#### 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

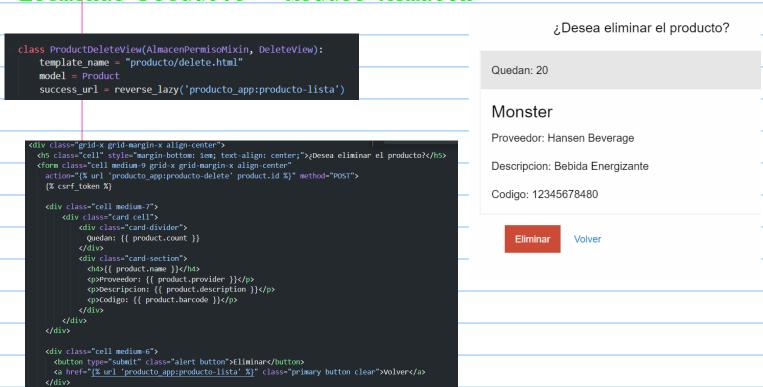
Guardar

</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

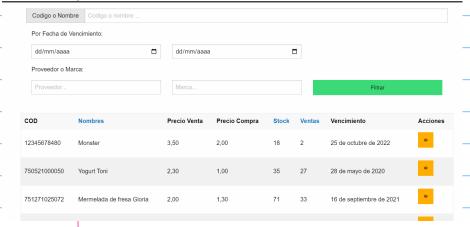
https://www.youtube.com/watch?v=N9iQm4N3H8s clase extra en youtube

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
```

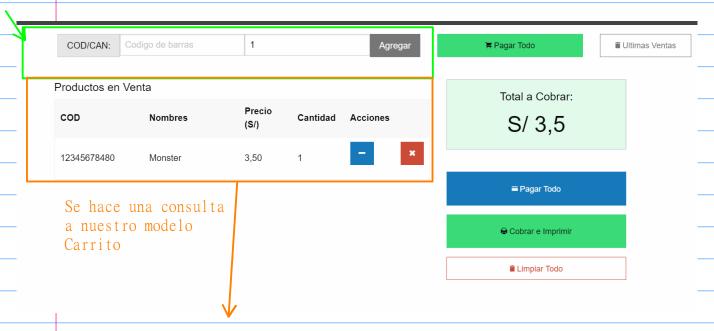
En este caso usa el objeto request dentro del HTML para recuperar toda la URL y le anida el parametro

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 %}





