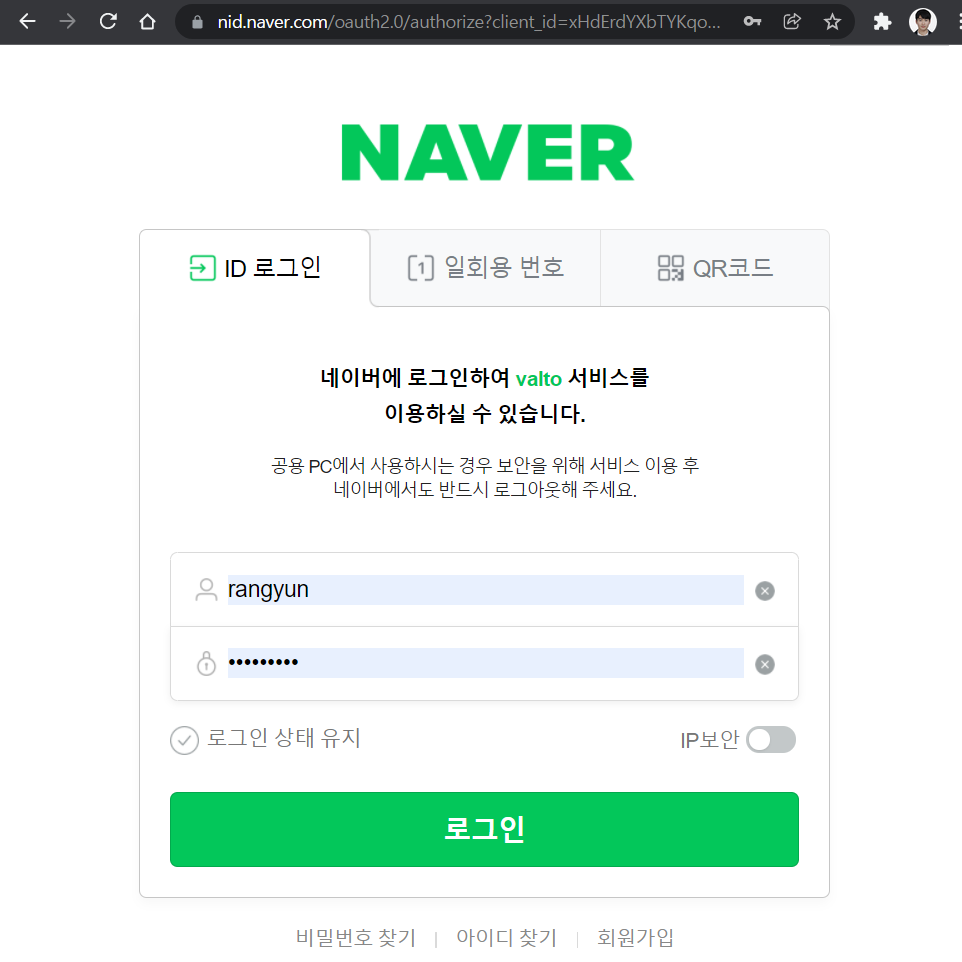
**네이버 간편 로그인:**

대학 커뮤니티(애브리타임), 소셜미디어(인스타그램), 네이버 스마트 스토어를 주된 홍보 및 유입 플랫폼으로 선정하였기에, 고객이 “스마트폰”으로 사이트에 접속할 확률이 높다고 예상하였습니다. 이에 따라 사이트 자체적으로 회원 기본 정보를 입력받고 “회원가입” 후 로그인을 하게 되면 고객이 구매를 위해 감수해야 하는 불편이 늘어난다고 판단하여, “네이버 간편 로그인 API”를 웹 사이트에 적용하였습니다.

네이버 간편 로그인을 통해 고객은 스마트폰에 등록된 네이버 계정을 통해 1초만에 로그인할 수 있으며, 네이버 스마트 스토어도 추가적인 로그인 없이 저희 웹 사이트와 자유롭게 이동하며 개인의 구매내역을 확인할 수 있는 효과를 가졌습니다.

결과적으로 실제 구매한 100%의 고객이 네이버 간편 로그인으로 로그인하여 일반 회원가입보다 선호하는 로그인 유형임을 확인할 수 있었습니다.

**구현된 화면: (**[**http://valto.shop/common/login**](http://valto.shop/common/login)**)**



**실제코드:**

프론트엔드(common/login.html) 로그인 부분 추출

{% load crispy\_forms\_tags %}

<!-- 로그인 창 구현-->

<div class="form-group">

<form method="post" class="post-form" action="{% url 'common:login' %}">

{% csrf\_token %}

<input type="hidden" name="next" value="{{ next }}">

{% include "./formerrors.html" %}

<input type="text" class="form-control" name="username" id="username"

placeholder="ID" value="{{ form.username.value|default\_if\_none:'' }}" style="width: 260px">

<input type="password" class="form-control" name="password" id="password"

placeholder="비밀번호" value="{{ form.password.value|default\_if\_none:'' }}"style="width: 260px">

