Modelos y Apps con las que trabaja el Proyecto

sale_price = self.cleaned_data['sale_price']
purchase_price = self.cleaned_data.get('purchase_price')

if not sale_price >= purchase_price:

return sale_price



Registrar Producto - Modulo Almacén

```
class ProductUpdateView(AlmacenPermisoMixin, UpdateView):
    template_name = "producto/form_producto.html"
    model = Product
    form_class = ProductForm
    success_url = reverse_lazy('producto_app:producto-lista')
```

Se maneja igual que el create ya que trabaja con el mismo formulario, sin embargo el HTML se cambia dependiendo si es para crear o actualizar ya que muestra un boton de eliminar en el actualizar

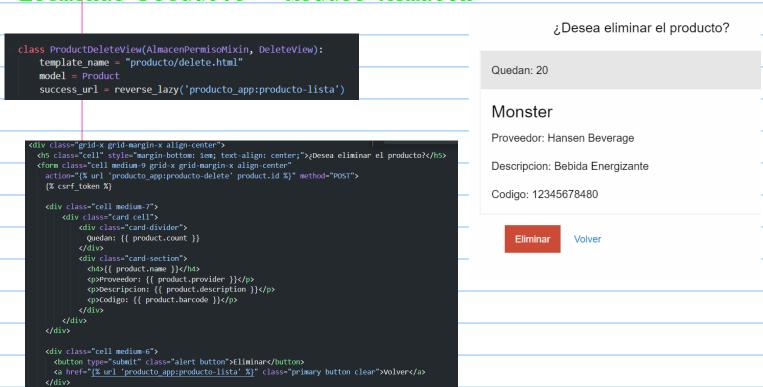
Guardar

</form>
</div>



Recordando que cuando se trabaja con update pues se puede acceder al objeto, entonces he usado éso para saber si hay un objeto y en caso de haberlo mostrar el botón

Eliminar Producto - Módulo Almacén



Detalle del Producto - Módulo Almacén

Precio Compra: 2,00
Precio Venta: 3,50
Numero de Ventas: 2
Descripcion: Bebida Energizante

Ventas en los ultimos 30 días

Fecha Producto Ventas

11 de octubre de 2022 Monster 2

La lista de las ventas de los ultimos 30 dias se la hace por medio de el get_context_data

- si que remos restar una fecha en días usamos el timedelta
- si que remos el producto creado en un rango de fechas usamos el created_range
- si que remos que por ejemplo que en un dia se ha vendido mas de 1 vez un producto pues que se acumule el numero de veces mas no que me liste una tupla nueva usamos el values

- si que remos hacer una operacion aritmetica adicional usamos el annotate

Fecha	Producto	Ventas
11 de octubre de 2022	Monster	2

Fina mente en el HTML se itera una vez mandado como contexto el resultado de mi manager

Sin embargo al final tengo un botón imprimir el cual me genera un PDF

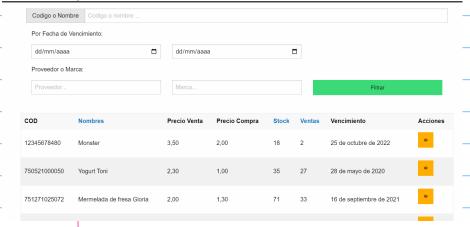
https://www.youtube.com/watch?v=N9iQm4N3H8s clase extra en youtube

