**ПРИЛОЖЕНИЕ А**

**Листинг программы**

Классы и функции представления:

from typing import Any

from django.db.models.query import QuerySet

from django.shortcuts import render, redirect

import json

from django.http import Http404

from django.core.serializers import serialize

from django.views.generic.base import TemplateView

from django.views.generic.list import ListView

from django.views.generic.detail import DetailView

from django.contrib.contenttypes.models import ContentType

from .models import Marker, ModelVillages, VillagesImageSet, ModelSights, ModelEvents, EventsImageSet

from .forms import EventForm, EventImageFormSet

class MarkersMapView(TemplateView):

template\_name = 'main/map.html'

def get\_context\_data(self, \*\*kwargs):

context = super().get\_context\_data(\*\*kwargs)

context['markers'] = json.loads(serialize('geojson', Marker.objects.all()))

context['markers\_obj'] = Marker.objects.exclude(village\_\_name = 'Хатынь')

return context

class IndexView(TemplateView):

template\_name = 'main/main-page.html'

def get\_context\_data(self, \*\*kwargs):

context = super().get\_context\_data(\*\*kwargs)

context['title'] = 'Main page'

context['villages\_obj'] = ModelVillages.objects.all()[:6]

context['villages\_obj\_inscriptions'] = ModelVillages.objects.exclude(stone\_inscription = "")[:12]

return context

class BellsView(TemplateView):

template\_name = 'main/bells-information.html'

def get\_context\_data(self, \*\*kwargs):

context = super().get\_context\_data(\*\*kwargs)

context['title'] = 'Bells information'

context['bells\_obj'] = ModelSights.objects.filter(name='Колокола Хатыни')

return context

class MonumentView(TemplateView):

template\_name= 'main/monument-information.html'

def get\_context\_data(self, \*\*kwargs):

context = super().get\_context\_data(\*\*kwargs)

context['title'] = 'Monument information'

context['monument\_obj'] = ModelSights.objects.filter(name='Памятник "Непокоренный"')

return context

class VillagesListView(ListView):

model = ModelVillages

paginate\_by = 8

context\_object\_name = 'villages\_obj'

template\_name = 'main/villages.html'

def get\_context\_data(self, \*, object\_list=None, \*\*kwargs):

context = super().get\_context\_data(object\_list=object\_list, \*\*kwargs)

context['title'] = 'Villages'

return context

def get\_queryset(self) -> QuerySet[Any]:

queryset = super().get\_queryset()

queryset = ModelVillages.objects.exclude(cover='')

return queryset

class VillageInfoView(DetailView):

model = ModelVillages

template\_name = 'main/village-information.html'

pk\_url\_kwarg = 'village\_id'

context\_object\_name = 'village\_obj'

def get\_context\_data(self, \*\*kwargs):

context = super().get\_context\_data(\*\*kwargs)

context['title'] = 'Village information'

context['img\_set'] = VillagesImageSet.objects.filter(post=self.get\_object().id)

return context

def create\_event(request):

if request.method == 'POST':

form = EventForm(request.POST)

formset = EventImageFormSet(request.POST, request.FILES, prefix='eventsimageset\_set')

if form.is\_valid() and formset.is\_valid():

event = form.save()

for i, form in enumerate(formset.forms):

if form.is\_valid() and f'eventsimageset\_set-{i}-image' in request.FILES:

image = request.FILES.getlist(f'eventsimageset\_set-{i}-image')[:6]

for img in image:

instance = EventsImageSet(post=event, image=img, content\_type=ContentType.objects.get\_for\_model(ModelEvents), object\_id=event.id)

instance.save()

return redirect('main:index')

else:

form = EventForm()

formset = EventImageFormSet(prefix='eventsimageset\_set')

context = {

'form': form,

'formset': formset,

'title': 'Create event'

}

return render(request, 'main/create-event.html', context)

class EventsListView(ListView):

model = ModelVillages

paginate\_by = 12

context\_object\_name = 'villages\_obj'

template\_name = 'main/events.html'

def get\_context\_data(self, \*, object\_list=None, \*\*kwargs):

context = super().get\_context\_data(object\_list=object\_list, \*\*kwargs)

context['title'] = 'Events'

context['events\_obj'] = ModelEvents.objects.filter(draft=False)

return context

def get\_queryset(self) -> QuerySet[Any]:

queryset = super().get\_queryset()

queryset = ModelVillages.objects.filter(draft=False)

return queryset

class EventInfoView(DetailView):

model = ModelEvents

template\_name = 'main/event-information.html'

pk\_url\_kwarg = 'event\_id'

context\_object\_name = 'event\_obj'

def get\_context\_data(self, \*\*kwargs):

context = super().get\_context\_data(\*\*kwargs)

context['title'] = 'Event information'

context['img\_set'] = EventsImageSet.objects.filter(post=self.get\_object().id)

return context

def get\_object(self, queryset=None):

event = super().get\_object(queryset=queryset)

if event.draft:

raise Http404("Event not found.")

return event