<button type="submit" class="submit-btn"style="width: 260pxs;font-family:GyeonggiTitleM; ;"><b>로그인</b></button>

</form>

</div>

<div class="links">

<a href="{% url 'common:signup' %}" style="font-family:GyeonggiTitleM; ;"><b>회원가입 / </b></a>

<a href="{% url 'kapchikachi:Notice\_info' %}" style="font-family:GyeonggiTitleM; ;"><b>이용약관</b></a>

</div>

<div class="links">

{% load socialaccount %}

{% if request.user.is\_authenticated %}

<p>{{ request.user.username }}</p>

<br>

<a href="{% url 'common:logout' %}" style="font-family:GyeonggiTitleM; ;">로그아웃</a>

<br>

{% else %}

<a href="{% provider\_login\_url 'naver' %}"><img src="{%static 'img/naver.png'%}" style="width: 25px;height: 25px;margin-left: 2px;"></a>

{% endif %}

</div>

</div>

</div>

<div class="col-sm-3"></div>

</div>

백엔드(Capstone/DB설정) *NAVER API토큰을 발급받아 서버와 통신 key설정*

텍스트, 스크린샷, 모니터, 화면이(가) 표시된 사진

자동 생성된 설명

백엔드(common/views.py) *일반 회원가입과 소셜로그인 POST요청 처리 알고리즘*

def signup(request):

if request.method=='POST':

form1 = UserForm(request.POST)

form2 = UserinfoForm(request.POST)

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')

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})

**구매 프로세스:**

사용자 편의를 위해 상단 고정 바를 추가하여, 모든 창에서 Shopping 버튼을 눌러 접근이 용이하게 했습니다.

**구현된 화면:**



**실제 코드:**

프론트엔드(kapchikachi/index.html)

<span class="site-heading"style="text-align :center;">

<div class="row"><a href="{% url 'kapchikachi:main' %}"><img src="{%static 'img/logo2-remove.png'%}" style="max-width: 200px;max-height: 200px;margin-left: auto;margin-right: auto;"></a></div>

<div class="row"style="background-color:blanchedalmond;padding-bottom:10px;padding-top: 10px;" >

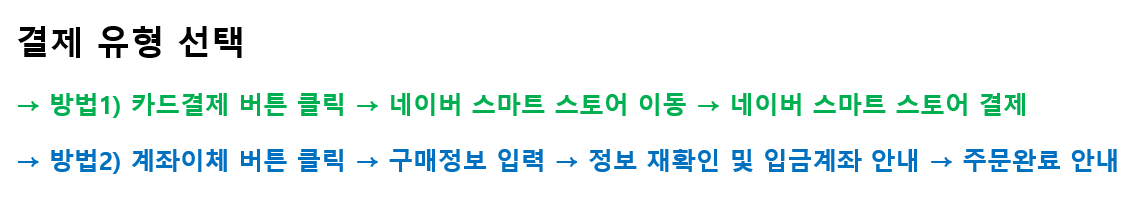
<div class="col-4"><a href="{% url 'kapchikachi:main' %}" style="font-family:GyeonggiTitleM;font-size:2.5vh;padding-top: 5px;text-decoration:none;"><b>Home</b></a></div>

<div class="col-4"><a href="{% url 'kapchikachi:Products' %}" style="font-family:GyeonggiTitleM;font-size:2.5vh;"><b>Products</b></a></div>

<div class="col-4"><a href="{% url 'kapchikachi:SelectPayOption' %}" id ="Shopping"style="font-family:GyeonggiTitleM;font-size:2.5vh;padding-top: 5px;text-decoration:none;"><b>Shopping</b></a></div>

</div>

</span>



**구현된 화면:**



**실제 코드:**

프론트엔드(kapchikachi/selectpayoption.html)

<div class="row" style="margin-top: 20px;">

<div class="col-6" style="text-align: center;font-family:GyeonggiTitleM ; " ><button class="btn btn-secondary" onclick=location.href="{% url 'kapchikachi:CommentCreate' %}"><b>계좌 이체</b></button>

<p style="text-align: center;font-family:GyeonggiTitleM; margin-top: 20px;font-size: 9pt; color: #3B170B;"><b>직접수령 선택 가능</b></p></div>

<div class="col-6" style="text-align: center;font-family:GyeonggiTitleM ;" ><button class="btn btn-secondary" onclick=location.href="{% url 'kapchikachi:PayByCard' %}"><b>카드 결제</b></button>

<p style="text-align: center;font-family:GyeonggiTitleM; margin-top: 20px;font-size: 9pt; color: #3B170B;"><b>네이버 스토어로 이동</b></p></div>

</div>

백엔드(kapchikachi/views.py) *계좌이체 방식의* ***‘입금완료’*** *버튼 누르면 실행되는 코드*

#입금완료 버튼 누를 경우 DB처리 로직

@login\_required(login\_url='common:login')

def Deposit(request):

# 사용자 DB설절

this\_user = request.user.id

user = User.objects.get(id=this\_user)

##DB추출 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 1000으로 하여 자동연산하도록 지정 --> 서버 실 운영 이후 DB수정 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 DB에서 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 DB는 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)

**\* 백엔드 코드는 양이 너무 많아 보고서에 모두 담지 못했습니다. 첨부 파일 참고 부탁드립니다.**

**고객 프로젝트 동참 유도를 위한 시각화된 데이터 노출**

**1) 유기동물 실태의 심각성 전달을 위한 차트 활용 + 현재 판매액 표기**

대중에게 유기동물의 실태에 대한 심각성을 글만으로 알리기보다는 구체적인 수치와 함께 동적 기능이 담긴 차트로 표현했습니다.