PENDIENTE

Generar PDF con Django

Reportes en Módulo Almacén

Para hacer el reporte en este caso de igual manera se trabaja con un ListView pero se mandan varios parametros



```
def filtrar(self, **filters):
    if not filters['date_start']:
        filters['date_start'] = '2020-01-01'

    if not filters['date_end']:
        filters['date_end'] = timezone.now().date() + timedelta(1080)

#
    consulta = self.filter(
        due_date__range=(filters['date_start'], filters['date_end'])
).filter(
        Q(name__icontains=filters['kword']) | Q(barcode=filters['kword']) -
).filter(
        marca__name__icontains=filters['marca'],
        provider__name__icontains=filters['provider'],
)

if filters['order'] == 'name':
        return consulta.order_by('name')
elif filters['order'] == 'stok':
        return consulta.order_by('count')
elif filters['order'] == 'num':
        return consulta.order_by('-num_sale')
else:
        return consulta.order_by('-created')
```

```
{% if request.path == request.get_full_path %}
    <a href="{{request.get_full_path}}?order=name"> Nombres</a>
    {% else %}
    <a href="{{request.get_full_path}}&order=name"> Nombres</a>
    {% endif %}

        >/th>
        Precio Venta
        + Compression of the compression of
```

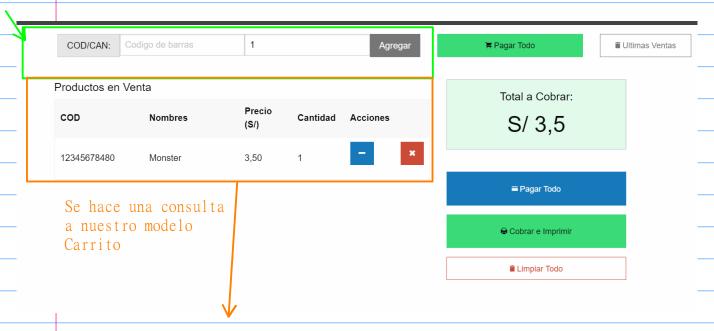
En este caso usa el objeto request dentro del HTML para recuperar toda la URL y le anida el parametro

127.0.0.1:8000/producto/reporte//?kword=o&date_start=&date_end=&provide=&marca=&order=stok

Análisis Pantalla Proceso Venta- Modulo Venta

Al momento de hacer lo que sería el carrito de compras

En éste caso, este codigo de barras buscará un producto y se añadirá la cantidad, a la final se resumen en un createView (POST) ya que agrega el producto a mi modelo Carshop a fin de que mi carrito se recuerde siempre para cada Usuario..



Acá lo que hago es listar cada producto guardado dentro de mi modelo carrito a fin de que se recuerde el carrito del usuario

Agregar Producto a Carrito de compra - Modulo Venta

Según la lógica que estuvimos analizando, lo que se tiene que hacer es tomar ese código de barras y enviarlo hacia el servidor a través del método Post. Para que? Para que mediante este código recuperemos un producto, en este caso recuperado este producto, recuperemos ese producto y luego lo registremos dentro de nuestro modelo Carrito de compras o Car Shop.

¿Cómo se Resuelve?

Tenemos 3 formas

- CreateView
- FormView => trabaja con un formulario, toma los datos de ese formulario y hace el registro
- Usar la Vista genérica View

No se puede usar el CreateView debido que para el create debemos estar vinculados a un modelo, tampoco el View debido a que necesitamos un formulario donde mediante el codigo de barras van a ingresar el productu y su cantidad.. En conclusión solo puedo usar el FormView

Recordando que el formview trabaja siempre con el formvalid, dentro del form_valid iría el código de barras y con éso recuperamos el producto y finalmente agregarlo al carrito

