프로젝트 실제 웹사이트:

http://valto.shop

(운영 중 중간에 서버 점검이나, 업데이트 시 일시적으로 접속이 원활하지 않을 수 있습니다 😊)

팀 내 수행역할:

Back-End구축, DB설계, Server구축 및 운영, Excel로 DB자동 연산함수 구축, 웹 사이트 구상/기획, 이미지 편집, Front 사용자 편의 증진을 위한 배치/ 기능 제안

전체 프로젝트 구조:

Capstone →전체 프로젝트
- templates
- __init__.py
- settings.py
- urls.py

common →회원가입, 로그인, 로그아웃 기능 APP
- migrations
- templates
- __init__.py
- admin.py
- forms.py
- models.py
- urls.py
- views.py

kapchikachi →전체 기능 제어 APP
- migrations
- static
- templates
- __init__.py
- admin.py
- forms.py
- models.py
- urls.py
- views.py

venv →가상환경 파일

db.sqlite3 →전체 DB

manage.py →python으로 서버를 제어하는 코드 역할

1) Back-End구축 및 DB설계

Python코드로 작성한 웹 사이트 내부 설계 코드:

settings.py : python 코드로 작성된 알고리즘을 웹으로 보여줄 수 있게 변환해주는 Framework의 세부 setting파일(필요한 라이브러리를 넣고 각종 제어 가능)

외부API 및 라이브러리를 활용하여 웹 사이트의 편의성과 심미성 증대

```
DEBUG = False
ALLOWED_HOSTS = '*'

INSTALLED_APPS = [
    'kapchikachi',
    'common',

    #엑셀로 데이터 입출력
    'import_export',

    #입력 창 이쁘게 변경
    'markdownx',
    'crispy_forms',

    #웹 페이지 로그인 기능
    'django.contrib.sites',
    'allauth',
    'allauth.account',
    'allauth.socialaccount',

    #소셜 로그인 구현할 사이트 리스트
    'allauth.socialaccount.providers.google',
```

```
    'allauth.socialaccount.providers.kakao',
    'allauth.socialaccount.providers.naver',

    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

]
```

Django Framework의 세부 setting 코드는 생략

```
#로그인 완료 후 기본적으로 이동할 페이지
LOGIN_REDIRECT_URL = 'kapchikachi:CommentCreate'

#로그아웃 후 기본적으로 이동할 페이지
LOGOUT_REDIRECT_URL = 'kapchikachi:main'


#소셜 로그인 API
AUTHENTICATION_BACKENDS = [

    # Needed to login by username in Django admin, regardless of `allauth`
    'django.contrib.auth.backends.ModelBackend',

    # `allauth` specific authentication methods, such as login by e-mail
    'allauth.account.auth_backends.AuthenticationBackend',

]

#네이버 로그인
NAVER_CLIENT_ID = 'xHdErdYXbTYKqoWfP7Xd'
NAVER_SECRET_KEY = '@@@@@@@@@@@@@@'#네이버 secret key는 @으로 가렸습니다:)

#마크다운x 사이즈 자동 조절
MARKDOWNX_EDITOR_RESIZABLE = True

#크리스피폼
CRISPY_TEMPLATE_PACK = 'bootstrap4'
```

Capstone 프로젝트 :

전체 프로젝트로서, 세부 App으로 kapchikachi와 common을 가집니다.

Capstone/urls.py : 프로젝트 전체의 중앙 url을 처리하는 역할

```
from django.contrib import admin
from django.urls import path, include
from django.conf.urls import url
from markdownx import urls as markdownx
from django.conf import settings
from django.conf.urls.static import static
from django.views.static import serve

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('kapchikachi.urls', namespace="kapchikachi")),
    path('markdownx/', include('markdownx.urls')),
    path('common/', include('common.urls', namespace="common")),

    #소셜로그인 기능
    path('accounts/', include('allauth.urls')),
```

```
    #정적파일처리 (css, 사진)
    url(r'^media/(?P<path>.*)$', serve,{'document_root':
settings.MEDIA_ROOT}),
    url(r'^static/(?P<path>.*)$', serve,{'document_root':
settings.STATIC_ROOT}),

]

urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

common 로그인 기능 구현을 위한 App

common/models.py : common앱의 DB를 정의하고 구축

```
from django.db import models
from django.contrib.auth.models import User
from django.contrib.auth.models import AbstractBaseUser, BaseUserManager


class Userinfo(models.Model):
    username = models.ForeignKey(User, on_delete=models.CASCADE)
    # username = models.CharField(max_length=200,blank=True)
    # created = models.DateTimeField(auto_now_add=True)  # 작성시간
    email = models.EmailField(blank=True)
    gender_method=(
        (None,'선택'),
        ('0','남성'),
        ('1','여성')
    )
    gender = models.CharField(max_length=1, choices=gender_method,
null=False, blank=False)
    birth = models.DateField(blank=True)
```

마케팅 및 판매 통계 분석을 위해, 성별 / 연령대 정보만 직접회원가입 시 입력하도록 설계

common/views.py : common앱의 "회원가입" 기능을 위한 실제 동작 알고리즘 코드

```
from django.shortcuts import render, redirect
from .forms import *
from django.contrib import auth
from django.contrib.auth.models import User
from django.contrib.auth.decorators import login_required
from kapchikachi.models import Comment
from django.views.generic import ListView
def signup(request):
    if request.method=='POST':
        form1 = UserForm(request.POST)
        form2 = UserinfoForm(request.POST)
        # if form1.is_valid():# and form2.is_valid():

        if request.POST['password1']==request.POST['password2']:

            user=User.objects.create_user(username=request.POST['username'],
password=request.POST['password1'])

            auth.login(request, user,
backend='django.contrib.auth.backends.ModelBackend')

            #저장된 User 호출
            this_user = request.user.id
            user = User.objects.get(id=this_user)

            tmp_birth_year =request.POST['birth_year']
            tmp_birth_month =request.POST['birth_month']
            tmp_birth_day = request.POST['birth_day']
            user_birth=tmp_birth_year+'-'+tmp_birth_month+'-'+tmp_birth_day
```

```
            userinfo = Userinfo.objects.create(username=user,
email=request.POST['email'], gender=request.POST['gender'],
birth=user_birth )
            userinfo.save()

            return redirect('kapchikachi:CommentCreate')

    else:
        form1 = UserForm()
        form2 = UserinfoForm()

    return render(request, 'common/signup.html', {'user_form':form1,
'userinfo_form':form2})
```

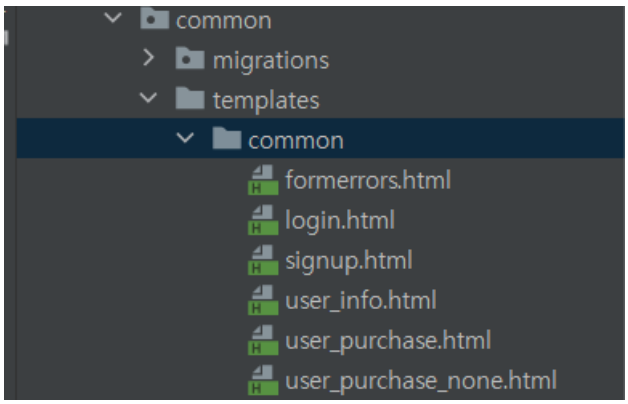
common/forms.py : 고객의 회원가입 정보를 입력받을 DB를 정의하고 데이터를 입력받는 역할

```
from django import forms
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User
from .models import Userinfo


YEARS= [x for x in range(1940,2022)]

class UserForm(UserCreationForm):

    class Meta:
        model = User
        fields = ("username",)


class UserinfoForm(forms.ModelForm):
    birth = forms.DateField(label='생년월일을 입력하세요.', initial="2021-11-16",
widget=forms.SelectDateWidget(years=YEARS))
    class Meta:
        model = Userinfo
        fields = ["gender", "birth", "email"]
        labels = {
            'gender' : '성별',
            'birth' : '생년월일',
            'email' : 'email'
        }
```

common/urls.py : 실제 사용자로부터 입력받을 상세 홈페이지 url 주소들과 views의 알고리즘 함수를 이어주는 역할

```
from django.urls import path, include
from django.contrib.auth import views as auth_views
from . import views, templates

app_name='common'

urlpatterns = [ #함수는 그냥 호출 / 클래스는 클래스명.as_view() 처리
    path('login/',
auth_views.LoginView.as_view(template_name='common/login.html'),
name='login'),
    path('signup/', views.signup, name='signup'),
    path('logout/', auth_views.LogoutView.as_view(), name='logout'),
]
```

Common/templates 내의 html파일: python으로 구축된 알고리즘이 작동하여 웹 페이지에서 "회원가입, 로그아웃, 로그인"관련 화면을 보여주는 역할



kapchikachi 웹 사이트의 핵심 기능을 담은 APP

실제 구매 항목을 입력받고, 가격을 산정하여 DB에 저장한 뒤, DB로부터 구매이력 및 배송조회, 전체 모금액 연산 후 반환, 구매자의 응원의 말 Random으로 3개 뽑아 반환, 각 html(웹 화면)의 이동을 제어

* 캡스톤디자인 valto팀의 프로젝트 명이 "같이가치"이므로 영문으로 프로젝트명 지정

kapchikachi/admin.py : 관리자(팀원들)이 DB를 sql형태가 아닌, 웹 사이트의 일반적인 관리가 가능하도록 데이터를 반환해주고 수정할 수 있게 해주는 창을 제어

```
from django.contrib import admin
from import_export.admin import ExportActionModelAdmin, ImportExportMixin,
ImportMixin
from .models import *

