




Modelos y Apps con las que trabaja el Proyecto

Por Nombre Por Fecha Por Stok → Notar que se puede ordenar por fecha, stock y nombre

COD	Nombres	Precio Venta	Precio Compra	Stok	Num Ventas	Acciones
750521000050	Yogurt Toni	2,30	1,00	35	27	 
702011075291	Milows	2,30	1,50	21	29	 

View

```
def get_queryset(self):
    kword = self.request.GET.get("kword", '')
    order = self.request.GET.get("order", '')
    queryset = Product.objects.buscar_producto(kword, order)
    return queryset
```

Manager

```
def buscar_producto(self, kword, order):
    consulta = self.filter(
        Q(name__icontains=kword) | Q(barcode=kword)
    )
    # verificamos en que orden se solicita
    if order == 'date':
        # ordenar por fecha
        return consulta.order_by('created')
    elif order == 'name':
        # ordenar por nombre
        return consulta.order_by('name')
    elif order == 'stok':
        return consulta.order_by('count')
    else:
        return consulta.order_by('-created')
```

Registrar Producto - Modulo Almacén

```
class ProductCreateView(AlmacenPermisoMixin, CreateView):
    template_name = "producto/form_producto.html"
    form_class = ProductForm
    success_url = reverse_lazy('producto_app:producto-lista')
```

```
# 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.get('purchase_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

Eliminar

```
<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

```
class ProductDeleteView(AlmacenPermisoMixin, DeleteView):
    template_name = "producto/delete.html"
    model = Product
    success_url = reverse_lazy('producto_app:producto-lista')
```

```
<div class="grid-x grid-margin-x align-center">
    <h5 class="cell" style="margin-bottom: 1em; text-align: center;">¿Desea eliminar el producto?</h5>
    <form class="cell medium-9 grid-x grid-margin-x align-center"
        action="{% url 'producto_app:producto-delete' product.id %}" method="POST">
        {% csrf_token %}

        <div class="cell medium-7">
            <div class="card cell">
                <div class="card-divider">
                    Quedan: {{ product.count }}
                </div>
                <div class="card-section">
                    <h4>{{ product.name }}</h4>
                    <p>Proveedor: {{ product.provider }}</p>
                    <p>Descripcion: {{ product.description }}</p>
                    <p>Codigo: {{ product.barcode }}</p>
                </div>
            </div>
        </div>

        <div class="cell medium-6">
            <button type="submit" class="alert button">Eliminar</button>
            <a href="{% url 'producto_app:producto-lista' %}" class="primary button clear">Volver</a>
        </div>
    </form>
</div>
```

¿Desea eliminar el producto?

Quedan: 20

Monster

Proveedor: Hansen Beverage

Descripcion: Bebida Energizante

Codigo: 12345678480

Eliminar

Volver

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 dias

Fecha	Producto	Ventas
11 de octubre de 2022	Monster	2

Imprimir

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

```
class ProductDetailView(AlmacenPermisoMixin, DetailView):
    template_name = "producto/detail.html"
    model = Product

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        #
        context["ventas_mes"] = SaleDetail.objects.ventas_mes_producto(
            self.kwargs['pk']
        )
        return context
```

```
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
```

```
<tbody>
{% for ventas in ventas_mes %}
<tr>
    <td>{{ ventas.sale_date_sale_date }}</td>
    <td>{{ ventas.product_name }}</td>
    <td>{{ ventas.cantidad_vendida }}</td>
</tr>
{% empty %}
<p style="color: green;">No hay ventas para este producto</p>
{% endfor %}
</tbody>
</table>
</div>
```

- si queremos restar una fecha en días usamos el `timedelta`
- si queremos el producto creado en un rango de fechas usamos el `created_range`
- si queremos 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 queremos hacer una operacion aritmetica adicional usamos el `annotate`

Fecha	Producto	Ventas
11 de octubre de 2022	Monster	2

Finalmente 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

[illegible]

Reportes en Módulo Almacén

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

Por Fecha de Vencimiento:

Proveedor o Marca:

Filtrar

COD	Nombres	Precio Venta	Precio Compra	Stock	Ventas	Vencimiento	Acciones
12345678480	Monster	3,50	2,00	18	2	25 de octubre de 2022	
750521000050	Yogurt Toni	2,30	1,00	35	27	28 de mayo de 2020	
751271025072	Mermelada de fresa Gloria	2,00	1,30	71	33	16 de septiembre de 2021	

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

    def get_queryset(self):

        queryset = Product.objects.filter(
            keyword=self.request.GET.get("keyword", ''),
            date_start=self.request.GET.get("date_start", ''),
            date_end=self.request.GET.get("date_end", ''),
            provider=self.request.GET.get("provider", ''),
            marca=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['keyword']) | Q(barcode=filters['keyword'])
    ).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>
<th>Precio Venta</th>
<th>Precio Compra</th>
<th>
{% if request.path == request.get_full_path %}
<a href="{{request.get_full_path}}?order=stok">Stock</a>
{% else %}
<a href="{{request.get_full_path}}?order=stok">Stock</a>
{% endif %}
</th>
```

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/?keyword=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..

COD/CAN: Codigo de barras 1 Agregar

Ultimas Ventas

Pagar Todo

Total a Cobrar:
S/ 3,5

Pagar Todo

Cobrar e Imprimir

Limpiar Todo

COD	Nombres	Precio (S/)	Cantidad	Acciones
12345678480	Monster	3,50	1	<div>-</div> <div>×</div>

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