Siempre necesitamos sobreescribir el formvalid al trabajar con el formview a fin de que se valide el formulario

```
VentaForm(forms.Form):
 barcode = forms.CharField(
                                                          a los widgets para el diseño..
    widget=forms.TextInput(
          'placeholder': 'Codigo de barras',
          'class': 'input-group-field',
                                 se valida que la cantidad sea mayor a
 count = forms.IntegerField(
    widget=forms.NumberInput(
          'class': 'input-group-field',
 def clean count(self):
    count = self.cleaned_data['count']
    if count < 1:</pre>
       raise forms.ValidationError('Ingrese una cantidad mayor a cero')
    return count
Ahora pasando a mi view y a mi form_valid ¿qué sucede si una vez ya está agregado un
producto al carrito y mando de nuevo el mismo codigo?
Pues debería simplemente subir la cantidad, más no añadirse otra tupla, entonces
para ello uso el get or create.. OjO que no estoy usando el update or create debido a que
 si por ejemplo en cantidad pongo que se añada 5 pues esto debe sumarse mas no actualizarse
 En el caso del get_or_create si existe lo recupera y si no existe lo crea
     form_valid(self, form):
     barcode = form.cleaned_data['barcode']
     count = form.cleaned_data['count']
     obj, created = CarShop.objects.get_or_create(
        barcode=barcode,
         defaults={
            'product': Product.objects.get(barcode=barcode),
                                                        Recordar que para este caso el succerss url
            'count': count
                                                        esta puesto '' lo que significa que me redireccio
                                                        na a la misma página
     if not created:
        obj.count = obj.count + count
        obj.save()
     return super(AddCarView, self).form valid(form)
                                                              Notar que con el get_context_data logro
  def get_context_data(self, **kwargs):
      context = super().get_context_data(**kwargs)
                                                              listar los productos del carrito
      context["productos"] = CarShop.objects.all()
      context["total_cobrar"] = CarShop.objects.total_cobrar()
      context['form voucher'] = VentaVoucherForm
      return context
```

context["productos"] = CarShop.objects.all()

{% for producto in productos %}

Calcular Monto Total de la Venta - Módulo Venta





Para calcular el monto total debemos de igual manera debemos mandar un contexto con el valor procesado, se suman/multiplica los valores y generamos el total

```
context["total_cobrar"] = CarShop.objects.total_cobrar()
```

Recordar que si queremos calcular un valor donde vamos a hacer operaciones aritméticas. ¿qué deberíamos hacer? pues un agregate

Eli<mark>minar productos del carrito de venta - Modulo Venta</mark>



¿Cómo hacemos para que se disminuya la cantidad cada que damos click en el botón azul y se actualize el total? Pues partamos de que es un updateview ya que estoy actualizando una cantidad de un registro que ya existe

Usamos el View en lugar
del updateView ya que
el updateV necesita
formularios para poder
trabajar..
El View no, lo que hace
es interceptar un método
POST y dentro de ésto podemos
indicar cualquier proceso

```
'carshop/update/<pk>/',
            views.CarShopUpdateView.as view(),
            name='carshop-update',
      <form action="{% url 'venta_app:carshop-update' producto.id %}" method="POST">{% csrf_token %}
        <button type="submit" class="button primary"><i class="fi-minus"></i></button>
      </form>
          Y de esta forma haría el update, pero cómo hago con el delete?
      form action="{% url 'venta_app:carshop-delete' producto.id %}" method="POST">{% csrf_token %}
                                                                                           Pues misma historia
       <button type="submit" class="button alert"><i class="fi-x"></i></button>
             class CarShopDeleteView(VentasPermisoMixin, DeleteView):
                                                                           carshop/delete/<pk>/',
                 model = CarShop
                                                                           views.CarShopDeleteView.as view(),
                 success_url = reverse_lazy('venta_app:venta-index')
                                                                           name='carshop-delete',
  Limpiar carrito de Venta - Modulo Venta
COD/CAN: Codigo

    ■ Pagar Todo
    ■ Ultimas Ventas
                                                                          COD/CAN: Codigo de barras
Productos en Venta
                                                                          Productos en Venta
                                     Total a Cobrar
                                                                                                                   Total a Cobrar
COD
                                                                          COD Nombres
                                                                                          Precio (S/) Cantidad Acciones
                                      S/ 8.1
                                                                                                                    S/ 0
                                                                                                                    ■ Limpiar Todo
              ¿Cómo hago para que al dar click en Limpiar Todo me limpie absolutamente todo?
              Pues no debo usar el deleteView debido a que debo mandar un pk por ende voy
              a usar el View
                class CarShopDeleteAll(VentasPermisoMixin, View):
                                                                         recordar que se intercepta el método
                    def post(self, request, *args, **kwargs):
                                                                         POST
                       CarShop.objects.all().delete()
                        return HttpResponseRedirect(
                           reverse(
                               'venta app:venta-index'
                         <form class="cell" action="{% url 'venta_app:carshop-delete_all' %}"</pre>
                           method="POST">{% csrf token %}
                           <button type="submit" class="cell hollow button alert">
                             <i class="fi-trash"></i></i></or>
                             <span>Limpiar Todo</span>
                           </button>
                         </form>
```