# Register your models here.


class CommentAdmin(ImportExportMixin, admin.ModelAdmin):
    pass
admin.site.register(Comment, CommentAdmin)

class OrderAdmin(ImportExportMixin, admin.ModelAdmin):
    pass
admin.site.register(Order, OrderAdmin)
```

kapchikachi/models.py : 고객의 구매정보 전반과 결제api연동을 위한 DB정의

```
from django.db import models
from django.contrib.auth.models import User
from django.core.validators import MinValueValidator, MaxValueValidator
from .iamport import validation_prepare, get_transaction
import hashlib, random, time
from django.db.models.signals import post_save
from markdownx.models import MarkdownxField

# Create your models here.

#구매자에게 입력 받을 코멘트
class Comment(models.Model):
    content = MarkdownxField(help_text='"석수" 관련 멘트를 남겨주시면 후원금을
석수에게 지급합니다!', blank=True) #구매자 코멘트 : 300자 이내로 제한
    created = models.DateTimeField(auto_now_add=True) #작성시간

    #유저가 누군지 입력 받게 할 지는 상의 필요
    author = models.ForeignKey(User, on_delete=models.CASCADE) #유저:
sns로그인 해야 이용 할 수 있게 하는 경우
```

```
    # #현재유저식별을 위한 사용자명

    #수취인 이름
    receiver_name = models.CharField(max_length=10, blank=False)

    #수취인 휴대폰 정보 입력 +정수형으로 할시 아마 0 은 생략될거같에서 문자열로 수정
    receiver_phone = models.CharField(max_length=12,blank=False)


    count_method=(
        (0, '선택(0 개)'),
        (1,1),
        (2,2),
        (3,3),
        (4,4),
        (5,5),
        (6,6),
        (7,7),
        (8,8),
        (9,9),
    )
    dog_case1 = models.IntegerField(default=0,choices=count_method,
blank=True)
    dog_case2 = models.IntegerField(default=0,choices=count_method,
blank=True)
    dog_keyring =models.IntegerField(default=0,choices=count_method,
blank=True)
    cat_case1 =models.IntegerField(default=0,choices=count_method,
blank=True)
    cat_case2 =models.IntegerField(default=0,choices=count_method,
blank=True)
    cat_keyring = models.IntegerField(default=0,choices=count_method,
blank=True )

    #1 인당 총 구매 금액  -> 배송비 + 원가는 나중에 빼기
    sell = models.IntegerField() #int 형으로 받게 설정 + 음수 불가하도록 설정
필요(재윤)

    #배송방법 선택
    shipping_method = (
        ('0','직접수령(    0 원)'),
        ('1','택배배송(2,800 원)')

    )
    shipping=models.CharField(max_length=1, choices=shipping_method,
blank=False, help_text="직접수령 선택시 담당자가 연락드립니다.")

    #배송지 #배송지 100 자 이내로 입력가능
    detail_address=MarkdownxField(blank=True, help_text="택배배송 선택시
입력해주세요.")

    #외대생 여부 확인
    hufs=models.BooleanField(default=False,blank=True)

    #결제 관련
    #입금자명
    deposit_name=models.CharField(max_length=10, blank=False)

    #구매번호(처리가 되었음을 알려주는 방법)
    order_num=models.IntegerField(blank=True,null=True)
```

```
    #입금완료여부
    deposit_check=models.IntegerField(blank=True,null=True)

    #실구매여부 확인 (구매 전: 0 , 구매 후:1)
    real_selled = models.IntegerField()  # 실구매여부 확인 (구매 전: 0 , 구매 후:1)

    #배송상태
    shipping_state_check=(
        ('0','준비중'),
        ('1','배송중'),
        ('2','배송완료')
    )
    shipping_state=models.CharField(max_length=1,
choices=shipping_state_check, default='0', blank=True)


    def __str__(self):
        return f'{self.author}'

    def get_donator_pk(self):
        return(self.pk)

class Order(models.Model):
    user_pk = models.IntegerField(blank=False) #사용자 식별 번호
    user_id = models.ForeignKey(User, on_delete=models.CASCADE)
    order_num = models.IntegerField(blank=False)  # 주문번호
    created = models.DateTimeField(auto_now_add=True, auto_now=False)  #
order_created
    real_selled = models.IntegerField()  # 실구매여부 확인 (구매 전: 0 , 구매 후:1)
    order_sell = models.IntegerField() #1 인당 총 구매 금액  -> 배송비 + 원가는
나중에 빼기


########주문 DB 모델#########
class OrderTransaction(models.Model):
    user_pk = models.IntegerField(blank=False) #사용자 식별 번호
    # user_id = models.CharField(max_length=300,blank=False) #사용자
아이디(식별용)
    user_id = models.ForeignKey(User, on_delete=models.CASCADE)
    # order_created =  models.DateTimeField(auto_now_add=True) #작성시간
    order_num = models.IntegerField(blank=False) #주문번호
    order_sell = models.IntegerField() #1 인당 총 구매 금액  -> 배송비 + 원가는
나중에 빼기
    real_selled = models.IntegerField() #실구매여부 확인 (구매 전: 0 , 구매 후:1)

    #######결제 시스템 연동 코드#######
    transaction_id = models.CharField(max_length=120, null=True, blank=True)
    order_id = models.CharField(max_length=120, unique=True) #order_num
    amount = models.PositiveIntegerField(default=0) #order_sell
    success = models.BooleanField(default=False)
    transaction_status = models.CharField(max_length=220, null=True,
blank=True)
    type = models.CharField(max_length=120)
    created = models.DateTimeField(auto_now_add=True,
auto_now=False)#order_created

    def __str__(self):
        return self.order_id

    class Meta:
        ordering = ['-created']
```

```
class OrderTransactionManager(models.Manager):
    # 새로운 트랜젝션 생성
    def create_new(self, user, amount, type, success=None,
transaction_status=None):
        if not user:
            raise ValueError("유저가 확인되지 않습니다.")
        short_hash = hashlib.sha1(str(random.random())).hexdigest()[:2]
        time_hash = hashlib.sha1(str(int(time.time()))).hexdigest()[-3:]
        base = str(user.email).split("@")[0]
        key = hashlib.sha1(short_hash + time_hash + base).hexdigest()[:10]
        new_order_id = "%s" % (key)
        # 아임포트 결제 사전 검증 단계
        validation_prepare(new_order_id, amount)
        # 트랜젝션 저장
        new_trans = self.model(
            user=user,
            order_id=new_order_id,
            amount=amount,
            type=type
        )
        if success is not None:
            new_trans.success = success
            new_trans.transaction_status = transaction_status
        new_trans.save(using=self._db)
        return new_trans.order_id

    # 생선된 트랜잭션 검증
    def validation_trans(self, merchant_id):
        result = get_transaction(merchant_id)
        if result['status'] is not 'paid':
            return result
        else:
            return None

    def all_for_user(self, user):
        return super(OrderTransactionManager, self).filter(user=user)

    def get_recent_user(self, user, num):
        return super(OrderTransactionManager, self).filter(user=user)[:num]

def new_point_trans_validation(sender, instance, created, *args, **kwargs):
    if instance.transaction_id:
        # 거래 후 아임포트에서 넘긴 결과
        v_trans = OrderTransaction.objects.validation_trans(
            merchant_id=instance.order_id
        )
        res_merchant_id = v_trans['merchant_id']
        res_imp_id = v_trans['imp_id']
        res_amount = v_trans['amount']
        # 데이터베이스에 실제 결제된 정보가 있는지 체크
        r_trans = OrderTransaction.objects.filter(
            order_id=res_merchant_id,
            transaction_id=res_imp_id,
            amount=res_amount
        ).exists()
        if not v_trans or not r_trans:
            raise ValueError('비정상적인 거래입니다.')
post_save.connect(new_point_trans_validation, sender=OrderTransaction)
```

* 간편결제를 위해 I'mport 사를 경유하여 kakaopay와 tosspayment를 적용하려 했으나, 인력부족(프론트 1인, 백엔드~서버~엑셀 DB연산 1인)과 시간적 여유 부족으로 인해 완벽히 적용하지 못했으나, 실제 구현한 back-end코드는 모두 첨부하였습니다😊

kapchikachi/views.py : 프로젝트의 로그인 관련 기능 외 모든 알고리즘, 동작 코드

```
import openpyxl
from django.shortcuts import render, redirect
from django.views import View
import random
from .models import * #모든 모델 상속
from django.views.generic import CreateView #Admin 에 띄우면 사용
from openpyxl import Workbook, load_workbook #엑셀 DB 연결
from .forms import CommentForm #forms 의 파일 상속
from django.utils import timezone #시간 설정 패키지
from django.contrib.auth.decorators import login_required
from django.contrib.auth.mixins import LoginRequiredMixin #'로그인'에 대한
조건을 담은 클래스
from django.http import HttpResponseRedirect
from django.shortcuts import render
from django.http import JsonResponse


#html 연결
def main_html(request): #재윤 기능 test html
    return render(request, 'kapchikachi/index.html')

def About_us(request):
    return render(request, 'kapchikachi/about.html')

def notfound_buy(request):
    return render(request, 'kapchikachi/notfound_buy.html')

def Notice_info(request):
    return render(request, 'kapchikachi/Notice_info.html')
def Qna(request):
    return render(request, 'kapchikachi/qna.html') #고동현 추가
#상품목록
def Airpoddog1(request):
    return render(request, 'kapchikachi/products_info1.html')
def Airpoddog2(request):
    return render(request, 'kapchikachi/products_info2.html')
def Airpodcat1(request):
    return render(request, 'kapchikachi/products_info3.html')
def Airpodcat2(request):
    return render(request, 'kapchikachi/products_info4.html')
def KeyringDog(request):
    return render(request, 'kapchikachi/products_info5.html')
def KeyringCat(request):
    return render(request, 'kapchikachi/products_info6.html')
def Products(request):
    return render(request, 'kapchikachi/products.html')

def SelectPayOption(request): #구매 방법 선택(카드결제: 네이버 스토이 이동 /
무통장입금: 우리 구매 페이지 이동)
    return render(request, 'kapchikachi/selectpayoption.html')


