```python
import random
import requests
from django.shortcuts import render
from django.views import generic
from django.db.models import Count
from rest_framework import viewsets
from .serializers import PeliTauluSerializer
from peli.models import PeliTaulu, TrelloTaulu
from p12LmmApi.ei_gittiin import TRELLO_KEY, TRELLO_TOKEN, TRELLO_ID_LIST



class PeliTauluView(viewsets.ModelViewSet):
    """Suoraviivainen (suojattu) readonly -rajapinta kaikkiin kohteisiin (Helsinki)."""
    queryset = PeliTaulu.objects.filter(kunta='Helsinki')
    serializer_class = PeliTauluSerializer



class PeliLista(generic.ListView):
    """Suomen kartta joka on lohkottu maakunnittain. Maakunnan valitsemalla aukeaa vastaava näkymä yhden maakunnan
     kaikkiin kuntiin. Hover näyttää oleellsisia tietoja maakunnasta.
     Satunnaismuuttujalla suoraan yllätyskohteeseen.
     Erityisen ylpeä olen hausta:'list(PeliTaulu.objects.all().order_by('maakunta')
     .values_list('maakunta', flat=True).distinct())', jolla haetaan koko tietokannasta 19 Suomen maakuntaa."""
    model = PeliTaulu
    template_name = 'peli/pelitaulu_list.html'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        tyypit = list(PeliTaulu.objects.filter().order_by('tyyppi').values_list('tyyppi', flat=True).distinct())
        mkt = list(PeliTaulu.objects.all().order_by('maakunta').values_list('maakunta', flat=True).distinct())
        total = PeliTaulu.objects.count()
        satlmm = random.randint(1, 1250)

        konteksti = {
            'total': total,
            'tyypit': tyypit,
            'mkt': mkt,
            'satlmm': satlmm,
            'context': context
        }
        return konteksti



class PeliTiedotMaakunta(generic.ListView):
    """Lista valitun maakunnan niistä kunnista, joista löytyy annettujen ehtojen mukaisia kohteita."""
    model = PeliTaulu
    template_name = 'peli/pelitaulu_list_maakunta.html'

    def get_context_data(self, **kwargs):
        # Call the base implementation first to get a context
        context = super().get_context_data(**kwargs)
        maakunta = self.kwargs['maakunta']
        context['kunnat'] = PeliTaulu.objects.filter(maakunta=maakunta).order_by('kunta').distinct().\
            values('kunta')
```

```
        total = PeliTaulu.objects.filter(maakunta=maakunta).count()
        konteksti = {
            'total': total,
            'maakunta': maakunta,
            'context': context
        }
        return konteksti


class PeliTiedotKunta(generic.ListView):
    """Lista valitun kunnan alueella olevista kohteista, jotka täyttävät annetut rajoittavat
ehdot."""
    model = PeliTaulu
    template_name = 'peli/pelitaulu_list_kunta.html'
    paginate_by = 10

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        kunta = self.kwargs['kunta']
        maakunta = PeliTaulu.objects.filter(kunta=kunta).order_by('kunta').values_list('maakunta',
flat=True).distinct()
        context['kohteet'] = PeliTaulu.objects.filter(kunta=kunta).order_by('nimi').\
            values('nimi', 'pk', 'tyyppi')
        konteksti = {
            'maakunta': maakunta[0],
            'kunta': kunta,
            'context': context
        }
        return konteksti


class PeliTiedotDetail(generic.DetailView):
    """Faktakartta yksittäisen kohteen tietoihin. Kahdeksan tietokokonaisuutta kustakin
kohteesta.