이에 더하여 저희 프로젝트의 목표 금액과 고객이 구매한 총 금액을 실시간으로 연산하여 그 수치를 메인 페이지에 막대그래프로 표현했습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명**구현된 화면:**

**실제코드:**

프론트엔드(kapchikachi/index.html) *‘목표액’ 을 노출하는 코드만 추출*

const chart3=myChart3 =new Chart(ctx3,{

type: 'horizontalBar',

data: {

labels: [''],

datasets: [{

label: '현재 금액',

data: [num],

borderColor: "rgba(139, 69, 19,1)",

backgroundColor: "rgba(139, 69, 19,0.5)",

fill: false,

}]

},

options: {

legend: {

display: false

},

title: {

display: true,

text: '목표액 대비 현재 총 금액'

},

tooltips: {

mode: 'index',

intersect: false,

},

responsive: true,

scales: {

display: false,

drawBorder: false,

xAxes: [{

gridLines:{

display:false,

drawBorder:false,

},

scaleLabel: {

display: true,

labelString: '액수 단위(만원)',

beginAtZero: true,

},

ticks: {

autoSkip: false,

min: 0,

max: 50,

},

}],

yAxes: [{

barPercentage:0.5,

display: true,

gridLines:{

display:false,

drawBorder:false

},

scaleLabel: {

labelString: '',

display: true,

}

}]

}

}

});

</script>

**실제 코드:**

백엔드(kapchikachi/views.py) *총 구매 금액을 연산하여 반환하는 코드만 추출*

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))

total\_selled=0

#총 판매액 연산

for a in range(db\_num):

total\_selled += int(all\_data[a].sell)

result={

'total\_selled':total\_selled, #총판매액

}

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

**2) 타인의 관심도를 노출하기 위한 코멘트 노출**

고객이 입력한 코멘트 중 랜덤으로 3개의 데이터를 추출하여 메인 페이지에 노출

프로젝트에 대한 사람들의 관심도를 시각적으로 보여주어 나도 참여해볼까? 하는 군중심리를 위한 기능

**구현된 화면:**

텍스트이(가) 표시된 사진

자동 생성된 설명

**실제 코드:**

백엔드(kapchikachi/views.py) *랜덤 코멘트 노출하는 코드만 추출*

#응원의 글(랜덤 출력) 3가지만 랜덤으로 출력 (앞의 코드는 생략)

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={

'ran\_comment1':ran\_comment[0],

'ran\_comment2':ran\_comment[1],

'ran\_comment3':ran\_comment[2],

}

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

**구매 내역 확인 창**

일반적인 form으로 주문을 받는 프로젝트는 구매한 고객들이 자신의 주문이 제대로 입력된 것인지 바로 확인이 되지 않아 주문의 접수여부와 배송 상태를 확인할 수 없어서 불편하다는 의견이 집계되었습니다. 이에 따라 저희 팀은 최대한의 고객 편의 제공을 위해, “주문상태” 와 “배송상태”를 실시간으로 주문 내역에서 확인할 수 있도록 사이트를 설계했습니다. 이때 주문 내역은 최근의 주문 3건이 노출되며 구매 정보(구매일시, 상품, 구매인, 연락처, 배송주소)와 주문상태, 배송상태를 조회할 수 있게 구축했습니다.

**텍스트이(가) 표시된 사진

자동 생성된 설명구현된 화면**: (아래의 주문은 예시 주문입니다. 화면이 길어 좌우로 나눴으니 참고 부탁드립니다.)

**실제 코드:**

백엔드(kapchikachi/views.py) *개인 구매이력 반환 코드 중 일부만 추출*

#####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/') #urls에서 notfound\_buy url과 함수 지정 '구매이력이 없습니다. 캠페인에 동참해주세요.' 문구 출력하는 페이지로 이동

elif personal\_buy\_num >3: #구매이력이 3번보다 많은 경우

personal\_buy\_num=3

else: #1~3번 구매한 경우

pass

#####구매정보#####(Comment Model DB)

not\_found=[]

personal\_receiver\_name=[]

personal\_receiver\_phone=[]

personal\_product=[]

personal\_shipping=[]

personal\_address=[] #shipping이 1인 경우 입력/ else인 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):