###############전역변수 선언###############
login_valid=0 #로그인 값이 유효한지 검사하는 전역변수
count=0 #조회수 체크용

###상품 가격 상품가격 수정(고동현)
price_dog_case1=8900
price_dog_case2=8900
price_dog_keyring=5000
```

```
price_cat_case1=8900
price_cat_case2=8900
price_cat_keyring=5000


###상품 원가
cost_dog_case1=6900 #2000원
cost_dag_case2=6900
cost_dog_keyring=4000 #1000원
cost_cat_case1=6900
cost_cat_case2=6900
cost_cat_keyring=4000

#배송비
shipping_fee=2800

#직접수령 장소
pickup_address="한국외국어대학교 글로벌캠퍼스 정문"
direct_pickup="한국외국어대학교 글로벌캠퍼스 정문"

###############기부자들이 로그인 -> 구매 과정###############
@login_required(login_url='common:login')
def CommentCreate(request):

    if request.method == 'POST':
        form = CommentForm(request.POST)

        if form.is_valid():
            comment=form.save(commit=False)
            comment.created=timezone.now()
            comment.author=request.user

            # 필수 정보 체크용(기존 정보 재 이용 시)
            if comment.receiver_name==None or comment.receiver_phone == None
or comment.shipping==None or comment.content==None:
                result = {
                    'form': form,
                    'warning': '입력해주세요!'
                }

                return render(request, 'kapchikachi/comment_form.html',
context=result)

            #값이 비어있으면 0으로 변경 null 오류 수정
            if comment.dog_case1 == '0':
                comment.dog_case1 =0

            if comment.dog_case2 == '0':
                comment.dog_case2 =0

            if comment.dog_keyring == '0':
                comment.dog_keyring =0

            if comment.cat_case1 == '0':
                comment.cat_case1 =0

            if comment.cat_case2 == '0':
                comment.cat_case2 =0

            if comment.cat_keyring == '0':
                comment.cat_keyring =0

            if
comment.dog_case1+comment.dog_case2+comment.dog_keyring+comment.cat_case1+co
```

```
mment.cat_case2+comment.cat_keyring == 0:
                result = {
                    'form': form,
                    'warning_product': '상품을 선택해주세요!'
                }

                return render(request, 'kapchikachi/comment_form.html',
context=result)

            #구매자 별 총 상품 금액 데이터형 오류 수정(고동현)

            selled_product = int(float(comment.dog_case1))*price_dog_case1 +
int(float(comment.dog_case2))*price_dog_case2 +
int(float(comment.dog_keyring))*price_dog_keyring +
int(float(comment.cat_case1))*price_cat_case1 +
int(float(comment.cat_case2))*price_cat_case2
+int(float(comment.cat_keyring))*price_cat_keyring

            #직접수령 시 배송비 x
            #delivery_fee=0

            if comment.shipping=='1': #택배 이용 시 배송비 책정
                # delivery_fee += int(shipping_fee)
                selled_product += 2800

            #배송방법에 따라 상세주소 입력 여부 결정
            if comment.shipping=='1' and comment.detail_address=='' or
comment.detail_address==None  :

                result = {
                    'form': form,
                    'warning_address': '배송받을 주소를 입력해주세요!'
                }

                return render(request, 'kapchikachi/comment_form.html',
context=result)

            if comment.shipping=='0':
                comment.detail_address=pickup_address

            #구매자 별 총 판매 금액
            total_cost=selled_product #+ delivery_fee

            #구매자 총 판매금액
            comment.sell=total_cost

            #'구매 전'임을 DB에 표시
            comment.real_selled=0

            # 상품 준비중 처리
            comment.shipping_state=0


            #구매자 정보 저장
            comment.save()

            return HttpResponseRedirect('/check/')

        else:
            return render(request, 'kapchikachi/comment_form.html', {'form':
form})
    else:
        form=CommentForm()
        result = {
```

```
            'form': form,
        }
        return render(request, 'kapchikachi/comment_form.html',context=result)


#####기 구매 이력이 있는 회원정보 호출#####
@login_required(login_url='common:login')
def Find_past_info(request):
    this_receiver_name=''
    this_receiver_phone=''

    if request.method != 'POST':
        this_user=request.user.id

        all_data = Comment.objects.all().order_by('-created') #주문 생성 시점
기준 '내림차순'정렬

        user = User.objects.get(id=this_user)
        before_sell=all_data.filter(author_id=user, real_selled=1)
        if int(before_sell.count())<1:
            form = CommentForm()
            result = {
                'form': form,
                'warning_info': '구매이력이 없습니다. 구매 부탁드립니다!',
            }
            return render(request, 'kapchikachi/comment_form.html',
context=result)
        else:
            before_info=before_sell.get()

            this_receiver_name=before_info.receiver_name
            this_receiver_phone=before_info.receiver_phone

            form = CommentForm()
            result = {
                'form':form,
                'this_receiver_name':this_receiver_name,
                'this_receiver_phone':this_receiver_phone,
            }
            return render(request, 'kapchikachi/comment_form2.html',
context=result)

    ####POST(Submit)받은 경우
    else:
        form = CommentForm(request.POST)

        if form.is_valid():
            comment=form.save(commit=False)
            comment.created=timezone.now()
            comment.author=request.user
            if this_receiver_name != '' and this_receiver_phone != '':
                comment.receiver_name=this_receiver_name
                comment.receiver_phone=this_receiver_phone

            # 필수 정보 체크용(기존 정보 재 이용 시)
            if comment.shipping==None or comment.content==None:
                result = {
                    'form': form,
                    'this_receiver_name':this_receiver_name,
                    'this_receiver_phone':this_receiver_phone,
                    'warnning': '입력해주세요!'
                }

                return render(request, 'kapchikachi/comment_form.html',
```

```
context=result)

            #값이 비어있으면 0 으로 변경
            if comment.dog_case1 == None:
                comment.dog_case1 =0

            if comment.dog_case2 == None:
                comment.dog_case2 =0

            if comment.dog_keyring == None:
                comment.dog_keyring =0

            if comment.cat_case1 == None:
                comment.cat_case1 =0

            if comment.cat_case2 == None:
                comment.cat_case2 =0

            if comment.cat_keyring == None:
                comment.cat_keyring =0

            #구매자 별 총 상품 금액
            selled_product = comment.dog_case1*price_dog_case1 +
comment.dog_case2*price_dog_case2 + comment.dog_keyring*price_dog_keyring +
comment.cat_case1*price_cat_case1 + comment.cat_case2*price_cat_case2 +
comment.cat_keyring*price_cat_keyring

            #직접수령 시 배송비 x
            deliverty_fee=0

            if comment.shipping=='1': #택배 이용 시 배송비 책정
                # deliverty_fee += shipping_fee
                deliverty_fee += 3500

            #배송방법에 따라 상세주소 입력 여부 결정
            if comment.shipping=='1' and comment.detail_address=='' :

                result = {
                    'form': form,
                    'warnning_address': '배송받을 주소를 입력해주세요!'
                }

                return render(request, 'kapchikachi/comment_form2.html',
context=result)

            if comment.shipping=='0':
                comment.detail_address=pickup_address

            #구매자 별 총 판매 금액
            total_cost=selled_product + deliverty_fee

            #구매자 총 판매금액
            comment.sell=total_cost+1000

            #'구매 전'임을 DB 에 표시
            comment.real_selled=0

            # 상품 준비중 처리
            comment.shipping_state=0

            #구매자 정보 저장
            comment.save()

            return HttpResponseRedirect('/check/')
```

```
#####구매전 재확인#####
@login_required(login_url='common:login')
def CheckBeforeSell(request):
    this_user=request.user.id

    all_data = Comment.objects.all().order_by('-created') #주문 생성 시점 기준
'내림차순'정렬

    user = User.objects.get(id=this_user)
    before_sell=all_data.filter(author_id=user, real_selled=0)

    if before_sell.count() ==0: #구매희망 내역이 없는 경우
        return render(request, 'kapchikapchi/not_selected.html') #구매희망
내역이 없다며, 구매창으로 이동할 수 있는 버튼 생성
    ####not_selected.html 에 구매창으로 넘길 수 있는 링크 걸기####

    selected_before_sell= before_sell.first()#가장 최신 구매희망 이력만 DB 로
불러오기

    #현재 유저가 구매할 정보 확인
    this_purchase_time=selected_before_sell.created #구매 신청 일자



    this_receiver_name =selected_before_sell.receiver_name  # 수취인 성함

    #수취인 번호
    if selected_before_sell.receiver_phone =='':
        this_receiver_phone = '수취인 번호를 다시 입력해주세요.'
    else:
        this_receiver_phone =selected_before_sell.receiver_phone  # 수취인 번호

    #배송방법 선택
    if selected_before_sell.shipping== '0':
        this_shipping = '직접수령'
    if selected_before_sell.shipping== '1':
        this_shipping = '택배배송'
    else:
        this_shipping = '배송방법을 다시 선택하세요.'

    if this_shipping == '1': #배송서비스 이용시 상세주소 입력
        this_receiver_address =selected_before_sell.detail_address  # 수취인
주소(상세주소)
    else:
        this_receiver_address = pickup_address #직접수령 장소

