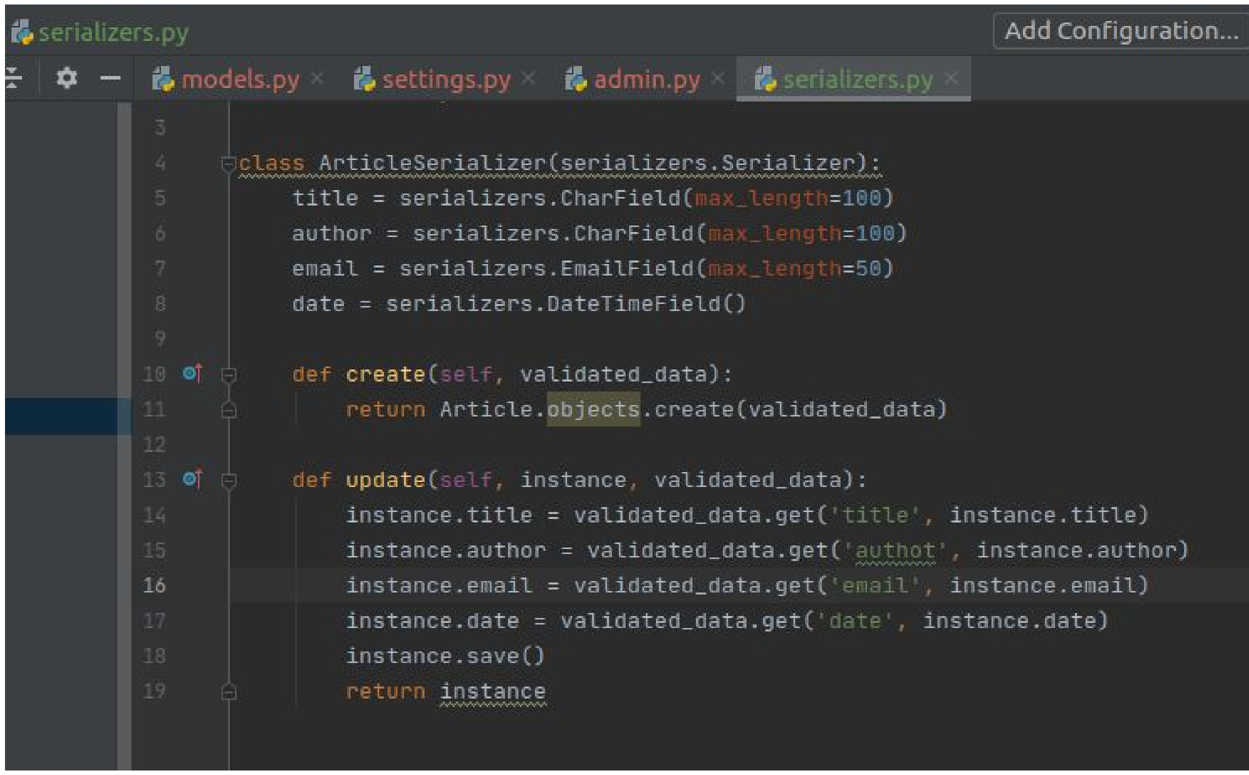
- Add rest_framework to INSTALLED_APPS.
- **Serializers**: To send data to clients, Serializers serialize the data into JSON format which is then sent to the client. Step by step guide to work with them:
    - Create serializers.py in your app.
    - Import your models in it.
    - Create <Model Name>Serializer class that inherits from serializers.Serializer.
    - Override its create and update methods as shown below:



```python
class ArticleSerializer(serializers.Serializer):
    title = serializers.CharField(max_length=100)
    author = serializers.CharField(max_length=100)
    email = serializers.EmailField(max_length=50)
    date = serializers.DateTimeField()

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

    def update(self, instance, validated_data):
        instance.title = validated_data.get('title', instance.title)
        instance.author = validated_data.get('authot', instance.author)
        instance.email = validated_data.get('email', instance.email)
        instance.date = validated_data.get('date', instance.date)
        instance.save()
        return instance
```

    - Go to the shell and do the following imports:
        - from <app name>.models import <model name>
        - from <app name>.serializers import <name of corresponding serializer>
        - from rest_framework.renderers import JSONRenderer
        - from rest_framework.parsers import JSONParser
    - To use serializer:
        - Create model objects.
        - Pass them to the serializer class you created:
            - serializer = ArticleSerializer(article_object)
            - >>> serializer: prints fields of serializer class.
            - >>> serializer.data: prints objects in dictionary.
        - This step converts the objects into a Python dictionary. To convert them into JSON objects:
            - content = JSONRenderer().render(Article.objects.all(), many = True)
            - >>> content: prints serialized objects in JSON form.

- **Model Serializers**: Instead of having to specify all fields in our serializer class, we can use serializer.ModelSerializer and define:
    - Class Meta:
        - model = Article
        - fields = [‘field1’, ‘field2’, ‘field3’]
- We get the same behaviour as with serializer.Serializer.
- To use Function-based views with serializer:

    ```
    from django.shortcuts import render
    from django.http import HttpResponse, JsonResponse
    from rest_framework.parsers import JSONParser
    from .models import Article
    from .serializers import ArticleSerializer
    ```