#####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 에어팟1,2세대 케이스 {tmp.dog\_case1}개\n'

if tmp.dog\_case2>=1:

tmp\_product += f'강아지 에어팟 pro 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 에어팟1,2세대 케이스 {tmp.cat\_case1}개\n'

if tmp.cat\_case2>=1:

tmp\_product += f'고양이 에어팟 pro 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')

**프로모션 광고 팝업 노출**

**실제화면:**

**실제코드:**

프론트앤드(kapchikachi/index.html)

<div id="myModal" class="modal" style=" display: flex; justify-content: center;">

<div class="modal-content" style="text-align: center;width: 400px;height: 400px;margin-left: auto; margin-right: auto;" >

<img src ="{%static 'img/specialsale.jpg'%}" class="img-fluid">

<button class="btn btn-secondary" style="font-size: 13pt;font-family:GyeonggiTitleM ;" onClick="close\_pop();" >

<b>닫기</b>

</button>

<script type="text/javascript">

let isElement = document.querySelector('#myModal');

isElement.style.opacity = 0.1;

function Effect\_With\_JS(){

let isCount = 0;

isElement.style.opacity = 0.1;

let period = setInterval(function(){

isElement.style.opacity = isCount;

isCount = isCount + 0.1;

if(isCount >1.0){

clearInterval(period);

}

},100);

}

Effect\_With\_JS();

//팝업 Close 기능

function close\_pop(flag) {

$('#myModal').fadeOut();

};

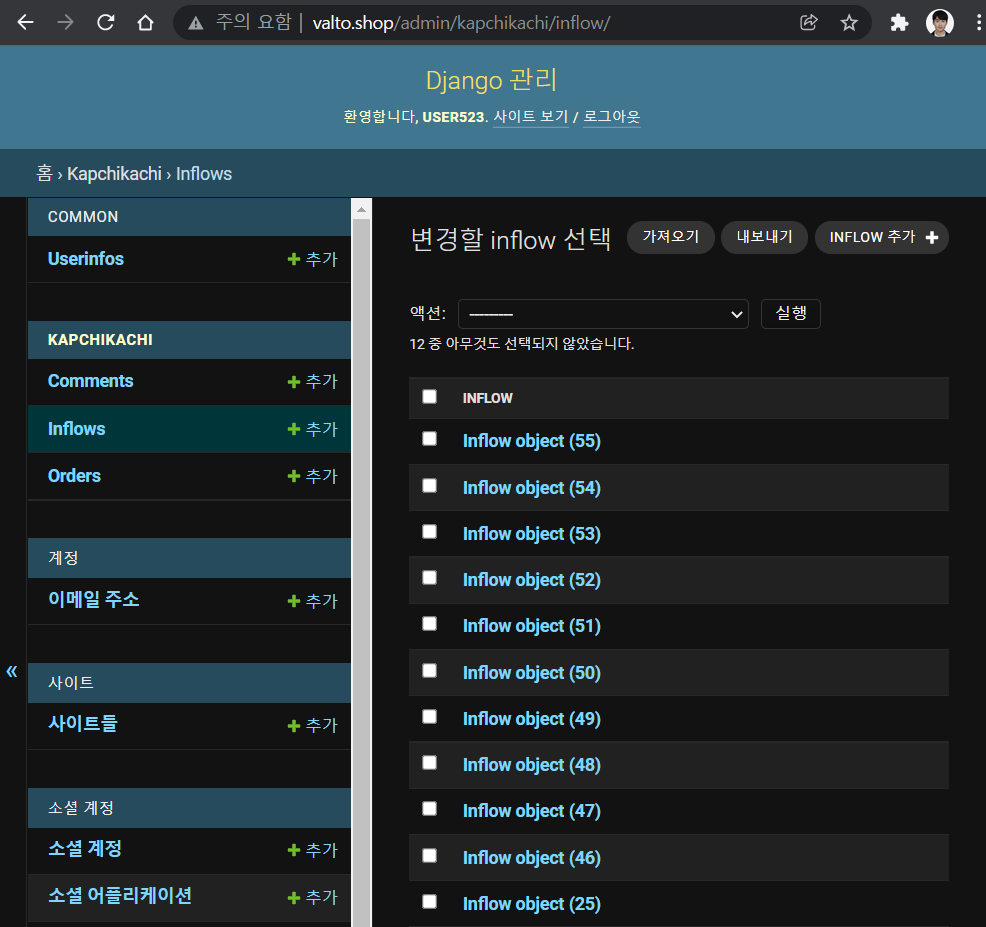
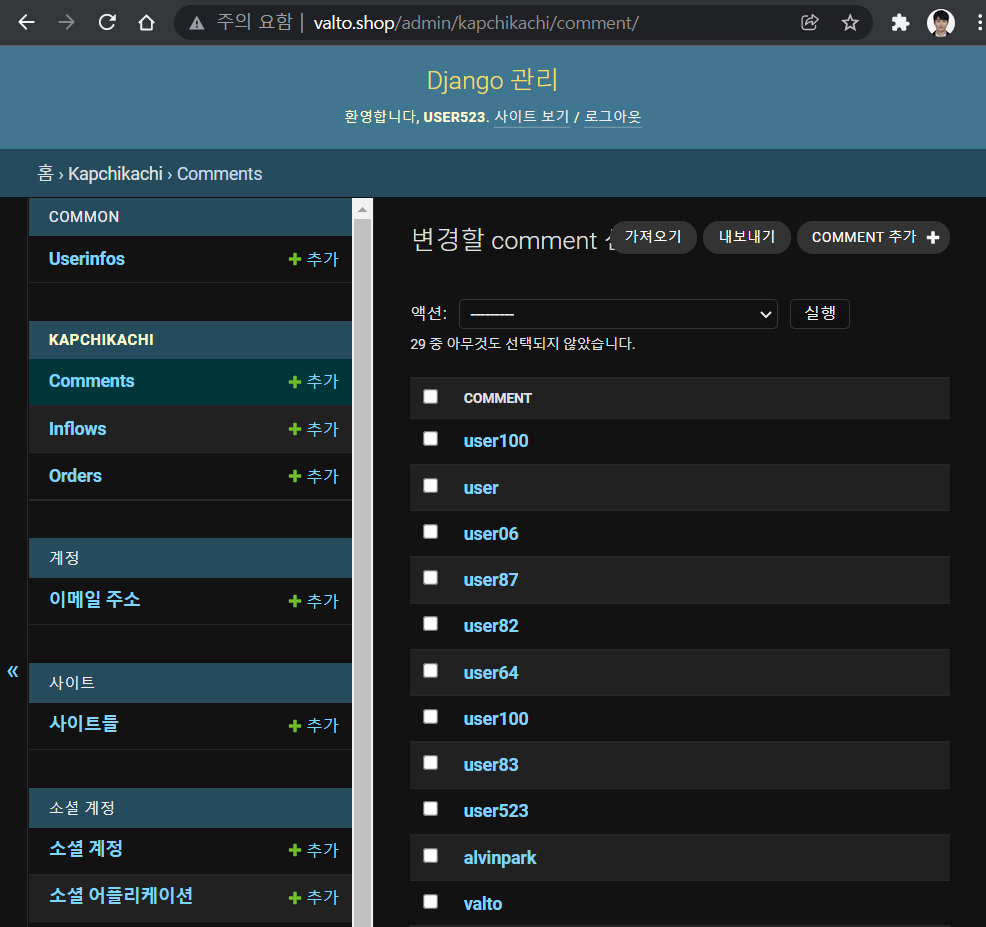
</script>