    1. Kunnan päätökset ja tämän ajallinen sijoittuminen niissä.
    2. Kaikki oleellinen (self -kentät).
    3. Tyyppi -kuva (piirros).
    4. Valokuva. Carousel, jos muita kuvia.
    5. Kartta. GoogleMaps -linkki.
    6. Päätösdokumentti. Linkki aukeaa pdf-dokumenttiin.
    7. Somefeed.
    8. tags. Indeksoidut tietotyypit (tyyppi, kunta, tie)"""
    model = PeliTaulu

    def get_context_data(self, **kwargs):
        # Call the base implementation first to get a context
        context = super().get_context_data(**kwargs)
        pk = self.kwargs['pk']
        tarksitatrello = PeliTaulu.objects.get(pk=pk).sometunniste
        context['trellossa'] = TrelloTaulu.objects.filter(sometunniste_trello=tarksitatrello)\
            .values_list('julkaistu_trello', flat=True)
        context['uusitunniste'] = PeliTaulu.objects.get(pk=pk).sometunniste[1:]
        return context


def uusitrellokortti(request, sometunniste):
    url = "https://api.trello.com/1/cards"
    uusi_kortti = sometunniste
    query = {
        'key': TRELLO_KEY,
        'token': TRELLO_TOKEN,
```

```
        'idList': TRELLO_ID_LIST,
        'name': uusi_kortti,
    }
    response = requests.request(
        "POST",
        url,
        params=query,
    )
    tallenna = TrelloTaulu(sometunniste_trello=uusi_kortti)
    tallenna.save()
    print(response.status_code)
    context = {
        'name': uusi_kortti,
    }
    return render(request, 'peli/uusitrello.html', context)


class PeliTiedotTyyppi(generic.ListView):
    """Haetaan listalle kaikki vailtun 'tyyppi' -tiedon kohteet. Truncate 40 (koska 498 mäntyä).
    Tästä voi hyvin tehdä yleisen hakemaan ,myös 'svuosi' ja 'tie'."""
    model = PeliTaulu
    template_name = 'peli/pelitaulu_list_tyyppi.html'

    def get_context_data(self, **kwargs):
        contexti = super().get_context_data(**kwargs)
        tyyppi = self.kwargs['tyyppi']
        contexti = PeliTaulu.objects.filter(tyyppi=tyyppi).values('pk', 'nimi')

        context = {
            'tyyppi': tyyppi,
            'contexti': contexti
        }

        return context


class PeliTiedotVuodet(generic.TemplateView):
    """Haetaan kaikki suojeluvuoden 'svuosi' kohteet. Näistä palautetaan (5 suurimman) tyyppi
    ja määrä ja lisäksi lasketaan kyseisen vuoden totaali. Templatella esitetään GoogleCraphs.
    Vaihtoehtoina voi käyttää esimerkiksi 'BarChart - barchart' tai 'PieChart - piechart'
yhdistelmiä,
    jotka siis toteutetaan helposti muuttamalla templatella olevan scriptin tekijöitä vastaavasti.
    suojelut : noutaa ko. vuoden kohteiden kokonaismäärän. tyypitv : noutaa ko. vuoden
tyyppiluokat(distinct).
    Lopuksi rakennetaan Googlen vaatima muoto kaavion data-syötteelle."""
    model = PeliTaulu
    template_name = 'peli/pelitaulu_vuodet.html'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        vuosi = self.kwargs['svuosi']
        suojelut = PeliTaulu.objects.filter(svuosi__exact=vuosi).values('tyyppi')

        tyypit = PeliTaulu.objects.filter(svuosi=vuosi).order_by('tyyppi').distinct().values('tyyppi')

        """Tyyppidatahaku haetaan oikeaan muotoon kannasta ja näin päästään eroon
kovakoodatusta datasta"""
        tyypit_kaikki = list(PeliTaulu.objects.filter().order_by('tyyppi').distinct().values_list('tyyppi',
flat=True))
```

```
        """Rakennetaan hauista data kaavioon. Ensin lista valitun vuoden kohteista vs. kaikki kohdetyypit, tulos"""
        tulos = []
        for tyyppi in tyypit_kaikki:
            kierros = (tyypit.filter(tyyppi=tyyppi).annotate(Count('tyyppi')))
            for item in kierros:
                tulos.append(item)

        """Sitten muokataan tulosta siten, että siihen lisätään otsikkotiedot ja tyypin data oikeassa muodossa, sd."""
        i = 0
        kk = len(tulos)
        sd = [['Suojeluvuosi', 'Uusia päätöksiä']]
        for kk in tulos:
            sd.append([kohdetyyppi for kohdetyyppi in tulos[i].values()])
            i = i+1

        context = {
            'sd': sd,
            'vuosi': vuosi,
            'tyypit': tyypit,
            'suojelut': suojelut,
        }
        return context
```