    # 구매상품 확인
    product_type = []  # 구매상품 종류
    if selected_before_sell.dog_case1 > 0:
        product_type.append(f'강아지 에어팟 1 세대
케이스{selected_before_sell.dog_case1}개')

    if selected_before_sell.dog_case2 > 0:
        product_type.append(f'강아지 에어팟 2 세대
케이스{selected_before_sell.dog_case2}개')

    if selected_before_sell.dog_keyring > 0:
        product_type.append(f'강아지 키링{selected_before_sell.dog_keyring}개')
```

```
    if selected_before_sell.cat_case1 > 0:
        product_type.append(f'고양이 에어팟 1 세대
케이스{selected_before_sell.cat_case1}개')

    if selected_before_sell.cat_case2 > 0:
        product_type.append(f'고양이 에어팟 2 세대
케이스{selected_before_sell.cat_case2}개')

    if selected_before_sell.cat_keyring > 0:
        product_type.append(f'고양이 키링{selected_before_sell.cat_keyring}개')

    this_purchase_product='' #상품을 담은 문자열
    for a in range(len(product_type)):
        if a!=0:
            this_purchase_product+=',\n'
        this_purchase_product+=product_type[a]

    #한마디
    this_purchase_comment = selected_before_sell.content

    # 구매금액
    this_purchase_amount = selected_before_sell.sell


    result={
        'this_purchase_time': this_purchase_time,
        'this_receiver_name':this_receiver_name,
        'this_receiver_phone':this_receiver_phone,
        'this_shipping':this_shipping, #배송방법 -> 동현 프론트추가 요청(재윤)
        'this_receiver_address':this_receiver_address,
        'this_purchase_product':this_purchase_product,
        'this_purchase_comment':this_purchase_comment,
        'this_purchase_amount':this_purchase_amount,

    }
    return render(request,'kapchikachi/check.html', context=result)

#입금완료 버튼 누를 경우 DB 처리 로직
@login_required(login_url='common:login')
def Deposit(request):
    # 사용자 DB 설절
    this_user = request.user.id
    user = User.objects.get(id=this_user)

    ##DB 추출 및 저장##
    all_data = Comment.objects.all().order_by('-created')  # 주문 생성 시점 기준
'내림차순'정렬

    try: #이미 구매처리가 된 경우엔 F5 버튼으로 새로고침해도 개인 구매내역으로 넘어가게
설정
        personal_data = all_data.filter(author_id=user, real_selled=0)
#사용자의 최신 구매요청 정보 추출
        personal_buy=personal_data[0]#.get()
    except:
        return HttpResponseRedirect('/personal/')

    this_pk = personal_buy.pk

    #주문번호를 첫주문을 1000 으로 하여 자동연산하도록 지정 --> 서버 실 운영 이후 DB 수정
불가능
    # 주문번호 (최초 시작 번호)
```

```
    this_order_num = 1000
    this_order_num += int(Order.objects.count())

    this_order_sell = personal_buy.sell

    #구매정보 저장
    this_order_data = Order.objects.create(user_pk = this_pk, user_id = user,
order_num = this_order_num, order_sell = this_order_sell, real_selled = 1)
    this_order_data.save()

    #comment DB 에서 판매완료 표시
    personal_buy.created = this_order_data.created
    personal_buy.real_selled = 1
    personal_buy.order_num = this_order_num # 주문번호
    personal_buy.deposit_check = 0 # 입금 확인 요청 상태
    personal_buy.save()

    ##사용자가 구매 완료한 시점이므로 구매요청했지만, 실구매가 이루어지지 않은 DB 는 모두
삭제
    delete_comment_data = Comment.objects.filter(author_id=user,
real_selled=0)
    if delete_comment_data.count() >= 0:
        delete_comment_data.delete()


    # 배송방법 출력
    if personal_buy.shipping == '0':
        this_data_shipping = "직접수령"
    if personal_buy.shipping == '1':
        this_data_shipping = "택배배송"


    # 관리자가 입금 내역 확인했는 지 여부 출력
    if personal_buy.deposit_check == 0:
        this_deposit_check = "입금확인 전"
    else:
        this_deposit_check = "입금확인 완료"

    result={
        'this_created': personal_buy.created, #주문시각
        'this_order_num': this_order_data.order_num, #주문번호
        'this_deposit_name': personal_buy.deposit_name, #예금주 명
        'this_deposit_check' : this_deposit_check, #입금확인 상태
        'this_receiver': personal_buy.receiver_name, #수취인 성함
        'this_receiver_phone': personal_buy.receiver_phone, #수취인 번호
        'this_shipping': this_data_shipping, #배송방법
        'this_address': personal_buy.detail_address, #상세주소
        'this_sell': personal_buy.sell, #판매가격
        'this_comment': personal_buy.content, #구매자의 한마디
    }

    return render(request, 'kapchikachi/checkbill.html', context=result)

#####개인구매이력 확인창#####
@login_required(login_url='common:login')
def Check_personal_buy(request):
    # 사용자 식별
    this_user = request.user.id
    user = User.objects.get(id=this_user)

    # 사용자의 구매정보 추출(내림차순)
```

```
    all_data=Comment.objects.order_by('-created') #내림차순 정렬
    personal_buy= all_data.filter(author_id=user, real_selled=1) #사용자의
구매정보 조회
    personal_buy_num= int(personal_buy.count()) #사용자의 구매횟수 조회

    # 사용자별 주문정보 추출(내림차순)
    all_order=Order.objects.order_by('-created')#내림차순 정렬
    personal_order= all_order.filter(user_id=user, real_selled=1)


    if personal_buy_num <1: #구매이력이 없는 경우
        return HttpResponseRedirect('/notfound_buy/') #urls 에서 notfound_buy
url 과 함수 지정 '구매이력이 없습니다. 캠페인에 동참해주세요.' 문구 출력하는 페이지로 이동
    elif personal_buy_num >3: #구매이력이 3 번보다 많은 경우
        personal_buy_num=3
    else: #1~3 번 구매한 경우
        pass

    #####구매정보#####(Comment Model DB)
    not_found=[]
    personal_receiver_name=[]
    personal_receiver_phone=[]
    personal_product=[]
    personal_shipping=[]
    personal_address=[] #shipping 이 1 인 경우 입력/ else 인 경우 직접수령
    personal_comment=[]
    personal_shipping_state = []
    personal_order_deposit_name = []
    personal_order_deposit_check = []


    #####주문정보#####(Order Model DB)
    personal_order_created=[]
    personal_order_num=[]
    time_str=[]




    for a in range(0,personal_buy_num): #장고 orm 에서 sql 의 값을 인덱싱 가능한데,
첫번째 튜플이 index=1 일 경우/ 만약 0 부터이면 for 문 값 바꾸기
        #####구매정보#####
        tmp = personal_buy[a]
        personal_receiver_name.append(f'수령인 명 : {tmp.receiver_name}')
        personal_receiver_phone.append(f'수령인 번호 : {tmp.receiver_phone}')
        if tmp.shipping == 1:
            personal_address.append(f'수령장소 : {tmp.detail_adress}')
            personal_shipping.append('수령방법 : 택배')
        else:
            personal_address.append(f'수령장소 : {pickup_address}')
            personal_shipping.append('수령방법 : 직접수령')
        personal_comment.append(f'응원의 말 : {tmp.content}')
        not_found.append('')

        personal_order_deposit_name.append(f'입금자 명: {tmp.deposit_name}')
        if tmp.deposit_check == 0:
            personal_order_deposit_check.append('주문확인 상태 : 입금확인 전')
        else:
            personal_order_deposit_check.append('주문확인 상태 : 입금확인 완료')

        tmp_product='구매상품 :\n'
```

```
        if tmp.dog_case1>=1:
            tmp_product+=f'강아지 에어팟 1 세대 케이스 {tmp.dog_case1}개\n'
        if tmp.dog_case2>=1:
            tmp_product += f'강아지 에어팟 2 세대 케이스 {tmp.dog_case2}개\n'
        if tmp.dog_keyring>=1:
            tmp_product += f'강아지 키링 {tmp.dog_keyring}개\n'
        if tmp.cat_case1>=1:
            tmp_product += f'고양이 에어팟 1 세대 케이스 {tmp.cat_case1}개\n'
        if tmp.cat_case2>=1:
            tmp_product += f'고양이 에어팟 2 세대 케이스 {tmp.cat_case2}개\n'
        if tmp.cat_keyring>=1:
            tmp_product += f'고양이 키링{tmp.cat_keyring}개\n'
        personal_product.append(tmp_product)

        if tmp.shipping_state =='0':
            personal_shipping_state.append('배송상태 : 준비중')
        elif tmp.shipping_state =='1':
            personal_shipping_state.append('배송상태 : 배송중')
        else:
            personal_shipping_state.append('배송상태 : 배송완료')

        #####주문정보#####
        tmp2 = personal_order[a]
        personal_order_created.append(tmp2.created)
        personal_order_num.append(f'주문번호 : {tmp2.order_num}')
        time_str.append('주문시각: ')


    cycle=3-len(personal_receiver_name)
    check_num=len(personal_receiver_name)
    if cycle>0:
        for b in range(1,cycle+1):
            #####구매정보#####
            not_found.append(f'{check_num+b}번째 구매이력이 없습니다.')
            personal_receiver_name.append('')
            personal_receiver_phone.append('')
            personal_product.append('')
            personal_shipping.append('')
            personal_address.append('')  # shipping 이 1 인 경우 입력/ else 인 경우
직접수령
            personal_comment.append('')
            personal_shipping_state.append('')

            personal_order_deposit_name.append('')
            personal_order_deposit_check.append('')
            # personal_cashreceipts.append('')
            # personal_cashreceipts_num.append('')