</div>

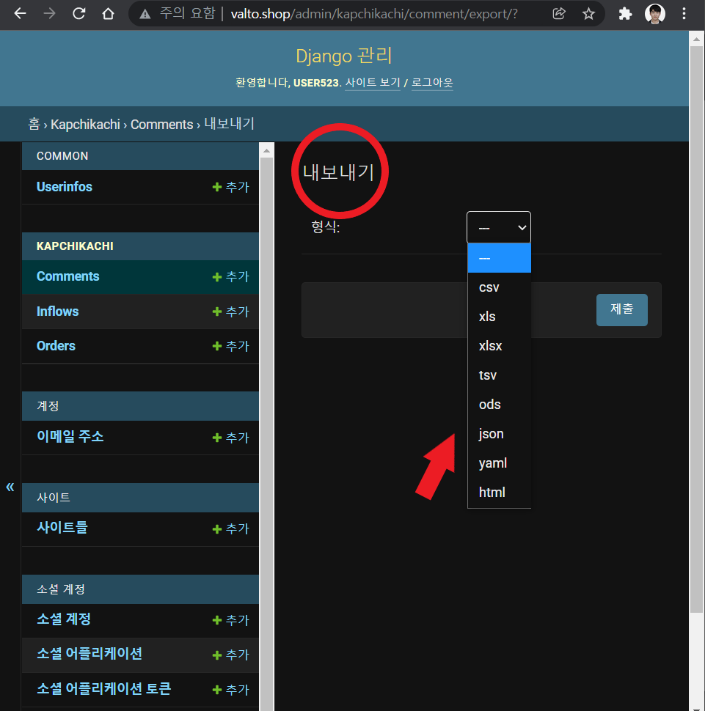
