|  |
| --- |
| **Club상세페이지, 검색 api** |
| class ClubfilterSet(FilterSet):      class Meta:          model = Clubs          fields = {'club\_type\_\_club\_type\_name':['exact']}  class ClubsList(generics.ListAPIView):      queryset = Clubs.objects.annotate(likes = Count('like\_user')).all()      serializer\_class = ClubsSerializer      filter\_backends = [DjangoFilterBackend, SearchFilter]      search\_fields = ['club\_name', 'club\_desc', 'established', 'club\_type\_\_club\_type\_name', 'club\_type\_\_club\_type\_desc']      filterset\_class = ClubfilterSet  class ClubDetail(APIView):      def get\_object(self, pk):  *try*:  *return* Clubs.objects.get(pk=pk)  *except* Clubs.DoesNotExist:  *raise* Http404      def get(self, request, pk, format=None):          clubs\_detail = self.get\_object(pk)          serializer = ClubsSerializer(clubs\_detail)  *return* Response(serializer.data) |

|  |
| --- |
| **게시물 댓글,대댓글 api** |
| *#to\_representation을 오버라이딩 해서 화면에 출력*  class RecursiveSerializer(serializers.Serializer):      def to\_representation(self, instance):          serializer = self.parent\_reply.parent\_reply.\_\_class\_\_(instance, context = self.context)  *return* serializer.data  class PostsRepliesSerializer(serializers.ModelSerializer):      reply = serializers.SerializerMethodField(read\_only= True)      class Meta:          model = PostsReplies          fields = '\_\_all\_\_'          read\_only\_fields= ['user', 'post', 'is\_deleted']      def get\_reply(self, instance):  *#self.\_\_class\_\_로 직렬화*  *#instance.reply를 통해 재귀*          serializer = self.\_\_class\_\_(instance.reply, many=True)  *#직렬화된 필드를 부모필드에 연결*          serializer.bind('', self)  *return* serializer.data |

|  |
| --- |
| **좋아요 api** |
| class PostsLikesAPIView(APIView):      def get\_object(self, pk):  *try*:  *return* Posts.objects.get(pk=pk)  *except* Posts.DoesNotExist:  *raise* Http404      def get(self, request, pk, format = None):  *# 로그인을 안했으면 401 에러를 발생시킨다.*  *if* not request.user.is\_authenticated:  *return* Response("Please Login First", status = status.HTTP\_401\_UNAUTHORIZED)  *#post라는 객체 정의해서 posts models에 pk(primary key)르 가져옴*  *# post = get\_object\_or\_404(Posts, pk = pk)*          post = Posts.objects.filter(pk=pk)  *# post가 있을때*  *if* post.exists():  *#models를 가져옴*              post\_like = PostsLike()  *#pk를 post\_like.posts에 정의*              post\_like.posts = post[0]  *# 요청받는 유저를 post\_like.user에 정의*              post\_like.user = request.user  *#좋아요가 이미 있을때 -> like.delete()*  *if* post[0].like.filter(user = request.user).exists():  *# post[0].like.remove(request.user)*                  post[0].like.filter(user=request.user).delete()  *# 좋아요가 없을 때 create*  *else*:                  post[0].like.add(post\_like.user)  *return* post\_like              post\_like.save()              serializer = PostsLikeSerializer(post\_like)  *return* Response(serializer.data, status = status.HTTP\_201\_CREATED)  *# post가 없을때*  *else*:  *return* Response(serializer.errors, status = status.HTTP\_404\_NOT\_FOUND)          temp = PostsLikeSerializer(post\_like)          serializer = PostsLikeSerializer(data = temp.data) |