            #####주문정보#####
            personal_order_created.append('')
            personal_order_num.append('')
            time_str.append('')


    result={
        #####첫번째 주문#####
        'first_not_found' : not_found[0],
        'first_personal_receiver_name' : personal_receiver_name[0],
        'first_personal_receiver_phone' : personal_receiver_phone[0],
        'first_personal_product' : personal_product[0],
        'first_personal_shipping' : personal_shipping[0],
        'first_personal_address' : personal_address[0], # shipping 이 1 인 경우
```

```
입력/ else인 경우 직접수령
        'first_personal_comment' : personal_comment[0],
        'first_personal_order_created' : personal_order_created[0],
        'first_time_str' : time_str[0],
        'first_personal_order_num' : personal_order_num[0],
        'first_personal_shipping_state' : personal_shipping_state[0],
        'first_personal_order_deposit_name':personal_order_deposit_name[0],
        'first_personal_order_deposit_check':personal_order_deposit_check[0],

        #####두번째 주문#####
        'second_not_found': not_found[1],
        'second_personal_receiver_name': personal_receiver_name[1],
        'second_personal_receiver_phone': personal_receiver_phone[1],
        'second_personal_product': personal_product[1],
        'second_personal_shipping': personal_shipping[1],
        'second_personal_address': personal_address[1],  # shipping이 1인 경우
입력/ else인 경우 직접수령
        'second_personal_comment': personal_comment[1],
        'second_personal_order_created': personal_order_created[1],
        'second_time_str': time_str[1],
        'second_personal_order_num': personal_order_num[1],
        'second_personal_shipping_state': personal_shipping_state[1],
        'second_personal_order_deposit_name': personal_order_deposit_name[1],
        'second_personal_order_deposit_check':
personal_order_deposit_check[1],
        # 'second_personal_cashreceipts':personal_cashreceipts[1],#현금영수증
발급 여부 -> 동현 프론트 추가 요청(재윤)
        #
'second_personal_cashreceipts_num':personal_cashreceipts_num[1],#현금영수증
번호 발급 여부 -> 동현 프론트 추가 요청(재윤)

        #####세번째 주문#####
        'third_not_found': not_found[2],
        'third_personal_receiver_name': personal_receiver_name[2],
        'third_personal_receiver_phone': personal_receiver_phone[2],
        'third_personal_product': personal_product[2],
        'third_personal_shipping': personal_shipping[2],
        'third_personal_address': personal_address[2],  # shipping이 1인 경우
입력/ else인 경우 직접수령
        'third_personal_comment': personal_comment[2],
        'third_personal_order_created': personal_order_created[2],
        'third_time_str': time_str[2],
        'third_personal_order_num': personal_order_num[2],
        'third_personal_shipping_state': personal_shipping_state[2],
        'third_personal_order_deposit_name': personal_order_deposit_name[2],
        'third_personal_order_deposit_check': personal_order_deposit_check[2],
        # 'third_personal_cashreceipts':personal_cashreceipts[2],#현금영수증
발급 여부 -> 동현 프론트 추가 요청(재윤)
        # 'third_personal_cashreceipts_num':personal_cashreceipts_num[2]
#현금영수증 번호 발급 여부 -> 동현 프론트 추가 요청(재윤)
    }

    return render(request, 'kapchikachi/personal_buy_list.html',
context=result)



###############그래프 표시###############
def GetChart(request):
    #실제 판매된 상품만 객체로 저장
    all_data = Comment.objects.filter(real_selled=1)
    order_data = Order.objects.filter(real_selled=1)
```

```
    db_num = int(len(all_data))

    #각 상품 별 판매액
    selled_dog_case1=0
    selled_dog_case2=0
    selled_dog_keyring=0
    selled_cat_case1=0
    selled_cat_case2=0
    selled_cat_keyring=0

    for b in range(db_num):
        selled_dog_case1 += int(all_data[b].dog_case1) * (price_dog_case1 -
cost_dog_case1)
        selled_dog_case2 += int(all_data[b].dog_case2) * (price_dog_case2 -
cost_dag_case2)
        selled_dog_keyring += int(all_data[b].dog_keyring) *
(price_dog_keyring -cost_dog_keyring)
        selled_cat_case1 += int(all_data[b].cat_case1) * (price_cat_case1 -
cost_cat_case1)
        selled_cat_case2 += int(all_data[b].cat_case2) * (price_cat_case2 -
cost_cat_case2)
        selled_cat_keyring += int(all_data[b].cat_keyring) *
(price_cat_keyring -cost_cat_keyring)

    # 총 모금액 (판매액 아님)
    total_selled =
selled_dog_case1+selled_dog_case2+selled_dog_keyring+selled_cat_case1+selled
_cat_case2+selled_cat_keyring

    #유기견 후원액
    money_for_dog=selled_dog_case1+selled_dog_case2+selled_dog_keyring

    #유기묘 후원액
    money_for_cat=selled_cat_case1+selled_cat_case2+selled_cat_keyring

    total_donation=money_for_cat+money_for_dog

    #응원의 글(랜덤 출력) 3 가지만 랜덤으로 출력
    ran_num = []
    ran_comment = ['','','']
    ran_comment_time = ['','','']

    if db_num>0:
        ran_num.append(random.randint(1, db_num)-1)
        if db_num>1:
            while True:
                tmp=random.randint(1, db_num)-1
                if (all_data[tmp].content!='' or all_data[tmp].content!=None)
and ran_num[0]!= tmp:
                    break
            ran_num.append(tmp)
            if db_num>2:
                while True:
                    tmp = random.randint(1, db_num) - 1
                    if (all_data[tmp].content != '' or all_data[tmp].content !=
None) and ran_num[1] != tmp and ran_num[0]!= tmp:
                        break
                ran_num.append(tmp)


    if len(ran_num)>0:
        for i in range(len(ran_num)):
            tmp=int(ran_num[i])
            ran_comment_time[i]=order_data[tmp].created
            tmp_name=all_data[tmp].receiver_name
```

```
            ran_name= tmp_name[:1] +'**'
            ran_comment[i] = f'{ran_name}: {all_data[tmp].content}'


    result={
        'total_selled':total_selled,
        'dog_case1':selled_dog_case1,
        'dog_case2':selled_dog_case2,
        'dog_keyring':selled_dog_keyring,
        'cat_case1':selled_cat_case1,
        'cat_case2':selled_cat_case2,
        'cat_keyring':selled_cat_keyring,
        'money_for_dog':money_for_dog,
        'money_for_cat':money_for_cat,
        'ran_comment1':ran_comment[0],
        'ran_comment2':ran_comment[1],
        'ran_comment3':ran_comment[2],
        'ran_comment_time1':ran_comment_time[0],
        'ran_comment_time2':ran_comment_time[1],
        'ran_comment_time3':ran_comment_time[2],
        'total_donation':total_donation
    }
    return render(request, 'kapchikachi/index.html',context=result)


##############################################################################

#####결제 시스템#####
@login_required(login_url='common:login')
class OrderCheckoutAjaxView(View):
    def post(self, request, *args, **kwargs):
        if not request.user.is_authenticated():
            return JsonResponse({}, status=401)
        user = request.user

        #사용자 DB 설정(내가 추가)
        all_data = Comment.objects.order_by('-created')
        this_user = request.user.id
        user = User.objects.get(id=this_user)

        personal_data = all_data.filter(author_id=user, real_selled=0)
#사용자의 최신 구매요청 정보 추출
        personal_buy=personal_data[0]#.get()

        this_pk = personal_buy.pk
        this_id = personal_buy.author_id

        #주문번호를 첫주문을 1000으로 하여 자동연산하도록 지정 --> 서버 실 운영 이후
DB 수정 불가능
        this_order_num = 1000
        this_order_num += int(OrderTransaction.objects.count())

        this_order_sell = personal_buy.sell

        # amount = request.POST.get('amount')
        amount = personal_buy.sell
        type = request.POST.get('type')
        try:
            trans = OrderTransaction.objects.create_new(
                user_id=user,
                amount=amount,
                type=type,
                user_pk=this_pk,
                order_num=this_order_num,
```

```
                order_sell=this_order_sell,
                real_selled=0,
            )
        except:
            trans = None
        if trans is not None:
            data = {
                "works": True,
                "merchant_id": trans
            }
            return JsonResponse(data)
        else:
            return JsonResponse({}, status=401)

@login_required(login_url='common:login')
class OrderImpAjaxView(View):
    def post(self, request, *args, **kwargs):
        if not request.user.is_authenticated():
            return JsonResponse({}, status=401)
        user = request.user
        merchant_id = request.POST.get('merchant_id')
        imp_id = request.POST.get('imp_id')
        amount = request.POST.get('amount')
        # amount =
        try:
            trans = OrderTransaction.objects.get(
                user_id=user,
                order_id=merchant_id,
                amount=amount
            )
        except:
            trans = None
        if trans is not None:
            trans.transaction_id = imp_id
            trans.success = True
            trans.save()
            data = {
                "works": True
            }
            return JsonResponse(data)
        else:
            return JsonResponse({}, status=401)

def purchase_order(request):
    template = 'kapchikachi/purchase_order.html'
    return render(request, template)



#함수의 연산 적용 예정
#####결제 완료 후 결제완료 창######
@login_required(login_url='common:login')
def check_bill(request):
    #사용자 식별
    this_user = request.user.id
    user = User.objects.get(id=this_user)

    ##현 함수에서 사용할 DB 추출
    unsorted_comment_data = Comment.objects.filter(author_id=user,
real_selled=0)
    comment_data = unsorted_comment_data.order_by('-created')
    this_comment_data = comment_data[0]#.get() #현 함수에서 사용할 db

    unsorted_order_data = OrderTransaction.objects.filter(user_id=user,
real_selled=0)
```

```
    order_data = unsorted_order_data.order_by('-created')
    if order_data.count()<=1:
        this_order_data = order_data
    else:
        this_order_data = order_data[0]#.get() #현 함수에서 사용할 db