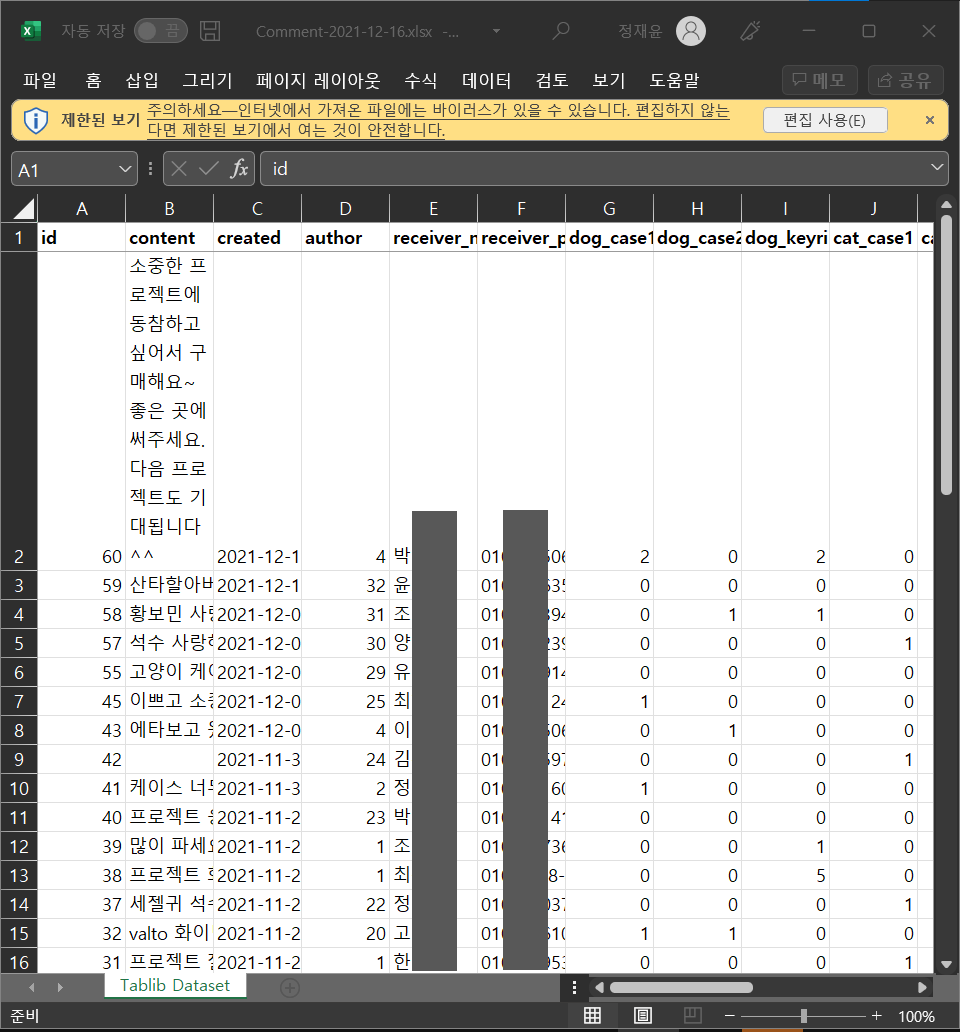
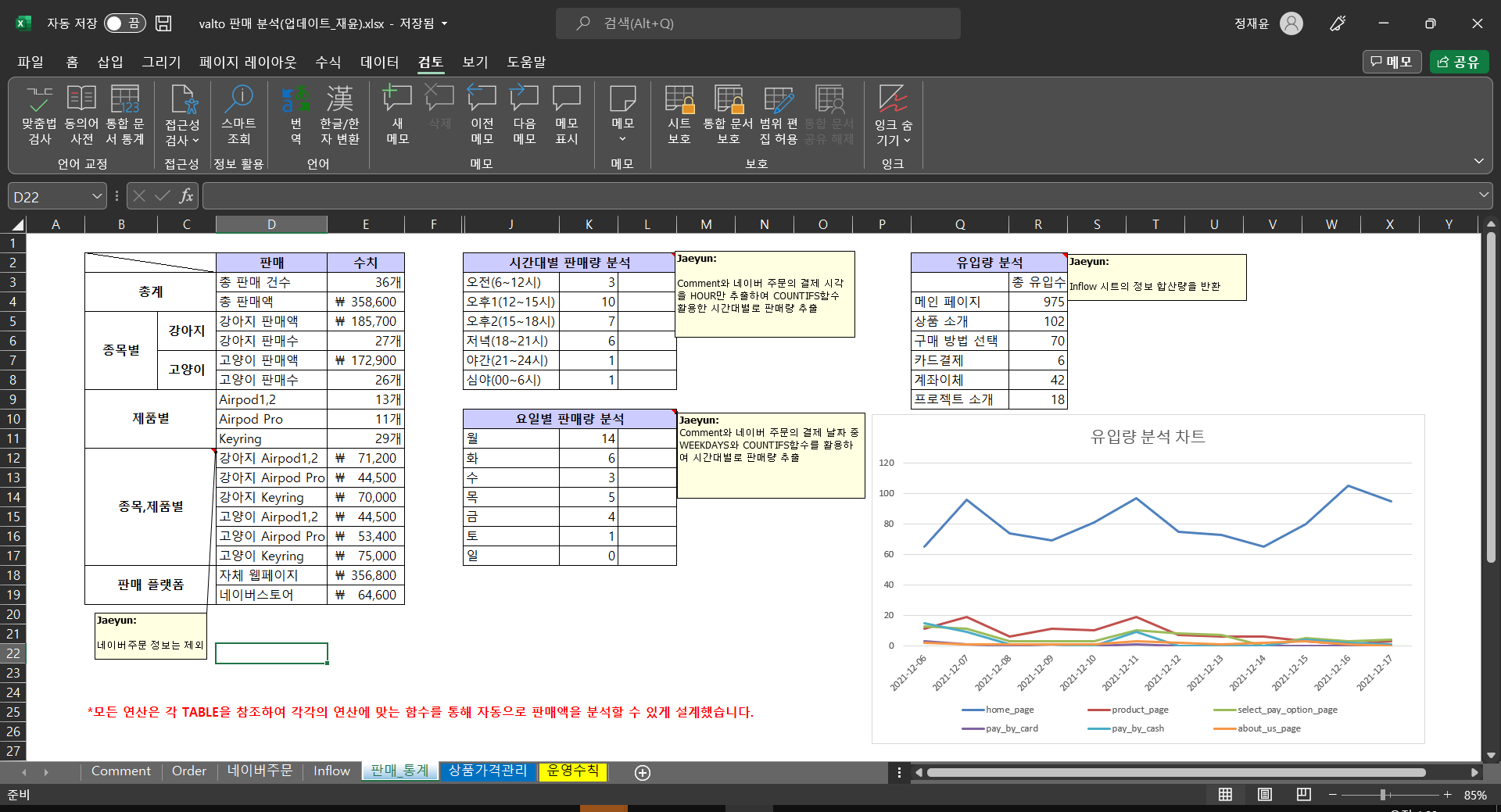
