Ответы на вопросы

1. Создал **”ViewSet”** для сущности **”Entity”** и переопределил методы **”perform_create”** и **”perform_update”**, добавив поле **”modified_by”** в коллекцию данных сериализатора:

```
class EntityViewSet(viewsets.ModelViewSet):
    '''View set of Entity class serializer
    '''
    queryset = Entity.objects.all()
    serializer_class = EntitySerializer
    permission_classes = [permissions.IsAuthenticatedOrReadOnly]

    def perform_create(self, serializer):
        serializer.validated_data['modified_by'] = self.request.user
        return super().perform_create(serializer)

    def perform_update(self, serializer):
        serializer.validated_data['modified_by'] = self.request.user
        return super().perform_update(serializer)
```

2. Создал класс поля **”AliasIntegerField”**, для которого можно задавать псевдоним имени поля данных, унаследовав его от **”IntegerField”**, и применил его в сериализаторе:

```
class AliasIntegerField(serializers.IntegerField):
    '''Integer field with possibility of creating a field name alias
    '''
    def __init__(self, field_name_alias: str=None, **kwargs):
        self.field_name_alias = field_name_alias
        super().__init__(**kwargs)

    def get_value(self, dictionary):
        try:
            dictionary[self.field_name] = dictionary.pop(self.field_name_alias)
        except KeyError:
            pass
        return super().get_value(dictionary)

class EntitySerializer(serializers.ModelSerializer):
    '''Entity class serializer
    '''
    value = AliasIntegerField(field_name_alias='data[value]')
    properties = serializers.SerializerMethodField(read_only=True)

    def get_properties(self, obj: Entity) -> dict:
        '''Returns entity's properties as a dictionary
        '''
```

```
        properties = obj.properties.all()
        return {property.key: property.value for property in properties}

    class Meta:
        model = Entity
        fields = ['value', 'properties']
```

3. Использовал поле **”SerializerMethodField”** для вывода properties в необходимом формате:

```
@extend_schema_serializer(
    examples=[
        OpenApiExample(
            'Example',
            summary='Example',
            description='Ordinal example',
            value={
                'value': 0
            },
            request_only=True,
            response_only=False,
        ),
        OpenApiExample(
            'Alias example',
            summary='Example with alias',
            description='Example of using alias',
            value={
                'data[value]': 0
            },
            request_only=True,
            response_only=False,
        ),
    ]
)
class EntitySerializer(serializers.ModelSerializer):
    '''Entity class serializer
    '''
    value = AliasIntegerField(field_name_alias='data[value]')
    properties = serializers.SerializerMethodField(read_only=True)

    def get_properties(self, obj: Entity) -> dict:
        '''Returns entity's properties as a dictionary
        '''
        properties = obj.properties.all()
        return {property.key: property.value for property in properties}

    class Meta:
        model = Entity
```

```
    fields = ['value', 'properties']
```