Отработка класса APIView

Создадим API для новостного сайта.
Для начала создадим модели

```python
from django.db import models

class Author(models.Model):
  name = models.CharField(max_length=255)
  email = models.EmailField()
```

```python
class Article(models.Model):
    title = models.CharField(max_length=120)
    description = models.TextField()
    body = models.TextField()
    author = models.ForeignKey('Author', related_name='articles', on_delete=models.CASCADE)
```

Добавим представление для получения Get запроса

```python
from rest_framework.response import Response
from rest_framework.views import APIView

from .models import Article
from .serializers import ArticleSerializer

class ArticleView(APIView):
    def get(self, request):
        articles = Article.objects.all()
        # the many param informs the serializer that it wil
        serializer = ArticleSerializer(articles, many=True)
        return Response({"articles": serializer.data})
```

И сериализатор

```python
from rest_framework import serializers

class ArticleSerializer(serializers.Serializer):
    title = serializers.CharField(max_length=120)
    description = serializers.CharField()
    body = serializers.CharField()
```

Добавим метод Post в представления

и метод create в сериализатор

```
def post(self, request):
    article = request.data.get('articles')

    # Create an article from the above data
    serializer = ArticleSerializer(data=article)
    if serializer.is_valid(raise_exception=True):
        article_saved = serializer.save()
    return Response({"success": "Article '{}' created successfully".format(article_saved.title)})
```

```
def create(self, validated_data):
    return Article.objects.create(**validated_data)
```

Добавим метод Put в представления

и метод update в сериализатор

```
def put(self, request, pk):
    saved_article = get_object_or_404(Article.objects.all(), pk=pk)
    data = request.data.get('articles')
    serializer = ArticleSerializer(instance=saved_article, data=data, partial=True)

    if serializer.is_valid(raise_exception=True):
        article_saved = serializer.save()

    return Response({
        "success": "Article '{}' updated successfully".format(article_saved.title)
    })
```

```
def update(self, instance, validated_data):
    instance.title = validated_data.get('title', instance.title)
    instance.description = validated_data.get('description', instance.description)
    instance.body = validated_data.get('body', instance.body)
    instance.author_id = validated_data.get('author_id', instance.author_id)

    instance.save()
    return instance
```

```
def delete(self, request, pk):
    # Get object with this pk
    article = get_object_or_404(Article.objects.all(), pk=pk)
    article.delete()
    return Response({
        "message": "Article with id `{}` has been deleted.".format(pk)
    }, status=204)
```

Не забываем указывать внутренние настройки

И правильные маршруты

Задание

Создать API Для хранения данных о товарах в корзине. Для этого необходимо создать таблицы Товары и Корзина. В Корзине есть информация о товарах (попробуйте сделать связь многие ко многим. На край просто указывайте id товаров, через запятую (Это уже тем у кого меньше 70 баллов)), общая стоимость заказа. Товар: Наименование, Описание, цена.
Добавляем CRUD для Корзины.