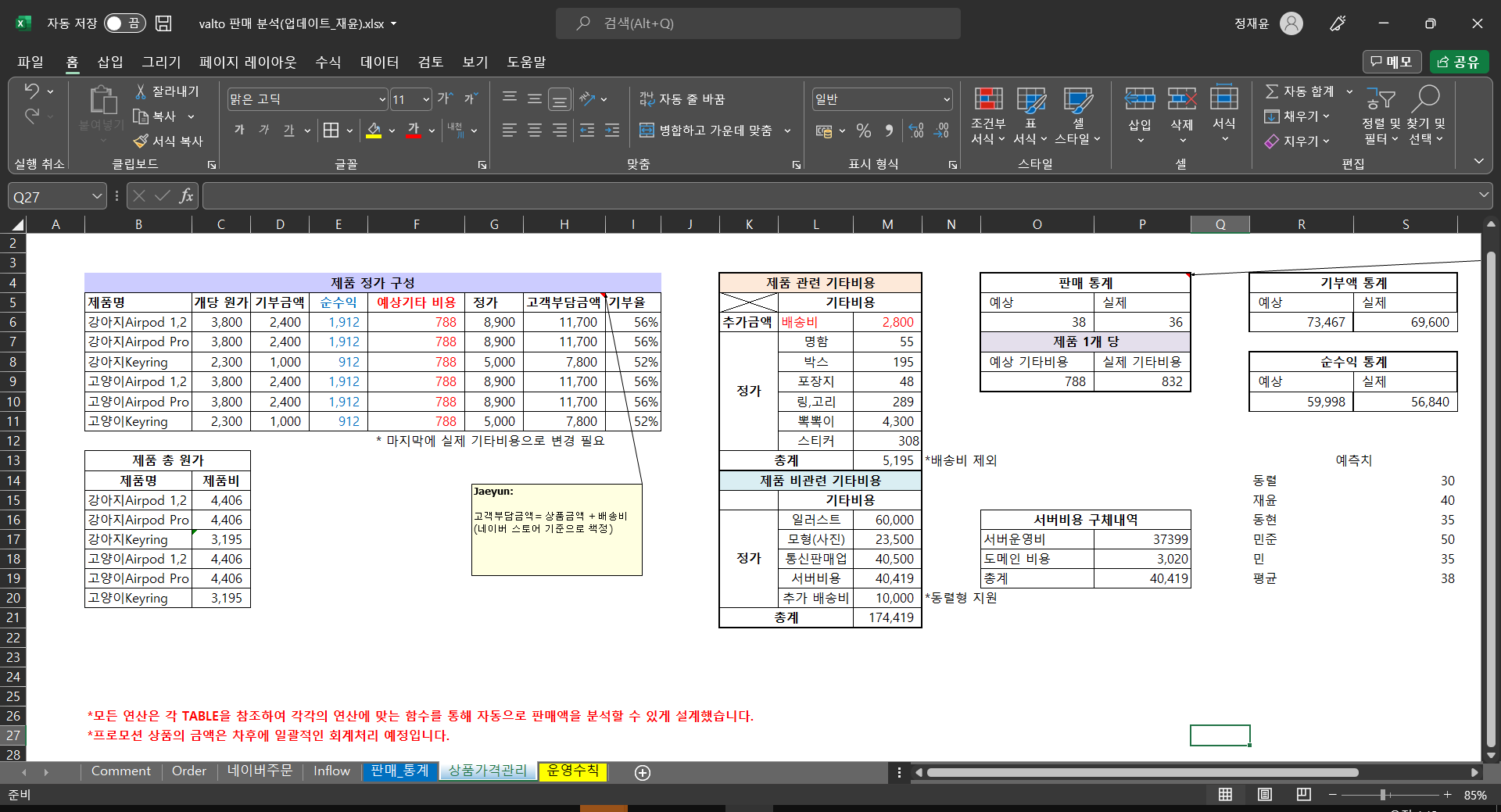
</div>