    #comment DB 에서 판매완료 표시
    this_comment_data.created = this_order_data.created
    this_comment_data.real_selled = 1
    this_comment_data.save()

    #order DB 에서 판매완료 표시
    this_order_data.real_selled = 1
    this_order_data.save()

    ##사용자가 구매 완료한 시점이므로 구매요청했지만, 실구매가 이루어지지 않은 DB 는 모두
삭제
    delete_comment_data = Comment.objects.filter(author_id=user,
real_selled=0)
    if delete_comment_data.count() >= 0:
        delete_comment_data.delete()

    delete_order_data = OrderTransaction.objects.filter(user_id=user,
real_selled=0)
    if delete_order_data.count() >= 0:
        delete_order_data.delete()

    result={
        'this_created': this_comment_data.created, #주문시각
        'this_order_num': this_order_data.order_num, #주문번호
        'this_receiver': this_comment_data.receiver_name, #수취인 성함
        'this_receiver_phone': this_comment_data.receiver_phone, #수취인 번호
        'this_shipping': this_comment_data.shipping, #배송방법
        'this_address': this_comment_data.detail_address, #상세주소
        'this_sell': this_comment_data.sell, #판매가격
        'this_comment': this_comment_data.content, #구매자의 한마디
    }

    return render(request, 'kapchikachi/checkbill.html', context=result)



#DB 내의 모든 데이터 삭제하기(관리용- 거의 쓸 일 없을 것으로 판단)
def DeleteData(request):
    Comment.objects.all().delete() #쿼리의 모든 Comment 데이터 삭제
    Order.objects.all().delete() #쿼리의 모든 Order 데이터 삭제
    return render(request, 'kapchikachi/delete.html')
```

kapchikachi/forms.py: 고객의 구매정보를 입력받기 위한 form → html의 input 칸에서 입력받을 DB 정의

```
from django import forms
from .models import Comment

class CommentForm(forms.ModelForm):
    class Meta:
        model= Comment
        fields=['receiver_name','receiver_phone','content', 'dog_case1',
'dog_case2', 'dog_keyring', 'cat_case1', 'cat_case2', 'cat_keyring',
```

```
'shipping', 'detail_address', 'hufs','deposit_name']
        labels= {
            'receiver_name': '수취인 성함', #배송받는 분
            'receiver_phone': '수취인 연락처',
            'content': '남기고 싶은 말',
            'dog_case1': '강아지 에어팟 1,2 케이스',
            'dog_case2': '강아지 에어팟프로 케이스',
            'dog_keyring': '강아지 키링',
            'cat_case1': '고양이 에어팟 1,2 케이스',
            'cat_case2': '고양이 에어팟프로 케이스',
            'cat_keyring': '고양이 키링',
            'shipping': '수령방법 선택', #배송비 책정용
            'detail_address': '배송지 입력',
            'hufs': '외대생 여부',
            'deposit_name': '예금주', #입금하는 사람 이름
        }
```

kapchikachi/urls.py : 모든 세부 기능을 제어하는 kapchikachi/views.py의 알고리즘과 홈페이지 url을 연결하는 역할

```
from django.urls import path, include
from . import views
from .views import*
from django.views import View

app_name='kapchikachi'


urlpatterns = [ #함수는 그냥 호출 / 클래스는 클래스명.as_view() 처리
    ###메인 페이지 노출& 그래프, 댓글 노출###
    path('',views.GetChart, name='main'),

    ###운영진 소개###
    path('aboutus/', views.About_us, name='aboutus'),

    ###결제 유형 선택 페이지###
    path('SelectPayOption/', views.SelectPayOption, name='SelectPayOption'),
#추가할 url 페이지

    ###구매 페이지 시스템###
    path('purchase/', views.CommentCreate, name='CommentCreate'),
    path('pastinfo/', views.Find_past_info, name='FindPastInfo'), #기존 구매
이력창으로 넘기는 url
    path('check/', views.CheckBeforeSell, name='check'),
    path('checkbill/', views.Deposit, name="check_bill"),

    ####결제 시스템####
    # path('pay/', views.pay, name="pay"),
    ####아엠포트####
    path('charge/', views.purchase_order, name='purchase_order'),
    path('checkout/', views.OrderCheckoutAjaxView, name='order_checkout'),
    path('validation/', views.OrderImpAjaxView, name='order_validation'),

    path('payment/', views.real_sell, name="real_sell"),
    # path('checkbill/', views.check_bill, name="check_bill"),

    ####구매이력확인####
    path('personal/', views.Check_personal_buy, name='Check_personal_buy'),
    path('notfound_buy/', views.notfound_buy, name='notfound_buy'),
```

```
    ####사업자정보고지용####
    path('noticeinfo/', views.Notice_info, name='Notice_info'),
    path('qna/', views.Qna, name='Qna'),#고동현이 추가
    ####관리자용 페이지####
    path('download/', views.GetData, name='GetData'),
    path('delete/', views.DeleteData, name='DeleteData'),

    #상품목록
    path('dogairpod1/',views.Airpoddog1, name='DogAirpod12'),
    path('dogairpod2/', views.Airpoddog2, name='DogAirpodpro'),
    path('catairpod1/', views.Airpodcat1, name='CatAirpod12'),
    path('catairpod2/', views.Airpodcat2, name='CatAirpodpro'),
    path('dogkeyring/', views.KeyringDog, name='DogKeyring'),
    path('catkeyring/', views.KeyringCat, name='CatKeyring'),
     path('products/', views.Products, name='Products'),
]
```

*** 현재 적용은 못했으나, 향후 연동할 간편결제 기능**

kapchikachi/iamport.py : iamport를 통한 API연동하기 위해 설계한 logic

```
import requests
import json
from django.conf import settings

def get_access_token():
    access_data = {
        'imp_key': settings.IAMPORT_KEY,
        'imp_secret': settings.IAMPORT_SECRET
    }
    url = "https://api.iamport.kr/users/getToken"
    req = requests.post(url, data=access_data)
    access_res = req.json()
    if access_res['code'] is 0:
        return access_res['response']['access_token']
    else:
        return None
def validation_prepare(merchant_id, amount, *args, **kwargs):
    access_token = get_access_token()
    if access_token:
        access_data = {
            'merchant_uid': merchant_id,
            'amount': amount
        }
        url = "https://api.iamport.kr/payments/prepare"
        headers = {
            'Authorization': access_token
        }
        req = requests.post(url, data=access_data, headers=headers)
        res = req.json()
        if res['code'] is not 0:
            raise ValueError("API 연결에 문제가 생겼습니다.")
    else:
        raise ValueError("인증 토큰이 없습니다.")