Vista para Venta Simple

Pagar Todo

Este botón lo que hace es tomar todo lo que está en mi carrito y registrarlo como una venta sin embargo nunca nos pide un comprobante, qué tipo de pago se hará, etc.

Esto es debido a que cuando no se alcanza un monto mínimo en una compra pues no es necesario emitir un comprobante de pago, no imprime ni na...

Esto es muy usual por lo que se recomienda siempre implementarlo

Entonces si estamos pensando en el proceso de registrar pues qué tipo de vista debemos de registrar?

-No tenemos un formulario esctrictamente basado en un modelo (VENTA) Descartado el formview y el createview...

OjO que no se debe confundir, los inputs del codigo son para el carrito

Usaremos el VIEW

- DECLARAR LA VISTA VIEW
- TOMAR PRODUCTOS DEL CARRITO Y AÑADIRLOS A VENTA
- ELIMINAMOS TODOS LOS ELEMENTOS DEL CARRITO DE COMPRAS INDICANDO QUE YA SE VENDIÓ TODO

Para este caso lo que hemos realizado y se recomi enda mucho es que si una vista tiene mucho codigo pues es mejor hacer uso de los functions luego simplemente lo llamo

```
lef procesar_venta(self, **params_venta):
   productos en car = CarShop.objects.all()
   if productos_en_car.count() > 0:
      venta = Sale.objects.create(
          type_payment=params_venta['type_payment'],
user=params_venta['user'],
      ventas_detalle = []
       for producto_car in productos_en_car:
           venta_detalle = SaleDetail
              product=producto_car.product,
              sale=venta,
              count=producto car.count.
              price_purchase=producto_car.product.purchase_price,
              price_sale=producto_car.product.sale_price,
          producto = producto car.product
          producto.count = producto.count - producto_car.count
          producto.num sale = producto.num sale + producto car.count
          ventas_detalle.append(venta_detalle)
          productos en venta.append(producto)
          venta.count = venta.count + producto car.count
          venta.amount = venta.amount + producto_car.count*producto_car.product.sale_price
      venta.save()
      SaleDetail.objects.bulk create(ventas detalle)
      Product.objects.bulk_update(productos_en_venta, ['count', 'num_sale'])
      productos en car.delete()
      return venta
```

Función para el Proceso Venta - Modulo Venta

```
procesar_venta(self, **params_venta):
productos en car = CarShop.objects.all()
if productos en car.count() > 0:
    venta = Sale.objects.create(
        date sale=timezone.now(),
        count=0,
        amount=0.
        type_invoce=params_venta['type_invoce'],
        type_payment=params_venta['type_payment'],
        user=params_venta['user'],
    ventas detalle = []
    productos_en_venta = []
    for producto car in productos en car:
        venta_detalle = SaleDetail(
            product=producto_car.product,
             sale=venta,
             count=producto car.count,
            price purchase=producto car.product.purchase price,
            price_sale=producto_car.product.sale_price,
             tax=0.18,
        producto = producto car.product
        producto.count = producto.count - producto car.count
        producto.num sale = producto.num sale + producto car.count
        ventas_detalle.append(venta_detalle)
        productos_en_venta.append(producto)
        venta.count = venta.count + producto_car.count
        venta.amount = venta.amount + producto car.count*producto car.product.sale price
    venta.save()
    SaleDetail.objects.bulk_create(ventas_detalle)
    Product.objects.bulk update(productos en venta, ['count', 'num sale'])
    productos en car.delete()
    return venta
    return None
```