**구매정보, 유입량 DB를 EXCEL로 반환하고, 데이터 분석 자동화 처리**

구매정보, 유입량을 관리자 페이지(<http://valto.shop/admin>)에서 확인하고 DB를 수정할 수 있습니다. 그렇지만 하기 사진들과 같이 이름과 객체 명만으로는 정확히 어떤 주문, 어느 날에 유입량이 얼마나 되는 지 명확하게 식별할 수 없었고 Mysql을 사용하는 “코드로 DB조작”이 아닌, 웹 페이지에서 마케팅 성과 측정 및 판매 관리를 하기에 어려움이 있을 것이라 판단했습니다.



따라서 웹 사이트 런칭 전부터 위와 같은 불편함을 최소화하고자, 모든 DB를 EXCEL로 다운로드 받을 수 있게 코드를 설계했습니다. 나아가 다운받은 EXCEL을 연산 함수로 모든 판매 이력, 구매 통계를 하나의 표로 확인할 수 있도록 함수를 설정해, 자동적으로 분석을 할 수 있게 만들었습니다.



**마케팅 분석용: 페이지 별 유입량 측정**

* 메인 페이지(홈페이지) / 프로젝트 소개 / 상품 소개 / 구매 선택 창 / 카드결제 / 계좌이체

각 파트별로 유입되는 트래픽(유입 수)를 DB에 매일 따로 객체를 만들어 기록하고 EXCEL로 일자 별 페이지마다의 유입량을 비교 분석할 수 있게 사이트의 알고리즘을 수정했습니다.

**관리자 페이지(valto.shop/admin)에 노출되는 화면**

\* 21.12.11의 유입량 통계

**텍스트, 모니터, 화면, 실내이(가) 표시된 사진

자동 생성된 설명**

**실제 코드:**

1) DB정의(kapchikachi/models.py)

#마케팅 분석용

class Inflow(models.Model):

created = models.DateField(auto\_now\_add=True) # 작성시간

home\_page=models.IntegerField(blank=True) # 1. 홈페이지 유입수

product\_page=models.IntegerField(blank=True) # 2. 제품창 유입수

select\_pay\_option\_page=models.IntegerField(blank=True) # 3. 주문결제유형 창 유입수

pay\_by\_card=models.IntegerField(blank=True) # 4. 카드결제 창 유입수

pay\_by\_cash=models.IntegerField(blank=True) # 5. 계좌이체 창 유입수

about\_us\_page=models.IntegerField(blank=True) # 6. 캠페인 취지 창 유입수

2) 알고리즘(kapchikachi/views.py)

def Products(request):

# 마케팅 분석용(제품목록 창 유입량)

tmp\_today = datetime.today().strftime("%Y-%m-%d") # 현재 날짜 가져오기

total\_inflow = Inflow.objects.order\_by('-created') # 유입량 데이터 불러오기

check\_inflow = 0 # 금일 DB가 존재하는 지 확인

check\_inflow\_index = '' # 오늘 날짜의 DB 위치 확인

for i in range(len(total\_inflow)):

if str(total\_inflow[i].created) == tmp\_today: # 오늘 날짜의 조회수 DB가 존재하는 경우

check\_inflow += 1

check\_inflow\_index = i

else:

check\_inflow += 0

if check\_inflow > 0: # 금일 DB가 존재하는 경우

total\_inflow[check\_inflow\_index].product\_page += 1

total\_inflow[check\_inflow\_index].save()

else: # 금일 DB가 없는 경우

Inflow.objects.create(home\_page=0, product\_page=1, select\_pay\_option\_page=0, pay\_by\_card=0, pay\_by\_cash=0,

about\_us\_page=0)

return render(request, 'kapchikachi/products.html')

**향후 추가 기능: 웹 사이트 내 간편결제**

지금은 웹 사이트의 판매처 다변화를 위해 “네이버 스마트 스토어”를 활용하며, 네이버 간편로그인 기능을 사용해, 고객이 “네이버 계정” 로그인 한 번만으로 자체 제작 웹 사이트에서 상품을 구매, 구매내역 확인, 네이버 스마트 스토어 이동 후 카드 결제가 모두 가능합니다. 그렇지만, 향후 웹 페이지 내에서도 네이버 스마트 스토어를 통하지 않고도 카드 결제가 가능하도록 import(간편결제 대행사) 서버와 통신하여 카카오페이 및 토스를 활용해 간편결제가 가능하도록 코드 일부를 구현해 두었습니다.

**실제코드:**

백엔드(kapchikachi/models.py) 결제 DB 중 일부

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]

백엔드(kapchikachi/views.py) 결제 알고리즘 중 일부

@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')

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)