def get_transaction(merchant_id, *args, **kwargs):
    access_token = get_access_token()
    if access_token:
        url = "https://api.iamport.kr/payments/find/" + merchant_id
        headers = {
            'Authorization': access_token
        }
        req = requests.post(url, headers=headers)
```

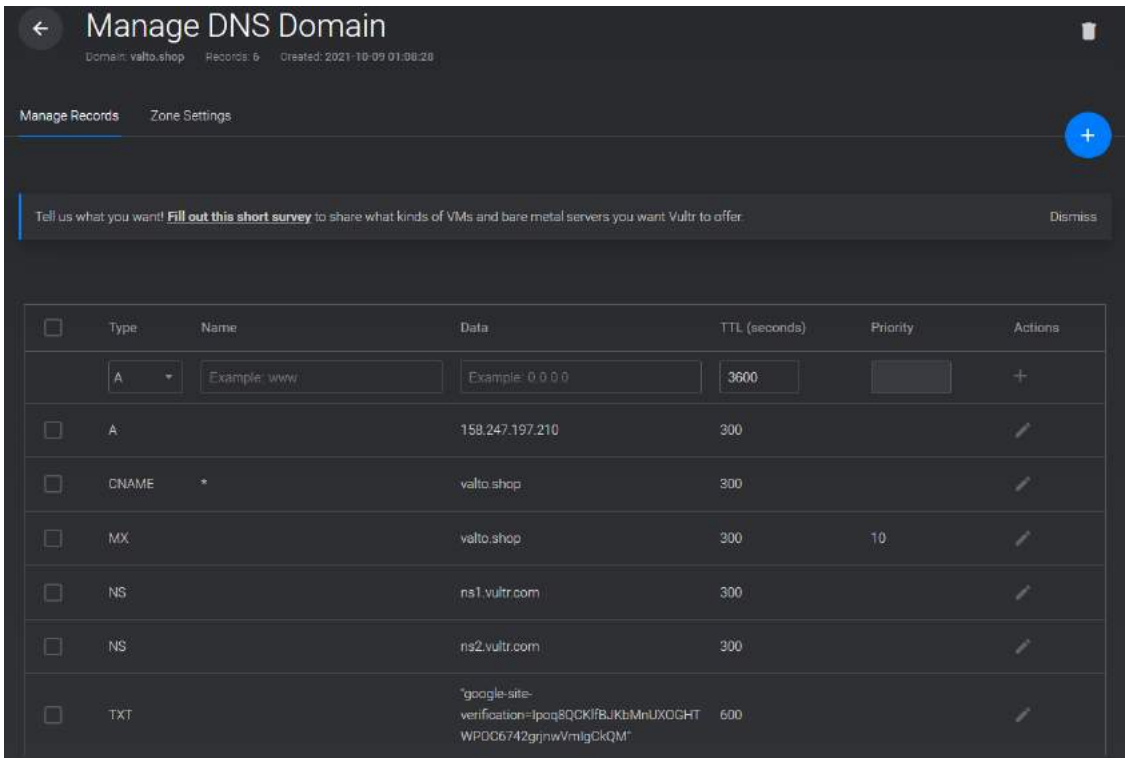
```
        res = req.json()
        if res['code'] is 0:
            context = {
                'imp_id': res['response']['imp_uid'],
                'merchant_id': res['response']['merchant_uid'],
                'amount': res['response']['amount'],
                'status': res['response']['status'],
                'type': res['response']['pay_method'],
                'receipt_url': res['response']['receipt_url']
            }
            return context
        else:
            return None
    else:
        raise ValueError("인증 토큰이 없습니다.")
```

2) Server구축 및 문제 해결

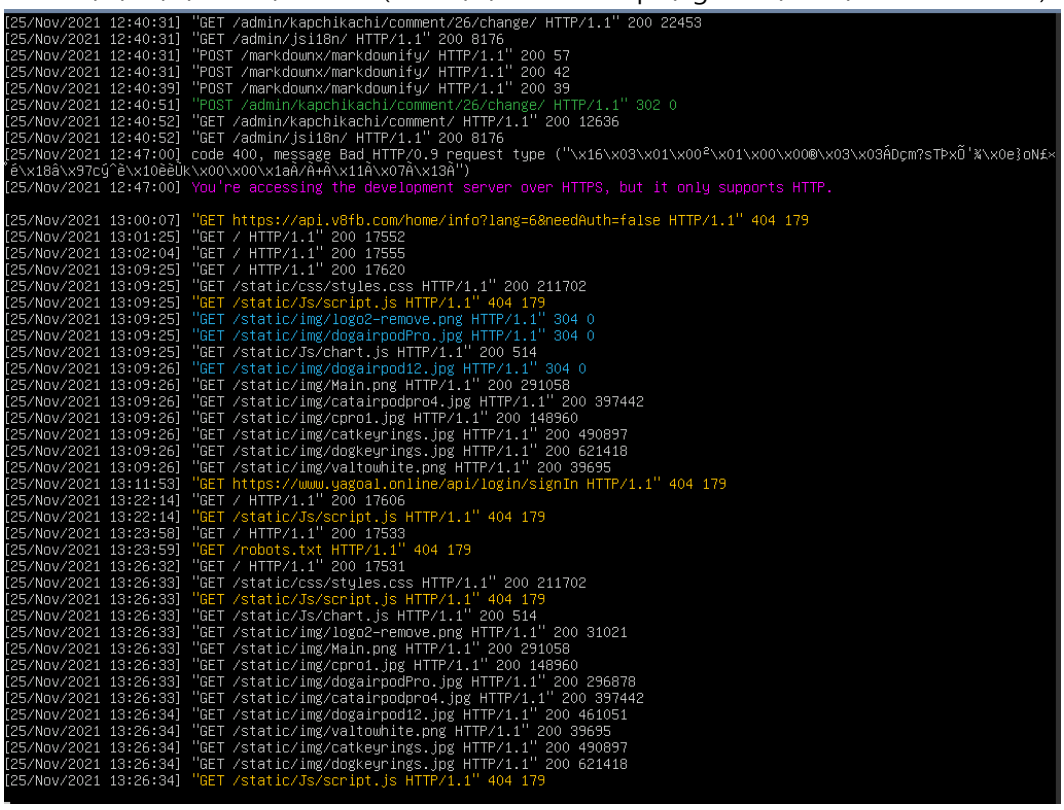
학교에서 linux언어를 사용해서 server를 실제로 구현하는 방법이나 error상황에 대처하는 방법을 배운 적이 없기에, 구글 검색을 하고 사설 강의를 통해 server를 구축하고 error를 제어했습니다.

* window server는 linux server에 비해 4~5배 이상의 가격이라, 최대한 예산을 절약하는 방법을 활요했습니다.

도메인 제공 업체 gabia을 통한 프로젝트 홈페이지의 url 보유(http://valto.shop)



Vultr 서버 내의 DNS주소 설정(vultr에서 발급받은 ip에 gabia의 도메인을 입힌 모습)



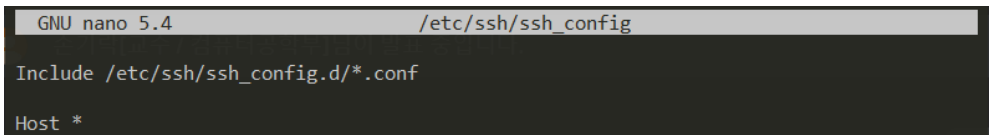
```
[25/Nov/2021 12:40:31] "GET /admin/kapchikachi/comment/26/change/ HTTP/1.1" 200 22453
[25/Nov/2021 12:40:31] "GET /admin/jsi18n/ HTTP/1.1" 200 8176
[25/Nov/2021 12:40:31] "POST /markdownx/markdownify/ HTTP/1.1" 200 57
[25/Nov/2021 12:40:31] "POST /markdownx/markdownify/ HTTP/1.1" 200 42
[25/Nov/2021 12:40:39] "POST /markdownx/markdownify/ HTTP/1.1" 200 39
[25/Nov/2021 12:40:51] "POST /admin/kapchikachi/comment/26/change/ HTTP/1.1" 302 0
[25/Nov/2021 12:40:52] "GET /admin/kapchikachi/comment/ HTTP/1.1" 200 12636
[25/Nov/2021 12:40:52] "GET /admin/jsi18n/ HTTP/1.1" 200 8176
[25/Nov/2021 12:47:00] code 400, message Bad HTTP/0.9 request type ("\x16\x03\x01\x00ª\x01\x00\x00¦\x03\x03ÂDçm?sTPxÕ'¾\x0e}oN£×
ê\x18â\x97cý^ê\x10ëèÙk\x00\x00\x1aÂ/À+À\x11À\x07À\x13Â")
[25/Nov/2021 12:47:00] You're accessing the development server over HTTPS, but it only supports HTTP.

