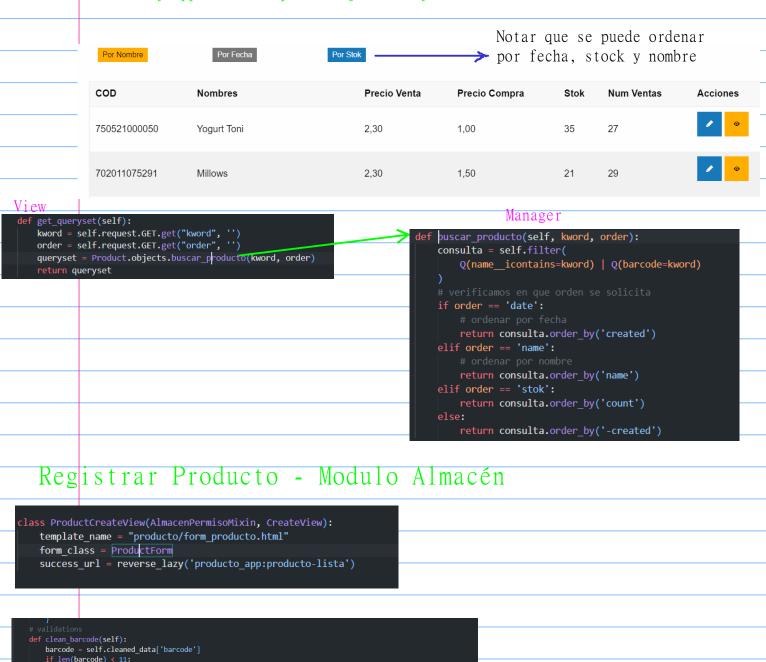
Modelos y Apps con las que trabaja el Proyecto



```
# validations

def clean_barcode(self):
    barcode = self.cleaned_data['barcode']
    if len(barcode) < 11:
        raise forms.ValidationError('Ingrese un codigo de barras correcto')

return barcode

def clean_purchase_price(self):
    purchase_price = self.cleaned_data['purchase_price']
    if not purchase_price > 0:
        raise forms.ValidationError('Ingrese un precio compra mayor a cero')

return purchase_price

def clean_sale_price(self):
    sale_price = self.cleaned_data['sale_price']
    purchase_price = self.cleaned_data['sale_price')
    if not sale_price >= purchase_price:
        raise_forms.ValidationError('El precio de venta debe ser mayor o igual que el precio de compra')

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

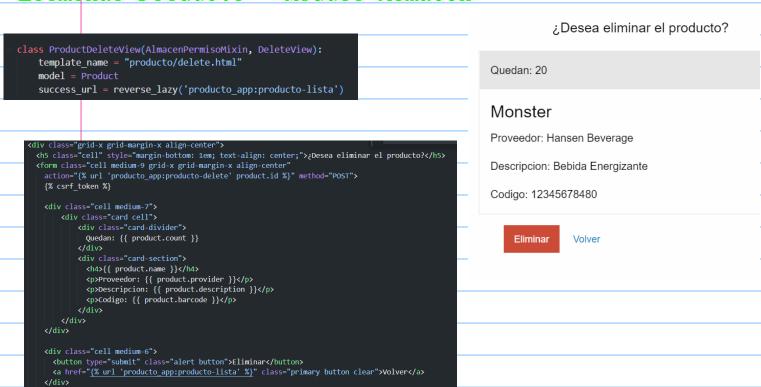
</form>
</div>



```
<div class="cell medium-6">
   <button type="submit" class="success button" style="width: 200px;">Guardar</button>
   {% if product %}
   <a href="{% url 'producto_app:producto-delete' product.id %}" class="alert button clear">Eliminar</a>
   {% endif %}
</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

```
def ventas_mes_producto(self, id_prod):
    # creamos rango de fecha
    end_date = timezone.now()
    start_date = end_date - timedelta(days=30)

    consulta = self.filter(
        sale_anulate=False,
        created_range=(start_date, end_date),
        product_pk=id_prod,
    ).values('sale_date_sale_date', 'product_name').annotate(
        cantidad_vendida=Sum('count'),
    )
    return consulta
```

```
{% for ventas in ventas_mes %}

{tr>
{{ ventas.sale__date_sale__date }}
{{ ventas.product__name }}
{{ ventas.product__name }}
{{ ventas.cantidad_vendida }}
{{ ventas.cantidad_vendida }}
{/tr>

{% empty %}
No hay ventas para este producto
{% endfor %}

</div>
```

- 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

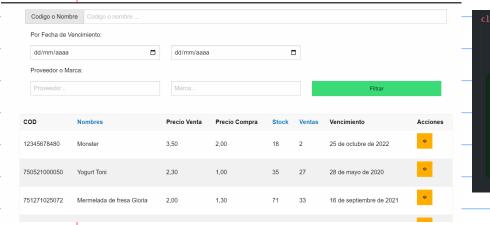
https://www.youtube.com/watch?v=N9iQm4N3H8s clase extra en youtube

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



```
class FiltrosProductListView(AlmacenPermisoMixin, ListView):
    template_name = "producto/filtros.html"
    context_object_name = 'productos'

def get_queryset(self):

    queryset = Product.objects.filtrar(
        kword=self.request.GET.get("kword", ''),
        date_start=self.request.GET.get("date_start", ''),
        date_end=self.request.GET.get("date_end", ''),
        provider=self.request.GET.get("provider", ''),
        order=self.request.GET.get("marca", ''),
        order=self.request.GET.get("order", ''),
    )
    return queryset
```

```
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')
```

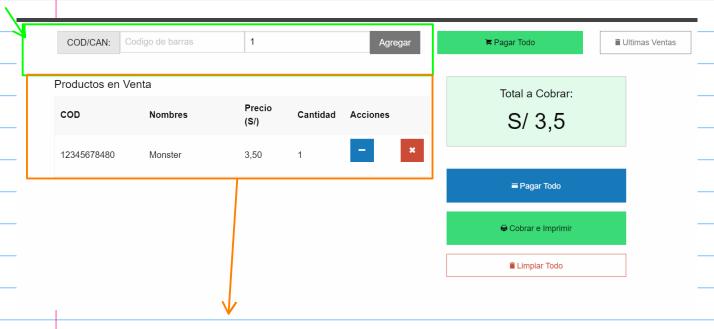
En este caso usa el objeto request dentro del HTML para recuperar toda la URL y le anida el parametro

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