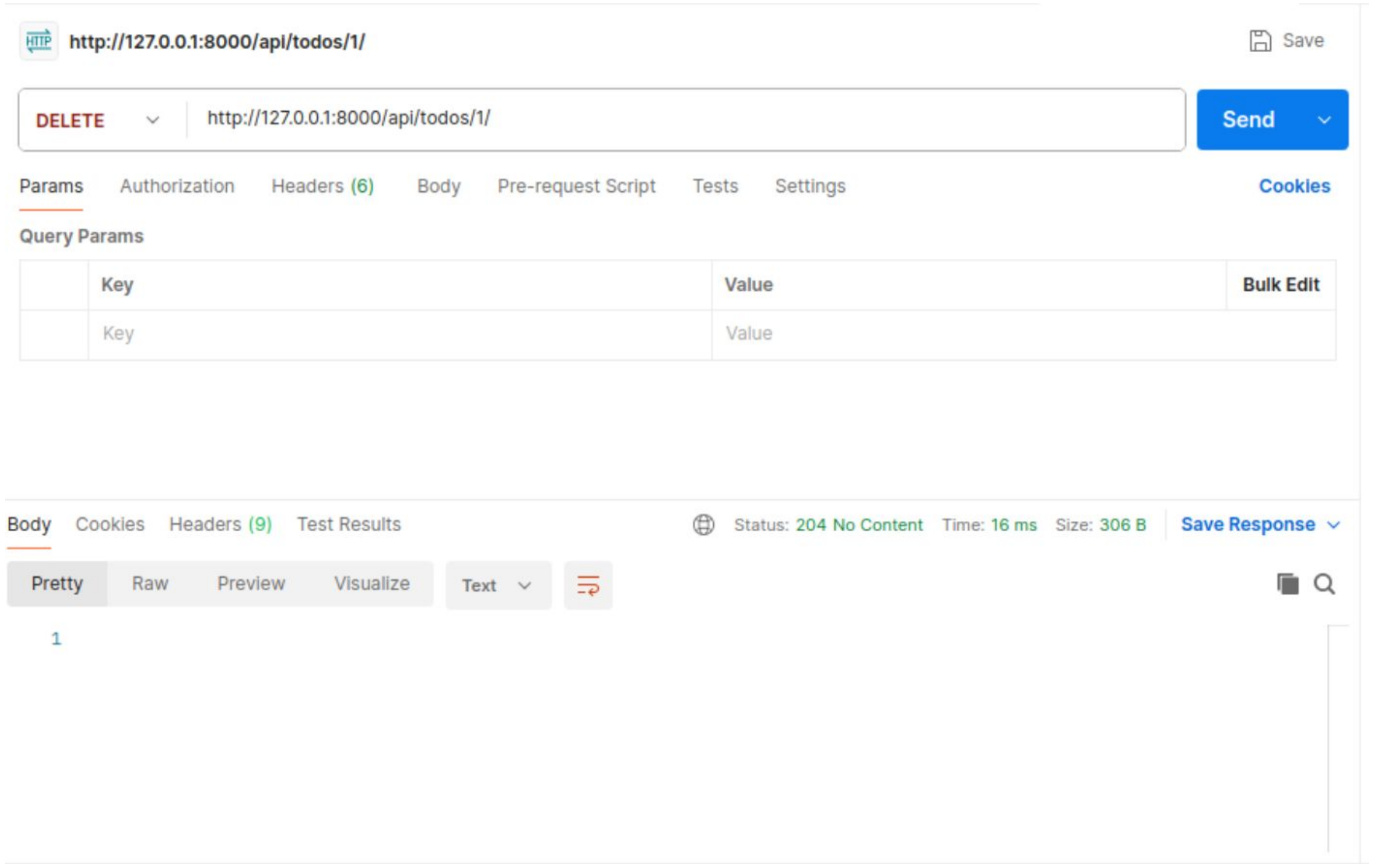
```
{
    "title" : "Todo - 1 updated",
    "completed" : true
}
```

Body Cookies Headers (10) Test Results

Status: 200 OK Time: 8 ms Size: 370 B

Save Response
Pretty Raw Preview Visualize JSON

```
{
    "success": "Todo updated successfully"
}
```


http://127.0.0.1:8000/api/todos/1/

DELETE http://127.0.0.1:8000/api/todos/1/

Params | Authorization | Headers (6) | Body | Pre-request Script | Tests | Settings | Cookies

Query Params

	Key	Value	Bulk Edit
	Key	Value	

Body | Cookies | Headers (9) | Test Results

Status: 204 No Content Time: 16 ms Size: 306 B

- So far we have written ‘Function based views’ (FBV)
- We’ll write ‘Class Based View’ (CBV)
- Something might be felt as magic! But, it may be!

Class Based View

```
# views.py
from rest_framework import generics
from .models import Todo
from .serializers import TodoSerializer

class TodoListCreate(generics.ListCreateAPIView):
    queryset = Todo.objects.all()
    serializer_class = TodoSerializer

class TodoRetrieveUpdateDestroy(generics.RetrieveUpdateDestroyAPIView):
    queryset = Todo.objects.all()
    serializer_class = TodoSerializer
```

Modify the urls.py

```
# todos/urls.py
from django.urls import path
from . import views

urlpatterns = [
    path("todos/", views.TodoListCreate.as_view() ,name="todos"),
    path("todos/<int:pk>/", views.TodoRetrieveUpdateDestroy.as_view() ,name="todos-details"),
]
```