[25/Nov/2021 13:00:07] "GET https://api.v8fb.com/home/info?lang=68needAuth=false HTTP/1.1" 404 179
[25/Nov/2021 13:01:25] "GET / HTTP/1.1" 200 17552
[25/Nov/2021 13:02:04] "GET / HTTP/1.1" 200 17555
[25/Nov/2021 13:09:25] "GET / HTTP/1.1" 200 17620
[25/Nov/2021 13:09:25] "GET /static/css/styles.css HTTP/1.1" 200 211702
[25/Nov/2021 13:09:25] "GET /static/Js/script.js HTTP/1.1" 404 179
[25/Nov/2021 13:09:25] "GET /static/img/logo2-remove.png HTTP/1.1" 304 0
[25/Nov/2021 13:09:25] "GET /static/img/dogairpodPro.jpg HTTP/1.1" 304 0
[25/Nov/2021 13:09:25] "GET /static/Js/chart.js HTTP/1.1" 200 514
[25/Nov/2021 13:09:26] "GET /static/img/dogairpod12.jpg HTTP/1.1" 304 0
[25/Nov/2021 13:09:26] "GET /static/img/Main.png HTTP/1.1" 200 291058
[25/Nov/2021 13:09:26] "GET /static/img/catairpodpro4.jpg HTTP/1.1" 200 397442
[25/Nov/2021 13:09:26] "GET /static/img/cpro1.jpg HTTP/1.1" 200 148960
[25/Nov/2021 13:09:26] "GET /static/img/catkeyrings.jpg HTTP/1.1" 200 490897
[25/Nov/2021 13:09:26] "GET /static/img/dogkeyrings.jpg HTTP/1.1" 200 621418
[25/Nov/2021 13:09:26] "GET /static/img/valtowhite.png HTTP/1.1" 200 39695
[25/Nov/2021 13:11:53] "GET https://www.yagoal.online/api/login/signIn HTTP/1.1" 404 179
[25/Nov/2021 13:22:14] "GET / HTTP/1.1" 200 17606
[25/Nov/2021 13:22:14] "GET /static/Js/script.js HTTP/1.1" 404 179
[25/Nov/2021 13:23:58] "GET / HTTP/1.1" 200 17533
[25/Nov/2021 13:23:59] "GET /robots.txt HTTP/1.1" 404 179
[25/Nov/2021 13:26:32] "GET / HTTP/1.1" 200 17531
[25/Nov/2021 13:26:33] "GET /static/css/styles.css HTTP/1.1" 200 211702
[25/Nov/2021 13:26:33] "GET /static/Js/script.js HTTP/1.1" 404 179
[25/Nov/2021 13:26:33] "GET /static/Js/chart.js HTTP/1.1" 200 514
[25/Nov/2021 13:26:33] "GET /static/img/logo2-remove.png HTTP/1.1" 200 31021
[25/Nov/2021 13:26:33] "GET /static/img/Main.png HTTP/1.1" 200 291058
[25/Nov/2021 13:26:33] "GET /static/img/cpro1.jpg HTTP/1.1" 200 148960
[25/Nov/2021 13:26:33] "GET /static/img/dogairpodPro.jpg HTTP/1.1" 200 296878
[25/Nov/2021 13:26:33] "GET /static/img/catairpodpro4.jpg HTTP/1.1" 200 397442
[25/Nov/2021 13:26:34] "GET /static/img/dogairpod12.jpg HTTP/1.1" 200 461051
[25/Nov/2021 13:26:34] "GET /static/img/valtowhite.png HTTP/1.1" 200 39695
[25/Nov/2021 13:26:34] "GET /static/img/catkeyrings.jpg HTTP/1.1" 200 490897
[25/Nov/2021 13:26:34] "GET /static/img/dogkeyrings.jpg HTTP/1.1" 200 621418
[25/Nov/2021 13:26:34] "GET /static/Js/script.js HTTP/1.1" 404 179
```

실제 운영하며 관리하고 있는 server의 사용자 접속 현황 log기록

초기에 Ubuntu의 linux setting방법을 정확히 몰라, 지속적으로

"Broken pip from ip번호" , "client_loop: send disconnect: Broken pipe", "TimeoutError" 등의 server와 client측 네트워크의 연결 문제가 서버를 가동하고 2~3시간 뒤에 발생했습니다.

이를 해결하고자 서버 통신을 찾아보고, 검색을 해보며 해결하였습니다.

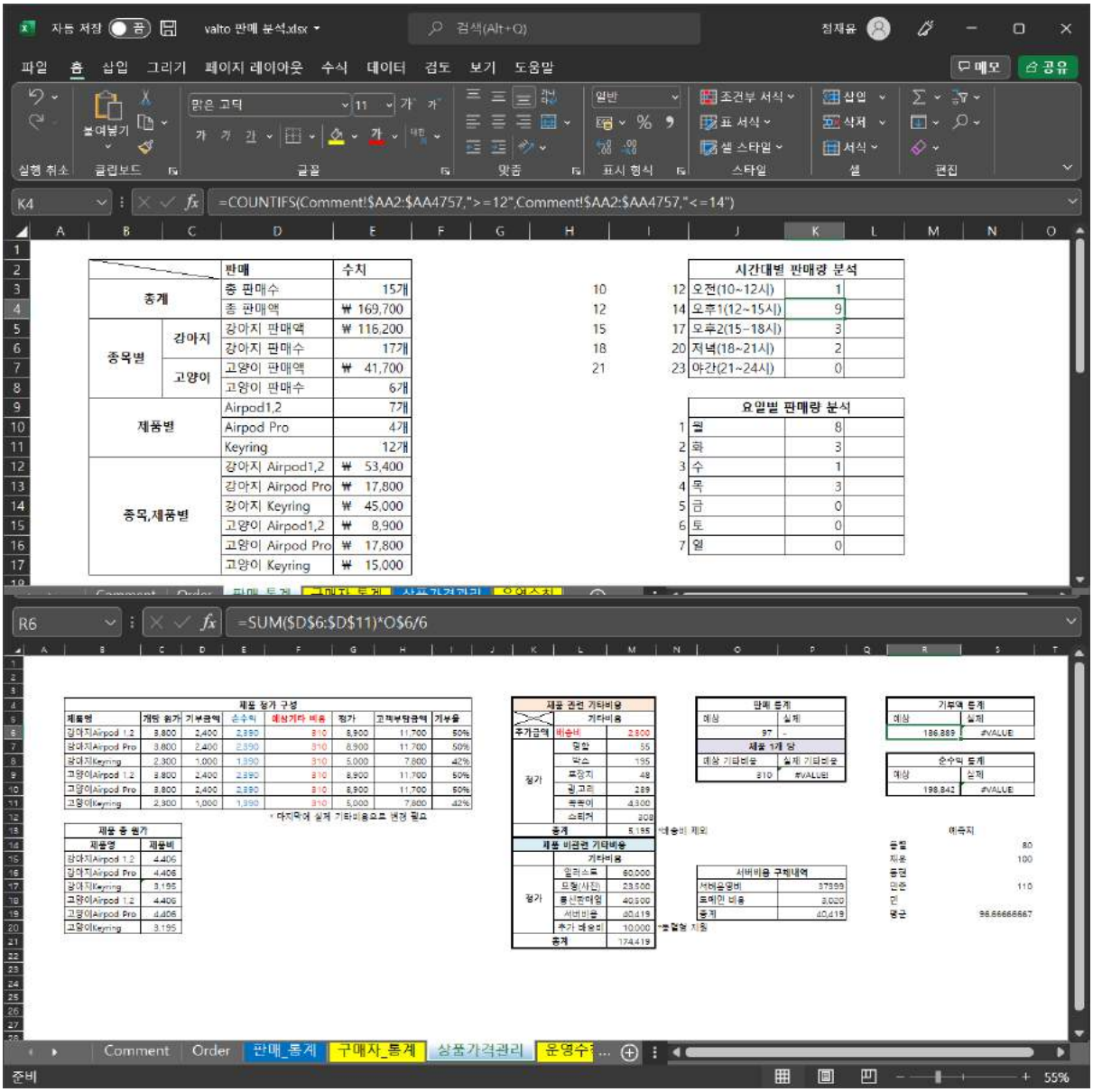


```
 GNU nano 5.4                    /etc/ssh/ssh_config

Include /etc/ssh/ssh_config.d/*.conf

Host *
```

```
    ServerAliveInterval 120
    SendEnv LANG LC_*
    HashKnownHosts yes
    GSSAPIAuthentication yes
```

server단과 client단의 설정을 각각 변경

3) Excel로 DB자동 연산함수 구축



각 수치는 Comment와 Order로 웹 사이트의 DB를 받고, 판매_통계/구매자_통계/상품가격관리에서 Comment와 Order의 정보를 자동으로 계산하여 수치로 표현하도록 Excel을 구축했습니다.

또한 운영수칙을 정하여, 각 팀원들이 웹 사이트와 고객센터를 운영할 때 효율성이 강화되도록 조치했습니다.

* 위의 자료는 11/25 13:45분까지의 판매이력입니다

4) 웹 사이트 구상/기획(기능, 전반 디자인)

+ Front 사용자 편의 증진을 위한 배치/ 기능 제안

* 아래 내용은 웹 디자인 설계 기획안 일부 발췌문 입니다.

디자인 핵심: Simple is the Best

* 우리 웹의 메인페이지는 사진과 글이 많은 페이지 -> 가독성이 가장 중요!

메인페이지의 웹 디자인 깔끔하게, 색감 구성은 색 배합을 3색 정도로 제한

(각각 색 통일: 버튼 색1, 배경색1, 콘텐츠박스색1)

특히 로고가 우리 팀을 상징하므로, 로고와 유사한 색배합

(배경: 흰색, 페이지 박스(글/사진의 테두리 박스): 옅은 갈색or 베이지 (로고와유사하게), 버튼: 검정)

디테일한 디자인은 사진 / 글로 눈에 띄도록 강조 가능

메인페이지의 역할:화려함x, 감성적+간결함(제품과 프로젝트의 핵심내용 전달 집중을 위해)

Point1: 우리가 전달하려는 메시지를 분명하게 전달(유기동물에 대한 관심 필요성)

Point2: 우리 제품 이미지 강조(한눈에 제품 디자인이 보이도록)

일반버튼 : 검정색 버튼 추천(가장 가시적으로 잘 보임 + 우리 색 배합에 가장 무난)

https://getbootstrap.com/docs/5.1/components/buttons/

메인페이지 노출 제품 소개(재윤 웹 디자인 기획안 참고)

아래링크 템플릿처럼 만들되, 메인에서는 각 제품의 모습이 "자동으로"넘어가게(carousel)적용

https://startbootstrap.com/template/shop-item(참고!)

1) 여러제품이므로 carousel 이용해서 시간 자동설정2sec 해서 넘어가게 설정

https://getbootstrap.com/docs/5.1/components/carousel/

2) 사진만 보이면 어떤 제품인지 알기 어려우니 card를 carousel 겉에 덧씌워 디자인

+ 제품들 전체를 확인하기 쉽도록 "상품 목록"을 볼 수 있는 창으로 갈 버튼 별도 생성하기

https://getbootstrap.com/docs/5.1/components/card/

3) 메인페이지에서 구매로 가는 버튼(Shopping)

이 버튼을 누르면 "구매 페이지로 이동한다는 내용 제공"

https://getbootstrap.com/docs/5.1/components/popovers/

구매페이지에서 "카드결제"와 "계좌이체" 버튼을 따로 두어, 우리 프로젝트가 실제로 API연동으로 간편결제를 수행하지는 않지만, 고객들이 네이버스토어를 통해 쉽게 간편결제가 가능하도록 유도

+(소비자 편의 증진 방안) 네이버로그인API를 썼기 때문에 고객들이 "네이버로 간편로그인"하면 추가적인 로그인 없이 "네이버스토어"에서 바로 제품 선택후 구매 가능

4) 로그인 버튼(드롭다운): 각각 하지말고, "로그인"하나 뜨고 누르면 2가지 선택가능하게 지정

https://getbootstrap.com/docs/5.1/components/dropdowns/

(일반로그인 및 회원가입/ 네이버로그인)
5) 그래프:
전체적 디자인:
아래 링크의 "차트 요소만 적용"
https://startbootstrap.com/previews/sb-admin-pro
(색깔은 유기견 /유기묘 2가지로 나누어 다른 색을 배합하고 각 동물별 수치(유기동물 수, 피해동물수, 보호받는 수, 모금액 등은 "채도"를 다르게 하여 그래프 구성 -> 색이 아예 달라버리면 고객 입장에선 대체 어떤 관계를 가 진 그래프인지 파악하기 어려움)

목표액과 현 모금액만 색을 조금 다르게 처리
+우리의 현재 모금액(고객 구매에 따라 변화하는 데이터 값)은 ing 상태이므로 "Animated stripes" 형식의 progress이용 + 색은(현재 진행중이라는 상황을 강조하기 위한 디테일)
https://getbootstrap.com/docs/5.1/components/progress/

전체적인 디자인은 코로나 라이브 처럼 웹 페이지의 배경과 구성은 simple하게 (패턴 빼버리기) 안에 세부 내용은 다른 "캠페인 사이트 참고"(디자인적 디테일을 웹 배경에 넣지 말기)

6) 랜덤 comment노출(고객이 홈페이지를 통해 결제할 때 응원의글을 남기면 랜덤으로 main에 출력) 사람들의 참여와 관심이 있음을 부각하기 위한 장치로 간결하게, 내용이 잘 보이도록 배치

7) 결제 완료 후, 감성적인 느낌과 감사의 말을 전하기 위해, 민 제작한 감사의 인사말이 마지막 창에 노출되도록 프론트 디자인 구성